



Sviluppo e distribuzione di moduli



Sviluppo e distribuzione di moduli

Nota

Prima di utilizzare queste informazioni, leggere le informazioni generiche nella sezione Avvisi alla fine di questo documento.

1 febbraio 2008

Questa edizione si applica alla versione 6, release 1, modifica 0, di WebSphere Process Server for Multiplatforms (numero prodotto 5724-L01) e a tutte le modifiche ed le release successive fino a quando non sarà diversamente indicato nelle nuove edizioni.

Per inviare i commenti relativi a questo documento, scrivere al seguente indirizzo: doc-comments@us.ibm.com. Attendiamo tali commenti.

L'IBM può utilizzare o distribuire le informazioni inviate dagli utenti secondo le modalità ritenute opportune senza incorrere in alcun obbligo nei confronti degli utenti stessi.

© Copyright International Business Machines Corporation 2005, 2008. Tutti i diritti riservati.

Indice

Figure **v**

Tabelle **vii**

Parte 1. Sviluppo delle applicazioni **1**

Capitolo 1. Panoramica della distribuzione di moduli **3**

Sviluppo di moduli del servizio 4

 Sviluppo dei componenti del servizio 5

 Richiamo dei componenti 7

 Panoramica relativa all'isolamento dei moduli e delle destinazioni 10

 Bind HTTP 14

Sostituzione dell'implementazione Service Component Architecture generata 15

Sostituzione di una conversione da Service Data Object a Java 16

Regole di runtime usate per le conversioni da Java a Service Data Object. 18

Capitolo 2. Sviluppo di applicazioni client per processi di business e attività **21**

Sviluppo di applicazioni client EJB per processi di business e attività umane. 21

 Accesso alle API EJB 22

 Interrogazione degli oggetti business-process e degli oggetti relativi alle attività 27

 Sviluppo delle applicazioni per i processi di business 63

 Sviluppo di applicazioni per attività umane 83

 Sviluppo di applicazioni per processi di business e attività umane 102

 Gestione delle eccezioni e degli errori 107

Sviluppo di applicazioni client per le API dei servizi Web 109

 Introduzione: servizi Web 109

 Componenti del servizio Web e sequenza di controllo 110

 Panoramica delle API dei servizi Web 110

 Requisiti per processi di business e attività umane. 111

 Sviluppo di applicazioni client. 111

 Copia di risorse 112

 Sviluppo di applicazioni client nell'ambiente dei servizi Web Java 120

 Sviluppo di applicazioni client nell'ambiente .NET 130

 Interrogazione degli oggetti business-process e degli oggetti relativi alle attività 134

Sviluppo di applicazioni client JMS 138

 Presentazione di JMS. 138

 Requisiti per processi di business. 139

 Accesso all'interfaccia JMS 139

 Struttura di un messaggio JMS Business Process Choreographer 141

 Autorizzazione per rendering JMS 142

 Panoramica della API JMS 143

 Sviluppo di applicazioni JMS 144

Sviluppo delle applicazioni Web per i processi di business e attività umane utilizzando i componenti JSF 146

 Aggiunta del componente Elenco ad un'applicazione JSF 152

 Aggiunta del componente Dettagli ad un'applicazione JSF 158

 Aggiunta del componente CommandBar ad un'applicazione JSF 161

 Aggiunta del componente Messaggio ad un'applicazione JSF 165

Sviluppo di pagine JSP per messaggi di attività e processo 168

 Frammenti JSP definiti dall'utente 169

Creazione di plug-in per personalizzare la funzionalità delle attività umane 170

 Creazione di gestori eventi API 171

 Creazione di gestori eventi di notifica 173

 Creazione di plug-in per la post-elaborazione dei risultati delle query persone 174

 Installazione di plug-in 176

 Registrazione di plug-in. 177

Parte 2. Distribuzione delle applicazioni **179**

Capitolo 3. Panoramica di preparazione e installazione dei moduli **181**

Panoramica delle librerie e dei file JAR. 181

Panoramica del file EAR. 183

Preparazione alla distribuzione su un server 184

Considerazioni sull'installazione delle applicazioni di servizio sui cluster. 185

Capitolo 4. Installazione di un modulo su un server di produzione **187**

Creazione di un file EAR che può essere installato utilizzando serviceDeploy 188

Distribuzione di applicazioni mediante le attività Apache Ant 188

Capitolo 5. Installazione di applicazioni di processo di business e attività umana **191**

Installazione di applicazioni di processo di business e attività umane in modalità interattiva 193

Configurazione dell'origine dati dell'applicazione del processo e delle impostazioni di riferimento definite	194
Disinstallazione delle applicazioni di processi di business e attività umane utilizzando la console di gestione	195
Disinstallazione delle applicazioni di processi di business e attività umane utilizzando i comandi di gestione	197
Capitolo 6. Installazione adattatori	199
Capitolo 7. Installazione delle applicazioni EIS	201
Distribuzione di un modulo dell'applicazione EIS esistente sulla piattaforma J2SE	202

Distribuzione di un modulo dell'applicazione EIS sulla piattaforma J2EE	203
---	-----

Capitolo 8. Risoluzione dei problemi relativi ad una distribuzione non riuscita	205
Eliminazione delle specifiche dell'attivazione J2C	206
Eliminazione delle destinazioni SIBus	207

Parte 3. Appendici 209

Informazioni particolari	211
---	------------

Figure

1. Modello di richiamo semplice	11	4. Modello di richiamo isolato che richiama UpdatedCalculateFinal.	14
2. Più applicazioni che richiamano un singolo servizio	12	5. Relazione tra modulo, componente e libreria	182
3. Modello di richiamo isolato che richiama UpdateCalculateFinal	13		

Tabelle

1.	Conversione da tipi WSDL a classi Java	19	26.	Metodi dell'API per il controllo del ciclo vita delle istanze di processo	82
2.	29	27.	Metodi dell'API per il controllo del ciclo vita delle istanze di attività	83
3.	Colonne nella vista ACTIVITY	41	28.	Metodi API per le variabili e le proprietà personalizzate	83
4.	Colonne nella vista ACTIVITY_ATTRIBUTE	42	29.	I metodi API per i modelli di attività	99
5.	Colonne nella vista ACTIVITY_SERVICE	43	30.	I metodi API per le istanze dell'attività	99
6.	Colonne nella vista APPLICATION_COMP	43	31.	Metodi API per effettuare operazioni con le escalation	100
7.	Colonne nella vista ESCALATION	44	32.	Metodi API per le variabili e le proprietà personalizzate	100
8.	Colonne nella vista ESCALATION_CPROP	46	33.	Associazione dei bind di riferimento ai nomi JNDI	148
9.	Colonne nella vista ESCALATION_DESC	46	34.	Modo in cui le interfacce di Business Process Choreographer sono associate agli oggetti del modello di client	151
10.	Colonne nella vista ESC_TEMPL	46	35.	Attributi bpe:list	157
11.	Colonne nella vista ESC_TEMPL_CPROP	48	36.	Attributi bpe:column	158
12.	Colonne nella vista ESC_TEMPL_DESC	48	37.	Attributi bpe:details	160
13.	Colonne nella vista PROCESS_ATTRIBUTE	49	38.	Attributi bpe:property	160
14.	Colonne nella vista PROCESS_INSTANCE	49	39.	Attributi bpe:commandbar	164
15.	Colonne nella vista PROCESS_TEMPLATE	50	40.	Attributi bpe:command	164
16.	Colonne nella vista QUERY_PROPERTY	51	41.	Attributi bpe:form	168
17.	Colonne nella vista TASK	52	42.	Associazione di bind con risorse J2EE	201
18.	Colonne nella vista TASK_CPROP	55	43.	Associazione di bind con risorse J2EE	203
19.	Colonna della vista TASK_DESC	55			
20.	Colonne nella vista TASK_TEMPL	55			
21.	Colonne nella vista TASK_TEMPL_CPROP	57			
22.	Colonne nella vista TASK_TEMPL_DESC	58			
23.	Colonne nella vista WORK_ITEM	58			
24.	Metodi API per i modelli di processo	81			
25.	I metodi API sono collegati all'avvio di istanze di processo	81			

Parte 1. Sviluppo delle applicazioni

Capitolo 1. Panoramica della distribuzione di moduli

Un modulo è l'unità di distribuzione di base per un'applicazione WebSphere Process Server. Un modulo contiene uno o più librerie del componente e i moduli di staging utilizzati dall'applicazione. Un componente può fare riferimento ad altri componenti del servizio. Lo sviluppo dei moduli assicura che i componenti, i moduli di staging e le librerie (raccolte di risorse cui il modulo fa riferimento) richiesti dall'applicazione siano disponibili sul server di produzione.

WebSphere Integration Developer è lo strumento principale per lo sviluppo dei moduli per la distribuzione in WebSphere Process Server. Sebbene sia possibile sviluppare i moduli in altri ambienti, è meglio utilizzare WebSphere Integration Developer.

WebSphere Process Server supporta due tipi di moduli di servizio: i moduli per i servizi di business e i moduli di mediazione. Un modulo per i servizi di business implementa la logica di un processo. Un modulo di mediazione consente la comunicazione tra le applicazioni trasformando il richiamo del servizio in un formato compreso dalla destinazione, inoltrando la richiesta alla destinazione e restituendo il risultato al creatore.

Le sezioni di seguito riportate illustrano il modo in cui implementare ed aggiornare i moduli su WebSphere Process Server.

Una sintesi dei componenti

Un componente è il blocco di creazione di base per integrare la logica di business riutilizzabile. Un componente del servizio viene associato alle interfacce, ai riferimenti a alle implementazioni. L'interfaccia definisce un contratto tra un componente del servizio e un componente di richiamo. Con WebSphere Process Server, un modulo del servizio può esportare un componente del servizio per l'utilizzo da parte di altri moduli o importare un componente del servizio da utilizzare. Per richiamare un componente del servizio, un modulo di richiamo fa riferimento all'interfaccia del componente del servizio. I riferimenti alle interfacce vengono risolti mediante la loro configurazione dal modulo di richiamo alle rispettive interfacce.

Per sviluppare un modulo, è necessario effettuare le seguenti attività:

1. Definire le interfacce per i componenti nel modulo
2. Definire, modificare o gestire gli oggetti business utilizzati dai componenti del servizio
3. Definire o modificare i componenti del servizio mediante le relative interfacce.

Nota: Un componente del servizio viene definito mediante la relativa interfaccia.

4. Facoltativamente, esportare o importare i componenti del servizio.
5. Creare un file EAR utilizzato per installare un modulo che utilizza i componenti. Creare il file utilizzando la funzione EAR di esportazione in WebSphere Integration Developer, o il comando serviceDeploy per creare un file EAR, per installare un modulo del servizio che utilizza i componenti del servizio.

Tipi di sviluppo

WebSphere Process Server fornisce un modello di programmazione del componente per semplificare un paradigma di programmazione orientato al servizio. Per utilizzare questo modello, un provider esporta le interfacce di un componente del servizio, in modo che un consumatore possa importare tali interfacce ed utilizzare il componente del servizio come se fosse locale. Uno sviluppatore utilizza le interfacce di tipo forte o dinamico per implementare o richiamare i componenti del servizio. Le interfacce e i relativi metodi sono descritti nella sezione Riferimenti all'interno di questo Infocenter.

Una volta installati i moduli del servizio nei server, è possibile utilizzare la console di gestione per modificare il componente di destinazione per un riferimento da un'applicazione. La nuova destinazione deve accettare lo stesso tipo di oggetto business ed eseguire la stessa operazione richiesta dal riferimento dell'applicazione.

Considerazioni di sviluppo di un componente del servizio

Quando si sviluppa un componente del servizio, prendere in considerazione i seguenti quesiti:

- Il componente del servizio verrà esportato ed utilizzato da un altro modulo?
In questo caso, assicurarsi che l'interfaccia definita per il componente possa essere utilizzata da un altro modulo.
- Il componente del servizio richiederà un intervallo di tempo relativamente lungo per essere eseguito?
In questo caso, considerare di implementare un'interfaccia asincrona al componente del servizio.
- È vantaggioso decentralizzare il componente del servizio?
Se dovesse esserlo, si consiglia di conservare una copia del componente del servizio in un modulo del servizio distribuito in un cluster o server per trarre vantaggio dall'elaborazione parallela.
- L'applicazione richiede una miscelanea delle risorse di commit fase 1 e fase 2?
In questo caso, assicurarsi di abilitare l'ultimo supporto partecipante per l'applicazione.

Nota: Se si crea l'applicazione utilizzando WebSphere Integration Developer o si crea il file EAR installabile utilizzando il comando `serviceDeploy`, tali strumenti abilitano automaticamente il supporto per l'applicazione. Consultare l'argomento, "Utilizzo delle risorse di commit a una o a due fasi nella stessa operazione" nel centro informazioni WebSphere Application Server Network Deployment.

Sviluppo di moduli del servizio

Un componente del servizio deve essere contenuto all'interno di un modulo del servizio. Lo sviluppo dei moduli del servizio, per contenere i componenti del servizio, è la chiave per fornire servizi agli altri moduli.

Prima di iniziare

Questa attività presuppone che da un'analisi dei requisiti risulti che l'implementazione di un componente del servizio per l'utilizzo da parte di altri moduli sia vantaggiosa.

About this task

Una volta analizzati i requisiti, è possibile stabilire se fornire ed utilizzare i componenti del servizio rappresenta un modo efficiente per elaborare le informazioni. Se si determina che i componenti del servizio riutilizzabili sono vantaggiosi per l'ambiente di cui si dispone, creare un modulo del servizio che contenga i componenti del servizio.

Procedure

1. Identificare i componenti del servizio che possono essere utilizzati dagli altri moduli.
Una volta identificati i componenti del servizio, continuare con l'argomento Sviluppo dei componenti del servizio.
2. Identificare i componenti del servizio all'interno di un'applicazione che potrebbe utilizzare i componenti del servizio in altri moduli di servizio.
Una volta identificati i componenti del servizio e i relativi componenti di destinazione, continuare con l'argomento Richiamo dei componenti.
3. Connettere i componenti del client con i componenti di destinazione mediante i collegamenti.

Sviluppo dei componenti del servizio

Sviluppare i componenti del servizio per fornire una logica riutilizzabile a più applicazioni all'interno del server.

Prima di iniziare

Ciò presuppone che l'elaborazione utile per più moduli sia già stata sviluppata ed identificata.

About this task

Più moduli possono utilizzare un componente del servizio. L'esportazione di un componente del servizio lo rende disponibile agli altri moduli che fanno riferimento al componente del servizio mediante un'interfaccia. Questa attività descrive il modo in cui creare il componente del servizio, in modo che possa essere utilizzato da altri moduli.

Nota: Un componente del servizio singolo può contenere più interfacce.

Procedure

1. Definire l'oggetto di dati per spostare i dati tra il chiamante e il componente del servizio.
L'oggetto di dati ed il relativo tipo è parte dell'interfaccia tra i chiamanti e il componente del servizio.
2. Definire un'interfaccia che i chiamanti utilizzeranno come riferimento al componente del servizio.
La definizione dell'interfaccia denomina il componente del servizio ed elenca eventuali metodi disponibili all'interno del componente del servizio.
3. Sviluppare la classe che definisce l'implementazione.
 - Se il componente è long running (o asincrono), continuare con il passo 4 a pagina 6.

- Se il componente non è long running (o sincrono), continuare con il passo 5.
4. Sviluppo di un'implementazione asincrona.

Importante: Un'interfaccia del componente asincrono non può disporre di una proprietà `joinTransaction` impostata su `true`.

- a. Definire l'interfaccia che rappresenta il componente del servizio sincrono.
 - b. Definire l'implementazione del componente del servizio.
 - c. Continuare con il passo 6.
5. Sviluppo di un'implementazione sincrona.
 - a. Definire l'interfaccia che rappresenta il componente del servizio sincrono.
 - b. Definire l'implementazione del componente del servizio.
 6. Salvare le interfacce del componente e le implementazioni nei file con un'estensione `.java`.
 7. Posizionare il modulo del servizio e le risorse necessarie in un file JAR.
Consultare l'argomento "Distribuzione di un modulo su un server di produzione" all'Infocenter per una descrizione dei passi da 7 a 9.
 8. Eseguire il comando `serviceDeploy` per creare un file EAR che può essere installato contenente l'applicazione.
 9. Installare l'applicazione sul nodo del server.
 10. Opzionale: Configurare i collegamenti tra i chiamanti e il componente del servizio corrispondente, se si richiama un componente del servizio in un altro modulo del servizio.

La sezione "Gestione" dell'Infocenter descrive la configurazione dei collegamenti.

Esempi dello sviluppo dei componenti

Questo esempio illustra un componente del servizio sincrono che implementa un metodo singolo, `CustomerInfo`. La prima sezione definisce l'interfaccia al componente del servizio che implementa un metodo denominato `getCustomerInfo`.

```
public interface CustomerInfo {  
    public Customer getCustomerInfo(String customerID);  
}
```

Il seguente blocco di codice implementa il componente del servizio.

```
public class CustomerInfoImpl implements CustomerInfo {  
    public Customer getCustomerInfo(String customerID) {  
        Customer cust = new Customer();  
  
        cust.setCustNo(customerID);  
        cust.setFirstName("Victor");  
        cust.setLastName("Hugo");  
        cust.setSymbol("IBM");  
        cust.setNumShares(100);  
        cust.setPostalCode(10589);  
        cust.setErrorMsg("");  
  
        return cust;  
    }  
}
```

Questo esempio sviluppa un componente del servizio asincrono. La prima sezione del codice definisce l'interfaccia sul componente del servizio che implementa un metodo denominato `getQuote`.


```
public interface StockQuote {
    public float getQuote(String symbol);
}
```

La sezione di seguito riportata è l'implementazione della classe associata a StockQuote.

```
public class StockQuoteImpl implements StockQuote {
    public float getQuote(String symbol) {
        return 100.0f;
    }
}
```

La sezione di codice di seguito riportata implementa l'interfaccia asincrona StockQuoteAsync.

```
public interface StockQuoteAsync {
    // risposta differita
    public Ticket getQuoteAsync(String symbol);
    public float getQuoteResponse(Ticket ticket, long timeout);
    // callback
    public Ticket getQuoteAsync(String symbol, StockQuoteCallback callback);
}
```

La sezione di seguito riportata è l'interfaccia StockQuoteCallback, che definisce il metodo onGetQuoteResponse,

```
public interface StockQuoteCallback {
    public void onGetQuoteResponse(Ticket ticket, float quote);
}
```

Operazioni successive

Richiamare il servizio.

Richiamo dei componenti

I componenti con i moduli possono utilizzare i componenti su qualunque nodo di un cluster WebSphere Process Server.

Prima di iniziare

Prima di richiamare un componente, assicurarsi che il modulo contenente il componente sia installato su WebSphere Process Server.

About this task

I componenti possono utilizzare qualunque componente del servizio disponibile all'interno di un cluster di WebSphere Process Server utilizzando il nome del componente e inoltrando il tipo di dati previsto dal componente stesso. Il richiamo di un componente nell'ambiente interessa la posizione e la creazione del riferimento al componente richiesto.

Nota: Un componente in un modulo può richiamare un componente all'interno dello stesso modulo, operazione nota come richiamo intra-modulo. Implementare i

richiami esterni (richiami interni del modulo) esportando l'interfaccia nel componente fornito e importando l'interfaccia nel componente di richiamo.

Importante: Durante il richiamo di un componente che si trova su un server diverso da quello su cui è in esecuzione il modulo di richiamo, è necessario eseguire ulteriori configurazioni ai server. Le configurazioni richieste dipendono dal richiamo sincrono o asincrono del componente. Il modo in cui configurare i server delle applicazioni in questo caso, viene descritto nelle attività correlate.

Procedure

1. Determinare i componenti richiesti dal modulo di richiamo.
Prendere nota del nome dell'interfaccia all'interno del componente e del tipo di dati richiesto dall'interfaccia.
2. Definire un oggetto di dati.
Sebbene l'input o la restituzione possa essere una classe Java, un oggetto dati del servizio rappresenta la soluzione ottimale.
3. Ricercare il componente.
 - a. Utilizzare la classe `ServiceManager` per ottenere i riferimenti disponibili al modulo di richiamo.
 - b. Utilizzare il metodo `locateService()` per ricercare il componente.
In base al componente, l'interfaccia può essere un tipo di porta WSDL (Web Service Descriptor Language) o un'interfaccia Java.
4. Richiamare il componente in modo sincrono o asincrono.
È possibile richiamare il componente mediante un'interfaccia Java oppure utilizzando il metodo `invoke()` per richiamare il componente in modo dinamico.
5. Elaborare la restituzione.
Il componente può restituire un'eccezione, pertanto il client deve elaborare questa possibilità.

Esempio del richiamo di un componente

L'esempio di seguito riportato crea una classe `ServiceManager`.

```
ServiceManager serviceManager = new ServiceManager();
```

Il seguente esempio utilizza la classe `ServiceManager` per ottenere un elenco di componenti da un file che contiene i riferimenti del componente.

```
InputStream myReferences = new FileInputStream("MyReferences.references");  
ServiceManager serviceManager = new ServiceManager(myReferences);
```

Il seguente codice ricerca un componente che implementa l'interfaccia Java `StockQuote`.

```
StockQuote stockQuote = (StockQuote)serviceManager.locateService("stockQuote");
```

Il seguente codice ricerca un componente che implementa un'interfaccia Java o di tipo porta WSDL. Il modulo di richiamo utilizza l'interfaccia `Service` per interagire con il componente.

Suggerimento: Se il componente implementa un'interfaccia Java, può essere richiamato mediante l'interfaccia o il metodo `invoke()`.

```
Service stockQuote = (Service)serviceManager.locateService("stockQuote");
```

L'esempio di seguito riportato illustra MyValue, il codice che richiama un altro componente.

```
public class MyValueImpl implements MyValue {

    public float myValue throws MyValueException {

        ServiceManager serviceManager = new ServiceManager();

        // variables
        Customer customer = null;
        float quote = 0;
        float value = 0;

        // invoke
        CustomerInfo cInfo =
        (CustomerInfo)serviceManager.locateService("customerInfo");
        customer = cInfo.getCustomerInfo(customerID);

        if (customer.getErrorMsg().equals("")) {

            // invoke
            StockQuoteAsync sQuote =
            (StockQuoteAsync)serviceManager.locateService("stockQuote");
            Ticket ticket = sQuote.getQuoteAsync(customer.getSymbol());
            // ... do something else ...
            quote = sQuote.getQuoteResponse(ticket, Service.WAIT);

            // assign
            value = quote * customer.getNumShares();
        } else {

            // throw
            throw new MyValueException(customer.getErrorMsg());
        }
        // reply
        return value;
    }
}
```

Operazioni successive

Configurare i collegamenti tra i riferimenti del modulo di richiamo e le interfacce del componente.

Richiamo dinamico di un componente

Quando un modulo richiama un componente che dispone di un'interfaccia di tipo porta WSDL (Web Service Descriptor Language), il modulo deve richiamare il componente in modo dinamico utilizzando il metodo invoke().

Prima di iniziare

Questa attività presuppone che un componente chiamante stia richiamando un componente in modo dinamico.

About this task

Con un'interfaccia di tipo porta WSDL, un componente chiamante deve utilizzare il metodo invoke() per richiamare il componente. Inoltre, un modulo di richiamo può richiamare un componente che dispone di un'interfaccia Java.

Procedure

1. Determinare il modulo che contiene il componente richiesto.
2. Determinare la matrice richiesta dal componente.
La matrice di input può essere di uno dei tre tipi che seguono:
 - Tipi Java in maiuscolo primitivi o matrici di questo tipo
 - Classi Java ordinarie o matrici delle classi
 - SDO (Service Data Objects)
3. Definire una matrice per contenere la risposta dal componente.
La matrice di risposta può essere dello stesso tipo della matrice di input.
4. Utilizzare il metodo `invoke()` per richiamare il componente richiesto e inoltrare l'oggetto della matrice al componente.
5. Elaborare il risultato.

Esempi di richiamo dinamico di un componente

Nel seguente esempio, un modulo utilizza il metodo `invoke()` per richiamare un componente che utilizza tipi di dati Java di lettere maiuscole di origine.

```
Service service = (Service)serviceManager.locateService("multiParamInf");
```

```
Reference reference = service.getReference();
```

```
OperationType methodMultiType =  
    reference.getOperationType("methodWithMultiParameter");
```

```
Type t = methodMultiType.getInputType();
```

```
BOFactory boFactory = (BOFactory)serviceManager.locateService  
    ("com/ibm/websphere/bo/BOFactory");
```

```
DataObject paramObject = boFactory.createbyType(t);
```

```
paramObject.set(0,"input1")  
paramObject.set(1,"input2")  
paramObject.set(2,"input3")
```

```
service.invoke("methodMultiParamater",paramObject);
```

L'esempio di seguito riportato utilizza il metodo `invoke` con un'interfaccia di tipo porta WSDL come destinazione.

```
Service serviceOne = (Service)serviceManager.locateService("multiParamInfWSDL");
```

```
DataObject dob = factory.create("http://MultiCallWSServerOne/bos", "SameBO");  
dob.setString("attribute1", stringArg);
```

```
DataObject wrapBo = factory.createElement  
    ("http://MultiCallWSServerOne/wsd1/ServerOneInf", "methodOne");  
wrapBo.set("input1", dob); //wrapBo include tutti i parametri di methodOne  
wrapBo.set("input2", "XXXX");  
wrapBo.set("input3", "yyyy");
```

```
DataObject resBo= (DataObject)serviceOne.invoke("methodOne", wrapBo);
```

Panoramica relativa all'isolamento dei moduli e delle destinazioni

Quando si sviluppano i moduli, verranno identificati i servizi che possono essere utilizzati da più moduli. Potenziando i servizi in questo modo, vengono minimizzati il ciclo di sviluppo ed i costi. Quando un servizio viene utilizzato da molti moduli, è necessario isolare i moduli di richiamo dalla destinazione, in modo che se venisse aggiornata la destinazione, il passaggio al nuovo servizio sia noto al

modulo di richiamo. Questo argomento confronta il modello di richiamo semplice e il modello di richiamo isolato, fornendo un esempio del modo in cui l'isolamento può essere utile. Questo esempio specifico non rappresenta il solo modo di isolare i moduli dalle destinazioni.

Modello di richiamo semplice

Durante lo sviluppo di un modulo, è possibile utilizzare i servizi che si trovano in altri moduli. È possibile effettuare tale operazione importando il servizio nel modulo e richiamando quel servizio. Il servizio importato è "collegato" al servizio esportato dall'altro modulo in WebSphere Integration Developer o mediante il bind del servizio nella console di gestione. Il modello di richiamo semplice illustra questo modello.

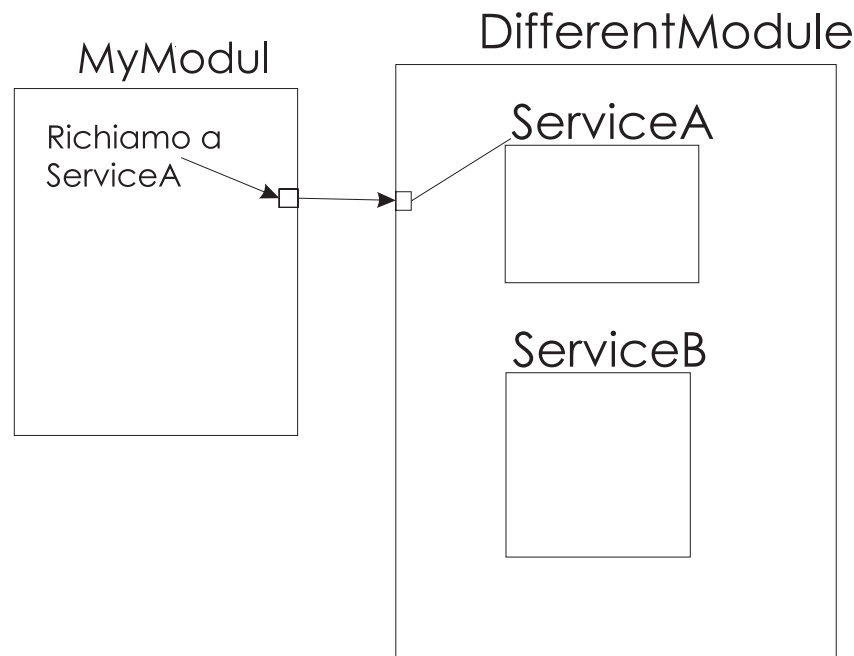


Figura 1. Modello di richiamo semplice

Modello di richiamo isolato

Per modificare la destinazione di un richiamo senza arrestare i moduli di richiamo, è possibile isolare i moduli di richiamo dalla destinazione del richiamo. Ciò consente ai moduli di continuare l'elaborazione mentre si modifica la destinazione, poiché non si sta modificando il modulo, ma la destinazione derivata. L'esempio di isolamento delle applicazioni illustra il modo in cui l'isolamento consente di modificare la destinazione senza condizionare lo stato del modulo di richiamo.

Esempio di isolamento delle applicazioni

Utilizzando il modello di richiamo semplice, più moduli che richiamano lo stesso servizio sembrerebbero Più applicazioni che richiamano un singolo servizio. MODA, MODB e MODC richiamano tutti CalculateFinalCost.

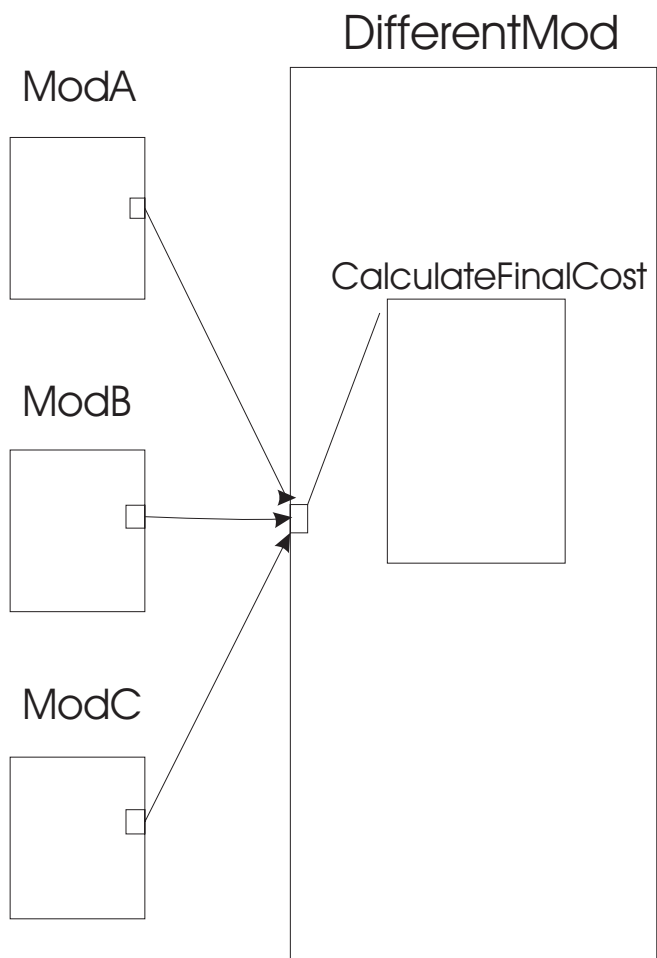


Figura 2. Più applicazioni che richiamano un singolo servizio

Il servizio fornito da CalculateFinalCost deve essere aggiornato, pertanto i nuovi costi riguardano tutti i moduli che utilizzano il servizio. Il team di sviluppo crea e verifica un nuovo servizio UpdatedCalculateFinal per incorporare le modifiche. Si è pronti a portare in produzione il nuovo servizio. Senza isolamento, sarebbe stato necessario aggiornare tutti i moduli che richiamano CalculateFinalCost per richiamare UpdatedCalculateFinal. Con l'isolamento, è necessario modificare solo il bind che collega il modulo di buffer alla destinazione.

Nota: La modifica del servizio in questo modo consente di continuare a fornire il servizio di origine agli altri moduli che possono necessitare.

Utilizzando l'isolamento, si crea un modulo di buffer tra le applicazioni e la destinazione (consultare Modello di richiamo isolato che richiama UpdateCalculateFinal).

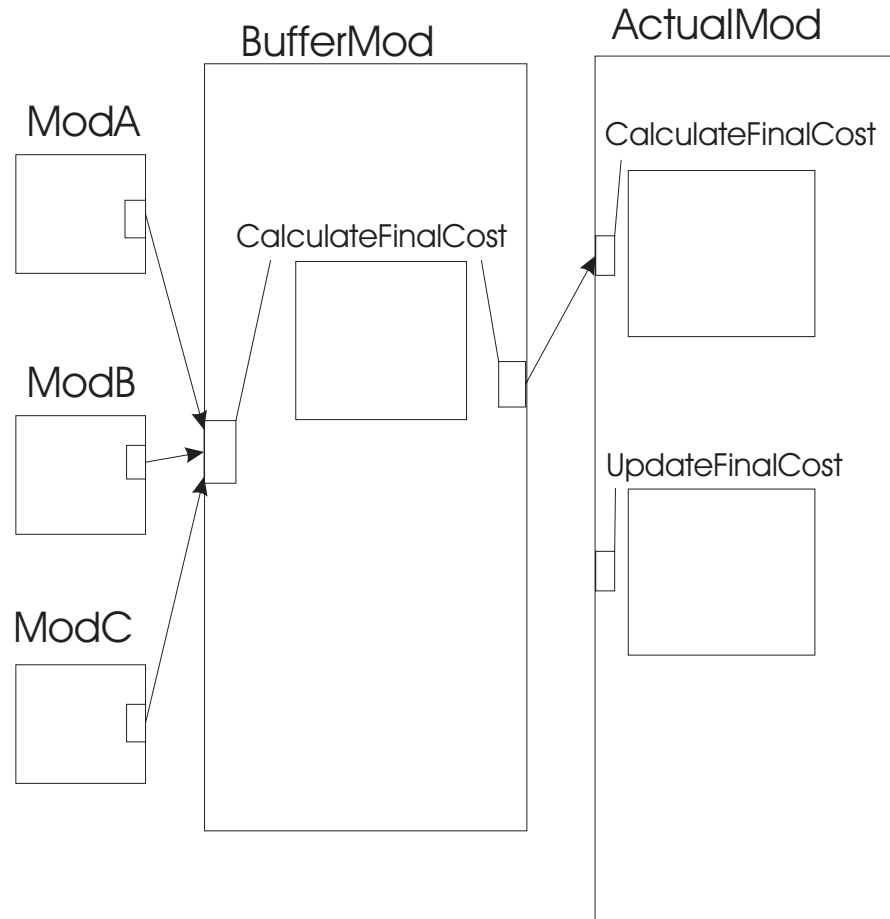


Figura 3. Modello di richiamo isolato che richiama UpdateCalculateFinal

Con questo modello, i moduli di richiamo non cambiano, è necessario solo modificare il bind dall'importazione del modulo di buffer alla destinazione (consultare Modello di richiamo isolato che richiama UpdatedCalculateFinal).

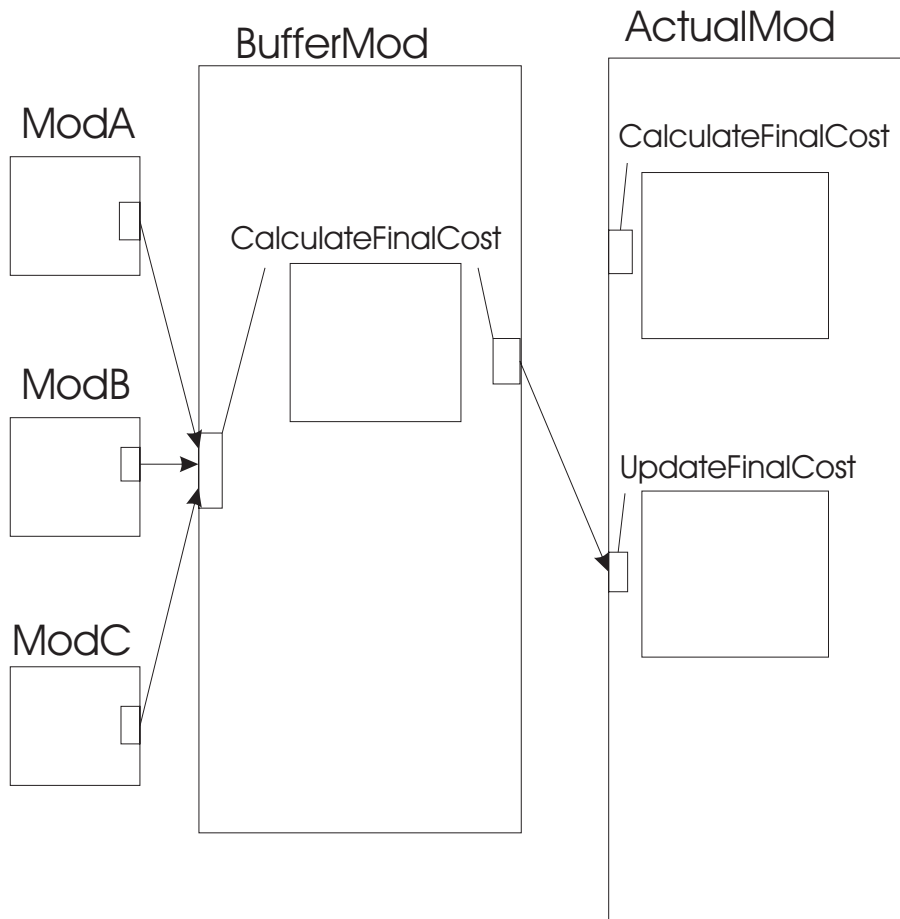


Figura 4. Modello di richiamo isolato che richiama UpdatedCalculateFinal

Se il modulo di buffer richiama la destinazione in modo sincronico, quando si riavvia il modulo di buffer (un modulo di mediazione o un servizio per il modulo di business) i risultati restituiti all'applicazione di origine provengono dalla nuova destinazione. Se il modulo di buffer richiama la destinazione in modo asincronico, i risultati restituiti all'applicazione di origine provengono dalla nuova destinazione al successivo richiamo.

Attività correlate

Modifica delle destinazioni

La modifica della destinazione di un riferimento conferisce alle applicazioni la flessibilità di poter approfittare dei miglioramenti dei componenti nel momento in cui vengono apportati, senza dover ricompilare e reinstallare l'applicazione.

Bind HTTP

Il bind HTTP è progettato per fornire connettività SCA (Service Component Architecture) al protocollo HTTP. Ciò consente ad applicazioni HTTP nuove o esistenti di partecipare ad ambienti Service Oriented Architecture (SOA).

Inoltre, una rete di ambienti di runtime SCA può comunicare attraverso un'infrastruttura HTTP esistente.

Il bind HTTP espone diverse funzioni HTTP:

- I messaggi vengono presentati ai componenti di mediazione in modo da conservare il formato HTTP e le informazioni dell'intestazione di messaggio. Ciò consente ad amministratori, utenti e programmatori di applicazioni HTTP di avere riferimento noto.
- Un framework esistente di bind dei dati viene esteso alle convenzioni HTTP e fornisce una associazione tra i messaggi SCA e intestazione e corpo dei messaggi HTTP.
- Impostazioni ed esportazioni possono essere configurate in modo da supportare una serie di comuni funzioni HTTP.
- Quando si installa un modulo SCA contenente importazioni o esportazioni HTTP, l'ambiente di runtime viene automaticamente configurato in modo da consentire la connettività HTTP.

Per maggiori informazioni su come creare importazioni ed esportazioni HTTP consultare il centro informazioni in **WebSphere Integration Developer > Sviluppo di applicazioni di integrazione > Bind dei dati HTTP**.

Attività correlate

 Visualizzazione dei bind HTTP

Dopo aver distribuito un'applicazione, è possibile esaminare i bind HTTP per assicurarsi che siano corretti.

 Modifica dei bind di esportazione HTTP

La console di gestione permette di modificare la configurazione dei bind di esportazione HTTP senza modificare la fonte originale e in seguito di ridistribuire l'applicazione.

 Modifica dei bind di importazione HTTP

La console di gestione permette di modificare la configurazione dei bind di importazione HTTP senza modificare la fonte originale e in seguito di ridistribuire l'applicazione.

Sostituzione dell'implementazione Service Component Architecture generata

A volte, la conversione creata dal sistema tra un codice Java e un Service Data Object (SDO) può non soddisfare le necessità dell'utente. Usare questa procedura per sostituire l'implementazione predefinita delle classi SCA (Service Component Architecture) con la propria.

Prima di iniziare

Assicurarsi di aver generato la conversione tipi da Java a WSDL (Web Services Definition Language) usando WebSphere Integration Developer oppure il comando genMapper.

About this task

È possibile sostituire un componente generato che associa un tipo Java su un tipo WSDL sostituendo il codice generato con il codice necessario alle proprie necessità. Si consiglia di utilizzare le proprie associazioni se sono state definite le proprie classi Java. Utilizzare questa procedura per apportare le modifiche.

Procedure

1. Individuare il componente generato. Il nome del componente è `classe_javaMapper.component`.
2. Modificare il componente usando un editor di testo.
3. Commentare il codice generato e inserire il proprio metodo.
Non modificare il nome del file che contiene l'implementazione del componente.

Il seguente è un esempio di componente generato da sostituire:

```
private DataObject javatodata_setAccount_output(Object myAccount) {  
  
    // Questo codice può essere sostituito per una associazione personalizzata.  
    // Commentare questo codice e scrivere il codice personalizzato.  
  
    // È inoltre possibile modificare il tipo Java che viene passato al  
    // convertitore e che il convertitore cerca di creare.  
  
    return SDOJavaObjectMediator.java2Data(myAccount);  
  
}
```

Copiare il componente e gli altri file nella directory in cui si trova il modulo contenitore, quindi collegare il componente in WebSphere Integration Developer oppure generare un file EAR (enterprise archive) con il comando `serviceDeploy`.

Concetti correlati

“Regole di runtime usate per le conversioni da Java a Service Data Object” a pagina 18

Per sostituire correttamente il codice generato, oppure per determinare le possibili eccezioni di runtime relative alle conversioni da Java a SDO (Service Data Object), è importante comprendere le regole coinvolte in tale processo. Per la maggior parte, le conversioni son semplici e dirette; tuttavia vi sono alcuni casi più complessi in cui il runtime offre le migliori possibilità per la conversione del codice generato.

Riferimenti correlati

 [Conversione da Java a XML](#)

Il sistema genera XML basati su tipi Java utilizzando regole predefinite.

 [Comando genMapper](#)

Utilizzare il comando `genMapper` per generare un componente che collega un riferimento Service Component Architecture (SCA) a un'interfaccia Java.

Sostituzione di una conversione da Service Data Object a Java

A volte, la conversione che il sistema crea tra un Service Data Object (SDO) e un oggetto di tipo Java può non soddisfare le necessità dell'utente. Usare questa procedura per sostituire l'implementazione predefinita con la propria.

Prima di iniziare

Assicurarsi di aver generato la conversione tipi da WSDL a Java usando WebSphere Integration Developer oppure il comando `genMapper`.

About this task

È possibile sostituire un componente generato che associa un tipo WSDL su un tipo Java sostituendo il codice generato con il codice necessario alle proprie necessità. Si consiglia di utilizzare le proprie associazioni se sono state definite le proprie classi Java. Utilizzare questa procedura per apportare le modifiche.

Procedure

1. Individuare il componente generato. Il nome del componente è *classe_javaMapper.component*.
2. Modificare il componente usando un editor di testo.
3. Commentare il codice generato e inserire il proprio metodo.
Non modificare il nome del file che contiene l'implementazione del componente.

Il seguente è un esempio di componente generato da sostituire:

```
private Object datatojava_get_customerAcct(DataObject myCustomerID,
    String integer)
{
    // Questo codice può essere sostituito per una associazione personalizzata.
    // Commentare questo codice e scrivere il codice personalizzato.

    // È inoltre possibile modificare il tipo Java che viene passato al
    // convertitore e che il convertitore cerca di creare.

    return SDOJavaObjectMediator.data2Java(customerID, integer) ;
}
```

Copiare il componente e gli altri file nella directory in cui si trova il modulo contenitore, quindi collegare il componente in WebSphere Integration Developer oppure generare un file EAR (enterprise archive) con il comando serviceDeploy.

Concetti correlati

“Regole di runtime usate per le conversioni da Java a Service Data Object” a pagina 18

Per sostituire correttamente il codice generato, oppure per determinare le possibili eccezioni di runtime relative alle conversioni da Java a SDO (Service Data Object), è importante comprendere le regole coinvolte in tale processo. Per la maggior parte, le conversioni son semplici e dirette; tuttavia vi sono alcuni casi più complessi in cui il runtime offre le migliori possibilità per la conversione del codice generato.

Riferimenti correlati

 Conversione da Java a XML

Il sistema genera XML basati su tipi Java utilizzando regole predefinite.

 Comando genMapper

Utilizzare il comando genMapper per generare un componente che collega un riferimento Service Component Architecture (SCA) a un'interfaccia Java.

Regole di runtime usate per le conversioni da Java a Service Data Object

Per sostituire correttamente il codice generato, oppure per determinare le possibili eccezioni di runtime relative alle conversioni da Java a SDO (Service Data Object), è importante comprendere le regole coinvolte in tale processo. Per la maggior parte, le conversioni son semplici e dirette; tuttavia vi sono alcuni casi più complessi in cui il runtime offre le migliori possibilità per la conversione del codice generato.

Tipi e classi di base

Il runtime esegue una conversione diretta tra tipi e classi di base Service Data Object e Java. I tipi e classi di base comprendono:

- Char o `java.lang.Character`
- Boolean
- `Java.lang.Boolean`
- Byte o `java.lang.Byte`
- Short o `java.lang.Short`
- Int o `java.lang.Integer`
- Long o `java.lang.Long`
- Float o `java.lang.Float`
- Double o `java.lang.Double`
- `Java.lang.String`
- `Java.math.BigInteger`
- `Java.math.BigDecimal`
- `Java.util.Calendar`
- `Java.util.Date`
- `Java.xml.namespace.QName`
- `Java.net.URI`
- `Byte[]`

Classi e array Java definiti dall'utente

Quando una classe o un array Java viene convertito in un SDO, il runtime crea un oggetto dati con un URI generato invertendo il nome di package del tipo Java e un tipo equivalente al nome della classe Java. Per esempio, la classe `Java.com.ibm.xsd.Customer` viene convertita in un SDO con URI `http://xsd.ibm.com` e tipo `Customer`. Il runtime esamina quindi il contenuto dei membri della classe Java e assegna valori alle proprietà dell'SDO.

Nella conversione di un tipo da SDO a Java, il runtime genera il nome del package invertendo l'URI, e il nome del tipo è equivalente al tipo dell'SDO. Per esempio, l'oggetto dati con tipo `Customer` e URI `http://xsd.ibm.com` genera un'istanza del package Java `com.ibm.xsd.Customer`. Il runtime estrae quindi i valori dalle proprietà dell'SDO e assegna tali proprietà ai campi dell'istanza della classe Java.

Quando la classe Java è un'interfaccia definita dall'utente, è necessario sostituire il codice generato e fornire una classe concreta che il runtime possa istanziare. Se il runtime non riesce a creare la classe concreta si verifica un'eccezione.

`Java.lang.Object`

Quando un tipo Java è `java.lang.Object`, il tipo generato è `xsd:anyType`. Un modulo può richiamare questa interfaccia con qualsiasi SDO. Il runtime tenterà di

istanziare una classe concreta analogamente a quanto avviene con classi e array Java definiti dall'utente, se il runtime riesce a individuare tale classe. Altrimenti, il runtime trasmette l'SDO all'interfaccia Java.

Anche nel caso in cui il metodo restituisca un tipo `java.lang.Object`, il runtime lo converte in SDO solo se il metodo restituisce un tipo concreto. Il runtime usa una conversione simile alla conversione di classi e array Java definiti dall'utente in SDO, come descritto nel paragrafo seguente.

Quando una classe o un array Java viene convertito in un SDO, il runtime crea un oggetto dati con un URI generato invertendo il nome di package del tipo Java e un tipo equivalente al nome della classe Java. Per esempio, la classe `com.ibm.xsd.Customer` viene convertita in un SDO con URI `http://xsd.ibm.com` e tipo `Customer`. Il runtime esamina quindi il contenuto dei membri della classe Java e assegna valori alle proprietà dell'SDO.

In entrambi i casi, se il runtime non riesce a completare la conversione si verifica un'eccezione.

Classi contenitore Java.util

Nella conversione a una classe contenitore Java concreta come `Vector`, `HashMap`, `HashSet` e simili, il runtime crea un'istanza della classe contenitore appropriata. Il runtime usa un metodo simile a quello usato per classi e array Java definiti dall'utente per popolare la classe contenitore. Se il runtime non riesce a individuare una classe Java concreta, popolerà la classe contenitore con l'SDO.

Quando si convertono classi contenitore Java concrete a SDO, il runtime usa gli schemi generati mostrati in "Java per la conversione XML."

Java.util interfaces

Per alcune interfacce contenitore nel package `java.util`, il runtime crea istanze delle seguenti classi concrete:

Tabella 1. Conversione da tipi WSDL a classi Java

Interfaccia	Classi concrete predefinite
Collection	HashSet
Map	HashMap
List	ArrayList
Set	HashSet

Attività correlate

"Sostituzione dell'implementazione Service Component Architecture generata" a pagina 15

A volte, la conversione creata dal sistema tra un codice Java e un Service Data Object (SDO) può non soddisfare le necessità dell'utente. Usare questa procedura per sostituire l'implementazione predefinita delle classi SCA (Service Component Architecture) con la propria.

"Sostituzione di una conversione da Service Data Object a Java" a pagina 16

A volte, la conversione che il sistema crea tra un Service Data Object (SDO) e un oggetto di tipo Java può non soddisfare le necessità dell'utente. Usare questa procedura per sostituire l'implementazione predefinita con la propria.

Riferimenti correlati

Conversione da Java a XML

Il sistema genera XML basati su tipi Java utilizzando regole predefinite.

Comando genMapper

Utilizzare il comando genMapper per generare un componente che collega un riferimento Service Component Architecture (SCA) a un'interfaccia Java.

Capitolo 2. Sviluppo di applicazioni client per processi di business e attività

È possibile utilizzare uno strumento di modellamento per creare e distribuire processi di business e attività. Tali processi e attività sono interconnessi al runtime, ad esempio, un processo viene avviato o le attività vengono richieste e completate. È possibile utilizzare Business Process Choreographer Explorer per interagire con processi e attività oppure utilizzare le API di Business Process Choreographer per sviluppare client personalizzati per tali interazioni.

About this task

Questi client possono essere i client Enterprise JavaBeans (EJB), i client del servizio Web o i client Web che presentano componenti JSF (JavaServer Faces) di Business Process Choreographer Explorer. Business Process Choreographer fornisce API Enterprise JavaBeans (EJB) e interfacce per servizi Web per poter sviluppare tali client. È possibile accedere alle API EJB da un'applicazione Java, inclusa un'altra applicazione EJB. Alle interfacce per servizi Web si può accedere da ambienti Java o da ambienti Microsoft .Net.

Sviluppo di applicazioni client EJB per processi di business e attività umane

Le API EJB forniscono una serie di metodi generici di sviluppo di applicazioni client EJB per lavorare con i processi di business e le attività umane installati su un WebSphere Process Server.

About this task

Con queste API Enterprise JavaBeans (EJB), è possibile creare applicazioni client per le seguenti operazioni:

- Gestione del ciclo di vita di processi e attività dall'avvio all'eliminazione, una volta completati
- Riparazione di attività e processi
- Gestione e distribuzione del carico di lavoro tra i membri di un gruppo di lavoro

Le API EJB vengono fornite come due bean enterprise di sessione senza stato:

- L'interfaccia `BusinessFlowManagerService` fornisce i metodi per le applicazioni del processo di business
- L'interfaccia `HumanTaskManagerService` fornisce i metodi per le applicazioni basate sulle attività

Per ulteriori informazioni sulle API EJB, consultare Javadoc nel package `com.ibm.bpe.api` e `com.ibm.task.api`.

La procedura che segue fornisce una panoramica delle azioni necessarie per sviluppare un'applicazione client EJB.

Procedure

1. Decidere le funzioni che devono essere fornite dall'applicazione.
2. Stabilire quali bean di sessione si desidera utilizzare.
In base agli scenari che si desidera implementare con l'applicazione, è possibile utilizzare uno o entrambi i bean di sessione.
3. Determinare le autorizzazioni necessarie agli utenti dell'applicazione.
Gli utenti dell'applicazione devono disporre dei ruoli di autorizzazione appropriati per richiamare i metodi inclusi nell'applicazione e visualizzare gli oggetti e gli attributi degli oggetti restituiti dai metodi. Quando un'istanza dell'appropriato bean di sessione viene creata, WebSphere Application Server associa un contesto all'istanza. Il contesto contiene informazioni sull'ID principale del chiamante, sull'elenco di appartenenti al gruppo e sui ruoli. Queste informazioni vengono utilizzate per controllare l'autorizzazione del chiamante per ogni chiamata.
Javadoc contiene informazioni sull'autorizzazione per ciascun metodo.
4. Stabilire come eseguire il rendering dell'applicazione.
Le API EJB possono essere richiamate localmente o in remoto.
5. Sviluppare l'applicazione.
 - a. Accedere all'API EJB.
 - b. Utilizzare l'API EJB per interagire con i processi o le attività.
 - Eseguire la query dei dati.
 - Eseguire operazioni con i dati.

Accesso alle API EJB

Le API Enterprise JavaBeans (EJB) sono fornite come due bean enterprise di sessione senza stato. Le applicazioni dei processi di business e le applicazioni delle attività accedono al bean enterprise di sessione appropriato mediante l'interfaccia home del bean.

About this task

L'interfaccia `BusinessFlowManagerService` fornisce i metodi per le applicazioni del processo di business e l'interfaccia `HumanTaskManagerService` fornisce i metodi per le applicazioni basate sulle attività. L'applicazione può essere una qualunque applicazione Java, compresa un'altra applicazione Enterprise JavaBeans (EJB).

Accedere all'interfaccia remota del bean di sessione

Un'applicazione client EJB accede all'interfaccia remota del bean di sessione attraverso l'interfaccia home remota del bean.

About this task

Il bean di sessione può essere il bean di sessione `BusinessFlowManager` per le applicazioni del processo o il bean di sessione `HumanTaskManager` per le applicazioni dell'attività.

Procedure

1. Aggiungere un riferimento all'interfaccia remota del bean di sessione al descrittore di distribuzione dell'applicazione. Aggiungere un riferimento ad uno dei seguenti file:
 - Il file `application-client.xml` per un'applicazione client Java 2 Platform, Enterprise Edition (J2EE)
 - Il file `web.xml`, per un'applicazione Web

- Il file `ejb-jar.xml` per un'applicazione Enterprise JavaBeans (EJB)

Il riferimento all'interfaccia home remota per le applicazioni del processo viene illustrato nel seguente esempio:

```
<ejb-ref>
  <ejb-ref-name>ejb/BusinessFlowManagerHome</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <home>com.ibm.bpe.api.BusinessFlowManagerHome</home>
  <remote>com.ibm.bpe.api.BusinessFlowManager</remote>
</ejb-ref>
```

Il riferimento all'interfaccia home remota per le applicazioni dell'attività è illustrato nel seguente esempio:

```
<ejb-ref>
  <ejb-ref-name>ejb/HumanTaskManagerHome</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <home>com.ibm.task.api.HumanTaskManagerHome</home>
  <remote>com.ibm.task.api.HumanTaskManager</remote>
</ejb-ref>
```

Se si utilizza WebSphere Integration Developer per aggiungere il riferimento EJB al descrittore di distribuzione, il bind del riferimento EJB viene creato automaticamente al momento della distribuzione dell'applicazione. Per ulteriori informazioni sull'aggiunta dei riferimenti EJB, fare riferimento alla documentazione di WebSphere Integration Developer.

2. Comprimere gli stub creati con l'applicazione.

Se l'applicazione viene eseguita su una JVM (Java Virtual Machine) diversa da quella in cui è in esecuzione l'applicazione BPEContainer o TaskContainer, effettuare le seguenti azioni.

- Per le applicazioni del processo, comprimere il file `<root_installazione>/ProcessChoreographer/client/bpe137650.jar` con il file EAR (enterprise archive) dell'applicazione.
- Per le applicazioni dell'attività, comprimere il file `<root_installazione>/ProcessChoreographer/client/task137650.jar` con il file EAR dell'applicazione di cui si dispone.
- Impostare il parametro **Classpath** nel file manifest del modulo dell'applicazione per includere il file JAR.
Il modulo dell'applicazione può essere un'applicazione J2EE, un'applicazione Web o un'applicazione EJB.
- Se si utilizzano tipi di dati complessi nel processo di business o nell'attività umana e il client non viene eseguito in un'applicazione EJB o in un'applicazione Web, comprimere i file XSD o WSDL corrispondenti con il file EAR dell'applicazione.

3. Ubicare l'interfaccia home remota del bean di sessione mediante il Java Naming and Directory Interface (JNDI).

L'esempio di seguito riportato illustra tale passo per un'applicazione del processo:

```
// Ottenere il contesto JNDI iniziale predefinito
InitialContext initialContext = new InitialContext();

// Ricercare l'interfaccia home remota del bean BusinessFlowManager
Object result =
    initialContext.lookup("java:comp/env/ejb/BusinessFlowManagerHome");

// Convertire il risultato della ricerca nel tipo adatto
BusinessFlowManagerHome processHome =
    (BusinessFlowManagerHome) javax.rmi.PortableRemoteObject.narrow
    (result, BusinessFlowManagerHome.class);
```

L'interfaccia home remota del bean di sessione contiene un metodo di creazione per oggetti EJB. Il metodo restituisce l'interfaccia remota del bean di sessione.

4. Accedere all'interfaccia remota del bean di sessione.

L'esempio di seguito riportato illustra tale passo per un'applicazione del processo:

```
BusinessFlowManager process = processHome.create();
```

L'accesso al bean di sessione non garantisce l'esecuzione da parte del chiamante di tutte le azioni fornite dal bean; inoltre, il chiamante deve essere autorizzato a eseguire tali azioni. Quando viene creata un'istanza del bean di sessione, un contesto viene associato all'istanza. Il contesto contiene l'ID principale del chiamante, l'elenco di appartenenti al gruppo e indica se il chiamante ha uno dei ruoli J2EE di Business Process Choreographer. Il contesto viene utilizzato per controllare l'autorizzazione del chiamante per ogni chiamata, anche quando non è impostata la sicurezza globale. Se la sicurezza globale non è impostata, l'ID principale del chiamante ha il valore UNAUTHENTICATED.

5. Richiamare le funzioni di business esposte dall'interfaccia del servizio.

L'esempio di seguito riportato illustra tale passo per un'applicazione del processo:

```
process.initiate("MyProcessModel",input);
```

Le chiamate dalle applicazioni vengono effettuate come transazioni. Una transazione viene stabilita e terminata in uno dei seguenti modi:

- Automaticamente da WebSphere Application Server (il descrittore di distribuzione specifica TX_REQUIRED).
- In modo esplicito dall'applicazione. È possibile raggruppare le chiamate dell'applicazione in un'unica transazione:

```
// Ottenere l'interfaccia transazione utente
UserTransaction transaction=
    (UserTransaction)initialContext.lookup("jta/usertransaction");

// Iniziare una transazione
transaction.begin();

// Chiamate delle applicazioni ...

// Sulla restituzione riuscita, eseguire il commit della transazione
transaction.commit();
```

Suggerimento: Per evitare conflitti di blocco del database, evitare di eseguire istruzioni simili alle seguenti in transazioni parallele:

```
// Ottenere l'interfaccia transazione utente
UserTransaction transaction=
    (UserTransaction)initialContext.lookup("jta/usertransaction");

transaction.begin();

//leggere l'istanza di attività
process.getActivityInstance(aiid);
//richiedere l'istanza di attività
process.claim(aiid);

transaction.commit();
```

Il metodo getActivityInstance e altre operazioni di lettura impostano un blocco di lettura. In questo esempio, viene eseguito l'upgrade da un blocco di lettura

sull'istanza di attività a un blocco aggiornato sull'istanza di attività. Questo può comportare un deadlock del database quando le transazioni vengono eseguite in parallelo.

Esempio

Di seguito viene riportato un esempio dei passi da 3 a 5 per l'applicazione di un'attività.

```
// Ottenere il contesto JNDI iniziale predefinito
InitialContext initialContext = new InitialContext();

// Ricercare l'interfaccia home remota del bean HumanTaskManager
Object result =
    initialContext.lookup("java:comp/env/ejb/HumanTaskManagerHome");

// Convertire il risultato della ricerca nel tipo adatto
HumanTaskManagerHome taskHome =
    (HumanTaskManagerHome)javax.rmi.PortableRemoteObject.narrow
    (result,HumanTaskManagerHome.class);

...

//Accedere all'interfaccia remota del bean di sessione.
HumanTaskManager task = taskHome.create();

...

//Richiamare le funzioni di business esposte dall'interfaccia del servizio.
task.callTask(tkiid,input);
```

Accedere all'interfaccia locale del bean di sessione

Un'applicazione client EJB accede all'interfaccia locale del bean di sessione attraverso l'interfaccia home locale del bean.

About this task

Il bean di sessione può essere il bean di sessione BusinessFlowManager per applicazioni di processo o il bean di sessione HumanTaskManager per applicazioni di attività umana.

Procedure

1. Aggiungere un riferimento all'interfaccia locale del bean di sessione al descrittore di distribuzione dell'applicazione. Aggiungere un riferimento ad uno dei seguenti file:

- Il file application-client.xml per un'applicazione client Java 2 Platform, Enterprise Edition (J2EE)
- Il file web.xml, per un'applicazione Web
- Il file ejb-jar.xml per un'applicazione Enterprise JavaBeans (EJB)

Il riferimento all'interfaccia home locale per le applicazioni del processo viene illustrato nel seguente esempio:

```
<ejb-local-ref>
  <ejb-ref-name>ejb/LocalBusinessFlowManagerHome</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <local-home>com.ibm.bpe.api.LocalBusinessFlowManagerHome</local-home>
  <local>com.ibm.bpe.api.LocalBusinessFlowManager</local>
</ejb-local-ref>
```

Il riferimento all'interfaccia home locale per le applicazioni dell'attività è illustrato nel seguente esempio:

```

<ejb-local-ref>
  <ejb-ref-name>ejb/LocalHumanTaskManagerHome</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <local-home>com.ibm.task.api.LocalHumanTaskManagerHome</local-home>
  <local>com.ibm.task.api.LocalHumanTaskManager</local>
</ejb-local-ref>

```

Se si utilizza WebSphere Integration Developer per aggiungere il riferimento EJB al descrittore di distribuzione, il bind del riferimento EJB viene creato automaticamente al momento della distribuzione dell'applicazione. Per ulteriori informazioni sull'aggiunta dei riferimenti EJB, fare riferimento alla documentazione di WebSphere Integration Developer.

2. Ubicare l'interfaccia home locale del bean di sessione attraverso Java Naming and Directory Interface (JNDI).

L'esempio di seguito riportato illustra tale passo per un'applicazione del processo:

```

// Ottenere il contesto JNDI iniziale predefinito
InitialContext initialContext = new InitialContext();

// Consultare l'interfaccia home locale del bean BusinessFlowManager

LocalBusinessFlowManagerHome processHome =
    (LocalBusinessFlowManagerHome)initialContext.lookup
    ("java:comp/env/ejb/LocalBusinessFlowManagerHome");

```

L'interfaccia home locale del bean di sessione contiene un metodo di creazione per oggetti EJB. Il metodo restituisce l'interfaccia locale del bean di sessione.

3. Accedere all'interfaccia locale del bean di sessione.

L'esempio di seguito riportato illustra tale passo per un'applicazione del processo:

```
LocalBusinessFlowManager process = processHome.create();
```

L'accesso al bean di sessione non garantisce l'esecuzione da parte del chiamante di tutte le azioni fornite dal bean; inoltre, il chiamante deve essere autorizzato a eseguire tali azioni. Quando viene creata un'istanza del bean di sessione, un contesto viene associato all'istanza. Il contesto contiene l'ID principale del chiamante, l'elenco di appartenenti al gruppo e indica se il chiamante ha uno dei ruoli J2EE di Business Process Choreographer. Il contesto viene utilizzato per controllare l'autorizzazione del chiamante per ogni chiamata, anche quando non è impostata la sicurezza globale. Se la sicurezza globale non è impostata, l'ID principale del chiamante ha il valore UNAUTHENTICATED.

4. Richiamare le funzioni di business esposte dall'interfaccia del servizio.

L'esempio di seguito riportato illustra tale passo per un'applicazione del processo:

```
process.initiate("MyProcessModel",input);
```

Le chiamate dalle applicazioni vengono effettuate come transazioni. Una transazione viene stabilita e terminata in uno dei seguenti modi:

- Automaticamente da WebSphere Application Server (il descrittore di distribuzione specifica TX_REQUIRED).
- In modo esplicito dall'applicazione. È possibile raggruppare le chiamate dell'applicazione in un'unica transazione:

```

// Ottenere l'interfaccia transazione utente
UserTransaction transaction=
    (UserTransaction)initialContext.lookup("jta/usertransaction");

// Iniziare una transazione
transaction.begin();

```

```
// Chiamate delle applicazioni ...

// Sulla restituzione riuscita, eseguire il commit della transazione
transaction.commit();
```

Suggerimento: Per evitare deadlock del database, evitare di eseguire istruzioni simili alle seguenti in transazioni parallele:

```
// Ottenere l'interfaccia transazione utente
UserTransaction transaction=
    (UserTransaction)initialContext.lookup("jta/usertransaction");

transaction.begin();

//leggere l'istanza di attività
process.getActivityInstance(aiid);
//richiedere l'istanza di attività
process.claim(aiid);

transaction.commit();
```

Il metodo `getActivityInstance` e altre operazioni di lettura impostano un blocco di lettura. In questo esempio, viene eseguito l'aggiornamento da un blocco di lettura sull'istanza di attività a un blocco aggiornato sull'istanza di attività. Questo può comportare un deadlock del database quando le transazioni vengono eseguite in parallelo

Esempio

Di seguito viene riportato un esempio dei passi da 2 a 4 per l'applicazione di un'attività.

```
// Ottenere il contesto JNDI iniziale predefinito
InitialContext initialContext = new InitialContext();

//Consultare l'interfaccia home locale del bean HumanTaskManager
LocalHumanTaskManagerHome taskHome =
    (LocalHumanTaskManagerHome)initialContext.lookup
    ("java:comp/env/ejb/LocalHumanTaskManagerHome");

...
//Accedere all'interfaccia locale del bean di sessione
LocalHumanTaskManager task = taskHome.create();

...
//Richiamare le funzioni di business esposte dall'interfaccia del servizio.
task.callTask(tkiid,input);
```

Interrogazione degli oggetti business-process e degli oggetti relativi alle attività

Le applicazioni client funzionano con gli oggetti business-process e con gli oggetti relativi alle attività. È possibile eseguire una query degli oggetti processo di business o degli oggetti relativi alle attività nel database per richiamare le proprietà specifiche di tali oggetti.

About this task

Durante la configurazione di Business Process Choreographer, un database relazionale è associato al contenitore di processi di business e al contenitore

dell'attività. Il database memorizza tutti i dati dei modelli e delle istanze (runtime) per la gestione dei processi di business e delle attività. Utilizzare la sintassi simile a SQL per eseguire la query di tali dati.

Eseguire una query offline per recuperare una particolare proprietà di un oggetto. Inoltre, salvare le query utilizzate spesso e includerle nell'applicazione.

Query sugli oggetti del processo di business e relativi alle attività

Utilizzare il metodo query o il metodo queryAll del servizio per richiamare le informazioni memorizzate sui processi di business e le attività.

Il metodo query può anche essere chiamato da tutti gli utenti, e riporta le proprietà degli oggetti per cui esistono elementi di lavoro. Il metodo queryAll può essere chiamato solo da utenti che hanno uno dei seguenti ruoli J2EE:

BPESystemAdministrator, TaskSystemAdministrator, BPESystemMonitor, o TaskSystemMonitor. Questo metodo riporta le proprietà di tutti gli oggetti memorizzati nel database.

Tutte le query API sono associate alle query SQL. La forma della conseguente query SQL dipende dai seguenti aspetti:

- Se una query è stata invocata da qualcuno con uno dei ruoli J2EE.
- Gli oggetti che sono richiesti nella query. Le viste del database predefinite sono fornite per eseguire la query delle proprietà dell'oggetto.
- L'inserimento di una clausola from, condizioni comuni e condizioni specifiche per utente per controllo accesso.

È possibile includere nelle query proprietà personalizzate e variabili. Se si includono varie proprietà personalizzate o variabili nella query, vengono generati self join nella tabella di database corrispondente. In base al sistema di database, le chiamate query() potrebbero avere implicazioni sulle prestazioni.

È possibile anche memorizzare le query nel database Business Process Choreographer utilizzando il metodo createStoredQuery. Quando si definisce la query memorizzata, vengono forniti i criteri di query. I criteri vengono applicati in modo dinamico quando la query memorizzata viene eseguita, vale a dire che i dati vengono assemblati al momento del runtime. Se la query memorizzata contiene parametri, anche i parametri vengono risolti al momento dell'esecuzione della query.

Per ulteriori informazioni sulle API di Business Process Choreographer, consultare Javadoc nel package com.ibm.bpe.api per i metodi relativi al processo e nel package com.ibm.task.api per i metodi relativi alle attività.

Sintassi del metodo di query API:

La sintassi delle query API Business Process Choreographer è simile a quella di query SQL. Una query può comprendere una clausola select, una clausola where, una clausola order-by, un parametro skip-tuples, un parametro threshold e un parametro time-zone.

La sintassi della query dipende dal tipo di oggetto. Nella tabella che segue viene riportata la sintassi di ciascuno dei diversi tipi di oggetto.

Tabella 2.

Oggetto	Sintassi
Modello processo	ProcessTemplateData[] queryProcessTemplates (java.lang.String whereClause, java.lang.String orderByClause, java.lang.Integer threshold, java.util.TimeZone timezone);
Modello attività	TaskTemplate[] queryTaskTemplates (java.lang.String whereClause, java.lang.String orderByClause, java.lang.Integer threshold, java.util.TimeZone timezone);
Dati correlati all'attività e al processo di business	QueryResultSet query (java.lang.String selectClause, java.lang.String whereClause, java.lang.String orderByClause, java.lang.Integer skipTuples java.lang.Integer threshold, java.util.TimeZone timezone);

Clausola Select:

La clausola select nella funzione di query identifica le proprietà dell'oggetto che devono essere restituite da una query.

La clausola select descrive il risultato query. Specifica un elenco di nomi che identificano le proprietà dell'oggetto (le colonne del risultato) da restituire. La sua sintassi è simile a quella di una clausola SELECT SQL; utilizzare virgole per separare parti della clausola. Ciascuna parte della clausola deve specificare una colonna da una delle viste predefinite. La colonna deve essere pienamente specificata per nome vista e nome colonna. Le colonne restituite nell'oggetto QueryResultSet compaiono nello stesso ordine delle colonne specificate nella clausola select.

La clausola select non supporta le funzioni di aggregazione SQL, come ad esempio AVG(), SUM(), MIN() o MAX().

Per selezionare le proprietà di più coppie nome-valore, come ad esempio le proprietà personalizzate e le proprietà delle variabili che possono essere interrogate, aggiungere un suffisso ad un carattere al nome visualizzato. Questo numero può avere un valore compreso tra 1 e 9.

Esempi delle clausole select

- "WORK_ITEM.OBJECT_TYPE, WORK_ITEM.REASON"
Richiama i tipi degli oggetti associati e i motivi di assegnazione degli elementi di lavoro.
- "DISTINCT WORK_ITEM.OBJECT_ID"
Ottiene tutti gli ID degli oggetti, senza duplicati, per cui il chiamante dispone di un elemento di lavoro.
- "ACTIVITY.TEMPLATE_NAME, WORK_ITEM.REASON"
Richiama i nomi delle attività di cui il chiamante ha gli elementi di lavoro e le relative cause di assegnazione.
- "ACTIVITY.STATE, PROCESS_INSTANCE.STARTER"
Richiama gli stati delle attività e gli starter delle istanze di processo associate.
- "DISTINCT TASK.TKIID, TASK.NAME"

Ottiene tutti gli ID e i nomi delle attività, senza duplicati, per cui il chiamante dispone di un elemento di lavoro.

- "TASK_CPROP1.STRING_VALUE, TASK_CPROP2.STRING_VALUE"
Ottiene i valori delle proprietà personalizzate che sono specificate ulteriormente nella clausola where.
- "QUERY_PROPERTY1.STRING_VALUE, QUERY_PROPERTY2.INT_VALUE"
Richiama i valori delle proprietà delle variabili che possono essere interrogate. Queste parti sono specificate ulteriormente nella clausola where.
- "COUNT(DISTINCT TASK.TKIID)"
Conta il numero di elementi di lavoro per le attività univoche che soddisfano la clausola where.

Clausola Where:

La clausola where nella query descrive i criteri di filtraggio da applicare al dominio della query.

La sintassi di una clausola where è simile a quella di una clausola WHERE SQL. Non è necessario aggiungere esplicitamente una clausola from SQL o unire predicati alla clausola where API, questi costrutti vengono aggiunti automaticamente quando viene eseguita la query. Se non si desidera applicare i criteri di filtro, è necessario specificare null per la clausola where.

La sintassi della clausola where supporta:

- Parole chiave: AND, OR, NOT
- Operatori di confronto: =, <=, <, <>, >, >=, LIKE
L'operazione LIKE supporta i caratteri jolly definiti per il database in cui viene eseguita la query.
- Operatori di impostazione: IN

Vengono applicate anche le seguenti regole:

- Specificare le costanti ID come ID('string-rep-of-oid').
- Specificare le costanti binarie come BIN('UTF-8 string').
- Utilizzare le costanti simboliche al posto delle enumerazioni di numeri interi. Ad esempio, invece di specificare un'espressione dello stato di attività ACTIVITY.STATE=2, specificare ACTIVITY.STATE=ACTIVITY.STATE.STATE_READY.
- Se il valore della proprietà nell'istruzione di confronto contiene gli apici ('), duplicare gli apici, ad esempio, "TASK_CPROP.STRING_VALUE='d'automatisation'".
- Fare riferimento alle proprietà delle coppie nome-valore, come ad esempio le proprietà personalizzate, aggiungendo un suffisso di una cifra al nome di visualizzazione. Ad esempio: "TASK_CPROP1.NAME='prop1' AND "TASK_CPROP2.NAME='prop2'".
- Specificare le costanti data/ora come TS('aaaa-mm-ggThh:mm:ss'). Per fare riferimento alla data corrente, specificare CURRENT_DATE come data/ora.
È necessario specificare almeno un valore data o ora nella data/ora:
 - Se si specifica solo una data, il valore ora è impostato su zero.
 - Se si specifica solo l'ora, la data viene impostata su quella corrente.
 - Se si specifica una data, l'anno deve essere costituito da quattro cifre; i valori mese e giorno sono facoltativi. I valori mese e giorno mancanti sono impostati su 01. Ad esempio, TS('2003') è uguale a TS('2003-01-01T00:00:00').

- Se si indica un'ora, questi valori vengono espressi nel sistema di 24 ore. Ad esempio, se la data corrente è 1 gennaio 2003, TS('T16:04') o TS('16:04') è uguale a TS('2003-01-01T16:04:00').

Esempi di clausole where

- Confronto di un ID oggetto con un ID esistente

```
"WORK_ITEM.WIID = ID('_WI:800c00ed.df8d7e7c.feffff80.38')"
```

Normalmente, questo tipo di clausola where viene creata in modo dinamico con un ID oggetto esistente di una chiamata precedente. Se questo ID oggetto viene memorizzato in una variabile *wiid1*, la clausola può essere costruita come:

```
"WORK_ITEM.WIID = ID('" + wiid1.toString() + '" )"
```

- Uso degli indicatori di data/ora

```
"ACTIVITY.STARTED >= TS('2002-06-1T16.00.00')"
```

- Uso delle costanti simboliche

```
"WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_OWNER"
```

- Utilizzo dei valori booleani true e false

```
"ACTIVITY.BUSINESS_RELEVANCE = TRUE"
```

- Utilizzo delle proprietà personalizzate

```
"TASK_CPROP1.NAME = 'prop1' AND " TASK_CPROP1.STRING_VALUE = 'v1' AND  
TASK_CPROP2.NAME = 'prop2' AND " TASK_CPROP2.STRING_VALUE = 'v2' "
```

Clausola Order-by:

La clausola order-by nella funzione di query specifica i criteri di ordinamento della serie di risultati della query.

È possibile specificare un elenco di colonne dalle viste da cui viene ordinato il risultato. Tali colonne devono essere interamente qualificate dal nome della vista e colonna. L'approccio migliore è quello di specificare colonne nella clausola select.

La sintassi della clausola order-by è simile a quella di una clausola order-by SQL; usare le virgole per separare ciascuna parte della clausola. È inoltre possibile specificare ASC per ordinare le colonne in ordine crescente e DESC per ordinare le colonne in ordine decrescente. Se non si desidera ordinare la serie di risultati della query, è necessario specificare null per la clausola order-by.

I criteri di ordinamento vengono applicati al server, ovvero la locale del server è usata per l'ordinamento. Se si specifica più di una colonna, la serie di risultati della query viene ordinata in base ai valori della prima colonna, quindi in base ai valori della seconda colonna e così via. Non è possibile specificare le colonne nella clausola order-by per posizione come in una query SQL.

Esempi delle clausole order-by

- "PROCESS_TEMPLATE.NAME"

Ordina i risultati della query alfabeticamente in base al nome del modello di processo.

- "PROCESS_INSTANCE.CREATED, PROCESS_INSTANCE.NAME DESC"

Ordina i risultati della query per data di creazione e, per una data specifica, sistema i risultati per nome istanza di processo, in ordine alfabetico inverso.

- "ACTIVITY.OWNER, ACTIVITY.TEMPLATE_NAME, ACTIVITY.STATE"

Ordina il risultato query per proprietario attività, per nome modello attività e per stato dell'attività.

Parametro skip-tuples:

Il parametro skip-tuples specifica il numero di tuple dall'inizio della serie dei risultati di query da ignorare e non restituire al chiamante nella serie dei risultati di query.

Utilizzare questo parametro con il parametro di soglia per implementare il paging in un'applicazione client, ad esempio, per richiamare i primi 20 elementi, quindi i successivi 20 elementi e così via.

Se questo parametro viene impostato su null e il parametro di soglia non viene impostato, vengono restituite tutte le tuple di qualifica.

Esempio di un parametro skip-tuples

- new Integer(5)

Specifica che le prime cinque tuple di qualifica non devono essere restituite.

Parametro Threshold:

Il parametro di soglia nella funzione query limita il numero di oggetti restituiti dal server al client nella serie di risultati della query.

Poiché gli insiemi di risultati della query negli scenari di produzione possono contenere migliaia o milioni di elementi, si consiglia di specificare sempre una soglia. Il parametro soglia può essere utile, ad esempio, in una GUI (graphical user interface) in cui dovrebbe essere visualizzato solo un piccolo numero di elementi. Se si imposta il parametro della soglia di conseguenza, la query al database è più veloce ed è necessario trasferire un numero minore di dati dal server al client.

Se questo parametro viene impostato su null e il parametro skip-tuples non viene impostato, vengono restituiti tutti gli oggetti di qualifica.

Esempio di un parametro di soglia

- new Integer(50)

Specifica che devono essere restituite 50 tuple di qualifica.

Parametro di fuso orario:

Il parametro relativo al fuso orario nella funzione di query definisce il fuso orario per le costanti data/ora nella query.

I fusi orari possono variare tra il client che avvia la query e il server che la elabora. Utilizzare il parametro fuso orario per specificare il fuso orario delle costanti data/ora usate nella clausola where, ad esempio, per specificare le ore locali. Le date restituite nella serie di risultati della query hanno lo stesso fuso orario di quelle specificate nella query.

Se il parametro è impostato su null, le costanti data/ora vengono considerate ore UTC (Coordinated Universal Time).

Esempi dei parametri di fuso orario

- process.query("ACTIVITY.AIID",
"ACTIVITY.STARTED > TS('2005-01-01T17:40')",
(String)null,
(Integer)null,
java.util.TimeZone.getDefault());

Restituisce gli ID degli oggetti per le attività che sono state avviate dopo le 17:40 ora locale il 1 gennaio 2005.

- `process.query("ACTIVITY.AIID",
"ACTIVITY.STARTED > TS('2005-01-01T17:40')",
(String)null, (Integer)null, (TimeZone)null);`

Restituisce ID di oggetti per le attività che sono state avviate dopo le 17:40 UTC del 1 gennaio 2005. In questo caso, ad esempio, si tratta di 6 ore precedenti l'ora standard orientale.

Parametri nelle query memorizzate:

Una query memorizzata è una query che viene memorizzata nel database ed identificata mediante un nome. Le tuple di qualifica vengono assemblate in modo dinamico al momento dell'esecuzione della query. Per rendere riutilizzabili le query memorizzate, è possibile utilizzare i parametri nella definizione di query risolti al momento del runtime.

Ad esempio, sono state definite proprietà personalizzate per memorizzare i nomi dei clienti. È possibile definire query per tornare le attività associate a un cliente particolare, ACME Co. Per svolgere la query di queste informazioni, la clausola dove nella query può assomigliare all'esempio che segue:

```
String whereClause =  
"TASK.STATE = TASK.STATE.STATE_READY  
AND WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER  
AND TASK_CPROP.NAME = 'company' AND TASK_CPROP.STRING_VALUE = 'ACME Co.'";
```

Per rendere questa query riutilizzabile in modo da potervi ricercare anche il cliente, BCME Ltd, è possibile utilizzare parametri relativi ai valori della proprietà personalizzata. Se si aggiungono parametri alla query dell'attività, il risultato potrebbe essere simile al seguente esempio:

```
String whereClause =  
"TASK.STATE = TASK.STATE.STATE_READY  
AND WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER  
AND TASK_CPROP.NAME = 'company' AND TASK_CPROP.STRING_VALUE = '@param1'";
```

Il parametro @param1 viene risolto al runtime dall'elenco di parametri trasferiti al metodo query. Le regole di seguito riportate vengono applicate all'utilizzo dei parametri nelle query:

- I parametri possono essere utilizzati solo nella clausola where.
- I parametri sono stringhe.
- I parametri vengono sostituiti al runtime mediante sostituzioni di stringa. Se sono necessari caratteri speciali, è necessario specificarli nella clausola where o trasferirli al runtime come parte del parametro.
- I nomi dei parametri sono costituiti dalla stringa @param concatenata al numero intero. Il numero più basso è 1, che indica il primo elemento nell'elenco di parametri trasferito all'API della query al runtime.
- Un parametro può essere utilizzato più volte in una clausola where, tutte le ricorrenze del parametro vengono sostituite dallo stesso valore.

Attività correlate

“Gestione delle query memorizzate” a pagina 60

Le query memorizzate forniscono una modalità di salvataggio delle query eseguite di frequente. La query memorizzata può essere una query disponibile per tutti gli utenti (query pubblica) o una query che appartiene a un utente specifico (query privata).

Risultati query:

Una serie di risultati della query contiene i risultati di una query.

Gli elementi della serie di risultati sono proprietà degli oggetti che soddisfano la clausola where data dal chiamante, e che il chiamante è autorizzato a vedere. È possibile leggere gli elementi in modo relativo mediante il metodo next API o in modo assoluto mediante i metodi first e last. Poiché il cursore implicito di una serie di risultati della query viene posizionato inizialmente davanti al primo elemento, è necessario richiamare il metodo first o next prima di leggere un elemento. È possibile utilizzare il metodo size per stabilire il numero di elementi nella serie.

Un elemento della serie di risultati della query comprende gli attributi selezionati degli elementi di lavoro e i relativi oggetti di riferimento associati, come ad esempio le istanze attività e le istanze del processo. Il primo attributo (colonna) di un elemento QueryResultSet indica il valore del primo attributo specificato nella clausola select della richiesta di query. Il secondo attributo (colonna) di un elemento QueryResultSet indica il valore del secondo attributo specificato nella clausola select della richiesta di query, e così via.

È possibile richiamare i valori degli attributi chiamando un metodo che sia compatibile con il tipo di attributo e specificando l'indice della colonna appropriata. La numerazione degli indici delle colonne inizia da 1.

Tipo di attributo	Metodo
String	getString
OID	getOID
Timestamp	getTimestamp getString getTimestampAsLong
Integer	getInteger getShort getLong getString getBoolean
Boolean	getBoolean getShort getInteger getLong getString
byte[]	getBinary

Esempio:

Viene eseguita la seguente query:

```
QueryResultSet resultSet =  
process.query("ACTIVITY.STARTED,  
ACTIVITY.TEMPLATE_NAME AS NAME,  
WORK_ITEM.WIID, WORK_ITEM.REASON",  
(String)null, (String)null,  
(Integer)null, (TimeZone)null);
```

La serie di risultati della query restituita ha quattro colonne:

- La colonna 1 è un indicatore di data/ora

- La colonna 2 è una stringa
- La colonna 3 è un ID oggetto
- La colonna 4 è un numero intero

È possibile utilizzare i seguenti metodi per richiamare i valori attributo:

```
while (resultSet.next())
{
    java.util.Calendar activityStarted = resultSet.getTimestamp(1);
    String templateName = resultSet.getString(2);
    WIID wiid = (WIID) resultSet.getOID(3);
    Integer reason = resultSet.getInteger(4);
}
```

È possibile utilizzare i nomi visualizzati della serie di risultati, ad esempio, come le intestazioni della tabella stampata. Questi nomi rappresentano i nomi colonna della vista o il nome definito dalla clausola AS nella query. È possibile utilizzare il seguente metodo per richiamare i nomi visualizzati nell'esempio:

```
resultSet.getColumnDisplayName(1) returns "STARTED"
resultSet.getColumnDisplayName(2) returns "NAME"
resultSet.getColumnDisplayName(3) returns "WIID"
resultSet.getColumnDisplayName(4) returns "REASON"
```

Condizioni di accesso specifiche dell'utente:

Le condizioni di accesso specifiche dell'utente vengono aggiunte quando l'istruzione SQL SELEZIONA è generata dalla query API. Tali condizioni garantiscono che siano restituiti al chiamante solo quegli oggetti che soddisfano la condizione specificata dal chiamante e a cui il chiamante è autorizzato.

La condizione di accesso aggiunta dipende dal fatto che l'utente sia o meno un amministratore di sistema.

Query chiamate da utenti che non sono amministratori di sistema

La clausola SQL WHERE generata combina le clausole dove API con una condizione per il controllo accesso specifica dell'utente. La query richiama solo quegli oggetti a cui l'utente è autorizzato ad accedere, ossia solo quegli oggetti per cui l'utente dispone di un elemento di lavoro. Un elemento di lavoro rappresenta l'assegnazione di un utente o gruppo utenti a un ruolo di autorizzazione di un oggetto business, quale un'attività o un processo. Se, per esempio, l'utente, John Smith, è un membro del ruolo di proprietario potenziale di una data attività, esiste un oggetto elemento di lavoro che rappresenta questo rapporto.

Per esempio, se un utente, che non è amministratore di sistema, effettua una query delle attività, la seguente condizione di accesso è aggiunta alla clausola WHERE se gli elementi di lavoro di gruppo non sono abilitati:

```
FROM TASK TA, WORK_ITEM WI
WHERE WI.OBJECT_ID = TA.TKIID
AND ( WI.OWNER_ID = 'user'
      OR WI.OWNER_ID = null AND WI.EVERYBODY = vero )
```

Così, se John Smith desidera un elenco delle attività per cui è proprietario potenziale, la clausola dove API apparirà come segue:

```
"WORK_ITEM.REASON == WORK_ITEM.REASON.REASON_POTENTIAL_OWNER"
```

Da tale clausola WHERE API deriva la seguente condizione di accesso nell'istruzione SQL:

```

FROM TASK TA, WORK_ITEM WI
WHERE WI.OBJECT_ID = TA.TKIID
AND ( WI.OWNER_ID = 'JohnSmith'
      OR WI.OWNER_ID = null AND WI.EVERYBODY = vero)
AND WI.REASON = 1

```

Questo significa inoltre che se John Smith desidera vedere le attività per cui è lettore di processo o un amministratore di processo e per cui non dispone di un elemento di lavoro, deve essere aggiunta una proprietà da PROCESS_INSTANCE, alla clausola SELEZIONA, DOVE, o ORDINATO PER della query, per esempio, PROCESS_INSTANCE.PIID.

Se sono abilitati elementi di lavoro di gruppo, viene aggiunta una condizione di accesso supplementare alla clausola WHERE che consente a un utente di accedere a oggetti a cui il gruppo ha accesso.

Query chiamate da utenti amministratori di sistema

Gli amministratori di sistema possono chiamare il metodo query per richiamare oggetti che hanno elementi di lavoro associati. In tal caso, viene aggiunto un join con la visualizzazione WORK_ITEM alla query SQL generata, ma nessuna condizione per il controllo accesso per il WORK_ITEM.OWNER_ID.

In tal caso, la query SQL per le attività contiene quanto segue:

```

FROM TASK TA, WORK_ITEM WI
WHERE WI.OBJECT_ID = TA.TKIID

```

Query queryAll

Questo tipo di query può essere chiamata solo da amministratori di sistema o controllori di sistema. Né condizioni per il controllo accesso né un join con la visualizzazione WORK_ITEM vengono aggiunti. Questo tipo di query rispetta tutti i dati per tutti gli oggetti.

Esempi dei metodi query e queryAll:

Questi esempi mostrano la sintassi di diverse query API tipiche e le istruzioni SQL associate che sono generate quando viene eseguita la query.

Esempio: attività di query in stato pronto:

Questo esempio mostra come usare il metodo query per richiamare attività con cui l'utente che ha effettuato l'accesso può lavorare.

John Smith vuole un elenco delle attività che gli sono state assegnate. Affinché un utente sia in grado di lavorare su un'attività, questa deve essere in stato pronto. L'utente che ha effettuato l'accesso deve inoltre avere un elemento di lavoro proprietario potenziale per l'attività. Il seguente frammento di codice mostra la chiamata del metodo query per questa query:

```

query( "DISTINCT TASK.TKIID",
      "TASK.KIND IN ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING )
      AND " +
      "TASK.STATE = TASK.STATE.STATE_READY AND " +
      "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
      (String)null, (String)null, (Integer)null, (TimeZone)null )

```

Le seguenti azioni vengono effettuate quando l'istruzione SQL SELEZIONA è generata:

- Una condizione per il comando di accesso viene aggiunta alla clausola dove. Questo esempio suppone che elementi di lavoro di gruppo non siano abilitati.
- Costanti quali TASK.STATE.STATE_READY, vengono sostituiti dai loro valori numerici.
- Sono aggiunte una clausola DA e condizioni join.

Il seguente frammento di codice mostra l'istruzione SQL generata dalla query API:

```
SELECT DISTINCT TASK.TKIID
FROM   TASK TA, WORK_ITEM WI,
WHERE  WI.OBJECT_ID = TA.TKIID
AND    TA.KIND IN ( 101, 105 )
AND    TA.STATE = 2
AND    WI.REASON = 1
E ( WI.OWNER_ID = 'JohnSmith' OR WI.OWNER_ID = null AND WI.EVERYBODY = vero )
```

Per limitare la query API ad attività per un processo specifico, per esempio, sampleProcess, la query apparirà come segue:

```
query( "DISTINCT TASK.TKIID",
      "PROCESS_TEMPLATE.NAME = 'sampleProcess' AND "+
      "TASK.KIND IN ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING )
      AND " +
      "TASK.STATE = TASK.STATE.STATE_READY AND " +
      "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
      (String)null, (String)null, (Integer)null, (TimeZone)null )
```

Esempio: attività di query nello stato richiesto:

Questo esempio mostra come usare il metodo query per richiamare attività che l'utente che ha effettuato l'accesso ha richiesto.

L'utente, John Smith, desidera ricercare attività che ha richiesto e che sono ancora nello stato richiesto. La condizione che specifica "richiesto da John Smith" è TASK.OWNER = 'JohnSmith'. Il seguente frammento di codice mostra la chiamata del metodo query per la query:

```
query( "DISTINCT TASK.TKIID",
      "TASK.STATE = TASK.STATE.STATE_CLAIMED AND " +
      "TASK.OWNER = 'JohnSmith'",
      (String)null, (String)null, (Integer)null, (TimeZone)null )
```

Il seguente frammento di codice mostra l'istruzione SQL generata dalla query API:

```
SELECT DISTINCT TASK.TKIID
FROM   TASK TA, WORK_ITEM WI,
WHERE  WI.OBJECT_ID = TA.TKIID
AND    TA.STATE = 8
AND    TA.OWNER = 'JohnSmith'
AND ( WI.OWNER_ID = 'JohnSmith' OR WI.OWNER_ID = null AND WI.EVERYBODY = true )
```

Quando viene richiesta un'attività, vengono creati elementi di lavoro per il proprietario dell'attività. Così, una maniera alternativa di formare la query per le attività richieste di John Smith è quella di aggiungere la seguente condizione alla query invece di usare TASK.OWNER = 'JohnSmith':

```
WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_OWNER
```

La query sarà quindi simile al seguente frammento di codice:

```
query( "DISTINCT TASK.TKIID",
      "TASK.STATE = TASK.STATE.STATE_CLAIMED AND " +
      "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_OWNER",
      (String)null, (String)null, (Integer)null, (TimeZone)null )
```

Le seguenti azioni vengono effettuate quando l'istruzione SQL SELEZIONA è generata:

- Una condizione per il comando di accesso viene aggiunta alla clausola dove. Questo esempio suppone che elementi di lavoro di gruppo non siano abilitati.
- Costanti quali TASK.STATE.STATE_READY, vengono sostituiti dai loro valori numerici.
- Sono aggiunte una clausola DA e condizioni join.

Il seguente frammento di codice mostra l'istruzione SQL generata dalla query API:

```
SELECT DISTINCT TASK.TKIID
  FROM  TASK TA, WORK_ITEM WI,
  WHERE WI.OBJECT_ID = TA.TKIID
  AND   TA.STATE = 8
  AND   WI.REASON = 4
  AND   ( WI.OWNER_ID = 'JohnSmith' OR WI.OWNER_ID = null AND WI.EVERYBODY = true )
```

John sta per andare in vacanza così il suo capo, Anne Grant, vuole controllare il suo carico di lavoro corrente. Anne ha diritti di amministratore di sistema. La query chiamata da lei è la stessa chiamata da John. Tuttavia, l'istruzione SQL generata è diversa in quanto Anne è un amministratore. Il seguente frammento di codice mostra l'istruzione SQL generata:

```
SELECT DISTINCT TASK.TKIID
  FROM  TASK TA, WORK_ITEM WI,
  WHERE TA.TKIID = WI.OBJECT_ID =
  AND   TA.STATE = 8
  AND   TA.OWNER = 'JohnSmith')
```

Poiché Anne è un amministratore, non viene aggiunta una condizione di controllo accesso alla clausola WHERE.

Esempio: escalation query:

Questo esempio mostra come utilizzare il metodo query per richiamare escalation per l'utente collegato.

Quando si effettua l'escalation di un'attività, viene creato un elemento di lavoro destinatario escalation. L'utente, Mary Jones desidera vedere un elenco delle sue attività su cui è stata effettuata l'escalation. Il seguente frammento di codice mostra la chiamata al metodo query per la query:

```
query( "DISTINCT ESCALATION.ESIID, ESCALATION.TKIID",
      "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_ESCALATION_RECEIVER",
      (String)null, (String)null, (Integer)null, (TimeZone)null )
```

Le seguenti azioni vengono effettuate quando l'istruzione SQL SELEZIONA è generata:

- Una condizione per il comando di accesso viene aggiunta alla clausola dove. Questo esempio suppone che elementi di lavoro di gruppo non siano abilitati.
- Costanti quali TASK.STATE.STATE_READY, vengono sostituiti dai loro valori numerici.
- Sono aggiunte una clausola DA e condizioni join.

Il seguente frammento di codice mostra l'istruzione SQL generata dalla query API:

```
SELECT DISTINCT ESCALATION.ESIID, ESCALATION.TKIID
  FROM  ESCALATION ESC, WORK_ITEM WI
  WHERE ESC.ESIID = WI.OBJECT_ID
```



```

AND    WI.REASON = 10
AND
( WI.OWNER_ID = 'MaryJones' OR WI.OWNER_ID = null AND WI.EVERYBODY = true )

```

Esempio: utilizzo del metodo queryAll:

Questo esempio mostra come utilizzare il metodo queryAll per richiamare tutte le attività che appartengono a un modello di processo.

Il metodo queryAll è disponibile solo per utenti aventi diritti di amministratore o controllore di sistema. Il seguente frammento di codice mostra la chiamata al metodo queryAll con cui la query richiama tutte le attività che appartengono al modello di processo, sampleProcess:

```

queryAll( "DISTINCT ACTIVITY.AIID",
          "PROCESS_TEMPLATE.NAME = 'sampleProcess'",
          (String)null, (String)null, (Integer)null, (TimeZone)null )

```

Il seguente frammento di codice mostra la query SQL generata dalla query API:

```

SELECT DISTINCT ACTIVITY.AIID
FROM   ACTIVITY AI, PROCESS_TEMPLATE PT
WHERE  AI.PTID = PT.PTID
AND    PT.NAME = 'sampleProcess'

```

Poiché la chiamata è effettuata dall'amministratore, non viene aggiunta una condizione di controllo accesso all'istruzione SQL generata. Non viene aggiunto nemmeno un join con la visualizzazione WORK_ITEM. Ciò significa che la query richiama tutte le attività per il modello di processo, comprese quelle attività che non presentano elementi di lavoro.

Esempio: inclusione di proprietà di query in una query:

Questo esempio mostra come usare il metodo query per richiamare attività che appartengono a un processo di business. Il processo ha proprietà query definite che si desidera includere nella ricerca.

Per esempio, si desidera cercare tutte le attività umane in stato pronto che appartengono al processo di business. Il processo ha una proprietà di query, **customerID**, con il valore CID_12345, e un namespace. Il seguente frammento di codice mostra la chiamata del metodo query per la query:

```

query ( " DISTINCT TASK.TKIID, TASK_TEMPL.NAME, TASK.STATE,
        PROCESS_INSTANCE.NAME",
        " QUERY_PROPERTY.NAME = 'customerID' AND " +
        " QUERY_PROPERTY.STRING_VALUE = 'CID_12345' AND " +
        " QUERY_PROPERTY.NAMESPACE =
        'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/' AND " +
        " TASK.KIND IN
        ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING ) AND " +
        " TASK.STATE = TASK.STATE.STATE_READY ",
        (Stringa)null, (Stringa)null, (Integer)null, (TimeZone)null );

```

Se si desidera aggiungere ora una seconda proprietà query alla query, per esempio, **Priorità**, con un dato namespace, la chiamata del metodo query per la query apparirà come segue:

```

query ( " DISTINCT TASK.TKIID, TASK_TEMPL.NAME, TASK.STATE,
        PROCESS_INSTANCE.NAME",
        " QUERY_PROPERTY1.NAME = 'customerID' AND " +
        " QUERY_PROPERTY1.STRING_VALUE = 'CID_12345' AND " +
        " QUERY_PROPERTY1.NAMESPACE =

```

```

        'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/' AND " +
" QUERY_PROPERTY2.NAME = 'Priority' AND " +
" QUERY_PROPERTY2.NAMESPACE =
        'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/' AND " +
" TASK.KIND IN
    ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING ) AND " +
" TASK.STATE = TASK.STATE.STATE_READY ",
(Stringa)null, (Stringa)null, (Integer)null, (TimeZone)null );

```

Se si aggiunge più di una proprietà query alla query, è necessario numerare ciascuna delle proprietà aggiunte come illustrato nel frammento di codice. Tuttavia, le proprietà personalizzate di query influiscono sulle prestazioni; queste diminuiscono con il numero di proprietà personalizzate nella query.

Esempio: inclusione di proprietà personalizzate in una query:

Questo esempio mostra come usare il metodo query per richiamare attività con proprietà personalizzate.

Per esempio, si desidera ricercare tutte le attività umane in stato pronto che hanno una proprietà personalizzata, **customerID**, con il valore CID_12345. Il seguente frammento di codice mostra la chiamata al metodo query per la query:

```

query ( " DISTINCT TASK.TKIID ",
        " TASK_CPROP.NAME = 'customerID' AND " +
        " TASK_CPROP.STRING_VALUE = 'CID_12345' AND " +
" TASK.KIND IN
    ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING ) AND " +
" TASK.STATE = TASK.STATE.STATE_READY ",
(Stringa)null, (Stringa)null, (Integer)null, (TimeZone)null );

```

Se si desidera richiamare le attività e le relative proprietà personalizzate, la chiamata al metodo query per la query apparirà come segue:

```

query ( " DISTINCT TASK.TKIID, TASK_CPROP.NAME, TASK_CPROP.STRING_VALUE",
        " TASK.KIND IN
    ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING ) AND " +
" TASK.STATE = TASK.STATE.STATE_READY ",
(Stringa)null, (Stringa)null, (Integer)null, (TimeZone)null );

```

L'istruzione SQL generata da questa query API è mostrata nel seguente frammento di codice:

```

SELECT DISTINCT TA.TKIID , TACP.NAME , TACP.STRING_VALUE
FROM TASK TA LEFT JOIN TASK_CPROP TACP ON (TA.TKIID = TACP.TKIID),
WORK_ITEM WI
WHERE WI.OBJECT_ID = TA.TKIID
AND TA.KIND IN ( 101, 105 )
AND TA.STATE = 2
AND (WI.OWNER_ID = 'JohnSmith' OR WI.OWNER_ID IS NULL AND WI.EVERYBODY = 1 )

```

Questa istruzione SQL contiene un join esterno tra la visualizzazione TASK e quella TASK_CPROP. Ciò significa che le attività che soddisfano la clausola WHERE vengono richiamate anche se non presentano proprietà personalizzate.

Viste predefinite per le query sugli oggetti processo di business o attività umane:

Le viste del database predefinite sono fornite per gli oggetti processo di business e attività umane. Utilizzare tali viste per eseguire le query dei dati di riferimento per tali oggetti.

Quando si utilizzano le viste predefinite, non è necessario aggiungere esplicitamente predicati di unione per le colonne della vista, poiché tali costrutti vengono aggiunti automaticamente. È possibile utilizzare la funzione di query generica dell'API del servizio (BusinessFlowManagerService o HumanTaskManagerService) per eseguire la query di questi dati. Inoltre, è possibile utilizzare il metodo corrispondente dell'API HumanTaskManagerDelegate oppure le query predefinite fornite dalle implementazioni dell'interfaccia ExecutableQuery.

Nota: Le viste possono contenere colonne che non sono descritte. Tali colonne sono solo per uso interno.

Vista ACTIVITY:

Utilizzare questa vista predefinita del database per le query sulle attività

Tabella 3. Colonne nella vista ACTIVITY

Nome colonna	Type	Commenti
PIID	ID	ID istanza di processo.
AIID	ID	ID istanza di attività.
PTID	ID	ID modello di processo.
ATID	ID	ID modello di attività.
KIND	Integer	Il tipo di attività. I valori possibili sono: KIND_INVOKE (21) KIND_RECEIVE (23) KIND_REPLY (24) KIND_THROW (25) KIND_RETHROW (46) KIND_TERMINATE (26) KIND_WAIT (27) KIND_COMPENSATE (29) KIND_SEQUENCE (30) KIND_EMPTY (3) KIND_SWITCH (32) KIND_WHILE (34) KIND_PICK (36) KIND_FLOW (38) KIND_SCOPE (40) KIND_SCRIPT (42) KIND_STAFF (43) KIND_ASSIGN (44) KIND_CUSTOM (45) KIND_FOR_EACH_PARALLEL (49) KIND_FOR_EACH_SERIAL (47)
COMPLETED	Timestamp	L'ora in cui l'attività viene completata.
ACTIVATED	Timestamp	L'ora in cui viene attivata l'attività.
FIRST_ACTIVATED	Timestamp	L'ora in cui l'attività è stata attivata per la prima volta.
STARTED	Timestamp	L'ora in cui viene avviata l'attività.

Tabella 3. Colonne nella vista ACTIVITY (Continua)

Nome colonna	Type	Commenti
STATE	Integer	Lo stato dell'attività. I valori possibili sono: STATE_INACTIVE (1) STATE_READY (2) STATE_RUNNING (3) STATE_PROCESSING_UNDO (14) STATE_SKIPPED (4) STATE_FINISHED (5) STATE_FAILED (6) STATE_TERMINATED (7) STATE_CLAIMED (8) STATE_TERMINATING (9) STATE_FAILING (10) STATE_WAITING (11) STATE_EXPIRED (12) STATE_STOPPED (13)
OWNER	String	ID principale del proprietario.
DESCRIPTION	String	Se la descrizione del modello di attività contiene segnaposti, questa colonna contiene la descrizione dell'istanza di attività con i segnaposti risolti.
TEMPLATE_NAME	String	Nome del modello attività associata.
TEMPLATE_DESCR	String	Descrizione del modello attività associata.
BUSINESS_RELEVANCE	Boolean	Specifica se l'attività è rilevante per il business. I valori possibili sono: TRUE L'attività è rilevante per il business. È possibile visualizzare lo stato dell'attività in Business Process Choreographer Explorer. FALSE L'attività non è rilevante per il business.
EXPIRES	Timestamp	La data e l'ora di scadenza dell'attività. Se l'attività è scaduta, la data e l'ora in cui si è verificato questo evento.

Vista ACTIVITY_ATTRIBUTE:

Utilizzare questa vista predefinita del database per le query relative alle proprietà personalizzate per le attività.

Tabella 4. Colonne nella vista ACTIVITY_ATTRIBUTE

Nome colonna	Type	Commenti
AIID	ID	L'ID dell'istanza di attività con una proprietà personalizzata.
NAME	String	Il nome della proprietà personalizzata.
VALUE	String	Il valore della proprietà personalizzata.

Vista ACTIVITY_SERVICE:

Utilizzare questa vista predefinita del database per le query sui servizi delle attività

Tabella 5. Colonne nella vista ACTIVITY_SERVICE

Nome colonna	Type	Commenti
EIID	ID	ID dell'istanza dell'evento.
AIID	ID	L'ID dell'istanza di attività in attesa dell'evento.
PIID	ID	L'ID dell'istanza di processo che contiene l'evento.
VTID	ID	ID del modello di servizio che descrive l'evento.
PORT_TYPE	String	Il nome del tipo di porta.
NAME_SPACE_URI	String	L'URI dello spazio dei nomi.
OPERATION	String	Il nome del servizio.

Vista APPLICATION_COMP:

Utilizzare la vista del database predefinita per eseguire le query dell'ID del componente dell'applicazione e delle impostazioni predefinite delle attività.

Tabella 6. Colonne nella vista APPLICATION_COMP

Nome colonna	Type	Commenti
ACOID	String	L'ID del componente dell'applicazione.
BUSINESS_RELEVANCE	Boolean	La politica di rilevanza di business dell'attività predefinita del componente. Questo valore può essere sovrascritto mediante una definizione nel modello di attività o nell'attività. L'attributo condiziona la registrazione della traccia di controllo. I valori possibili sono: TRUE L'attività è rilevante per il business e viene controllata. FALSE L'attività non è rilevante per il business e non viene controllata.
NAME	String	Il nome del componente dell'applicazione.
SUPPORT_AUTOCLAIM	Boolean	La politica di richiesta automatica predefinita del componente. Se questo attributo viene impostato su TRUE, l'attività viene richiesta automaticamente se un singolo utente è il potenziale proprietario. Questo valore può essere sovrascritto da una definizione nel modello di attività o nell'attività.

Tabella 6. Colonne nella vista APPLICATION_COMP (Continua)

Nome colonna	Type	Commenti
SUPPORT_CLAIM_SUSP	Boolean	L'impostazione predefinita del componente che determina se le attività sospese possono essere richieste. Se questo attributo viene impostato su TRUE, le attività sospese possono essere richiamate. Questo valore può essere sovrascritto mediante una definizione nel modello di attività o nell'attività.
SUPPORT_DELEGATION	Boolean	La politica di supporto deleghe dell'attività predefinita del componente. Se questo attributo è impostato su TRUE, le assegnazioni dell'elemento di lavoro per l'attività possono essere modificati. Ciò significa che gli elementi di lavoro possono essere creati, eliminati o trasferiti.
SUPPORT_FOLLOW_ON	Boolean	La politica dell'attività follow-on predefinita del componente. Se questo attributo è impostato su TRUE, le attività follow-on possono essere create per le attività. Questo valore può essere sovrascritto mediante una definizione nel modello di attività o nell'attività.
SUPPORT_SUB_TASK	Boolean	La politica dell'attività secondaria predefinita del componente. Se questo attributo è impostato su TRUE, le attività secondarie possono essere create per le attività. Questo valore può essere sovrascritto mediante una definizione nel modello di attività o nell'attività.

Vista ESCALATION:

Utilizzare la vista del database predefinita per eseguire le query dei dati per le escalation.

Tabella 7. Colonne nella vista ESCALATION

Nome colonna	Type	Commenti
ESIID	String	L'ID dell'istanza dell'escalation.
AZIONE	Integer	L'azione di cui viene eseguito il trigger da parte dell'escalation. I valori possibili sono: ACTION_CREATE_WORK_ITEM (1) Crea un elemento di lavoro per ciascun destinatario dell'escalation. ACTION_SEND_EMAIL (2) Invia un e-mail a ciascun destinatario dell'escalation. ACTION_CREATE_EVENT (3) Crea e pubblica un evento.

Tabella 7. Colonne nella vista ESCALATION (Continua)

Nome colonna	Type	Commenti
ACTIVATION_STATE	Integer	L'istanza di un'escalation viene creata se l'attività corrispondente raggiunge uno dei seguenti stati: ACTIVATION_STATE_READY (2) Specifica che attività umane o l'attività partecipante è pronto ad essere richiamato. ACTIVATION_STATE_RUNNING (3) Specifica che l'attività di origine avviata ed in esecuzione. ACTIVATION_STATE_CLAIMED (8) Specifica che l'attività viene richiesta. ACTIVATION_STATE_WAITING_FOR_SUBTASK (20) Specifica che l'attività è in attesa del completamento delle attività secondarie.
ACTIVATION_TIME	Timestamp	L'ora in cui è stata attivata l'escalation.
AT_LEAST_EXP_STATE	Integer	Lo stato dell'attività previsto dall'escalation. Se si verifica un timeout, lo stato dell'attività è confrontato al valore di questo attributo. I valori possibili sono: AT_LEAST_EXPECTED_STATE_CLAIMED (8) Specifica che l'attività viene richiesta. AT_LEAST_EXPECTED_STATE_ENDED (20) Specifica che l'attività è in uno stato finale (FINISHED, FAILED, TERMINATED o EXPIRED). AT_LEAST_EXPECTED_STATE_SUBTASKS_COMPLETED (21) Specifica che tutte le attività secondarie dell'attività sono state completate.
ESTID	String	L'ID del modello di escalation corrispondente.
FIRST_ESIID	String	L'ID della prima escalation della catena.
INCREASE_PRIORITY	Integer	Indica il modo in cui la proprietà dell'attività verrà incrementata. I valori possibili sono: INCREASE_PRIORITY_NO (1) La priorità dell'attività non è incrementata. INCREASE_PRIORITY_ONCE (2) La priorità dell'attività viene incrementata ogni volta di uno. INCREASE_PRIORITY_REPEATED (3) La priorità viene incrementata di uno ogni volta che si ripete l'escalation.
NAME	String	Il nome dell'escalation.
STATE	Integer	Lo stato dell'escalation. I valori possibili sono: STATE_INACTIVE (1) STATE_WAITING (2) STATE_ESCALATED (3) STATE_SUPERFLUOUS (4)

Tabella 7. Colonne nella vista ESCALATION (Continua)

Nome colonna	Type	Commenti
TKIID	String	L'ID dell'istanza dell'attività cui appartiene l'escalation.

Vista ESCALATION_CPROP:

Utilizzare la vista del database predefinita per eseguire le query delle proprietà personalizzate per le escalation.

Tabella 8. Colonne nella vista ESCALATION_CPROP

Nome colonna	Type	Commenti
ESIID	String	L'ID dell'escalation.
NAME	String	Il nome della proprietà.
DATA_TYPE	String	Il tipo di classe per le proprietà personalizzate non string.
STRING_VALUE	String	Il valore delle proprietà personalizzate del tipo String.

Vista ESCALATION_DESC:

Utilizzare la vista del database predefinita per eseguire le query di dati descrittivi multilingue per le escalation.

Tabella 9. Colonne nella vista ESCALATION_DESC

Nome colonna	Type	Commenti
ESIID	String	L'ID dell'escalation.
LOCALE	String	Il nome della locale associata alla descrizione o al nome di visualizzazione.
DESCRIPTION	String	Una descrizione del modello di attività.
DISPLAY_NAME	String	Il nome descrittivo dell'escalation.

Vista ESC_TEMPL:

Utilizzare la vista predefinita del database per eseguire la query dei dati per i modelli di escalation.

Tabella 10. Colonne nella vista ESC_TEMPL

Nome colonna	Type	Commenti
ESTID	String	L'ID del modello di escalation.

Tabella 10. Colonne nella vista ESC_TEMPL (Continua)

Nome colonna	Type	Commenti
AZIONE	Integer	<p>L'azione di cui viene eseguito il trigger da parte dell'escalation. I valori possibili sono:</p> <p>ACTION_CREATE_WORK_ITEM (1) Crea un elemento di lavoro per ciascun destinatario dell'escalation.</p> <p>ACTION_SEND_EMAIL (2) Invia un e-mail a ciascun destinatario dell'escalation.</p> <p>ACTION_CREATE_EVENT (3) Crea e pubblica un evento.</p>
ACTIVATION_STATE	Integer	<p>L'istanza di un'escalation viene creata se l'attività corrispondente raggiunge uno dei seguenti stati:</p> <p>ACTIVATION_STATE_READY (2) Specifica che attività umana o l'attività partecipante è pronto ad essere richiamato.</p> <p>ACTIVATION_STATE_RUNNING (3) Specifica che l'attività di origine avviata ed in esecuzione.</p> <p>ACTIVATION_STATE_CLAIMED (8) Specifica che l'attività viene richiesta.</p> <p>ACTIVATION_STATE_WAITING_FOR_SUBTASK (20) Specifica che l'attività è in attesa del completamento delle attività secondarie.</p>
AT_LEAST_EXP_STATE	Integer	<p>Lo stato dell'attività previsto dall'escalation. Se si verifica un timeout, lo stato dell'attività è confrontato al valore di questo attributo. I valori possibili sono:</p> <p>AT_LEAST_EXPECTED_STATE_CLAIMED (8) Specifica che l'attività viene richiesta.</p> <p>AT_LEAST_EXPECTED_STATE_ENDED (20) Specifica che l'attività è in uno stato finale (FINISHED, FAILED, TERMINATED o EXPIRED).</p> <p>AT_LEAST_EXPECTED_STATE_SUBTASKS_COMPLETED (21) Specifica che tutte le attività secondarie dell'attività sono state completate.</p>
CONTAINMENT_CTX_ID	String	<p>Se il modello di escalation appartiene a un modello di attività in linea, il contesto di contenimento è il modello di processo. Se il modello del contesto di escalation appartiene a un modello di attività autonomo, il contesto di contenimento è il modello di attività.</p>
FIRST_ESTID	String	<p>L'ID del primo modello di escalation in una catena di modelli di escalation.</p>

Tabella 10. Colonne nella vista ESC_TEMPL (Continua)

Nome colonna	Type	Commenti
INCREASE_PRIORITY	Integer	Indica il modo in cui la proprietà dell'attività verrà incrementata. I valori possibili sono: INCREASE_PRIORITY_NO (1) La priorità dell'attività non è incrementata. INCREASE_PRIORITY_ONCE (2) La priorità dell'attività viene incrementata ogni volta di uno. INCREASE_PRIORITY_REPEATED (3) La priorità viene incrementata di uno ogni volta che si ripete l'escalation.
NAME	String	Il nome del modello di escalation.
PREVIOUS_ESTID	String	L'ID del precedente modello di escalation in una catena di modelli di escalation.
TKTID	String	L'ID del modello dell'attività a cui appartiene il modello di escalation.

Vista ESC_TEMPL_CPROP:

Utilizzare la vista predefinita del database per eseguire la query delle proprietà personalizzate per modelli di escalation.

Tabella 11. Colonne nella vista ESC_TEMPL_CPROP

Nome colonna	Type	Commenti
ESTID	String	L'ID del modello di escalation.
NAME	String	Il nome della proprietà.
TKTID	String	L'ID del modello dell'attività a cui appartiene il modello di escalation.
DATA_TYPE	String	Il tipo di classe per le proprietà personalizzate non string.
VALUE	String	Il valore delle proprietà personalizzate del tipo String.

ESC_TEMPL_DESC view:

Utilizzare la vista predefinita del database per eseguire la query dei dati descrittivi multilingue per modelli di escalation.

Tabella 12. Colonne nella vista ESC_TEMPL_DESC

Nome colonna	Type	Commenti
ESTID	String	L'ID del modello di escalation.
LOCALE	String	Il nome della locale associata alla descrizione o al nome di visualizzazione.
TKTID	String	L'ID del modello dell'attività a cui appartiene il modello di escalation.
DESCRIPTION	String	Una descrizione del modello di attività.
DISPLAY_NAME	String	Il nome descrittivo dell'escalation.

Vista PROCESS_ATTRIBUTE:

Utilizzare questa vista del database predefinita per le query relative alle proprietà personalizzate per i processi.

Tabella 13. Colonne nella vista PROCESS_ATTRIBUTE

Nome colonna	Type	Commenti
PIID	ID	L'ID dell'istanza processo con una proprietà personalizzata.
NAME	String	Il nome della proprietà personalizzata.
VALUE	String	Il valore della proprietà personalizzata.

Vista PROCESS_INSTANCE:

Utilizzare questa vista predefinita del database per le query nelle istanze di processo.

Tabella 14. Colonne nella vista PROCESS_INSTANCE

Nome colonna	Type	Commenti
PTID	ID	ID modello di processo.
PIID	ID	ID istanza di processo.
NAME	String	Il nome dell'istanza di processo.
STATE	Integer	Lo stato dell'istanza di processo. I valori possibili sono: STATE_READY (1) STATE_RUNNING (2) STATE_FINISHED (3) STATE_COMPENSATING (4) STATE_INDOUBT (10) STATE_FAILED (5) STATE_TERMINATED (6) STATE_COMPENSATED (7) STATE_COMPENSATION_FAILED (12) STATE_TERMINATING (8) STATE_FAILING (9) STATE_SUSPENDED (11)
CREATED	Timestamp	L'ora in cui viene creata l'istanza di processo.
STARTED	Timestamp	L'ora in cui è stata avviata l'istanza di processo.
COMPLETED	Timestamp	L'ora in cui è stata completata l'istanza di processo.
PARENT_PIID	ID	L'ID dell'istanza di processo parent.
PARENT_NAME	String	Il nome dell'istanza di processo parent.
TOP_LEVEL_PIID	ID	L'ID dell'istanza di processo dell'istanza di processo del livello superiore. Se non esiste alcuna istanza di processo del livello superiore, si tratta dell'ID dell'istanza di processo dell'istanza corrente.
TOP_LEVEL_NAME	String	Il nome dell'istanza di processo del livello superiore. Se non esiste alcuna istanza di livello superiore, è il nome dell'istanza del processo corrente.

Tabella 14. Colonne nella vista *PROCESS_INSTANCE* (Continua)

Nome colonna	Type	Commenti
STARTER	String	L'ID principale dello starter dell'istanza di processo.
DESCRIPTION	String	Se la descrizione del modello di processo contiene dei segnaposti, questa colonna contiene la descrizione dell'istanza di processo con i segnaposti risolti.
TEMPLATE_NAME	String	Il nome del modello di processo associato.
TEMPLATE_DESCR	String	Descrizione del modello di processo associato.
RESUMES	Timestamp	L'ora in cui l'istanza di processo deve essere ripresa automaticamente.

Vista *PROCESS_TEMPLATE*:

Utilizzare questa vista predefinita del database per le query sui modelli di processo.

Tabella 15. Colonne nella vista *PROCESS_TEMPLATE*

Nome colonna	Type	Commenti
PTID	ID	ID modello di processo.
NAME	String	Il nome del modello di processo.
VALID_FROM	Timestamp	L'ora a partire dalla quale è possibile creare un'istanza del modello di processo.
TARGET_NAMESPACE	String	Lo spazio dei nomi di destinazione del modello di processo.
APPLICATION_NAME	String	Il nome dell'applicazione di business a cui appartiene il modello di processo.
VERSION	String	Versione definita dall'utente.
CREATED	Timestamp	L'ora in cui viene creato il modello di processo nel database.
STATE	Integer	Specifica se il modello di processo è disponibile per creare istanze di processo. I valori possibili sono: STATE_STARTED (1) STATE_STOPPED (2)
EXECUTION_MODE	Integer	Specifica le modalità con cui è possibile eseguire le istanze di processo derivate da questo modello di processo. I valori possibili sono: EXECUTION_MODE_MICROFLOW (1) EXECUTION_MODE_LONG_RUNNING (2)
DESCRIPTION	String	Descrizione del modello di processo.

Tabella 15. Colonne nella vista *PROCESS_TEMPLATE* (Continua)

Nome colonna	Type	Commenti
COMP_SPHERE	Integer	Specifica la funzione di compensazione delle istanze dei microflussi nel modello di processo, è possibile unire una sfera di compensazione esistente oppure crearne una. I valori possibili sono: COMP_SPHERE_REQUIRED (2) COMP_SPHERE_SUPPORTS (4)
DISPLAY_NAME	String	Il nome descrittivo del processo.

Vista *QUERY_PROPERTY*:

Utilizzare questa vista predefinita del database per le query sulle variabili a livello di processo.

Tabella 16. Colonne nella vista *QUERY_PROPERTY*

Nome colonna	Type	Commenti
PIID	ID	ID istanza di processo.
VARIABLE_NAME	String	Il nome della variabile a livello di processo.
NAME	String	Il nome della proprietà della query.
NAMESPACE	String	Lo spazio nomi della proprietà della query.
GENERIC_VALUE	String	Una rappresentazione di stringa per i tipi di proprietà che non sono associati a uno dei tipi definiti: <i>STRING_VALUE</i> , <i>NUMBER_VALUE</i> , <i>DECIMAL_VALUE</i> o <i>TIMESTAMP_VALUE</i> .
STRING_VALUE	String	Se un tipo di proprietà è associato a un tipo di stringa, questo è il valore della stringa.
NUMBER_VALUE	Integer	Se un tipo di proprietà è associato a un tipo di numero intero, questo è il valore del numero intero.
DECIMAL_VALUE	Decimal	Se un tipo di proprietà è associato a un tipo di virgola mobile, questo è il valore del numero decimale.
TIMESTAMP_VALUE	Timestamp	Se un tipo di proprietà è associato a un tipo di data/ora, questo è il valore della data/ora.

Vista *TASK*:

Utilizzare questa vista predefinita del database per le query sugli oggetti dell'attività.

Tabella 17. Colonne nella vista TASK

Nome colonna	Type	Commenti
TKIID	ID	ID dell'istanza dell'attività
ACTIVATED	Timestamp	L'ora in cui è stata attivata l'attività.
APPLIC_DEFAULTS_ID	ID	L>ID del componente dell'applicazione che specifica i valori predefiniti dell'attività.
APPLIC_NAME	String	Il nome dell'applicazione enterprise cui appartiene l'attività.
BUSINESS_RELEVANCE	Boolean	Specifica se l'attività è rilevante per il business. L'attributo condiziona la registrazione della traccia di controllo. I valori possibili sono: TRUE L'attività è rilevante per il business e viene controllata. FALSE L'attività non è rilevante per il business e non viene controllata.
COMPLETED	Timestamp	L'ora in cui è stata completata l'attività.
CONTAINMENT_CTX_ID	ID	Il contesto di contenimento per questa attività. Questo attributo determina il ciclo vita dell'attività. Quando il contesto di contenimento di un'attività viene eliminato, viene eliminata anche l'attività.
CTX_AUTHORIZATION	Integer	Consente al proprietario dell'attività di accedere al contesto dell'attività. I valori possibili sono: AUTH_NONE Nessun privilegio per l'oggetto contestuale associato. AUTH_READER Le operazioni che possono essere eseguite sull'oggetto di contesto associato richiedono privilegi di lettura, ad esempio, la lettura delle proprietà di un'istanza del processo.
DUE	Timestamp	L'ora in cui è attesa l'attività.
EXPIRES	Timestamp	La data di scadenza dell'attività.
FIRST_ACTIVATED	Timestamp	L'ora in cui l'attività era stata attivata per la prima volta.
FOLLOW_ON_TKIID	ID	L>ID dell'istanza dell'attività follow-on.
HIERARCHY_POSITION	Integer	I valori possibili sono: HIERARCHY_POSITION_TOP_TASK (0) L'attività di livello superiore nella gerarchia di attività. HIERARCHY_POSITION_SUB_TASK (1) L'attività è un'attività secondaria nella gerarchia di attività. HIERARCHY_POSITION_FOLLOW_ON_TASK (2) L'attività è un'attività follow-on nella gerarchia di attività.
IS_AD_HOC	Boolean	Indica se questa attività è stata creata in modo dinamico al momento del runtime o da un modello di attività.

Tabella 17. Colonne nella vista TASK (Continua)

Nome colonna	Type	Commenti
IS_ESCALATED	Boolean	Indica se si è verificata un'escalation di questa attività.
IS_INLINE	Boolean	Indica se l'attività è in linea in un processo di business.
IS_WAIT_FOR_SUB_TK	Boolean	Indica se l'attività principale è in attesa di un'attività secondaria per raggiungere uno stato finale.
KIND	Integer	<p>Il tipo di attività. I valori possibili sono:</p> <p>KIND_HUMAN (101) Indica che l'attività è un'attività di collaborazione che viene creata ed elaborata da un utente di tipo umano.</p> <p>KIND_WPC_STAFF_ACTIVITY (102) Indica che l'attività è di tipo attività umana, ovvero un'attività di staff di un processo di business WebSphere Business Integration Server Foundation, versione 5.</p> <p>KIND_ORIGINATING (103) Indica che l'attività è un'attività di chiamata che supporta interazioni utente-computer, che consentono di creare, inizializzare e avviare i servizi.</p> <p>KIND_PARTICIPATING (105) Indica che l'attività è un'attività da fare che supporta interazioni computer-utente, che consentono di implementare un servizio.</p> <p>KIND_ADMINISTRATIVE (106) Indica che l'attività è un'attività amministrativa.</p>
LAST_MODIFIED	Timestamp	L'ora in cui è stata modificata l'attività per l'ultima volta.
LAST_STATE_CHANGE	Timestamp	L'ora in cui è stato modificato lo stato di un'attività per l'ultima volta.
NAME	String	Il nome dell'attività.
NAME_SPACE	String	Lo spazio dei nomi utilizzato per classificare l'attività.
ORIGINATOR	String	L'ID principale del creatore dell'attività.
OWNER	String	L'ID principale del proprietario dell'attività.
PARENT_CONTEXT_ID	String	Il contesto parent di questa attività. Questo attributo fornisce una chiave al contesto corrispondente nel richiamo del componente dell'applicazione. Il contesto parent viene impostato dal componente dell'applicazione che crea l'attività.
PRIORITY	Integer	La priorità dell'attività.
RESUMES	Timestamp	L'ora in cui l'attività deve essere ripresa automaticamente.
STARTED	Timestamp	L'ora in cui l'attività era stata avviata (STATE_RUNNING, STATE_CLAIMED).

Tabella 17. Colonne nella vista TASK (Continua)

Nome colonna	Type	Commenti
STARTER	String	L'ID principale dello starter dell'attività.
STATE	Integer	Lo stato dell'attività. I valori possibili sono: STATE_READY (2) Indica che l'attività è pronta per essere richiesta. STATE_RUNNING (3) Indica che l'attività è avviata e in esecuzione. STATE_FINISHED (5) indica che l'attività è stata terminata correttamente. STATE_FAILED (6) Indica che l'attività non è terminata correttamente. STATE_TERMINATED (7) Indica che l'attività è stata terminata a causa di una richiesta esterna o interna. STATE_CLAIMED (8) Indica che l'attività è richiesta. STATE_EXPIRED (12) Indica che l'attività è terminata poiché supera la durata specificata. STATE_FORWARDED (101) Indica che l'attività è stata completata con un'attività follow-on.
SUPPORT_AUTOCLAIM	Boolean	Indica se questa attività viene richiesta automaticamente se viene assegnata ad un singolo utente.
SUPPORT_CLAIM_SUSP	Boolean	Indica se questa attività può essere richiesta se viene sospesa.
SUPPORT_DELEGATION	Boolean	Indica se questa attività supporta la delega del lavoro mediante la creazione, l'eliminazione o il trasferimento degli elementi di lavoro.
SUPPORT_FOLLOW_ON	Boolean	Indica se questa attività supporta la creazione di attività follow-on.
SUPPORT_SUB_TASK	Boolean	Indica se questa attività supporta la creazione di attività secondarie.
SUSPENDED	Boolean	Indica se l'attività viene sospesa.
TKTID	ID	ID modello di attività
TOP_TKIID	ID	L'ID istanza dell'attività principale superiore se si tratta di un'attività secondaria.
TYPE	String	Il tipo utilizzato per classificare l'attività.

Vista TASK_CPROP:

Utilizzare la vista del database predefinita per eseguire le query delle proprietà personalizzate per gli oggetti dell'attività.

Tabella 18. Colonne nella vista *TASK_CPROP*

Nome colonna	Type	Commenti
TKIID	String	ID istanza di attività.
NAME	String	Il nome della proprietà.
DATA_TYPE	String	Il tipo di classe per le proprietà personalizzate non string.
STRING_VALUE	String	Il valore delle proprietà personalizzate del tipo String.

Vista *TASK_DESC*:

Utilizzare la vista del database predefinita per eseguire le query di dati descrittivi multilingue per gli oggetti dell'attività.

Tabella 19. Colonna della vista *TASK_DESC*

Nome colonna	Type	Commenti
TKIID	String	ID istanza di attività.
LOCALE	String	Il nome della locale associata alla descrizione o al nome di visualizzazione.
DESCRIPTION	String	Una descrizione dell'attività.
DISPLAY_NAME	String	Il nome descrittivo dell'attività.

Vista *TASK_TEMPL*:

la vista del database predefinita che contiene i dati da utilizzare per eseguire l'istanza delle attività.

Tabella 20. Colonne nella vista *TASK_TEMPL*

Nome colonna	Type	Commenti
TKTID	String	ID modello di attività
VALID_FROM	Timestamp	L'ora in cui il modello di attività diventa disponibile per l'esecuzione dell'istanza.
APPLIC_DEFAULTS_ID	String	L>ID del componente dell'applicazione che specifica i valori predefiniti del modello di attività.
APPLIC_NAME	String	Il nome dell'applicazione enterprise cui appartiene il modello di attività.
BUSINESS_RELEVANCE	Boolean	Specifica se il modello di attività è rilevante per il business. L'attributo condiziona la registrazione della traccia di controllo. I valori possibili sono: TRUE L'attività è rilevante per il business e viene controllata. FALSE L'attività non è rilevante per il business e non viene controllata.
CONTAINMENT_CTX_ID	ID	Il contesto di contenimento per questo modello di attività. Questo attributo determina il ciclo vita del modello di attività. Quando il contesto di contenimento di un'attività viene eliminato, viene eliminata anche il modello di attività.

Tabella 20. Colonne nella vista TASK_TEMPL (Continua)

Nome colonna	Type	Commenti
CTX_AUTHORIZATION	Integer	<p>Consente al proprietario dell'attività di accedere al contesto dell'attività. I valori possibili sono:</p> <p>AUTH_NONE Nessun privilegio per l'oggetto contestuale associato.</p> <p>AUTH_READER Le operazioni che possono essere eseguite sull'oggetto di contesto associato richiedono privilegi di lettura, ad esempio, la lettura delle proprietà di un'istanza del processo.</p>
DEFINITION_NAME	String	Il nome della definizione del modello di attività nel file Task Execution Language (TEL).
DEFINITION_NS	String	Il namespace della definizione del modello di attività nel file TEL.
IS_AD_HOC	Boolean	Indica se questo modello di attività è stato creato in modo dinamico al momento del runtime o quando l'attività è stata distribuita come parte di un file EAR.
IS_INLINE	Boolean	Indica se questo modello di attività viene modellato come attività all'interno di un processo di business.
KIND	Integer	<p>Il tipo di attività derivate da questo modello di attività. I valori possibili sono:</p> <p>KIND_HUMAN (101) Indica che l'attività è un'attività di collaborazione che viene creata ed elaborata da un utente di tipo umano.</p> <p>KIND_ORIGINATING (103) Indica che l'attività è un'attività di chiamata che supporta interazioni utente-computer, che consentono di creare, inizializzare e avviare i servizi.</p> <p>KIND_PARTICIPATING (105) Indica che l'attività è un'attività da fare che supporta interazioni computer-utente, che consentono di implementare un servizio.</p> <p>KIND_ADMINISTRATIVE (106) Indica che l'attività è un'attività amministrativa.</p>
NAME	String	Il nome del modello di attività.
NAMESPACE	String	Lo spazio dei nomi utilizzato per classificare il modello di attività.
PRIORITY	Integer	La priorità del modello di attività.

Tabella 20. Colonne nella vista TASK_TEMPL (Continua)

Nome colonna	Type	Commenti
STATE	Integer	Lo stato del modello di attività. I valori possibili sono: STATE_STARTED (1) Specifica che il modello di attività è disponibile per la creazione di istanze dell'attività. STATE_STOPPED (2) Specifica che il modello di attività si è arrestato. Le istanze dell'attività non possono essere create dal modello di attività in questo stato.
SUPPORT_AUTOCLAIM	Boolean	Indica se le attività derivate da questo modello di attività possono essere richieste automaticamente se vengono assegnate ad un singolo utente.
SUPPORT_CLAIM_SUSP	Boolean	Indica se le attività derivate da questo modello di attività possono essere richieste se vengono sospese.
SUPPORT_DELEGATION	Boolean	Indica se l'attività derivata da questo modello di attività supportano la delega del lavoro utilizzando la creazione, l'eliminazione o il trasferimento degli elementi di lavoro.
SUPPORT_FOLLOW_ON	Boolean	Indica se il modello di attività supporta la creazione di attività follow-on.
SUPPORT_SUB_TASK	Boolean	Indica se il modello di attività supporta la creazione di attività secondarie.
TYPE	String	Il tipo utilizzato per classificare il modello di attività.

Vista TASK_TEMPL_CPROP:

Utilizzare la vista del database predefinita per eseguire le query delle proprietà personalizzate per i modelli di attività.

Tabella 21. Colonne nella vista TASK_TEMPL_CPROP

Nome colonna	Type	Commenti
TKTID	String	ID modello di attività
NAME	String	Il nome della proprietà.
DATA_TYPE	String	Il tipo di classe per le proprietà personalizzate non string.
STRING_VALUE	String	Il valore delle proprietà personalizzate del tipo String.

Vista TASK_TEMPL_DESC:

Utilizzare la vista del database predefinita per eseguire le query di dati descrittivi multilingue per gli oggetti del modello di attività.

Tabella 22. Colonne nella vista *TASK_TEMPL_DESC*

Nome colonna	Type	Commenti
TKTID	String	ID modello di attività
LOCALE	String	Il nome della locale associata alla descrizione o al nome di visualizzazione.
DESCRIPTION	String	Una descrizione del modello di attività.
DISPLAY_NAME	String	Il nome descrittivo del modello di attività.

Vista *WORK_ITEM*:

Utilizzare questa vista predefinita del database per le query sugli elementi di lavoro e i dati di autorizzazione per le attività, i processi e le escalation.

Tabella 23. Colonne nella vista *WORK_ITEM*

Nome colonna	Type	Commenti
WIID	ID	ID elemento di lavoro.
OWNER_ID	String	ID principale del proprietario.
GROUP_NAME	String	Il nome della worklist del gruppo associato.
EVERYBODY	Boolean	Specifica se tutti possiedono questo elemento di lavoro.
OBJECT_TYPE	Integer	<p>Il tipo di oggetto associato. I valori possibili sono:</p> <p>OBJECT_TYPE_ACTIVITY (1) Specifica che l'elemento di lavoro era stato creato per un'attività.</p> <p>OBJECT_TYPE_PROCESS_INSTANCE (3) Specifica che l'elemento di lavoro era stato creato per un'istanza del processo.</p> <p>OBJECT_TYPE_TASK_INSTANCE (5) Specifica che l'elemento di lavoro era stato creato per un'attività.</p> <p>OBJECT_TYPE_TASK_TEMPLATE (6) Specifica che l'elemento di lavoro era stato creato per un modello di attività.</p> <p>OBJECT_TYPE_ESCALATION_INSTANCE (7) Specifica che l'elemento di lavoro era stato creato per un'istanza di escalation.</p> <p>OBJECT_TYPE_APPLICATION_COMPONENT (9) Specifica che l'elemento di lavoro era stato creato per un componente dell'applicazione.</p>
OBJECT_ID	ID	L'ID dell'oggetto associato, ad esempio, il processo o l'attività associata.

Tabella 23. Colonne nella vista WORK_ITEM (Continua)

Nome colonna	Type	Commenti
ASSOC_OBJECT_TYPE	Integer	Il tipo di oggetto cui fa riferimento l'attributo ASSOC_OID, ad esempio, task, process, o oggetti esterni. Utilizzare i valori per l'attributo OBJECT_TYPE.
ASSOC_OID	ID	L'ID dell'oggetto associato all'elemento di lavoro. Ad esempio, il PIID (process instance ID) dell'istanza di processo contenente l'istanza di attività per la quale è stato creato questo elemento di lavoro.
REASON	Integer	Il motivo per l'assegnazione dell'elemento di lavoro. I valori possibili sono: REASON_POTENTIAL_STARTER (5) REASON_POTENTIAL_INSTANCE_CREATOR (11) REASON_POTENTIAL_STARTER (1) REASON_EDITOR (2) REASON_READER (3) REASON_ORIGINATOR (9) REASON_OWNER (4) REASON_STARTER (6) REASON_ESCALATION_RECEIVER (10) REASON_ADMINISTRATOR (7)
CREATION_TIME	Timestamp	La data e l'ora in cui è stato creato l'elemento di lavoro.

Esecuzione del filtro sui dati mediante l'utilizzo delle variabili nelle query

Una query restituisce gli oggetti che corrispondono ai criteri di query. È possibile eseguire il filtro di tali risultati sui valori delle variabili.

About this task

È possibile definire le variabili utilizzate da un processo al momento del runtime nel modello del processo. Per queste variabili, vengono indicate le parti per cui è possibile eseguire una query.

Ad esempio, John Smith, chiama il numero dell'assistenza della propria compagnia di assicurazioni per capire l'andamento del reclamo relativo alla propria automobile danneggiata. L'amministratore reclami utilizza l'ID cliente per individuare il reclamo.

Procedure

1. Opzionale: Elencare le proprietà delle variabili in un processo per cui è possibile eseguire una query.

Utilizzare l'ID modello di processo per identificare il processo. È possibile ignorare questo passo se già si conoscono le variabili per cui è possibile eseguire una query.

```
Elencare variableProperties = process.getQueryProperties(ptid);
per (int i = 0; i < variableProperties.size(); i++)
{
    QueryProperty queryData = (QueryProperty)variableProperties.get(i);
    String variableName = queryData.getVariableName();
}
```

```

String name      = queryData.getName();
int mappedType  = queryData.getMappedType();
...
}

```

2. Elencare le istanze del processo con le variabili che corrispondono ai criteri di filtro.

Per questo processo, l'ID cliente viene modellato come parte della variabile customerClaim per cui è possibile eseguire una query. È possibile quindi utilizzare l'ID cliente per individuare il reclamo.

```

QueryResultSet result = process.query
("PROCESS_INSTANCE.NAME, QUERY_PROPERTY.STRING_VALUE",
"QUERY_PROPERTY.VARIABLE_NAME = 'customerClaim' AND " +
"QUERY_PROPERTY.NAME = 'customerID' AND " +
"QUERY_PROPERTY.STRING_VALUE like 'Smith%'",
(String)null, (Integer)null,
(Integer)null, (TimeZone)null );

```

Questa azione restituisce una serie di risultati di query contenente i nomi delle istanze del processo e i valori degli ID cliente per i clienti i cui ID iniziano con Smith.

Gestione delle query memorizzate

Le query memorizzate forniscono una modalità di salvataggio delle query eseguite di frequente. La query memorizzata può essere una query disponibile per tutti gli utenti (query pubblica) o una query che appartiene a un utente specifico (query privata).

About this task

Una query memorizzata è una query che viene memorizzata nel database ed identificata mediante un nome. Una query memorizzata privata e una query memorizzata pubblica e query memorizzate private di proprietari diversi possono avere lo stesso nome.

È possibile disporre di query memorizzate per gli oggetti del processo di business, oggetti di attività o una combinazione di questi due tipi di oggetti.

Concetti correlati

“Parametri nelle query memorizzate” a pagina 33

Una query memorizzata è una query che viene memorizzata nel database ed identificata mediante un nome. Le tuple di qualifica vengono assemblate in modo dinamico al momento dell'esecuzione della query. Per rendere riutilizzabili le query memorizzate, è possibile utilizzare i parametri nella definizione di query risolti al momento del runtime.

Gestione di query memorizzate pubbliche:

Le query memorizzate pubbliche vengono create dall'amministratore di sistema. Queste query sono disponibili per tutti gli utenti.

About this task

Come amministratore di sistema, è possibile creare, visualizzare ed eliminare le query memorizzate pubbliche. Se non si specifica un ID utente nella chiamata API, si presume che la query memorizzata sia una query memorizzata pubblica.

Procedure

1. Creare una query memorizzata pubblica.

Ad esempio, il seguente frammento di codice crea una query memorizzata per le istanze del processo e la salva con il nome `CustomerOrdersStartingWithA`.

```
process.createStoredQuery("CustomerOrdersStartingWithA",
    "DISTINCT PROCESS_INSTANCE.PIID, PROCESS_INSTANCE.NAME",
    "PROCESS_INSTANCE.NAME LIKE 'A%'",
    "PROCESS_INSTANCE.NAME",
    (Integer)null, (TimeZone)null);
```

Il risultato della query memorizzata è un elenco ordinato di tutti i nomi delle istanze di processo che iniziano con la lettera A, con i PIID (process instance ID) associati.

2. Eseguire la query definita dalla query memorizzata.

```
QueryResultSet result = process.query("CustomerOrdersStartingWithA",
    new Integer(0));
```

Questa operazione restituisce gli oggetti che corrispondono ai criteri. In questo caso, tutti gli ordini del cliente che iniziano con la lettera A.

3. Elencare i nomi delle query memorizzate pubbliche disponibili.

Il seguente frammento di codice illustra come limitare l'elenco delle query restituite solo alle query pubbliche.

```
String[] storedQuery = process.getStoredQueryNames(StoredQueryData.KIND_PUBLIC);
```

4. Opzionale: Verificare la query definita da una determinata query memorizzata.

Una query memorizzata privata può avere lo stesso nome di una query memorizzata pubblica. Se i nomi sono gli stessi, viene restituita la query memorizzata privata. Il seguente frammento di codice illustra come restituire solo la query pubblica con il nome specificato. Se si desidera eseguire questa query per gli oggetti basati sull'attività, specificare `StoredQuery` come tipo di oggetto restituito anziché `StoredQueryData`.

```
StoredQueryData storedQuery = process.getStoredQuery
    (StoredQueryData.KIND_PUBLIC, "CustomerOrdersStartingWithA");
String selectClause = storedQuery.getSelectClause();
String whereClause = storedQuery.getWhereClause();
String orderByClause = storedQuery.getOrderByClause();
Integer threshold = storedQuery.getThreshold();
String owner = storedQuery.getOwner();
```

5. Eliminare una query memorizzata pubblica.

Il seguente frammento di codice illustra il modo in cui eliminare la query memorizzata creata al passo 1.

```
process.deleteStoredQuery("CustomerOrdersStartingWithA");
```

Gestione di query memorizzate private per altri utenti:

Le query private possono essere create da qualsiasi utente. Queste query sono disponibili solo per il proprietario della query e l'amministratore di sistema.

About this task

Come amministratore di sistema, è possibile gestire le query memorizzate private che appartengono a uno specifico utente.

Procedure

1. Creare una query memorizzata privata per l'ID utente Smith.

Ad esempio, il seguente frammento di codice crea una query memorizzata per le istanze del processo e la salva con il nome `CustomerOrdersStartingWithA` per l'ID utente Smith.

```
process.createStoredQuery("Smith", "CustomerOrdersStartingWithA",
    "DISTINCT PROCESS_INSTANCE.PIID, PROCESS_INSTANCE.NAME",
    "PROCESS_INSTANCE.NAME LIKE 'A%'",
    "PROCESS_INSTANCE.NAME",
    (Integer)null, (TimeZone)null,
    (List)null, (String)null);
```

Il risultato della query memorizzata è un elenco ordinato di tutti i nomi delle istanze di processo che iniziano con la lettera A, con i PIID (process instance ID) associati.

2. Eseguire la query definita dalla query memorizzata.

```
QueryResultSet result = process.query
    ("Smith", "CustomerOrdersStartingWithA",
    (Integer)null, (Integer)null, (List)null);
new Integer(0));
```

Questa operazione restituisce gli oggetti che corrispondono ai criteri. In questo caso, tutti gli ordini del cliente che iniziano con la lettera A.

3. Richiamare un elenco dei nomi delle query private che appartengono a uno specifico utente.

Ad esempio, il seguente frammento di codice illustra come richiamare un elenco delle query private che appartengono all'utente Smith.

```
String[] storedQuery = process.getStoredQueryNames("Smith");
```

4. Visualizzare i dettagli di una specifica query.

Il seguente frammento di codice illustra come richiamare i dettagli della query CustomerOrdersStartingWithA di proprietà dell'utente Smith.

```
StoredQuery storedQuery = process.getStoredQuery
    ("Smith", "CustomerOrdersStartingWithA");
String selectClause = storedQuery.getSelectClause();
String whereClause = storedQuery.getWhereClause();
String orderByClause = storedQuery.getOrderByClause();
Integer threshold = storedQuery.getThreshold();
String owner = storedQuery.getOwner();
```

5. Eliminare una query memorizzata privata.

Il seguente frammento di codice illustra come eliminare una query privata di proprietà dell'utente Smith.

```
process.deleteStoredQuery("Smith", "CustomerOrdersStartingWithA");
```

Utilizzo delle query memorizzate private:

Se non si è un amministratore di sistema, è possibile creare, eseguire ed eliminare le query memorizzate private. È possibile anche utilizzare le query memorizzate pubbliche create dall'amministratore di sistema.

Procedura

1. Creare una query memorizzata privata.

Ad esempio, il seguente frammento di codice crea una query memorizzata per le istanze del processo e la salva con un determinato nome. Se non è specificato un ID utente, si presume che la query memorizzata sia una query memorizzata privata per l'utente collegato.

```
process.createStoredQuery("CustomerOrdersStartingWithA",
    "DISTINCT PROCESS_INSTANCE.PIID, PROCESS_INSTANCE.NAME",
    "PROCESS_INSTANCE.NAME LIKE 'A%'",
    "PROCESS_INSTANCE.NAME",
    (Integer)null, (TimeZone)null);
```

Questa query restituisce un elenco ordinato di tutti i nomi delle istanze di processo che iniziano con la lettera A, con i PIID (process instance ID) associati.

2. Eseguire la query definita dalla query memorizzata.

```
QueryResultSet result = process.query("CustomerOrdersStartingWithA",  
    new Integer(0));
```

Questa operazione restituisce gli oggetti che corrispondono ai criteri. In questo caso, tutti gli ordini del cliente che iniziano con la lettera A.

3. Richiamare un elenco dei nomi delle query memorizzate a cui l'utente collegato può accedere.

Il seguente frammento di codice illustra come richiamare le query memorizzate private e pubbliche a cui l'utente può accedere.

```
String[] storedQuery = process.getStoredQueryNames();
```

4. Visualizzare i dettagli di una specifica query.

Il seguente frammento di codice illustra come richiamare i dettagli della query CustomerOrdersStartingWithA di proprietà dell'utente Smith.

```
StoredQuery storedQuery = process.getStoredQuery  
    ("CustomerOrdersStartingWithA");  
String selectClause = storedQuery.getSelectClause();  
String whereClause = storedQuery.getWhereClause();  
String orderByClause = storedQuery.getOrderByClause();  
Integer threshold = storedQuery.getThreshold();  
String owner = storedQuery.getOwner();
```

5. Eliminare una query memorizzata privata.

Il seguente frammento di codice illustra come eliminare una query memorizzata privata.

```
process.deleteStoredQuery("CustomerOrdersStartingWithA");
```

Sviluppo delle applicazioni per i processi di business

Un processo di business è costituito da una serie di attività relative al business richiamate in una sequenza specifica per raggiungere un obiettivo di business. Di seguito sono riportati gli esempi che illustrano il modo in cui è possibile sviluppare le applicazioni per azioni o processi tipici.

About this task

Un processo di business può essere un microflusso o un processo long-running:

- I microflussi sono processi di business short-running eseguiti in modo sincrono. Dopo un breve intervallo di tempo, il risultato viene restituito al chiamante.
- I processi long-running, che possono essere interrotti, vengono eseguiti come una sequenza di attività che sono legate insieme. L'uso di determinati costrutti in un processo determina interruzioni nel flusso del processo, ad esempio, richiamando un'attività umana, un servizio che utilizza un bind sincrono o utilizzando attività a tempo.

Rami paralleli del processo sono in genere esplorati in modo asincrono, vale a dire che attività in rami paralleli sono eseguite contemporaneamente. In base al tipo e all'impostazione della transazione di un'attività, quest'ultima può essere eseguita dalla relativa transazione.

Ruoli richiesti per le azioni sulle istanze del processo

L'accesso all'interfaccia BusinessFlowManager non garantisce che il chiamante possa eseguire tutte le azioni su un processo. Il chiamante deve essere collegato all'applicazione client con un ruolo che dispone dell'autorizzazione ad eseguire l'azione.

La tabella di seguito riportata illustra le azioni che possono essere effettuate su un'istanza del processo da un determinato ruolo.

Azione	Ruolo principale del chiamante		
	Lettores	Starter	Amministratore
createMessage	x	x	x
createWorkItem			x
delete			x
deleteWorkItem			x
forceTerminate			x
getActiveEventHandlers	x		x
getActivityInstance	x		x
getAllActivities	x		x
getAllWorkItems	x		x
getClientUISettings	x	x	x
getCustomProperties	x	x	x
getCustomProperty	x	x	x
getCustomPropertyNames	x	x	x
getFaultMessage	x	x	x
getInputClientUISettings	x	x	x
getInputMessage	x	x	x
getOutputClientUISettings	x	x	x
getOutputMessage	x	x	x
getProcessInstance	x	x	x
getVariable	x	x	x
getWaitingActivities	x	x	x
getWorkItems	x		x
restart			x
resume			x
setCustomProperty		x	x
setVariable			x
suspend			x
transferWorkItem			x

Ruoli richiesti per le azioni sulle attività business-process

L'accesso all'interfaccia BusinessFlowManager non garantisce che il chiamante possa eseguire tutte le azioni su un'attività. Il chiamante deve essere collegato all'applicazione client con un ruolo che dispone dell'autorizzazione ad eseguire l'azione.

La tabella di seguito riportata illustra le azioni che possono essere effettuate su un'istanza dell'attività da un determinato ruolo.

Azione	Ruolo principale del chiamante				
	Lettores	Editor	Proprietario potenziale	Proprietario	Amministratore
cancelClaim				x	x

Azione	Ruolo principale del chiamante				
	Letto	Editor	Proprietario potenziale	Proprietario	Amministratore
claim			x		x
complete				x	x
createMessage	x	x	x	x	x
createWorkItem					x
deleteWorkItem					x
forceComplete					x
forceRetry					x
getActivityInstance	x	x	x	x	x
getAllWorkItems	x	x	x	x	x
getClientUISettings	x	x	x	x	x
getCustomProperties	x	x	x	x	x
getCustomProperty	x	x	x	x	x
getCustomPropertyNames	x	x	x	x	x
getFaultMessage	x	x	x	x	x
getFaultNames	x	x	x	x	x
getInputMessage	x	x	x	x	x
getOutputMessage	x	x	x	x	x
getVariable	x	x	x	x	x
getVariableNames	x	x	x	x	x
getInputVariableNames	x	x	x	x	x
getOutputVariableNames	x	x	x	x	x
getWorkItems	x	x	x	x	x
setCustomProperty		x		x	x
setFaultMessage		x		x	x
setOutputMessage		x		x	x
setVariable					x
transferWorkItem				x Solo per gli amministratori o i proprietari potenziali	x

Gestione del ciclo di vita di un processo di business

Un'istanza di processo comincia ad esistere quando viene richiamato un metodo dell'API di Business Process Choreographer che può avviare un processo. La navigazione dell'istanza del processo continua fino a quando tutte le attività non sono in stato di fine. Nell'istanza del processo possono essere intraprese varie azioni per gestire il relativo ciclo di vita.

About this task

Di seguito sono riportati gli esempi che illustrano il modo in cui è possibile sviluppare le applicazioni per le seguenti azioni o processi del ciclo vitale tipico.

Avvio dei processi di business:

Il modo in cui viene avviato un processo di business dipende da se tale processo è un microflusso o un processo di lunga esecuzione. Il servizio che avvia il processo è anche importante per il modo in cui il processo viene avviato, il processo può disporre di un servizio di avvio univoco o di vari servizi di avvio.

About this task

Di seguito sono riportati gli esempi che illustrano il modo in cui è possibile sviluppare le applicazioni per gli scenari tipici per l'avvio dei microflussi e dei processi long-running.

Esecuzione di un microflusso che contiene un servizio di avvio univoco:

Un microflusso può essere avviato da un'attività di ricezione o di selezione. Il servizio di avvio è univoco se il microflusso viene avviato con un'attività receive o quando l'attività pick dispone solo di una definizione onMessage.

About this task

Se il microflusso implementa un'operazione richiesta-risposta, ovvero se il processo contiene una risposta, è possibile utilizzare il metodo call per eseguire il processo inoltrando il nome del modello di processo come parametro di richiamo.

Se il microflusso è un'operazione di sola andata, utilizzare il metodo sendMessage per eseguire il processo. Questo metodo non è illustrato nell'esempio.

Procedure

1. Opzionale: Elencare i modelli di processo per trovare il nome del processo che si desidera eseguire.

Se si conosce già il nome del processo, questa operazione è facoltativa.

```
ProcessTemplateData[] processTemplates = process.queryProcessTemplates
("PROCESS_TEMPLATE.EXECUTION_MODE =
PROCESS_TEMPLATE.EXECUTION_MODE.EXECUTION_MODE_MICROFLOW",
"PROCESS_TEMPLATE.NAME",
new Integer(50),
(TimeZone)null);
```

I risultati vengono ordinati per nome. La query restituisce una matrice contenente i primi 50 modelli da avviare mediante il metodo call.

2. Avviare il processo con un messaggio di input del tipo appropriato.

Quando si crea un messaggio, è necessario specificare il nome del tipo di messaggio, in modo che contenga la definizione del messaggio.

```
ProcessTemplateData template = processTemplates[0];
//creare un messaggio per la singola attività di avvio ricezione
ClientObjectWrapper input = process.createMessage
(template.getID(),
template.getInputMessageType());
DataObject myMessage = null;
if ( input.getObject() != null && input.getObject() instanceof DataObject )
{
myMessage = (DataObject)input.getObject();
//impostare le stringhe nel messaggio, ad esempio, un nome cliente
myMessage.setString("CustomerName", "Smith");
```

```

}

//eseguire il processo
ClientObjectWrapper output = process.call(template.getName(), input);
DataObject myOutput = null;
if ( output.getObject() != null && output.getObject() instanceof DataObject )
{
    myOutput = (DataObject)output.getObject();
    int order = myOutput.getInt("OrderNo");
}

```

Questa operazione crea un'istanza del modello di processo, CustomerTemplate, e invia alcuni dati del cliente. L'operazione prevede restituzioni solo quando il processo è completo. Il risultato del processo, OrderNo, viene restituito al chiamante.

Esecuzione di un microflusso che contiene un servizio di avvio non univoco:

Un microflusso può essere avviato da un'attività di ricezione o di selezione. Il servizio di avvio non è univoco se il microflusso viene avviato con un'attività pick con più definizioni onMessage.

About this task

Se il microflusso implementa un'operazione richiesta-risposta, ovvero se il processo contiene una risposta, è possibile utilizzare il metodo di richiamo per eseguire il processo inoltrando l'ID del servizio di avvio nella chiamata.

Se il microflusso è un'operazione di sola andata, utilizzare il metodo sendMessage per eseguire il processo. Questo metodo non è illustrato nell'esempio.

Procedure

1. Opzionale: Elencare i modelli di processo per trovare il nome del processo che si desidera eseguire.

Se si conosce già il nome del processo, questa operazione è facoltativa.

```

ProcessTemplateData[] processTemplates = process.queryProcessTemplates
("PROCESS_TEMPLATE.EXECUTION_MODE =
    PROCESS_TEMPLATE.EXECUTION_MODE.EXECUTION_MODE_MICROFLOW",
"PROCESS_TEMPLATE.NAME",
new Integer(50),
(TimeZone)null);

```

I risultati vengono ordinati per nome. La query restituisce una matrice contenente i primi 50 modelli da avviare come microflussi.

2. Determinare il servizio di avvio da richiamare.

Questo esempio utilizza il primo modello che viene trovato.

```

ProcessTemplateData template = processTemplates[0];
ActivityServiceTemplateData[] startActivities =
    process.getStartActivities(template.getID());

```

3. Avviare il processo con un messaggio di input del tipo appropriato.

Quando si crea un messaggio, è necessario specificare il nome del tipo di messaggio, in modo che contenga la definizione del messaggio.

```

ActivityServiceTemplateData activity = startActivities[0];
// creare un messaggio per il servizio da richiamare
ClientObjectWrapper input =
    process.createMessage(activity.getServiceTemplateID(),
        activity.getActivityTemplateID(),
        activity.getInputMessageType());
DataObject myMessage = null;

```

```

if ( input.getObject() != null && input.getObject() instanceof DataObject )
{
    myMessage = (DataObject)input.getObject();
    //impostare le stringhe nel messaggio, ad esempio, un nome cliente
    myMessage.setString("CustomerName", "Smith");
}
//eseguire il processo
ClientObjectWrapper output = process.call(activity.getServiceTemplateID(),
                                           activity.getActivityTemplateID(),
                                           input);
//verificare l'output del processo, ad esempio, un numero di ordine
DataObject myOutput = null;
if ( output.getObject() != null && output.getObject() instanceof DataObject )
{
    myOutput = (DataObject)output.getObject();
    int order = myOutput.getInt("OrderNo");
}

```

Questa operazione crea un'istanza del modello di processo, CustomerTemplate, e invia alcuni dati del cliente. L'operazione prevede restituzioni solo quando il processo è completo. Il risultato del processo, OrderNo, viene restituito al chiamante.

Avvio di un processo di lunga esecuzione che contiene un servizio di avvio univoco:

Se il servizio di avvio è univoco, è possibile utilizzare il metodo `initiate` e inviare il nome del modello di processo come parametro. Questo è il caso in cui il processo di lunga durata viene avviato con una singola attività di ricezione o di selezione e nel caso in cui la singola attività di selezione abbia solo una definizione per `onMessage`.

Procedure

1. Opzionale: Elencare i modelli di processo per trovare il nome del processo che si desidera avviare.

Se si conosce già il nome del processo, questa operazione è facoltativa.

```

ProcessTemplateData[] processTemplates = process.queryProcessTemplates
("PROCESS_TEMPLATE.EXECUTION_MODE =
PROCESS_TEMPLATE.EXECUTION_MODE.EXECUTION_MODE_LONG_RUNNING",
"PROCESS_TEMPLATE.NAME",
new Integer(50),
(TimeZone)null);

```

I risultati vengono ordinati per nome. La query restituisce una matrice contenente i primi 50 modelli che possono essere avviati dal metodo `initiate`.

2. Avviare il processo con un messaggio di input del tipo appropriato.

Quando si crea un messaggio, è necessario specificare il nome del tipo di messaggio, in modo che contenga la definizione del messaggio. Se si specifica un nome istanza di processo, questo non deve iniziare con un carattere di sottolineatura. Se non si specifica alcun nome dell'istanza di processo, si utilizza il PIID (Process Instance ID) in formato stringa.

```

ProcessTemplateData template = processTemplates[0];
//creare un messaggio per la singola attività di avvio ricezione
ClientObjectWrapper input = process.createMessage
(template.getID(),
template.getInputMessageType());
DataObject myMessage = null;
if ( input.getObject() != null && input.getObject() instanceof DataObject )
{
    myMessage = (DataObject)input.getObject();
    //impostare le stringhe nel messaggio, ad esempio, un nome cliente
    myMessage.setString("CustomerName", "Smith");
}

```

```

}
//avviare il processo
PIID piid = process.initiate(template.getName(), "CustomerOrder", input);

```

Questa operazione crea un'istanza, CustomerOrder, e invia alcuni dati del cliente. All'avvio del processo, l'operazione restituisce al chiamante l'ID oggetto della nuova istanza di processo.

L'avvio dell'istanza di processo è impostato sul chiamante della richiesta.

Questa persona riceve un elemento di lavoro per l'istanza del processo.

Vengono stabiliti gli amministratori del processo, i lettori e gli editor dell'istanza di processo e questi ricevono gli elementi di lavoro per l'istanza stessa. Vengono stabilite le istanze dell'attività follow-on. Queste vengono avviate automaticamente o, se si tratta di attività umane, o attività di ricezione o selezione, vengono creati elementi di lavoro per i potenziali proprietari.

Avvio di un processo di esecuzione prolungata che contiene un servizio di avvio non univoco:

Un processo di esecuzione prolungata può essere avviato mediante più attività di ricezione o di selezione. È possibile utilizzare il metodo initiate per avviare il processo. Se il servizio di avvio non è univoco, ad esempio, se il processo viene avviato con più attività pick o receive oppure un'attività pick dispone di più definizioni onMessage, è necessario identificare il servizio da richiamare.

Procedure

1. Opzionale: Elencare i modelli di processo per trovare il nome del processo che si desidera avviare.

Se si conosce già il nome del processo, questa operazione è facoltativa.

```

ProcessTemplateData[] processTemplates = process.queryProcessTemplates
("PROCESS_TEMPLATE.EXECUTION_MODE =
PROCESS_TEMPLATE.EXECUTION_MODE.EXECUTION_MODE_LONG_RUNNING",
"PROCESS_TEMPLATE.NAME",
new Integer(50),
(TimeZone)null);

```

I risultati vengono ordinati per nome. La query restituisce una matrice contenente i primi 50 modelli da avviare come processi long-running.

2. Determinare il servizio di avvio da richiamare.

```

ProcessTemplateData template = processTemplates[0];
ActivityServiceTemplateData[] startActivities =
process.getStartActivities(template.getID());

```

3. Avviare il processo con un messaggio di input del tipo appropriato.

Quando si crea un messaggio, è necessario specificare il nome del tipo di messaggio, in modo che contenga la definizione del messaggio. Se si specifica un nome istanza di processo, questo non deve iniziare con un carattere di sottolineatura. Se non si specifica alcun nome dell'istanza di processo, si utilizza il PIID (Process Instance ID) in formato stringa.

```

ActivityServiceTemplateData activity = startActivities[0];
// creare un messaggio per il servizio da richiamare
ClientObjectWrapper input = process.createMessage
(activity.getServiceTemplateID(),
activity.getActivityTemplateID(),
activity.getInputMessageType());

DataObject myMessage = null;
if ( input.getObject() != null && input.getObject() instanceof DataObject )
{
myMessage = (DataObject)input.getObject();
//impostare le stringhe nel messaggio, ad esempio, un nome cliente

```

```

    myMessage.setString("CustomerName", "Smith");
}
//avviare il processo
PIID piid = process.sendMessage(activity.getServiceTemplateID(),
                               activity.getActivityTemplateID(),
                               input);

```

Questa azione crea un'istanza ed inoltra alcuni dati del cliente. All'avvio del processo, l'operazione restituisce al chiamante l'ID oggetto della nuova istanza di processo.

L'avvio dell'istanza di processo è impostato sul chiamante della richiesta e riceve un elemento di lavoro per l'istanza stessa. Vengono stabiliti gli amministratori del processo, i lettori e gli editor dell'istanza di processo e questi ricevono gli elementi di lavoro per l'istanza stessa. Vengono stabilite le istanze dell'attività follow-on. Queste vengono avviate automaticamente o, se si tratta di attività umane, o attività di ricezione o selezione, vengono creati elementi di lavoro per i potenziali proprietari.

Sospensione e ripresa di un processo di business:

È possibile sospendere un'istanza di processo di livello superiore a esecuzione prolungata, mentre è in esecuzione e riprenderla per completarla.

Prima di iniziare

Il chiamante deve essere un amministratore dell'istanza di processo o di un processo di business. Per sospendere un'istanza di processo, quest'ultima deve essere in stato di esecuzione o in errore.

About this task

È possibile che si desideri sospendere un'istanza di processo, ad esempio, in modo che sia possibile configurare l'accesso a un sistema di back-end utilizzato in seguito nel processo. Quando i prerequisiti per il processo corrispondono, è possibile riprendere l'istanza del processo. È possibile che si desideri inoltre sospendere un processo per risolvere un problema che causa un errore nell'istanza di processo e riprendere quindi il processo una volta risolto il problema.

Procedure

1. Ottenere il processo in esecuzione, CustomerOrder, che si desidera sospendere.

```

ProcessInstanceData processInstance =
    process.getProcessInstance("CustomerOrder");

```

2. Sospendere l'istanza del processo.

```

PIID piid = processInstance.getID();
process.suspend( piid );

```

Questa azione sospende l'istanza del processo di livello superiore specificata. Lo stato dell'istanza di processo diventa sospesa. I processi secondari con l'attributo di autonomia impostato su child sono sospesi anche se sono in stato di esecuzione, in stato di errore, terminati o in stato compensazione. Anche le attività in linea associate a questa istanza di processo vengono sospese; le attività autonome associate a questa istanza di processo non vengono sospese.

In questo stato, le attività avviate possono sempre essere completate ma non vengono avviate nuove attività, per esempio un'operazione dell'azione umana nello stato richiesto può essere completata.

3. Riprendere l'istanza di processo.

```

process.resume( piid );

```


Questa azione colloca l'istanza del processo e i relativi processi secondari negli stati in cui si trovavano prima di essere sospesi.

Riavvio di un processo di business:

È possibile riavviare un'istanza di processo in stato terminato, finito, in errore o compensato.

Prima di iniziare

Il chiamante deve essere un amministratore dell'istanza di processo o di un processo di business.

About this task

Il riavvio di un'istanza di processo è simile all'avvio di un'istanza di processo. Tuttavia, quando viene riavviata un'istanza di processo, l'ID dell'istanza di processo è nota ed il messaggio di input per l'istanza è disponibile.

Se il processo dispone di più di un'attività pick o receive (nota anche come attività receive choice) che può creare l'istanza del processo, tutti i messaggi che appartengono a tali attività vengono utilizzati per riavviare l'istanza del processo. Se una qualunque di queste attività implementa un'operazione di richiesta-risposta, la risposta viene inviata di nuovo quando viene esplorata l'attività di risposta associata.

Procedure

1. Ottenere il processo che si desidera riavviare.

```
ProcessInstanceData processInstance =  
    process.getProcessInstance("CustomerOrder");
```

2. Riavviare l'istanza del processo.

```
PIID piid = processInstance.getID();  
process.restart( piid );
```

Questa azione riavvia l'istanza del processo specificata.

Chiusura di un'istanza del processo:

Talvolta, chi dispone dei privilegi di amministratore deve terminare un'istanza di processo di livello superiore che si trova in uno stato irreversibile. Poiché un'istanza del processo termina immediatamente, senza attendere i processi secondari o le attività in sospeso, terminare un'istanza di processo solo in situazioni eccezionali.

Procedure

1. Richiamare l'istanza di processo da terminare.

```
ProcessInstanceData processInstance =  
    process.getProcessInstance("CustomerOrder");
```

2. Terminare l'istanza di processo.

Se si termina un'istanza di processo, è possibile terminare l'istanza di processo con o senza compensazione.

Per terminare l'istanza del processo con la compensazione:

```
PIID piid = processInstance.getID();  
process.forceTerminate(piid, CompensationBehaviour.INVOKE_COMPENSATION);
```

Per terminare l'istanza del processo senza compensazione:

```
PIID piid = processInstance.getID();
process.forceTerminate(piid);
```

Se si termina l'istanza del processo con la compensazione, viene eseguita la compensazione del processo come se si fosse verificato un errore al livello superiore. Se si termina l'istanza del processo senza alcuna compensazione, l'istanza del processo viene terminata immediatamente senza attendere che le attività, attività da fare, o attività di chiamata in linea siano completate normalmente.

Le applicazioni avviate dal processo e le attività autonome correlate al processo non sono terminate dalla richiesta per forzare l'arresto. Se queste applicazioni stanno per essere terminate, è necessario aggiungere delle istruzioni all'applicazione del processo che terminano esplicitamente le applicazioni avviate dal processo.

Eliminazione delle istanze del processo:

Le istanze di processo completate vengono automaticamente eliminate dal database di Business Process Choreographer se la proprietà corrispondente viene impostata per il modello di processo nel modello di processo. È possibile voler conservare le istanze di processo nel database, ad esempio per eseguire la query di dati dalle istanze di processo che non sono scritte nel log di controllo. I dati memorizzati delle istanze di processo, tuttavia, non solo influiscono sullo spazio su disco e sulle prestazioni, ma impediscono anche la creazione di istanze di processo che utilizzano gli stessi valori di correlazione. È necessario quindi eliminare regolarmente i dati dell'istanza di processo dal database.

About this task

Per eliminare un'istanza di processo, è necessario disporre dei privilegi di amministratore del processo e l'istanza del processo deve essere di livello superiore.

Il seguente esempio illustra come eliminare tutte le istanze di un processo finito.

Procedure

1. Elencare le istanze di processo terminate.

```
QueryResultSet result =
    process.query("DISTINCT PROCESS_INSTANCE.PIID",
        "PROCESS_INSTANCE.STATE =
            PROCESS_INSTANCE.STATE.STATE_FINISHED",
        (String)null, (Integer)null, (TimeZone)null);
```

Questa azione restituisce una serie di risultati di query che elenca le istanze di processo terminate.

2. Eliminare le istanze di processo che sono terminate.

```
while (result.next() )
{
    PIID piid = (PIID) result.getOID(1);
    process.delete(piid);
}
```

Questa azione elimina l'istanza di processo selezionata e le relative attività in linea dal database.

Elaborazione di operazioni dell'attività umana

Le operazioni dell'attività umana in processi di business sono assegnate a diversi utenti di business mediante gli elementi di lavoro. Quando un processo viene avviato, vengono creati degli elementi di lavoro per i potenziali proprietari.

About this task

Quando un'attività umana viene attivata, vengono create un'istanza di attività e un'associata attività da fare. La gestione dell'attività umana e la gestione degli elementi di lavoro sono delegate a Human Task Manager. Qualsiasi cambiamento di stato dell'istanza di attività viene rispecchiato nell'istanza di attività e vice versa.

Un potenziale proprietario richiede l'attività. Questo utente è responsabile della fornitura di informazioni rilevanti e del completamento dell'attività.

Procedure

1. Elencare le attività che appartengono a un utente collegato e pronte per essere utilizzate:

```
QueryResultSet result =
    process.query("ACTIVITY.AIID",
        "ACTIVITY.STATE = ACTIVITY.STATE.STATE_READY AND
        ACTIVITY.KIND = ACTIVITY.KIND.KIND_STAFF AND
        WORK_ITEM.REASON =
            WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
        (String)null, (Integer)null, (TimeZone)null);
```

Questa operazione restituisce una serie di risultati della query che contiene le attività che possono essere elaborate dall'utente connesso.

2. Richiedere le attività su cui lavorare:

```
if (result.size() > 0)
{
    result.first();
    AIID aaid = (AIID) result.getOID(1);
    ClientObjectWrapper input = process.claim(aaid);
    DataObject activityInput = null ;
    if ( input.getObject() != null && input.getObject() instanceof DataObject )
    {
        activityInput = (DataObject)input.getObject();
        // leggere i valori
        ...
    }
}
```

Una volta richiesta l'attività, viene restituito il messaggio di input dell'attività stessa.

3. Una volta terminato il lavoro sull'attività, completare quest'ultima. L'attività può essere completata correttamente o con un messaggio di errore. Se l'attività ha esito positivo, viene inoltrato un messaggio di output. In caso contrario, l'attività passa nello stato di errore e viene inviato un messaggio di errore. Per queste azioni, è necessario creare i messaggi appropriati. Quando si crea un messaggio, è necessario specificare il nome del tipo di messaggio, in modo tale che contenga la definizione del messaggio.

- a. Per completare correttamente l'attività, creare un messaggio di output.

```
ActivityInstanceData activity = process.getActivityInstance(aaid);
ClientObjectWrapper output =
    process.createMessage(aaid, activity.getOutputMessageTypeName());
DataObject myMessage = null ;
```

```

if ( output.getObject() != null && output.getObject() instanceof DataObject )
{
    myMessage = (DataObject)output.getObject();
    //impostare le parti nel messaggio, ad esempio, un numero di ordine
    myMessage.setInt("OrderNo", 4711);
}

//completare l'attività
process.complete(aiid, output);

```

Questa azione imposta un messaggio di output contenente il numero dell'ordine.

- b. Per completare l'attività quando si verifica un errore, creare un messaggio di errore.

```

//recuperare gli errori modellati per l'attività attività umana
List faultNames = process.getFaultNames(aiid);

//creare un messaggio del tipo appropriato
ClientObjectWrapper myFault =
    process.createMessage(aiid, faultNames.get(0));

// impostare le parti nel messaggio di errore, ad esempio, un numero di ordine
DataObject myMessage = null ;
if ( myFault.getObject() != null && input.getObject() instanceof DataObject )
{
    myMessage = (DataObject)myFault.getObject();
    //impostare le parti nel messaggio, ad esempio, un nome cliente
    myMessage.setInt("error",1304);
}

process.complete(aiid, (String)faultNames.get(0), myFault);

```

Questa azione imposta l'attività in stato non riuscito o arrestato. Se il parametro **continueOnError** dell'attività nel modello di processo viene impostata su true, l'attività viene posizionata in stato non riuscito e la navigazione continua. Se il parametro **continueOnError** viene impostato su false e l'errore non viene individuato nell'ambito circostante, lo stato dell'attività diventa Arrestato. In questo stato l'attività può essere riparata utilizzando i comandi per forzare l'arresto o ritentare.

Elaborazione del flusso di lavoro di una sola persona

Alcuni flussi di lavoro vengono eseguiti da una sola persona, ad esempio, ordinare libri da una libreria online. Questo tipo di flusso di lavoro non ha percorsi paralleli. L'API `completeAndClaimSuccessor` supporta l'elaborazione di questo tipo di flusso di lavoro.

About this task

In una libreria online, per ordinare un libro l'acquirente completa una sequenza di azioni. Questa sequenza di azioni può essere effettuata come una serie di operazioni dell'attività umana (attività da fare). Se l'acquirente decide di acquistare più libri, questa attività equivale a richiedere l'operazione dell'attività umana successiva. Questo tipo di flusso di lavoro è conosciuto anche come *flusso di pagine*, in quanto le definizioni dell'interfaccia utente sono associate alle attività per controllare il flusso delle finestre di dialogo nell'interfaccia utente.

L'API `completeAndClaimSuccessor` completa un'operazione dell'attività umana e richiede l'attività successiva nella stessa istanza di processo per l'utente collegato. Tale API restituisce informazioni sulla successiva attività richiesta, incluso il messaggio di input da utilizzare. Poiché l'attività successiva è messa a disposizione

nella stessa transazione dell'attività completata, il comportamento transazionale di tutte le operazioni dell'attività umana nel modello di processo deve essere impostato su `participates`.

Confrontare questo esempio con l'esempio che utilizza sia la API di Business Flow Manager che la API di Human Task Manager.

Procedure

1. Richiedere la prima attività della sequenza di attività.

```
//
//Eeguire una query dell'elenco di attività che possono essere richieste all'utente collegato
//
QueryResultSet result =
    process.query("ACTIVITY.AIID",
        "PROCESS_INSTANCE.NAME = 'CustomerOrder' AND
        ACTIVITY.STATE = ACTIVITY.STATE.STATE_READY AND
        ACTIVITY.KIND = ACTIVITY.KIND.KIND_STAFF AND
        WORK_ITEM.REASON =
            WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
        (String)null, (Integer)null, (TimeZone)null);

...
//
//Richiedere la prima attività
//
if (result.size() > 0)
{
    result.first();
    AIID aaid = (AIID) result.getOID(1);
    ClientObjectWrapper input = process.claim(aaid);
    DataObject activityInput = null ;
    if ( input.getObject() != null && input.getObject() instanceof DataObject )
    {
        activityInput = (DataObject)input.getObject();
        // leggere i valori
        ...
    }
}
```

Una volta richiesta l'attività, viene restituito il messaggio di input dell'attività stessa.

2. Una volta terminato il lavoro sull'attività, completare l'attività e richiedere l'attività successiva.

Per completare l'attività, trasmettere un messaggio di output. Quando si crea il messaggio di output, è necessario specificare il nome del tipo di messaggio, in modo tale che contenga la definizione del messaggio.

```
ActivityInstanceData activity = process.getActivityInstance(aaid);
ClientObjectWrapper output =
    process.createMessage(aaid, activity.getOutputMessageTypeName());
DataObject myMessage = null ;
if ( output.getObject() != null && output.getObject() instanceof DataObject )
{
    myMessage = (DataObject)output.getObject();
    //impostare le parti nel messaggio, ad esempio, un numero di ordine
    myMessage.setInt("OrderNo", 4711);
}

//completare l'attività e richiedere l'attività successiva
CompleteAndClaimSuccessorResult successor =
    process.completeAndClaimSuccessor(aaid, output);
```

Questa azione imposta un messaggio di output che contiene il numero di ordine e richiede l'attività successiva nella sequenza. Se `AutoClaim` è impostato

per attività in successione e se esistono più percorsi da seguire, vengono richiamate tutte le attività in successione e un'attività casuale viene restituita come attività successiva. Se non esistono altre attività in successione che possono essere assegnate a questo utente, viene restituito il valore Null.

Se il processo contiene percorsi paralleli che possono essere seguiti e questi percorsi contengono attività umane per cui l'utente collegato è un potenziale proprietario di più di una di queste attività, un'attività casuale viene richiesta automaticamente e restituita come attività successiva.

3. Utilizzare l'attività successiva.

```
String name = successor.getActivityName();

ClientObjectWrapper nextInput = successor.getInputMessage();
if ( nextInput.getObject() !=
    null && nextInput.getObject() instanceof DataObject )
{
    activityInput = (DataObject)input.getObject();
    // leggere i valori
    ...
}

aiid = successor.getAIID();
```

4. Continuare con il passo 2 per completare l'attività.

Attività correlate

"Elaborazione del flusso di lavoro di una sola persona che include attività umana" a pagina 105

Alcuni flussi di lavoro vengono eseguiti da una sola persona, ad esempio, ordinare libri da una libreria online. Questo esempio mostra come effettuare la sequenza di azioni per ordinare il libro come una serie di operazioni dell'attività umana (attività da fare). Sia il Business Flow Manager che le API di Human Task Manager sono utilizzate per elaborare il flusso di lavoro.

Invio di un messaggio a un'attività in attesa

È possibile utilizzare le attività dei messaggi in arrivo (attività di ricezione, attività onMessage in pick, onEvent nei gestori eventi) per sincronizzare un processo in esecuzione con eventi del "mondo esterno". Ad esempio, la ricezione di un'e-mail da un cliente in risposta ad una richiesta di informazioni potrebbe rappresentare tale evento.

About this task

È possibile utilizzare le attività di origine per inviare il messaggio all'attività.

Procedure

1. Elencare i modelli del servizio delle attività che sono in attesa di un messaggio da parte dell'utente collegato in un'istanza di processo con un ID istanza di processo specifico.

```
ActivityServiceTemplateData[] services = process.getWaitingActivities(piid);
```

2. Inviare un messaggio al primo servizio in attesa.

Si presume che il primo servizio sia quello che si desidera utilizzare. Il chiamante deve essere un potenziale iniziatore dell'attività che riceve il messaggio o un amministratore dell'istanza del processo.

```
VTID vtid = services[0].getServiceTemplateID();
ATID atid = services[0].getActivityTemplateID();
String inputType = services[0].getInputMessageType();
```

```
// creare un messaggio per il servizio da richiamare
```

```

ClientObjectWrapper message =
    process.createMessage(vtid, atid, inputMessageTypeName);
DataObject myMessage = null;
if ( message.getObject() != null && message.getObject() instanceof DataObject )
{
    myMessage = (DataObject)message.getObject();
    //impostare le stringhe nel messaggio, ad esempio, ordinare della cioccolata
    myMessage.setString("Order", "chocolate");
}

// inviare il messaggio all'attività in attesa
process.sendMessage(vtid, atid, message);
}

```

Questa azione invia il messaggio specificato all'Activity Service in attesa e inoltra alcuni dati degli ordini.

Inoltre, è possibile specificare l'ID dell'istanza di processo, per verificare che il messaggio sia stato inviato all'istanza di processo specificata. Se non viene specificato l'ID dell'istanza di processo, il messaggio viene inviato all'Activity Service e l'istanza di processo identificata dai valori di correlazione nel messaggio. Se tale ID viene specificato, l'istanza di processo rilevata utilizzando i valori di correlazione viene controllata per verificare se dispone dell'ID dell'istanza di processo specificato.

Gestione eventi

Un processo di business completo e ciascuno dei relativi ambiti può essere associato ai gestori eventi richiamati se si verificano eventi associati. I gestori eventi sono simili per ricevere attività pick o receive, poiché un processo può fornire le operazioni dei servizi Web utilizzando i gestori eventi.

About this task

È possibile richiamare un gestore eventi tutte le volte in cui è in esecuzione l'ambito corrispondente. Inoltre, più istanze di un gestore eventi possono essere attivate simultaneamente.

Il seguente frammento del codice illustra il modo in cui ottenere i gestori eventi attivi per una determinata istanza di processo e il modo in cui inviare un messaggio di input.

Procedure

1. Determinare i dati dell'ID dell'istanza del processo ed elencare il gestore eventi attivo per il processo.

```

ProcessInstanceData processInstance =
    process.getProcessInstance( "CustomerOrder2711");
EventHandlerTemplateData[] events = process.getActiveEventHandlers(
    processInstance.getID() );

```

2. Inviare il messaggio di input.

Questo esempio utilizza il primo gestore eventi che viene trovato.

```

EventHandlerTemplateData event = null;
if ( events.length > 0 )
{
    event = events[0];

    // creare un messaggio per il servizio da richiamare
    ClientObjectWrapper input = process.createMessage(
        event.getID(), event.getInputMessageTypeName());

    if (input.getObject() != null && input.getObject() instanceof DataObject )
    {

```

```

        DataObject inputMessage = (DataObject)input.getObject();
        // impostare il contenuto del messaggio, ad esempio, un nome cliente,
il numero di ordine
        inputMessage.setString("CustomerName", "Smith");
        inputMessage.setString("OrderNo", "2711");

        // inviare il messaggio
        process.sendMessage( event.getProcessTemplateName(),
                            event.getPortTypeNamespace(),
                            event.getPortTypeName(),
                            event.getOperationName(),

        input );
    }
}

```

Questa azione invia il messaggio specificato al gestore eventi attivo per il processo.

Analisi dei risultati di un processo

Un processo può presentare operazioni di servizi Web modellate come operazioni di sola andata o di richiesta-risposta WSDL (Web Services Description Language). I risultati di processi di esecuzione prolungata con interfacce univoche non possono essere recuperate mediante il metodo `getOutputMessage`, perché il processo non ha output. Tuttavia, è possibile eseguire la query dei contenuti di variabili.

About this task

I risultati del processo vengono memorizzati nel database solo se il modello di processo, dal quale deriva l'istanza di processo, non specifica l'eliminazione automatica delle istanze di processo derivate.

Procedure

Analizzare i risultati del processo, ad esempio, controllare il numero dell'ordine.

```

QueryResultSet result = process.query
    ("PROCESS_INSTANCE.PIID",
    "PROCESS_INSTANCE.NAME = 'CustomerOrder' AND
    PROCESS_INSTANCE.STATE =
    PROCESS_INSTANCE.STATE.STATE_FINISHED",
    (String)null, (Integer)null, (TimeZone)null);
if (result.size() > 0)
{
    result.first();
    PIID piid = (PIID) result.getOID(1);
    ClientObjectWrapper output = process.getOutputMessage(piid);
    DataObject myOutput = null;
    if ( output.getObject() != null && output.getObject() instanceof DataObject )
    {
        myOutput = (DataObject)output.getObject();
        int order = myOutput.getInt("OrderNo");
    }
}

```

Ripristino delle attività

Un processo di lunga esecuzione può contenere attività anch'esse di lunga esecuzione. Queste attività potrebbero rilevare degli errori ed entrare in stato di arresto. Un'attività in stato di esecuzione potrebbe non rispondere. In entrambi i casi, un amministratore di processo può agire sull'attività secondo varie modalità, in modo che la navigazione del processo possa continuare.

About this task

L'API di Business Process Choreographer fornisce i metodi `forceRetry` e `forceComplete` per il ripristino delle attività. Di seguito sono riportati degli esempi che illustrano il modo in cui è possibile aggiungere delle azioni di ripristino per le attività delle applicazioni di cui si dispone.

Operazione per forzare il completamento di un'attività: About this task

Le attività nei processi di lunga esecuzione possono talvolta rilevare degli errori. Se tali errori non sono rilevati da un gestore errori nell'ambito associato e ed il relativo modello di attività specifico che l'attività termina quando si verifica un errore, lo stato dell'attività diventa Arrestata, in modo che possa essere riparata. In questo stato, è possibile forzare il completamento dell'attività.

Inoltre, è possibile forzare il completamento delle attività in stato di esecuzione, se ad esempio un'attività non risponde.

Esistono ulteriori requisiti per alcuni tipi di attività.

Operazione dell'attività umana

È possibile inoltrare i parametri alla chiamata che forza il completamento, come ad esempio il messaggio che avrebbe dovuto essere inviato o l'errore che avrebbe dovuto essere rilevato.

Attività script

Non è possibile inoltrare parametri nel richiamo forza completamento. Tuttavia, è necessario impostare le variabili che devono essere ripristinate.

Attività di chiamata

Inoltre, è possibile forzare il completamento delle attività di chiamata che richiamano un servizio asincrono che non è un processo secondario se tali attività si trovano in stato di esecuzione. Potrebbe essere necessario effettuare tale operazione, ad esempio, se il servizio asincrono viene richiamato e non risponde.

Procedure

1. Elencare le attività arrestate in stato terminato.

```
QueryResultSet result =
    process.query("DISTINCT ACTIVITY.AIID",
        "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND
        PROCESS_INSTANCE.NAME='CustomerOrder'",
        (String)null, (Integer)null, (TimeZone)null);
```

Questa operazione restituisce le attività arrestate per l'istanza del processo CustomerOrder.

2. Completare l'attività, per esempio, un'operazione dell'attività umana arrestate.

In questo esempio, viene inoltrato un messaggio di output.

```
if (result.size() > 0)
{
    result.first();
    AIID aaid = (AIID) result.getOID(1);
    ActivityInstanceData activity = process.getActivityInstance(aaid);
    ClientObjectWrapper output =
        process.createMessage(aaid, activity.getOutputMessageType());
    DataObject myMessage = null;
    if ( output.getObject() != null && output.getObject() instanceof DataObject )
    {
        myMessage = (DataObject)output.getObject();
        //impostare le parti nel messaggio, ad esempio, un numero di ordine
        myMessage.setInt("OrderNo", 4711);
    }
}
```

```

        boolean continueOnError = true;
        process.forceComplete(aiid, output, continueOnError);
    }

```

Questa operazione completa l'attività. Se si verifica un errore, il parametro **continueOnError** determina l'azione da eseguire se l'errore viene fornito con la richiesta `forceComplete`.

Nell'esempio, **continueOnError** è `true`. Questo valore indica che, se viene fornito un errore, lo stato dell'attività diventa non riuscita. L'errore viene propagato agli ambiti associati dell'attività fino a quando non viene gestito o non si raggiunge l'ambito del processo. In seguito, il processo viene messo in stato di errore e, infine, raggiunge lo stato di errore.

Ripetizione dell'esecuzione di un'attività arrestata: About this task

Se un'attività in un processo di esecuzione prolungata rileva un errore nell'ambito associato e se il modello dell'attività associata specifica che l'attività si arresta al verificarsi di un errore, lo stato dell'attività diventa arrestato, in modo da poter essere ripristinata. È possibile ripetere l'esecuzione dell'attività.

È possibile impostare delle variabili utilizzate dall'attività. Ad eccezione delle attività di script, è possibile anche inoltrare i parametri alla chiamata su cui è stato forzato il completamento, ad esempio il messaggio previsto dall'attività.

Procedure

1. Elencare le attività arrestate.

```

QueryResultSet result =
    process.query("DISTINCT ACTIVITY.AIID",
        "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND
        PROCESS_INSTANCE.NAME='CustomerOrder'",
        (String)null, (Integer)null, (TimeZone)null);

```

Questa operazione restituisce le attività arrestate per l'istanza del processo `CustomerOrder`.

2. Riprovare l'esecuzione dell'attività, per esempio, un'operazione dell'attività umana arrestata.

```

if (result.size() > 0)
{
    result.first();
    AIID aiid = (AIID) result.getOID(1);
    ActivityInstanceData activity = process.getActivityInstance(aiid);
    ClientObjectWrapper input =
        process.createMessage(aiid, activity.getOutputMessageTypeName());
    DataObject myMessage = null;
    if ( input.getObject() != null && input.getObject() instanceof DataObject )
    {
        myMessage = (DataObject)input.getObject();
        //impostare le stringhe nel messaggio, ad esempio, ordinare della cioccolata
        myMessage.setString("OrderNo", "chocolate");
    }

    boolean continueOnError = true;
    process.forceRetry(aiid, input, continueOnError);
}

```

Questa operazione consente di ripetere l'attività. Se si verifica un errore, il parametro **continueOnError** determina l'azione da eseguire se si verifica un errore durante l'elaborazione della richiesta `forceRetry`.

Nell'esempio, **continueOnError** è `true`. Questo indica che se si verifica un errore durante l'elaborazione della richiesta `forceRetry`, l'attività viene collocata

nello stato di errore. L'errore viene propagato agli ambiti associati dell'attività fino a quando non viene gestito o non si raggiunge l'ambito del processo. Il processo viene quindi messo in stato di errore e viene eseguito un gestore di errori sul livello del processo prima che lo stato del processo finisca nella stato di errore.

Interfaccia BusinessFlowManagerService

L'interfaccia BusinessFlowManagerService mostra le funzioni del processo di business che possono essere richiamate da un'applicazione client.

I metodi che possono essere richiamati dall'interfaccia BusinessFlowManagerService dipendono dallo stato del processo o dall'attività e dai privilegi dell'utente che utilizza l'applicazione contenente il metodo. I metodi principali per la modifica degli oggetti del processo di business sono elencati di seguito. Per ulteriori informazioni su tali metodi ed altri metodi che sono disponibili nell'interfaccia BusinessFlowManagerService, consultare Javadoc nel package com.ibm.bpe.api.

Modelli del processo

Un modello di processo è un modello di processo con versione, distribuito ed installato che contiene la specifica di un processo di business. È possibile eseguire l'istanza di tale modello ed avviarlo immettendo le richieste appropriate, ad esempio, sendMessage(). L'esecuzione di un'istanza di processo viene guidata automaticamente dal server.

Tabella 24. Metodi API per i modelli di processo

Metodo	Descrizione
getProcessTemplate	Richiama il modello di processo specificato.
queryProcessTemplates	Richiama i modelli di processo memorizzati nel database.

Istanze del processo

I metodi API di seguito riportati sono collegati all'avvio di istanze di processo.

Tabella 25. I metodi API sono collegati all'avvio di istanze di processo

Metodo	Descrizione
call	Crea ed esegue un microflusso.
callWithReplyContext	Crea ed esegue un microflusso con un servizio di avvio univoco o un processo di lunga esecuzione con un servizio di avvio univoco dal modello di processo specificato. la chiamata attende in modo sincronico il risultato.
callWithUISettings	Crea ed esegue un microflusso, quindi restituisce il messaggio di output e le impostazioni dell'interfaccia utente del client.

Tabella 25. I metodi API sono collegati all'avvio di istanze di processo (Continua)

Metodo	Descrizione
initiate	Crea un'istanza di processo e ne inizia l'elaborazione. Utilizzare questo metodo per i processi di lunga esecuzione. Inoltre, è possibile utilizzare questo metodo per i microflussi che si desidera far scattare e dimenticare.
sendMessage	Invia il messaggio specificato al servizio dell'attività specificata e all'istanza del processo. Se non esiste un'istanza del processo con gli stessi valori di correlazione, l'istanza viene creata. Il processo può avere servizi di avvio univoci o non univoci.
getStartActivities	Restituisce le informazioni sulle attività che possono avviare un'istanza di processo dal modello di processo specificato.
getActivityServiceTemplate	Richiama il modello del servizio dell'attività specificata.

Tabella 26. Metodi dell'API per il controllo del ciclo vita delle istanze di processo

Metodo	Descrizione
suspend	Sospende l'esecuzione di un'istanza di processo di lunga esecuzione di livello superiore in stato di esecuzione o in errore.
resume	Riprende l'esecuzione di un'istanza del processo di lunga esecuzione di livello superiore in stato sospeso.
restart	Riavvia un'istanza di processo di livello superiore di lunga esecuzione, che si trova nello stato terminato o non riuscito.
forceTerminate	Termina l'istanza del processo di livello superiore specificata, i relativi processi secondari con l'autonomia child e le relative attività in esecuzione, richieste o in attesa.
delete	Elimina l'istanza di processo di livello superiore specificata ed i relativi processi secondari con l'autonomia child.
query	Richiama le proprietà dal database che corrispondono ai criteri di ricerca.

Attività

Per le attività di chiamata, è possibile specificare nel modello di processo che tali attività continuino in situazioni di errore. Se il flag `continueOnError` viene impostato su `false` e si verifica un errore non gestito, lo stato dell'attività diventa `Arrestata`. Un amministratore del processo può quindi ripristinare l'attività. Il flag `continueOnError` e le relative funzioni di ripristino possono, ad esempio, essere utilizzati in un processo di di lunga esecuzione, dove un'attività di chiamata non viene completata correttamente con frequenza occasionale, ma lo sforzo per modellare la compensazione e la gestione dell'errore è troppo elevato.

I seguenti metodi sono disponibili per le attività di ripristino.

Tabella 27. Metodi dell'API per il controllo del ciclo vita delle istanze di attività

Metodo	Descrizione
claim	Richiede l'istanza di un'attività pronta affinché un utente possa effettuare operazioni sull'attività.
cancelClaim	Annulla la richiesta dell'istanza di attività.
complete	Completa l'istanza dell'attività
completeAndClaimSuccessor	Completa l'istanza di attività e richiama la successiva nella stessa istanza di processo per l'utente collegato.
forceComplete	Forza il completamento di un'istanza dell'attività che è in stato di esecuzione o arrestato.
forceRetry	Forza la ripetizione di un'istanza dell'attività che è in stato di esecuzione o arrestato.
query	Richiama le proprietà dal database che corrispondono ai criteri di ricerca.

Variabili e proprietà personalizzate

L'interfaccia fornisce i metodi get e set per richiamare ed impostare i valori per le variabili. Inoltre, è possibile associare le proprietà denominate e richiamare le proprietà denominate dalle istanze del processo e dell'attività. I nomi delle proprietà personalizzate ed i valori devono essere del tipo java.lang.String.

Tabella 28. Metodi API per le variabili e le proprietà personalizzate

Metodo	Descrizione
getVariable	Richiama la variabile specificata.
setVariable	Imposta la variabile specificata.
getCustomProperty	Richiama la proprietà personalizzata denominata dell'attività specificata o dell'istanza del processo.
getCustomProperties	Richiama le proprietà personalizzate dell'attività o dell'istanza di processo specificata.
getCustomPropertyNames	Richiama i nomi delle proprietà personalizzate per l'attività specificata o per l'istanza di processo.
setCustomProperty	Memorizza i valori specifici della personalizzazione per una determinata attività o istanza di processo.

Sviluppo di applicazioni per attività umane

Un'attività è il mezzo mediante il quale i componenti richiamano gli utenti come servizi oppure mediante il quale gli utenti stessi richiamano i servizi. Di seguito sono riportati gli esempi relativi alle applicazioni tipiche di attività umana.

About this task

Per ulteriori informazioni sull'API di Human Task Manager, consultare Javadoc nel package `com.ibm.task.api`.

Avvio di un'attività di chiamata che invoca un'interfaccia asincrona

Un'attività di chiamata è associata a un componente Service Component Architecture (SCA). Quando viene avviata l'attività, invoca il componente SCA. Avviare un'attività di chiamata in modalità asincrona solo se il componente SCA associato può essere invocato in modalità asincrona.

About this task

Tale componente SCA può, per esempio, essere implementato come un processo microflusso o come una semplice classe Java.

Questo scenario crea un'istanza di un modello di attività e inoltra alcuni dati del cliente. L'attività resta nello stato di esecuzione fino a quando non torna l'operazione di andata e ritorno. Il risultato dell'attività, `OrderNo`, viene restituita al chiamante.

Procedure

1. Opzionale: Elencare i modello di attività per trovare il nome dell'attività di chiamata che si desidera eseguire

Questo passo è facoltativo se già si conosce il nome dell'attività.

```
TaskTemplate[] taskTemplates = task.queryTaskTemplates(
    "TASK_TEMPL.KIND=TASK_TEMPL.KIND.KIND_ORIGINATING",
    "TASK_TEMPL.NAME",
    new Integer(50),
    (TimeZone)null);
```

I risultati vengono ordinati per nome. La query restituisce una matrice contenente i primi 50 modelli di origine memorizzati.

2. Creare un messaggio di input di tipo appropriato.

```
TaskTemplate template = taskTemplates[0];

// creare un messaggio per l'attività selezionata
ClientObjectWrapper input = task.createInputMessage( template.getID());
DataObject myMessage = null ;
if ( input.getObject() != null && input.getObject() instanceof DataObject )
{
    myMessage = (DataObject)input.getObject();
    //impostare le parti nel messaggio, ad esempio, un nome cliente
    myMessage.setString("CustomerName", "Smith");
}
```

3. Creare l'attività ed eseguirla in modo sincrono.

Affinché un'attività venga eseguita in modo sincrono, deve essere un'operazione in due tempi. L'esempio utilizza il metodo `createAndCallTask` per creare ed eseguire l'attività.

```
ClientObjectWrapper output = task.createAndCallTask( template.getName(),
                                                    template.getNamespace(),
                                                    input);
```

4. Analizzare il risultato dell'attività.

```
DataObject myOutput = null;
if ( output.getObject() != null && output.getObject() instanceof DataObject )
{
    myOutput = (DataObject)output.getObject();
    int order = myOutput.getInt("OrderNo");
}
```

Avvio di un'attività di chiamata che invoca un'interfaccia asincrona

Un'attività di chiamata è associata a un componente Service Component Architecture (SCA). Quando viene avviata l'attività, invoca il componente SCA. Avviare un'attività di chiamata in modalità asincrona solo se il componente SCA associato può essere invocato in modalità asincrona.

About this task

Tale componente SCA può, per esempio, essere implementato come un processo di esecuzione prolungata o un'operazione univoca.

Questo scenario crea un'istanza di un modello di attività e inoltra alcuni dati del cliente.

Procedure

1. Opzionale: Elencare i modello di attività per trovare il nome dell'attività di chiamata che si desidera eseguire

Questo passo è facoltativo se già si conosce il nome dell'attività.

```
TaskTemplate[] taskTemplates = task.queryTaskTemplates
("TASK_TEMPL.KIND=TASK_TEMPL.KIND.KIND_ORIGINATING",
 "TASK_TEMPL.NAME",
 new Integer(50),
 (TimeZone)null);
```

I risultati vengono ordinati per nome. La query restituisce una matrice contenente i primi 50 modelli di origine memorizzati.

2. Creare un messaggio di input di tipo appropriato.

```
TaskTemplate template = taskTemplates[0];

// creare un messaggio per l'attività selezionata
ClientObjectWrapper input = task.createInputMessage( template.getID());
DataObject myMessage = null ;
if ( input.getObject() != null && input.getObject() instanceof DataObject )
{
    myMessage = (DataObject)input.getObject();
    //impostare le parti nel messaggio, ad esempio, un nome cliente
    myMessage.setString("CustomerName", "Smith");
}
```

3. Creare l'attività ed eseguirla in modo asincrono.

L'esempio utilizza il metodo `createAndStartTask` per creare ed eseguire l'attività.

```
task.createAndStartTask( template.getName(),
                        template.getNamespace(),
                        input,
                        (ReplyHandlerWrapper)null);
```

Creazione e avvio di un'istanza di attività

Questo scenario illustra come creare un'istanza di un modello di attività che definisce un'attività di collaborazione (nota anche come *attività umana* nell'API) e avviare l'istanza di attività.

Procedure

1. Opzionale: Elencare i modelli di attività per trovare il nome dell'attività di collaborazione che si desidera eseguire.

Questo passo è facoltativo se già si conosce il nome dell'attività.

```

TaskTemplate[] taskTemplates = task.queryTaskTemplates
("TASK_TEMPL.KIND=TASK_TEMPL.KIND.KIND_HUMAN",
 "TASK_TEMPL.NAME",
 new Integer(50),
 (TimeZone)null);

```

I risultati vengono ordinati per nome. La query restituisce un array che contiene i primi 50 modelli di attività ordinati.

2. Creare un messaggio di input di tipo appropriato.

```

TaskTemplate template = taskTemplates[0];

// creare un messaggio per l'attività selezionata
ClientObjectWrapper input = task.createInputMessage( template.getID());
DataObject myMessage = null ;
if ( input.getObject() != null && input.getObject() instanceof DataObject )
{
    myMessage = (DataObject)input.getObject();
    //impostare le parti nel messaggio, ad esempio, un nome cliente
    myMessage.setString("CustomerName", "Smith");
}

```

3. Creare e avviare l'attività di collaborazione; un gestore di risposte non è specificato in questo esempio.

L'esempio utilizza il metodo `createAndStartTask` per creare ed avviare l'attività.

```

TKIID tkiid = task.createAndStartTask( template.getName(),
                                       template.getNamespace(),
                                       input,
                                       (ReplyHandlerWrapper)null);

```

Gli elementi di lavoro vengono creati per gli utenti che effettuano operazioni con l'istanza dell'attività. Ad esempio, un proprietario potenziale può richiedere la nuova istanza di attività.

4. Richiedere l'istanza dell'attività.

```

ClientObjectWrapper input2 = task.claim(tkiid);
DataObject taskInput = null ;
if ( input2.getObject() != null && input2.getObject() instanceof DataObject )
{
    taskInput = (DataObject)input2.getObject();
    // leggere i valori
    ...
}

```

Una volta richiesta l'istanza dell'attività, viene restituito il messaggio di input dell'attività stessa.

Elaborazione di attività da fare o di collaborazione

Le attività da fare (conosciute anche come *attività di partecipazione* nell'API) o le attività di collaborazione (conosciute anche come *attività umane* nell'API) sono assegnate a diverse persone dell'azienda attraverso elementi di lavoro. Le attività da fare e gli elementi di lavoro associati vengono creati, per esempio, quando un processo naviga verso un'operazione dell'attività umana.

About this task

Uno dei potenziali proprietari richiede l'attività associata all'elemento di lavoro. Questo utente è responsabile della fornitura di informazioni rilevanti e del completamento dell'attività.

Procedure

1. Elencare le attività che appartengono a un utente collegato e pronte per essere utilizzate.


```

QueryResultSet result =
    task.query("TASK.TKIID",
              "TASK.STATE = TASK.STATE.STATE_READY AND
              (TASK.KIND = TASK.KIND.KIND_PARTICIPATING OR
              TASK.KIND = TASK.KIND.KIND_HUMAN)AND
              WORK_ITEM.REASON =
              WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
              (String)null, (Integer)null, (TimeZone)null);

```

Questa operazione restituisce una serie di risultati della query che contiene le attività che possono essere elaborate dall'utente connesso.

2. Richiedere le attività su cui lavorare.

```

if (result.size() > 0)
{
    result.first();
    TKIID tkiid = (TKIID) result.getOID(1);
    ClientObjectWrapper input = task.claim(tkiid);
    DataObject taskInput = null ;
    if ( input.getObject() != null && input.getObject() instanceof DataObject )
    {
        taskInput = (DataObject)input.getObject();
        // leggere i valori
        ...
    }
}

```

Una volta richiesta l'attività, viene restituito il messaggio di input dell'attività stessa.

3. Una volta terminato il lavoro sull'attività, completare quest'ultima.

L'attività può essere completata correttamente o con un messaggio di errore. Se l'attività ha esito positivo, viene inoltrato un messaggio di output. Se l'attività non è riuscita correttamente, viene restituito un messaggio di errore. Per queste azioni, è necessario creare i messaggi appropriati.

- a. Per completare correttamente l'attività, creare un messaggio di output.

```

ClientObjectWrapper output =
    task.createOutputMessage(tkiid);
DataObject myMessage = null ;
if ( output.getObject() != null && output.getObject() instanceof DataObject )
{
    myMessage = (DataObject)output.getObject();
    //impostare le parti nel messaggio, ad esempio, un numero di ordine
    myMessage.setInt("OrderNo", 4711);
}

//completare l'attività
task.complete(tkiid, output);

```

Questa azione imposta un messaggio di output contenente il numero dell'ordine. Lo stato dell'attività diventa terminata.

- b. Per completare l'attività quando si verifica un errore, creare un messaggio di errore.

```

//richiamare gli errori modellati per l'attività
List faultNames = task.getFaultNames(tkiid);

//creare un messaggio del tipo appropriato
ClientObjectWrapper myFault =
    task.createFaultMessage(tkiid, (String)faultNames.get(0));

// impostare le parti nel messaggio di errore, ad esempio, un numero di ordine
DataObject myMessage = null ;
if ( myFault.getObject() != null && myFault.getObject() instanceof DataObject )
{
    myMessage = (DataObject)myFault.getObject();
}

```

```
//impostare le parti nel messaggio, ad esempio, un nome cliente
myMessage.setInt("error",1304);
}
```

```
task.complete (tkiid, (String)faultNames.get(0), myFault);
```

Questa azione imposta un messaggio di errore contenente il codice errore. Lo stato dell'attività diventa non riuscita.

Sospensione e ripresa di un'istanza di attività

È possibile sospendere istanze di attività di collaborazione (conosciute anche come *attività umane* nell'API) o istanze di attività da fare (conosciute anche come *attività di partecipazione* nell'API).

Prima di iniziare

L'istanza dell'attività può essere in stato pronto o richiesto. È possibile effettuarvi un'escalation. Il chiamante deve essere il proprietario, il creatore o l'amministratore dell'istanza dell'attività.

About this task

È possibile sospendere un'istanza dell'attività quando è in esecuzione. Se si desidera effettuare tale operazione, ad esempio, in modo che sia possibile raccogliere le informazioni necessarie per completare l'attività. Quando sono disponibili le informazioni, è possibile riprendere l'istanza dell'attività.

Procedure

1. Ottenere un elenco di attività che sono richieste dall'utente al momento collegato.

```
QueryResultSet result = task.query("DISTINCT TASK.TKIID",
                                   "TASK.STATE = TASK.STATE.STATE_CLAIMED",
                                   (String)null,
                                   (Integer)null,
                                   (TimeZone)null);
```

Questa operazione restituisce una serie di risultati della query che contiene un elenco di attività richieste dall'utente al momento collegato.

2. Sospendere l'istanza dell'attività.

```
if (result.size() > 0)
{
    result.first();
    TKIID tkiid = (TKIID) result.getOID(1);
    task.suspend(tkiid);
}
```

Questa azione sospende l'istanza dell'attività specificata. Lo stato dell'istanza dell'attività diventa sospesa.

3. Riprendere l'istanza di processo.

```
task.resume( tkiid );
```

Questa azione riprende l'istanza dell'attività nello stato che aveva prima di essere sospesa.

Analisi dei risultati di un'attività

Un'attività da fare (conosciuta anche come un'attività di *partecipazione* nell'API) o un'attività di collaborazione (conosciuta anche come *attività umana* nell'API) viene eseguita in modalità asincrona. Se un gestore di risposte viene specificato all'avvio

dell'attività, il messaggio di output viene restituito automaticamente una volta completata l'attività. Se non è specificato alcun gestore risposte, il messaggio deve essere recuperato esplicitamente.

About this task

I risultati dell'attività vengono memorizzati nel database solo se il modello di attività, da cui deriva l'istanza dell'attività, non specifica l'eliminazione automatica delle istanze di attività derivate.

Procedure

Analizzare i risultati dell'attività.

L'esempio illustra il modo in cui verificare il numero d'ordine di un'attività completata correttamente.

```
QueryResultSet result = task.query("DISTINCT TASK.TKIID",
                                   "TASK.NAME = 'CustomerOrder' AND
                                   TASK.STATE = TASK.STATE.STATE_FINISHED",
                                   (String)null, (Integer)null, (TimeZone)null);

if (result.size() > 0)
{
    result.first();
    TKIID tkiid = (TKIID) result.getOID(1);
    ClientObjectWrapper output = task.getOutputMessage(tkiid);
    DataObject myOutput = null;
    if ( output.getObject() != null && output.getObject() instanceof DataObject)
    {
        myOutput = (DataObject)output.getObject();
        int order = myOutput.getInt("OrderNo");
    }
}
```

Chiusura di un'istanza dell'attività

A volte chi dispone dei privilegi di amministratore deve terminare un'istanza dell'attività in stato irreversibile. Poiché l'istanza dell'attività viene terminata immediatamente, si consiglia di terminare un'istanza dell'attività solo in situazioni eccezionali.

Procedure

1. Richiamare l'istanza dell'attività da terminare.

```
Task taskInstance = task.getTask(tkiid);
```

2. Terminare l'istanza dell'attività.

```
TKIID tkiid = taskInstance.getID();
task.terminate(tkiid);
```

L'istanza dell'attività viene terminata immediatamente senza dover attendere alcuna attività in sospeso.

Eliminazione delle istanze di attività

Le istanze di attività vengono eliminate automaticamente solo quando sono completate e se questa opzione viene specificata nel modello di attività associato da cui derivano le istanze stesse. L'esempio di seguito riportato illustra il modo in cui eliminare tutte le istanze di attività che sono terminate e non sono state eliminate automaticamente.

Procedure

1. Elencare le istanze di attività terminate.

```

QueryResultSet result =
    task.query("DISTINCT TASK.TKIID",
              "TASK.STATE = TASK.STATE.STATE_FINISHED",
              (String)null, (Integer)null, (TimeZone)null);

```

Questa operazione restituisce una serie di risultati di query che elenca le istanze di attività terminate.

2. Eliminare le istanze di attività che sono terminate.

```

while (result.next() )
{
    TKIID tkiid = (TKIID) result.getOID(1);
    task.delete(tkiid);
}

```

Rilascio di un'attività richiesta

Quando un proprietario potenziale richiede un'attività, questa persona è responsabile del completamento dell'attività stessa. A volte, tuttavia, l'attività richiesta deve essere rilasciata, in modo che un altro proprietario potenziale possa richiederla.

About this task

Talvolta, chi dispone dei privilegi di amministratore deve rilasciare un'attività richiesta. Questa situazione può verificarsi, ad esempio, quando un'attività deve essere completata ma il proprietario di questa è assente. Il proprietario dell'attività può, inoltre, rilasciare un'attività richiesta.

Procedure

1. Elenca le attività richieste appartenenti ad una determinata persona, ad esempio, Smith.

```

QueryResultSet result =
    task.query("DISTINCT TASK.TKIID",
              "TASK.STATE = TASK.STATE.STATE_CLAIMED AND
              TASK.OWNER = 'Smith'",
              (String)null, (Integer)null, (TimeZone)null);

```

In questo modo viene restituita una serie di risultati della query che elenca le attività richieste dalla persona specificata, Smith.

2. Rilasciare l'attività richiesta.

```

if (result.size() > 0)
{
    result.first();
    TKIID tkiid = (TKIID) result.getOID(1);
    task.cancelClaim(tkiid, true);
}

```

In questo modo l'attività viene riportata allo stato pronto e può essere richiesta da uno degli altri potenziali proprietari. Qualsiasi dato di output o di errore impostato dal proprietario originale viene conservato.

Gestione degli elementi di lavoro

Durante il ciclo vita di un'istanza di attività, la serie di utenti associati all'oggetto può cambiare, ad esempio un utente è in vacanza, vengono assunti nuovi impiegati o il carico di lavoro deve essere distribuito in modo diverso. Per autorizzare le modifiche, è possibile sviluppare applicazioni per creare, eliminare o trasferire elementi di lavoro.

About this task

Un elemento di lavoro rappresenta l'assegnazione di un oggetto ad un utente o un gruppo di utenti per un motivo particolare. L'oggetto è solitamente un'istanza di operazione dell'attività umana, un'istanza di processo o un'istanza di attività. I motivi derivano dal ruolo che l'utente ha per l'oggetto. Un oggetto può avere elementi di lavoro multipli perché un utente può avere diversi ruoli in associazione con l'oggetto, e viene creato un elemento di lavoro per ciascuno di questi ruoli. Per esempio, un'istanza di attività da fare può avere un elemento di lavoro amministratore, lettore, editor, e proprietario contemporaneamente.

Le azioni che possono essere effettuate per gestire gli elementi di lavoro dipendono dal ruolo dell'utente, ad esempio, un amministratore può creare, eliminare e trasferire gli elementi di lavoro, ma il proprietario dell'attività può trasferire solo gli elementi di lavoro.

- Creare un elemento di lavoro.

```
// eseguire la query dell'istanza dell'attività per cui è necessario specificare
// è necessario specificare un amministratore
QueryResultSet result = task.query("TASK.TKIID",
                                   "TASK.NAME='CustomerOrder'",
                                   (String)null, (Integer)null,
                                   (TimeZone)null);

if ( result.size() > 0 )
{
    result.first();
    // creare l'elemento di lavoro
    task.createWorkItem((TKIID)(result.getOID(1)),
                       WorkItem.REASON_ADMINISTRATOR, "Smith");
}
```

Questa azione crea un elemento di lavoro per l'utente Smith che dispone del ruolo di amministratore.

- Cancellare un elemento di lavoro.

```
// eseguire la query dell'istanza dell'attività per cui è necessario cancellare
// un elemento di lavoro
QueryResultSet result = task.query("TASK.TKIID",
                                   "TASK.NAME='CustomerOrder'",
                                   (String)null, (Integer)null,
                                   (TimeZone)null);

if ( result.size() > 0 )
{
    result.first();
    // cancellare l'elemento di lavoro
    task.deleteWorkItem((TKIID)(result.getOID(1)),
                       WorkItem.REASON_READER, "Smith");
}
```

Questa azione elimina l'elemento di lavoro per l'utente Smith che dispone del ruolo di amministratore.

- Trasferire un elemento di lavoro.

```
// eseguire la query dell'attività da ripianificare
QueryResultSet result =
    task.query("DISTINCT TASK.TKIID",
              "TASK.NAME='CustomerOrder' AND
              TASK.STATE=TASK.STATE.STATE_READY AND
              WORK_ITEM.REASON=WORK_ITEM.REASON.REASON_POTENTIAL_OWNER AND
              WORK_ITEM.OWNER_ID='Miller'",
              (String)null, (Integer)null, (TimeZone)null);
if ( result.size() > 0 )
{
    result.first();
    // trasferire l'elemento di lavoro dall'utente Miller all'utente Smith
```

```

// in modo che Smith possa lavorare sull'attività
task.transferWorkItem((TKIID)(result.getOID(1)),
                      WorkItem.REASON_POTENTIAL_OWNER,"Miller","Smith");
}

```

In questo modo, si trasferisce l'elemento di lavoro all'utente Smith affinché possa lavorare su di esso.

Creazione dei modelli di attività e delle istanze di attività al runtime

In genere, utilizzare uno strumento di modellamento, come ad esempio WebSphere Integration Developer per creare i modelli di attività. Quindi, installare i modelli di attività in WebSphere Process Server e creare delle istanze da tali modelli, ad esempio utilizzando Business Process Choreographer Explorer. Tuttavia, è possibile creare anche istanze di attività umane o partecipanti o modelli al runtime.

About this task

Se si desidera effettuare tale operazione, ad esempio, quando la definizione dell'attività non è disponibile quando l'applicazione viene sviluppata, le attività che fanno parte di un flusso di lavoro non sono ancora note o è necessario che un'attività soddisfi alcune collaborazioni ad-hoc tra un gruppo di persone.

È possibile modellare attività da fare o di Collaborazione ad-hoc creando istanze della classe `com.ibm.task.api.TaskModel`, e utilizzandole per creare un modello di attività riutilizzabile o creare direttamente un'istanza di attività run-once. Per creare un'istanza della classe `TaskModel` è disponibile una serie di metodi di factory nella classe di factory `com.ibm.task.api.ClientTaskFactory`. Il modellamento di attività umane al runtime si basa sull'Eclipse Modeling Framework (EMF).

Procedure

1. Creare un `org.eclipse.emf.ecore.resource.ResourceSet` utilizzando il metodo `factory createResourceSet`.
2. Opzionale: Se si desidera usare tipi di messaggi complessi, è possibile definirli mediante `org.eclipse.xsd.XSDFactory` che si può ottenere usando il metodo `factory getXSDFactory()`, o importare direttamente uno schema XML esistente usando il metodo `factory loadXSDSchema`.
Per rendere accessibili i tipi complessi al WebSphere Process Server, distribuirli come parte di un'enterprise application.
3. Creare o importare una definizione Web Services Definition Language (WSDL) del tipo `javax.wsdl.Definition`.

È possibile creare una nuova definizione WSDL usando il metodo `createWSDLDefinition`. In seguito è possibile aggiungervi un tipo di porta e un'operazione. È anche possibile importare direttamente una definizione WSDL esistente usando il metodo `factory loadWSDLDefinition`.

4. Creare la definizione dell'attività usando il metodo `factory createTTask`.
Se si desidera aggiungere o manipolare più elementi di attività complesse, è possibile usare la classe `com.ibm.wbit.tel.TaskFactory` che si può richiamare tramite il metodo `factory getTaskFactory`.
5. Creare il modello di attività usando il metodo `factory createTaskModel` e passarlo al bundle risorse creato nel passaggio 1 e che riunisce tutte le risorse create nel frattempo.
6. Opzionale: Convalidare il modello usando il metodo `TaskModel convalida`.

Results

Utilizzare uno dei metodi API EJB di Human Task Manager crea che presentano un parametro **TaskModel** per creare un modello di attività riutilizzabile o un'istanza di attività run-once.

Creazione di attività di runtime che utilizzano tipi Java semplici:

In questo esempio viene creata un'attività di runtime che utilizza solo i tipi Java semplici nella relativa interfaccia, ad esempio, un oggetto String.

About this task

L'esempio viene eseguito solo all'interno del contesto dell'applicazione di business chiamante, per cui vengono caricate le risorse.

Procedure

1. Accedere a ClientTaskFactory, quindi creare una serie di risorse che contenga le definizioni del nuovo modello di attività.

```
ClientTaskFactory factory = ClientTaskFactory.newInstance();
ResourceSet resourceSet = factory.createResourceSet();
```

2. Creare la definizione WSDL, quindi aggiungere le descrizioni delle operazioni.

```
// creare l'interfaccia WSDL
Definition definition = factory.createWSDLDefinition
    ( resourceSet, new QName( "http://www.ibm.com/task/test/", "test" ) );
```

```
// creare un tipo di porta
PortType portType = factory.createPortType( definition, "doItPT" );
```

```
// creare un'operazione, i messaggi di input o di output sono di tipo String:
// un messaggio di errore non è specificato
Operation operation = factory.createOperation
    ( definition, portType, "doIt",
      new QName( "http://www.w3.org/2001/XMLSchema", "string" ),
      new QName( "http://www.w3.org/2001/XMLSchema", "string" ),
      (Map)null );
```

3. Creare il modello EMF della nuova attività umana.

Se si sta creando un'istanza di attività, non è richiesta una data Valido da (UTCDate).

```
TTask humanTask = factory.createTTask( resourceSet,
    TTaskKinds.HTASK_LITERAL,
    "TestTask",
    new UTCDate( "2005-01-01T00:00:00" ),
    "http://www.ibm.com/task/test/",
    portType,
    operation );
```

Questo passo inizializza le proprietà del modello di attività con i valori predefiniti.

4. Modificare le proprietà del modello di attività umana.

```
// utilizzare i metodi dal package com.ibm.wbit.tel, ad esempio,
humanTask.setBusinessRelevance( TBoolean, YES_LITERAL );
```

```
// richiamare la produzione attività per creare o modificare gli elementi
dell'attività compositi
TaskFactory taskFactory = factory.getTaskFactory();
```

```
// specificare le impostazioni di escalation
TVerb verb = taskFactory.createTVerb();
verb.setName("John");
```

```
// creare escalationReceiver, quindi aggiungere il verbo
```

```

TEscalationReceiver escalationReceiver =
    taskFactory.createTEscalationReceiver();
escalationReceiver.setVerb(verb);

```

```

// creare l'escalation, quindi aggiungere il destinatario dell'escalation
TEscalation escalation = taskFactory.createTEscalation();
escalation.setEscalationReceiver(escalationReceiver);

```

5. Creare il modello dell'attività che contiene tutte le definizioni della risorsa.

```
TaskModel taskModel = ClientTaskFactory.createTaskModel( resourceSet );
```

6. Convalidare il modello dell'attività, quindi correggere eventuali problemi di convalida che vengono trovati.

```
ValidationProblem[] validationProblems = taskModel.validate();
```

7. Creare l'istanza o il modello di attività di runtime.

Utilizzare l'interfaccia `HumanTaskManagerService` per creare l'istanza o il modello di attività. Poiché l'applicazione utilizza solo tipi Java semplici, non è necessario specificare un nome dell'applicazione.

- Il frammento di seguito riportato crea un'istanza dell'attività:

```
atask.createTask( taskModel, (String)null, "HTM" );
```

- Il frammento di seguito riportato crea un modello di attività:

```
task.createTaskTemplate( taskModel, (String)null );
```

Results

Se viene creata un'istanza di attività di runtime, ora può essere avviata. Se viene creato un modello di attività di runtime, ora è possibile creare istanze di attività dal modello.

Creazione di attività di runtime che utilizzano tipi complessi:

In questo esempio viene creata un'attività di runtime che utilizza tipi complessi nella relativa interfaccia. I tipi complessi sono già definiti, ovvero il file system locale sul client dispone di file XSD contenenti la descrizione dei tipi complessi.

About this task

L'esempio viene eseguito solo all'interno del contesto dell'applicazione di business chiamante, per cui vengono caricate le risorse.

Procedure

1. Accedere a `ClientTaskFactory`, quindi creare una serie di risorse che contenga le definizioni del nuovo modello di attività.

```
ClientTaskFactory factory = ClientTaskFactory.newInstance();
ResourceSet resourceSet = factory.createResourceSet();
```

2. Aggiungere le definizioni XSD dei tipi complessi alla serie di risorse in modo che siano disponibili quando si definiscono le operazioni.

I file si trovano nella posizione in cui viene eseguito il codice.

```
factory.loadXSDSchema( resourceSet, "InputBO.xsd" );
factory.loadXSDSchema( resourceSet, "OutputBO.xsd" );
```

3. Creare la definizione WSDL, quindi aggiungere le descrizioni delle operazioni.

```
// creare l'interfaccia WSDL
Definition definition = factory.createWSDLDefinition
    ( resourceSet, new QName( "http://www.ibm.com/task/test/", "test" ) );
```

```
// creare un tipo di porta
PortType portType = factory.createPortType( definition, "doItPT" );
```



```

// creare un'operazione, il messaggio di input è un InputBO e
// il messaggio di output è un OutputBO;
// un messaggio di errore non è specificato
Operation operation = factory.createOperation
    ( definition, portType, "doIt",
      new QName( "http://Input", "InputBO" ),
      new QName( "http://Output", "OutputBO" ),
      (Map)null );

```

4. Creare il modello EMF della nuova attività umana.

Se si sta creando un'istanza di attività, non è richiesta una data Valido da (UTCDate).

```

TTask humanTask = factory.createTTask( resourceSet,
                                       TTaskKinds.HTASK_LITERAL,
                                       "TestTask",
                                       new UTCDate( "2005-01-01T00:00:00" ),
                                       "http://www.ibm.com/task/test/",
                                       portType,
                                       operation );

```

Questo passo inizializza le proprietà del modello di attività con i valori predefiniti.

5. Modificare le proprietà del modello di attività umana.

```

// utilizzare i metodi dal package com.ibm.wbit.tel, ad esempio,
humanTask.setBusinessRelevance( TBoolean, YES_LITERAL );

// richiamare la produzione attività per creare o modificare gli elementi
dell'attività composti
TaskFactory taskFactory = factory.getTaskFactory();

// specificare le impostazioni di escalation
TVerb verb = taskFactory.createTVerb();
verb.setName("John");

// creare escalationReceiver, quindi aggiungere il verbo
TEscalationReceiver escalationReceiver =
    taskFactory.createTEscalationReceiver();
escalationReceiver.setVerb(verb);

// creare l'escalation, quindi aggiungere il destinatario dell'escalation
TEscalation escalation = taskFactory.createTEscalation();
escalation.setEscalationReceiver(escalationReceiver);

```

6. Creare il modello dell'attività che contiene tutte le definizioni della risorsa.

```

TaskModel taskModel = ClientTaskFactory.createTaskModel( resourceSet );

```

7. Convalidare il modello dell'attività, quindi correggere eventuali problemi di convalida che vengono trovati.

```

ValidationProblem[] validationProblems = taskModel.validate();

```

8. Creare l'istanza o il modello di attività di runtime.

Utilizzare l'interfaccia HumanTaskManagerService per creare l'istanza o il modello di attività. È necessario fornire un nome dell'applicazione contenente le definizioni del tipo di dati, in modo che sia possibile accedervi. Inoltre, l'applicazione deve contenere un'attività o un processo fittizio, in modo che l'applicazione sia caricata da Business Process Choreographer.

- Il frammento di seguito riportato crea un'istanza dell'attività:


```
task.createTask( taskModel, "BOapplication", "HTM" );
```
- Il frammento di seguito riportato crea un modello di attività:


```
task.createTaskTemplate( taskModel, "BOapplication" );
```

Results

Se viene creata un'istanza di attività di runtime, ora può essere avviata. Se viene creato un modello di attività di runtime, ora è possibile creare istanze di attività dal modello.

Creazione di attività di runtime che utilizzano un'interfaccia esistente:

In questo esempio viene creata un'attività di runtime che utilizza un'interfaccia che è già definita, ovvero il file system locale sul client dispone di un file che contiene la descrizione dell'interfaccia.

About this task

L'esempio viene eseguito solo all'interno del contesto dell'applicazione di business chiamante, per cui vengono caricate le risorse.

Procedure

1. Accedere a ClientTaskFactory, quindi creare una serie di risorse che contenga le definizioni del nuovo modello di attività.

```
ClientTaskFactory factory = ClientTaskFactory.newInstance();
ResourceSet resourceSet = factory.createResourceSet();
```

2. Accedere alla definizione WSDL e alle descrizioni delle operazioni.

La descrizione dell'interfaccia si trova nella posizione in cui viene eseguito il codice.

```
Definition definition = factory.loadWSDLDefinition(
    resourceSet, "interface.wsdl" );
PortType portType = definition.getPortType(
    new QName( definition.getTargetNamespace(), "doItPT" ) );
Operation operation = portType.getOperation(
    "doIt", (String)null, (String)null);
```

3. Creare il modello EMF della nuova attività umana.

Se si sta creando un'istanza di attività, non è richiesta una data Valido da (UTCDate).

```
TTask humanTask = factory.createTTask( resourceSet,
    TTaskKinds.HTASK_LITERAL,
    "TestTask",
    new UTCDate( "2005-01-01T00:00:00" ),
    "http://www.ibm.com/task/test/",
    portType,
    operation );
```

Questo passo inizializza le proprietà del modello di attività con i valori predefiniti.

4. Modificare le proprietà del modello di attività umana.

```
// utilizzare i metodi dal package com.ibm.wbit.tel, ad esempio,
humanTask.setBusinessRelevance( TBoolean, YES_LITERAL );
```

```
// richiamare la produzione attività per creare o modificare gli elementi
dell'attività compositi
```

```
TaskFactory taskFactory = factory.getTaskFactory();
```

```
// specificare le impostazioni di escalation
TVerb verb = taskFactory.createTVerb();
verb.setName("John");
```

```
// creare escalationReceiver, quindi aggiungere il verbo
TEscalationReceiver escalationReceiver =
    taskFactory.createTEscalationReceiver();
escalationReceiver.setVerb(verb);
```

- ```
// creare l'escalation, quindi aggiungere il destinatario dell'escalation
TEscalation escalation = taskFactory.createTEscalation();
escalation.setEscalationReceiver(escalationReceiver);
```
5. Creare il modello dell'attività che contiene tutte le definizioni della risorsa.

```
TaskModel taskModel = ClientTaskFactory.createTaskModel(resourceSet);
```
  6. Convalidare il modello dell'attività, quindi correggere eventuali problemi di convalida che vengono trovati.

```
ValidationProblem[] validationProblems = taskModel.validate();
```
  7. Creare l'istanza o il modello di attività di runtime.

Utilizzare l'interfaccia `HumanTaskManagerService` per creare l'istanza o il modello di attività. È necessario fornire un nome dell'applicazione contenente le definizioni del tipo di dati, in modo che sia possibile accedervi. Inoltre, l'applicazione deve contenere un'attività o un processo fittizio, in modo che l'applicazione sia caricata da `Business Process Choreographer`.

    - Il frammento di seguito riportato crea un'istanza dell'attività:

```
task.createTask(taskModel, "B0application", "HTM");
```
    - Il frammento di seguito riportato crea un modello di attività:

```
task.createTaskTemplate(taskModel, "B0application");
```

## Results

Se viene creata un'istanza di attività di runtime, ora può essere avviata. Se viene creato un modello di attività di runtime, ora è possibile creare istanze di attività dal modello.

## Creazione di attività di runtime che utilizzano un'interfaccia dall'applicazione chiamante:

In questo esempio viene creata un'attività di runtime che utilizza un'interfaccia che è parte dell'applicazione chiamante. Ad esempio, l'attività di runtime viene creata in un frammento Java di un processo di business ed utilizza un'interfaccia dell'applicazione del processo.

## About this task

L'esempio viene eseguito solo all'interno del contesto dell'applicazione di business chiamante, per cui vengono caricate le risorse.

## Procedure

1. Accedere a `ClientTaskFactory`, quindi creare una serie di risorse che contenga le definizioni del nuovo modello di attività.

```
ClientTaskFactory factory = ClientTaskFactory.newInstance();

// specificare il caricatore di classe del contesto, in modo che vengano trovate
// le seguenti risorse
ResourceSet resourceSet = factory.createResourceSet
 (Thread.currentThread().getContextClassLoader());
```
2. Accedere alla definizione WSDL e alle descrizioni delle operazioni.

Specificare il percorso contenente il file JAR del package.

```
Definition definition = factory.loadWSDLDefinition(resourceSet,
 "com/ibm/workflow/metaflow/interface.wsdl");
PortType portType = definition.getPortType(
 new QName(definition.getTargetNamespace(), "doItPT"));
Operation operation = portType.getOperation
 ("doIt", (String)null, (String)null);
```

3. Creare il modello EMF della nuova attività umana.

Se si sta creando un'istanza di attività, non è richiesta una data Valido da (UTCDate).

```
TTask humanTask = factory.createTTask(resourceSet,
 TTaskKinds.HTASK_LITERAL,
 "TestTask",
 new UTCDate("2005-01-01T00:00:00"),
 "http://www.ibm.com/task/test/",
 portType,
 operation);
```

Questo passo inizializza le proprietà del modello di attività con i valori predefiniti.

4. Modificare le proprietà del modello di attività umana.

```
// utilizzare i metodi dal package com.ibm.wbit.tel, ad esempio,
humanTask.setBusinessRelevance(TBoolean, YES_LITERAL);

// richiamare la produzione attività per creare o modificare gli elementi
dell'attività compositi
TaskFactory taskFactory = factory.getTaskFactory();

// specificare le impostazioni di escalation
TVerb verb = taskFactory.createTVerb();
verb.setName("John");

// creare escalationReceiver, quindi aggiungere il verbo
TEscalationReceiver escalationReceiver =
 taskFactory.createTEscalationReceiver();
escalationReceiver.setVerb(verb);

// creare l'escalation, quindi aggiungere il destinatario dell'escalation
TEscalation escalation = taskFactory.createTEscalation();
escalation.setEscalationReceiver(escalationReceiver);
```

5. Creare il modello dell'attività che contiene tutte le definizioni della risorsa.

```
TaskModel taskModel = ClientTaskFactory.createTaskModel(resourceSet);
```

6. Convalidare il modello dell'attività, quindi correggere eventuali problemi di convalida che vengono trovati.

```
ValidationProblem[] validationProblems = taskModel.validate();
```

7. Creare l'istanza o il modello di attività di runtime.

Utilizzare l'interfaccia HumanTaskManagerService per creare l'istanza o il modello di attività. È necessario fornire un nome dell'applicazione contenente le definizioni del tipo di dati, in modo che sia possibile accedervi.

- Il frammento di seguito riportato crea un'istanza dell'attività:  
task.createTask( taskModel, "WorkflowApplication", "HTM" );
- Il frammento di seguito riportato crea un modello di attività:  
task.createTaskTemplate( taskModel, "WorkflowApplication" );

## Results

Se viene creata un'istanza di attività di runtime, ora può essere avviata. Se viene creato un modello di attività di runtime, ora è possibile creare istanze di attività dal modello.

## Interfaccia HumanTaskManagerService

L'interfaccia HumanTaskManagerService dispone delle funzioni relative all'attività che possono essere richiamate da un client locale o remoto.

I metodi che possono essere richiamati dipendono dallo stato dell'attività e l'autorizzazione della persona che utilizza l'applicazione contenente il metodo. I metodi principali per la modifica degli oggetti dell'attività sono elencati di seguito. Per ulteriori informazioni su tali metodi ed altri metodi che sono disponibili nell'interfaccia HumanTaskManagerService, consultare Javadoc nel package com.ibm.task.api.

## Modelli di attività

I metodi di seguito riportati sono disponibili per effettuare operazioni con questi modelli di attività.

Tabella 29. I metodi API per i modelli di attività

| Metodo             | Descrizione                                                                                                                                    |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| getTaskTemplate    | Richiama il modello di attività specificata.                                                                                                   |
| createAndCallTask  | Crea ed esegue un'istanza di attività dal modello di attività specificato ed attende il risultato in modo sincrono.                            |
| createAndStartTask | Crea e avvia un'istanza dell'attività dal modello di attività specificato.                                                                     |
| createTask         | Crea un'istanza dell'attività dal modello di attività specificato.                                                                             |
| createInputMessage | Crea un messaggio di input per il modello di attività specificato. Ad esempio, un messaggio che può essere utilizzato per avviare un'attività. |
| queryTaskTemplates | Richiama i modelli di attività memorizzati nel database.                                                                                       |

## Istanze dell'attività

I metodi di seguito riportati sono disponibili per effettuare operazioni con queste istanze dell'attività.

Tabella 30. I metodi API per le istanze dell'attività

| Metodo    | Descrizione                                                                                                                                      |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| getTask   | Richiama un'istanza di attività che può essere in qualunque stato.                                                                               |
| callTask  | Avvia un'attività di chiamata in modalità sincrona.                                                                                              |
| startTask | Avvia un'attività che è già stata creata.                                                                                                        |
| suspend   | Sospende l'attività di collaborazione o da fare.                                                                                                 |
| resume    | Riprende l'attività di collaborazione o da fare.                                                                                                 |
| terminate | Termina l'istanza di attività specificata. Se viene terminata un'attività di chiamata, questa azione non ha alcun impatto sul servizio invocato. |
| delete    | Elimina l'istanza dell'attività specificata.                                                                                                     |
| claim     | Richiede l'attività per l'elaborazione.                                                                                                          |

Tabella 30. I metodi API per le istanze dell'attività (Continua)

| Metodo           | Descrizione                                                                                        |
|------------------|----------------------------------------------------------------------------------------------------|
| update           | Aggiorna l'istanza dell'attività.                                                                  |
| complete         | Completa l'istanza dell'attività.                                                                  |
| cancelClaim      | Rilascia un'istanza di attività richiesta, in modo che il potenziale proprietario possa lavorarci. |
| createWorkItem   | Crea un elemento di lavoro per l'istanza dell'attività.                                            |
| transferWorkItem | Trasferisce l'elemento di lavoro ad un proprietario specificato.                                   |
| deleteWorkItem   | Elimina l'elemento di lavoro.                                                                      |

## Escalation

I metodi di seguito riportati sono disponibili per effettuare operazioni con le escalation.

Tabella 31. Metodi API per effettuare operazioni con le escalation.

| Metodo        | Descrizione                                   |
|---------------|-----------------------------------------------|
| getEscalation | Richiama l'istanza di escalation specificata. |

## Proprietà personalizzate

Le attività, i modelli di attività e le escalation possono avere tutte proprietà personalizzate. L'interfaccia fornisce i metodi get e set per richiamare ed impostare i valori per le proprietà personalizzate. Inoltre è possibile associare le proprietà denominate e richiamarle dalle istanze dell'attività. I nomi delle proprietà personalizzate ed i valori devono essere del tipo java.lang.String. I metodi che seguono sono validi per le attività, i modelli di attività e le escalation.

Tabella 32. Metodi API per le variabili e le proprietà personalizzate

| Metodo                 | Descrizione                                                                                   |
|------------------------|-----------------------------------------------------------------------------------------------|
| getCustomProperty      | Richiama la proprietà personalizzata denominata dell'istanza dell'attività specificata.       |
| getCustomProperties    | Richiama le proprietà personalizzate dell'istanza dell'attività specificata.                  |
| getCustomPropertyNames | Richiama i nomi delle proprietà personalizzate per l'istanza dell'attività.                   |
| setCustomProperty      | Memorizza i valori specifici della personalizzazione per l'istanza dell'attività specificata. |

## Azioni consentite per le attività:

Le azioni che possono essere effettuate su un'attività dipendono dall'attività, se è un'attività da fare, un'attività di collaborazione, un'attività di chiamata o un'attività di gestione.

Non è possibile usare tutte le azioni fornite dall'interfaccia HumanTaskManager per tutti i tipi di attività. La tabella di seguito riportata illustra le azioni che è possibile effettuare su ciascun tipo di attività.

| Azione                                | Tipo di attività |                            |                      |                      |
|---------------------------------------|------------------|----------------------------|----------------------|----------------------|
|                                       | Attività da fare | Attività di collaborazione | Attività di chiamata | Attività di gestione |
| callTask                              |                  |                            | X                    |                      |
| cancelClaim                           | X                | X <sup>1</sup>             |                      |                      |
| richiesta                             | X                | X <sup>1</sup>             |                      |                      |
| completa                              | X                | X <sup>1</sup>             |                      | X                    |
| completeWithFollowOnTask <sup>4</sup> | X                | X <sup>1</sup>             |                      |                      |
| completeWithFollowOnTask <sup>5</sup> |                  | X <sup>3</sup>             | X <sup>3</sup>       |                      |
| createFaultMessage                    | X                | X                          | X                    | X                    |
| createInputMessage                    | X                | X                          | X                    | X                    |
| createOutputMessage                   | X                | X                          | X                    | X                    |
| createWorkItem                        | X                | X <sup>1</sup>             | X                    | X                    |
| elimina                               | X <sup>1</sup>   | X <sup>1</sup>             | X                    | X <sup>1</sup>       |
| deleteWorkItem                        | X                | X <sup>1</sup>             | X                    | X                    |
| getCustomProperty                     | X                | X <sup>1</sup>             | X                    | X                    |
| getDocumentation                      | X                | X <sup>1</sup>             | X                    | X                    |
| getFaultNames                         | X                | X <sup>1</sup>             |                      |                      |
| getFaultMessage                       | X                | X <sup>1</sup>             | X                    |                      |
| getInputMessage                       | X                | X <sup>1</sup>             | X                    |                      |
| getOutputMessage                      | X                | X <sup>1</sup>             | X                    |                      |
| getUsersInRole                        | X                | X <sup>1</sup>             | X                    | X                    |
| getTask                               | X                | X <sup>1</sup>             | X                    | X                    |
| getUISettings                         | X                | X <sup>1</sup>             | X                    | X                    |
| riprendi                              | X                | X <sup>1</sup>             |                      |                      |
| setCustomProperty                     | X                | X <sup>1</sup>             | X                    | X                    |
| setFaultMessage                       | X                | X <sup>1</sup>             |                      |                      |
| setOutputMessage                      | X                | X <sup>1</sup>             |                      |                      |
| startTask                             | X <sup>1</sup>   | X <sup>1</sup>             | X                    | X                    |
| startTaskAsSubtask <sup>6</sup>       | X                | X <sup>1</sup>             |                      |                      |
| startTaskAsSubtask <sup>7</sup>       |                  | X <sup>3</sup>             | X <sup>3</sup>       |                      |
| sospendi                              | X                | X <sup>1</sup>             |                      |                      |
| suspendWithCancelClaim                | X                | X <sup>1</sup>             |                      |                      |
| termina                               | X <sup>1</sup>   | X <sup>1</sup>             | X <sup>1</sup>       |                      |
| transferWorkItem                      | X                | X <sup>1</sup>             | X                    | X                    |
| aggiorna                              | X                | X <sup>1</sup>             | X                    | X                    |

| Azione                                                                                          | Tipo di attività |                            |                      |                      |
|-------------------------------------------------------------------------------------------------|------------------|----------------------------|----------------------|----------------------|
|                                                                                                 | Attività da fare | Attività di collaborazione | Attività di chiamata | Attività di gestione |
| <b>Note:</b>                                                                                    |                  |                            |                      |                      |
| 1. Solo per le attività autonome, le attività ad-hoc e i modelli di attività                    |                  |                            |                      |                      |
| 2. Per le attività autonome, le attività in linea nei processi di business e le attività ad-hoc |                  |                            |                      |                      |
| 3. Solo per le attività autonome e per le attività ad-hoc                                       |                  |                            |                      |                      |
| 4. I tipi di attività che possono avere attività follow-on                                      |                  |                            |                      |                      |
| 5. I tipi di attività che possono essere utilizzati come attività follow-on                     |                  |                            |                      |                      |
| 6. I tipi di attività che possono avere attività secondarie                                     |                  |                            |                      |                      |
| 7. I tipi di attività che possono essere utilizzati come attività secondarie                    |                  |                            |                      |                      |

## Sviluppo di applicazioni per processi di business e attività umane

Nella maggior parte degli scenari di processi di business sono implicate persone. Per esempio, un processo di business richiede l'interazione di persone nell'avvio o nell'amministrazione del processo o durante l'esecuzione delle operazioni dell'attività umana. Per supportare tali scenari, è necessario usare sia la API di Business Flow Manager che la API di Human Task Manager.

### About this task

Per coinvolgere persone negli scenari di processi di business, è possibile includere i seguenti tipi di attività nel processo di business:

- Un'attività di chiamata in linea (conosciuta anche come *attività di origine* nella API).  
È possibile fornire un'attività di chiamata per ciascuna attività di ricezione, per ciascun elemento onMessage di un'attività pick, e per ciascun elemento onEvent di un gestore eventi. Questa attività controlla quindi chi è autorizzato ad avviare un processo o comunicare con un'istanza del processo in esecuzione.
- Un'attività di amministrazione.  
È possibile fornire un'attività di amministrazione per specificare chi è autorizzato ad amministrare il processo o eseguire operazioni amministrative nelle attività non riuscite del processo.
- Un'attività da fare (conosciuta anche come *attività di partecipazione* nella API).  
Un'attività da fare effettua un'operazione dell'attività umana. Questo tipo di attività consente di coinvolgere persone nel processo.

Le operazioni dell'attività umana nel processo di business rappresentano le attività da fare che la persona esegue nello scenario del processo di business. È possibile usare sia la API di Business Flow Manager che la API di Human Task Manager per realizzare questi scenario:

- Il processo di business è il contenitore di tutte le attività che appartengono al processo, comprese le operazioni dell'attività umana rappresentate dalle attività da fare. Quando viene creata un'istanza del processo, viene assegnato un unico ID oggetto (PIID).
- Quando viene attivata un'operazione dell'attività umana durante l'esecuzione di un'istanza del processo, viene creata un'istanza dell'attività, identificata dal rispettivo ID oggetto (AIID) unico. Contemporaneamente, viene creata anche



un'istanza dell'attività da fare in linea identificata dal rispettivo ID oggetto (TKIID). Il rapporto dell'operazione dell'attività umana con l'istanza dell'attività è ottenuto tramite gli ID oggetto:

- L'ID dell'attività da fare dell'istanza dell'attività è impostato sul TKIID dell'attività da fare associata.
  - L'ID del contesto di contenimento dell'istanza dell'attività è impostato sul PIID dell'istanza del processo che contiene l'istanza dell'attività associata.
  - L'ID del contesto principale dell'istanza dell'attività è impostato sul AIID dell'istanza dell'attività associata.
- I cicli di vita di tutte le istanze di attività da fare in linea sono gestite dall'istanza del processo. Quando viene eliminata l'istanza del processo, vengono eliminate anche le istanze dell'attività. In altre parole, tutte le attività con l'ID del contesto di contenimento impostato sul PIID dell'istanza di processo vengono automaticamente eliminate.

### **Determinazione dei modelli o attività di processo che possono essere avviati**

Un processo di business può essere avviato chiamando i metodi chiama, avvia, o sendMessage della API di Business Flow Manager. Se il processo ha una sola attività di avvio, è possibile usare la firma del metodo che richiede il nome del modello di processo come parametro. Se il processo ha più di un'attività di avvio, è necessario identificare esplicitamente l'attività di avvio.

#### **About this task**

Quando è modellato un processo di business, il modellatore può decidere che solo un sottoinsieme di utenti può creare una versione del processo dal modello di processo. Questo avviene associando un'attività di chiamata in linea a un'attività di avvio del processo e specificando le limitazioni di autorizzazione per quell'attività. Solo a coloro che sono potenziali iniziatori o amministratori dell'attività è consentito creare una versione dell'attività, e quindi una versione del modello di processo.

Se un'attività di chiamata in linea non è associata all'attività di avvio, o se non vengono specificate le limitazioni di autorizzazione per l'attività, chiunque può creare una versione del processo utilizzando l'attività di avvio.

Un processo può avere più di un'attività di avvio, ognuna con diverse query persone per iniziatori o amministratori potenziali. Ciò significa che un utente può essere autorizzato ad avviare un processo utilizzando l'attività A ma non l'attività B.

#### **Procedure**

1. Utilizzare la API di Business Flow Manager per creare un elenco delle versioni correnti dei modelli di processo che si trovano nello stato avviato.

**Suggerimento:** Il metodo queryProcessTemplates esclude solo quei modelli di processo che fanno parte di applicazioni non ancora avviate. Così, se si utilizza questo metodo senza filtrare i risultati, il metodo restituisce tutte le versioni dei modelli di processo a prescindere dal loro stato.

```
// current timestamp in UTC format, converted to yyyy-mm-ddThh:mm:ss
String now = (new UTCDate()).toXsdString();
String whereClause = "PROCESS_TEMPLATE.STATE =
 PROCESS_TEMPLATE.STATE.STATE_STARTED AND
 PROCESS_TEMPLATE.VALID_FROM =
```

```
(SELECT MAX(VALID_FROM) FROM PROCESS_TEMPLATE
WHERE NAME=PROCESS_TEMPLATE.NAME AND
VALID_FROM <= TS('" + now + "')");
```

```
ProcessTemplateData[] processTemplates = process.queryProcessTemplates
(whereClause,
 "PROCESS_TEMPLATE.NAME",
 (Integer)null, (TimeZone)null);
```

I risultati vengono ordinati per nome del modello di processo.

2. Creare l'elenco dei modelli di processo e l'elenco delle attività di avvio per cui l'utente è autorizzato.

L'elenco dei modelli di processo contiene quei modelli di processo che hanno un'unica attività di avvio. Tali attività o non sono protette o l'utente che ha effettuato l'accesso è autorizzato ad avviarle. Alternativamente, è possibile raccogliere i modelli di processo che possono essere avviati da almeno una delle attività di avvio.

**Suggerimento:** Anche un amministratore di sistema può avviare una versione di processo. Per ottenere un elenco completo dei modelli, è inoltre necessario leggere il modello dell'attività di amministrazione associato al modello di processo, e controllare se l'utente che ha effettuato l'accesso è un amministratore.

```
List authorizedProcessTemplates = new ArrayList();
List authorizedActivityServiceTemplates = new ArrayList();
```

3. Determinare le attività di avvio per ciascuno dei modelli di processo.

```
for(int i=0; i<processTemplates.length; i++)
{
 ProcessTemplateData template = processTemplates[i];
 ActivityServiceTemplateData[] startActivities =
 process.getStartActivities(template.getID());
```

4. Per ciascuna attività di avvio, richiamare l'ID del modello dell'attività di chiamata in linea associata.

```
for(int j=0; j<startActivities.length; j++)
{
 ActivityServiceTemplateData activity = startActivities[j];
 TKTID tktid = activity.getTaskTemplateID();
```

- a. Se il modello di un'attività di chiamata non esiste, il modello di processo non è protetto da questa attività di avvio.

In tal caso, chiunque può creare una versione di processo utilizzando questa attività di avvio.

```
boolean isAuthorized = false;
 if (tktid == null)
 {
 isAuthorized = true;
 authorizedActivityServiceTemplates.add(activity);
 }
```

- b. Se il modello dell'attività di chiamata esiste, utilizzare la API di Human Task Manager per controllare l'autorizzazione per l'utente che ha effettuato l'accesso.

Nell'esempio, l'utente che ha effettuato l'accesso è Smith. Questi deve essere un potenziale iniziatore dell'attività di chiamata o un amministratore.

```
if (tktid != null)
{
 isAuthorized =
 task.isUserInRole
 (tkid, "Smith", WorkItem.REASON_POTENTIAL_STARTER) ||
 task.isUserInRole(tktid, "Smith", WorkItem.REASON_ADMINISTRATOR);
```

```

 if (isAuthorized)
 {
 authorizedActivityServiceTemplates.add(activity);
 }
 }

```

Se l'utente ha il ruolo specificato, o se non vengono specificati i criteri di assegnazione persone per il ruolo, il metodo `isUserInRole` restituisce il valore `true`.

- Controllare se il processo può essere avviato utilizzando solo il nome `process` del modello di processo.

```

 if (isAuthorized && startActivities.length == 1)
 {
 authorizedProcessTemplates.add(template);
 }

```

- Terminare i loop.

```

 } // fine del loop per ciascun modello di servizio attività
} // fine del loop per ciascun modello di processo

```

## Elaborazione del flusso di lavoro di una sola persona che include attività umana

Alcuni flussi di lavoro vengono eseguiti da una sola persona, ad esempio, ordinare libri da una libreria online. Questo esempio mostra come effettuare la sequenza di azioni per ordinare il libro come una serie di operazioni dell'attività umana (attività da fare). Sia il Business Flow Manager che le API di Human Task Manager sono utilizzate per elaborare il flusso di lavoro.

### About this task

In una libreria online, per ordinare un libro l'acquirente completa una sequenza di azioni. Questa sequenza di azioni può essere effettuata come una serie di operazioni dell'attività umana (attività da fare). Se l'acquirente decide di acquistare più libri, questa attività equivale a richiedere l'operazione dell'attività umana successiva. Informazioni relative alla sequenza di attività sono mantenute da Business Flow Manager, mentre le attività stesse sono mantenute da Human Task Manager.

Confrontare questo esempio con l'esempio che utilizza solo la API di Business Flow Manager.

### Procedure

- Utilizzare la API di Business Flow Manager per ottenere la versione del processo su cui si desidera lavorare.

In questo esempio, una versione del processo `CustomerOrder`.

```

ProcessInstanceData processInstance =
 process.getProcessInstance("CustomerOrder");
String piid = processInstance.getID().toString();

```

- Utilizzare la API di Human Task Manager per effettuare una query sulle attività da fare pronte (`kind participating`) che fanno parte della versione del processo specificata.

Utilizzare l'ID del contesto di contenimento dell'attività per specificare la versione del processo contenente. Per il flusso di lavoro di una singola persona, la query restituisce le attività da fare associate alla prima operazione dell'attività umana nella sequenza di operazioni dell'attività umana.

```

//
// Eseguire una query dell'elenco di attività da fare che possono essere richieste all'utente co
// per la versione del processo specificata
//
QueryResultSet result =
 task.query("DISTINCT TASK.TKIID",
 "TASK.CONTAINMENT_CTX_ID = ID('" + piid + "') AND
 TASK.STATE = TASK.STATE.STATE_READY AND
 TASK.KIND = TASK.KIND.KIND_PARTICIPATING AND
 WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
 (String)null, (Integer)null, (TimeZone)null);

```

3. Richiedere l'attività da fare che viene restituita.

```

if (result.size() > 0)
{
 result.first();
 TKIID tkiid = (TKIID) result.getOID(1);
 ClientObjectWrapper input = task.claim(tkiid);
 DataObject activityInput = null ;
 if (input.getObject() != null && input.getObject() instanceof DataObject)
 {
 taskInput = (DataObject)input.getObject();
 // leggere i valori
 ...
 }
}

```

Una volta richiesta l'attività, viene restituito il messaggio di input dell'attività stessa.

4. Determinare l'operazione dell'attività umana associata con l'attività da fare.

È possibile utilizzare uno dei seguenti metodi per correlare le attività alle relative task.

- Il metodo `task.getActivityID`:

```

AIID aiid = task.getActivityID(tkiid);

```
- L'ID del contesto principale che fa parte dell'oggetto attività:

```

AIID aiid = null;
Task taskInstance = task.getTask(tkiid);

OID oid = taskInstance.getParentContextID();
if (oid != null and oid instanceof AIID)
{
 aiid = (AIID)oid;
}

```

5. Una volta terminato di lavorare sull'attività, utilizzare la API di Business Flow Manager per completare l'attività e l'operazione dell'attività umana associata, e richiedere la successiva operazione dell'attività umana nella versione del processo.

Per completare l'operazione dell'attività umana, viene creato un messaggio output. Quando si crea il messaggio di output, è necessario specificare il nome del tipo di messaggio, in modo tale che contenga la definizione del messaggio.

```

ActivityInstanceData activity = process.getActivityInstance(aiid);
ClientObjectWrapper output =
 process.createMessage(aiid, activity.getOutputMessageType());
DataObject myMessage = null ;
if (output.getObject() != null && output.getObject() instanceof DataObject)
{
 myMessage = (DataObject)output.getObject();
 //impostare le parti nel messaggio, ad esempio, un numero di ordine
 myMessage.setInt("OrderNo", 4711);
}

```

```
//completare l'operazione dell'attività umana e l'attività da fare associata,
// e richiedere la successiva operazione dell'attività umana
CompleteAndClaimSuccessorResult successor =
 process.completeAndClaimSuccessor(aiid, output);
```

Questa sezione imposta un messaggio di output che contiene il numero dell'ordine e richiede la successiva operazione dell'attività umana nella sequenza. Se AutoClaim è impostato per attività in successione e se esistono più percorsi da seguire, vengono richiamate tutte le attività in successione e un'attività casuale viene restituita come attività successiva. Se non esistono altre attività in successione che possono essere assegnate a questo utente, viene restituito il valore Null.

Se il processo contiene percorsi paralleli che possono essere seguiti e questi percorsi contengono attività umane per cui l'utente collegato è un potenziale proprietario di più di una di queste attività, un'attività casuale viene richiesta automaticamente e restituita come attività successiva.

6. Lavorare sulla successiva operazione dell'attività umana.

```
ClientObjectWrapper nextInput = successor.getInputMessage();
if (nextInput.getObject() !=
 null && nextInput.getObject() instanceof DataObject)
{
 activityInput = (DataObject)input.getObject();
 // leggere i valori
 ...
}

aiid = successor.getAIID();
```

7. Continuare con il passaggio 5 per completare l'operazione dell'attività umana e richiamare la successiva operazione dell'attività umana.

#### Attività correlate

“Elaborazione del flusso di lavoro di una sola persona” a pagina 74

Alcuni flussi di lavoro vengono eseguiti da una sola persona, ad esempio, ordinare libri da una libreria online. Questo tipo di flusso di lavoro non ha percorsi paralleli. L'API completeAndClaimSuccessor supporta l'elaborazione di questo tipo di flusso di lavoro.

## Gestione delle eccezioni e degli errori

Un processo BPEL può rilevare un errore in diversi punti del processo.

#### About this task

Gli errori di BPEL (Business Process Execution Language) sono originati da:

- Chiamate del servizio Web (errori WSDL (Web Services Description Language))
- Esecuzione di attività
- Errori BPEL standard riconosciuti da Business Process Choreographer

Esistono dei meccanismi per gestire questi errori. Utilizzare uno dei seguenti meccanismi per gestire gli errori generati da un'istanza del processo.

- Inoltro del controllo ai gestori degli errori corrispondenti
- Compensazione del lavoro precedente nel processo
- Arresto del processo per consentire di risolvere il problema (force-retry, force-complete)

Un processo BPEL può inoltre restituire errori a un chiamante di un'operazione fornita dal processo. È possibile modellare l'errore nel processo come un'attività di replica con un nome errore e dati di errore. Questi errori vengono restituiti al chiamante API come eccezioni verificate.

Se un processo BPEL non gestisce un errore BPEL o se si verifica un'eccezione API, un'eccezione di runtime viene restituita al chiamante API. Un'eccezione API si verifica ad esempio quando il modello di processo, da cui deve essere creata un'istanza, non esiste.

La gestione degli errori e delle eccezioni viene descritta nelle seguenti attività.

## Gestione delle eccezioni API

### About this task

Se un metodo nell'interfaccia `BusinessFlowManagerService` o `HumanTaskManagerService` non viene completato correttamente, viene restituita un'eccezione che denota la causa dell'errore. È possibile gestire questa eccezione specificatamente per fornire una guida al chiamante.

Tuttavia, è pratica comune gestire solo un sottogruppo di eccezioni e fornire una guida generale per le altre potenziali eccezioni. Tutte le eccezioni specifiche ereditano da una `ProcessException` o `TaskException` generica. La *procedura ottimale* è di rilevare le eccezioni generiche con un'istruzione finale `catch(ProcessException)` o `catch(TaskException)`. Questa istruzione consente di verificare la compatibilità del programma dell'applicazione in quanto tiene conto delle altre eccezioni che possono verificarsi.

## Verifica dell'errore impostato per un'attività

### Procedure

1. Elencare le attività di operazioni che si trovano in uno stato di arresto o di errore.

```
QueryResultSet result =
 process.query("ACTIVITY.AIID",
 "(ACTIVITY.STATE = ACTIVITY.STATE.STATE_FAILED OR
 ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED) AND
 ACTIVITY.KIND=ACTIVITY.KIND.KIND_STAFF",
 (String)null, (Integer)null, (TimeZone)null);
```

Questa operazione consente di restituire un insieme di risultati della query che contiene le attività in errore o in arresto.

2. Leggere il nome dell'errore.

```
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 ClientObjectWrapper faultMessage = process.getFaultMessage(aiid);
 DataObject fault = null ;
 if (faultMessage.getObject() != null && faultMessage.getObject()
 instanceof DataObject)
 {
 fault = (DataObject) faultMessage.getObject();
 Type type = fault.getType();
 String name = type.getName();
 String uri = type.getURI();
 }
}
```

Restituisce il nome dell'errore. Inoltre, è possibile analizzare l'eccezione non gestita per un'attività arrestata invece di richiamare il nome dell'errore.

## Verifica dell'errore verificatosi per un'attività

### About this task

Se un'attività causa un errore, il tipo di errore determina le operazioni che è possibile effettuare per ripristinare l'attività.

### Procedure

1. Elencare le operazioni dell'attività umana che si trovano in uno stato di arresto.

```
QueryResultSet result =
 process.query("ACTIVITY.AIID",
 "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND
 ACTIVITY.KIND=ACTIVITY.KIND.KIND_INVOKE",
 (String)null, (Integer)null, (TimeZone)null);
```

Questa operazione consente di restituire una serie di risultati della query che contiene le attività di chiamata.

2. Leggere il nome dell'errore.

```
if (result.size() > 0)
{
 result.first();
 AIID aaid = (AIID) result.getOID(1);
 ActivityInstanceData activity = process.getActivityInstance(aaid);

 ProcessException excp = activity.getUnhandledException();
 if (excp instanceof ApplicationFaultException)
 {
 ApplicationFaultException fault = (ApplicationFaultException)excp;
 String faultName = fault.getFaultName();
 }
}
```

---

## Sviluppo di applicazioni client per le API dei servizi Web

È possibile sviluppare applicazioni client che accedono ad applicazioni di processi di business e applicazioni di attività umane mediante le API dei servizi Web.

### About this task

Le applicazioni client possono essere sviluppate in un qualsiasi ambiente client dei servizi Web, inclusi i servizi Web Java e Microsoft .NET.

## Introduzione: servizi Web

I servizi Web sono applicazioni enterprise basate sul Web che utilizzano standard aperti basati su XML e protocolli di trasporto per lo scambio di dati con le applicazioni client. I servizi Web consentono l'utilizzo di un modello di programmazione neutrale per la lingua e l'ambiente.

I servizi Web utilizzano le seguenti tecnologie principali:

- XML (Extensible Markup Language). XML risolve il problema dell'indipendenza dei dati. Viene utilizzato per descrivere i dati e per associare i dati all'interno e all'esterno di qualsiasi applicazione o linguaggio di programmazione
- WSDL (Web Services Description Language). Questo linguaggio basato su XML viene utilizzato per creare una descrizione di un'applicazione sottostante. Questa

descrizione converte un'applicazione in un servizio Web, agendo come interfaccia tra l'applicazione sottostante e altre applicazioni abilitate al Web.

- SOAP (Simple Object Access Protocol). SOAP è il protocollo di comunicazione principale per il Web e molti servizi Web utilizzano questo protocollo per comunicare tra loro.

## Componenti del servizio Web e sequenza di controllo

Diversi componenti lato client e lato server fanno parte della sequenza di controllo che rappresenta una richiesta e una risposta di un servizio Web.

Di seguito viene riportata una sequenza di controllo tipica:

1. Sul lato client:
  - a. Un'applicazione client (fornita dall'utente) esegue una richiesta per un servizio Web.
  - b. Un client proxy (fornito dall'utente, ma che può essere generato automaticamente mediante utilità lato client) inserisce la richiesta di servizio in una busta di richiesta SOAP.
  - c. L'infrastruttura di sviluppo lato client inoltra la richiesta a un URL definito come endpoint del servizio Web.
2. La rete trasmette la richiesta all'endpoint del servizio Web mediante HTTP o HTTPS.
3. Sul lato server:
  - a. L'API generica dei servizi Web riceve e decodifica la richiesta.
  - b. La richiesta viene gestita direttamente dal componente generico di Business Flow Manager o Human Task Manager o inoltrata al processo di business o all'attività umana specificati.
  - c. I dati restituiti vengono inseriti nella busta di risposta SOAP.
4. La rete trasmette la risposta all'ambiente lato client mediante HTTP o HTTPS.
5. Sul lato client:
  - a. L'infrastruttura di sviluppo lato client apre la busta di risposta SOAP.
  - b. Il client proxy estrae i dati dalla risposta SOAP e li trasferisce all'applicazione client.
  - c. L'applicazione client elabora i dati restituiti come necessario.

## Panoramica delle API dei servizi Web

Le API dei servizi Web consentono di sviluppare applicazioni client che utilizzano servizi Web per accedere a processi di business e attività umane in esecuzione nell'ambiente di Business Process Choreographer.

Le API dei servizi Web di Business Process Choreographer forniscono due interfacce diverse (tipi di porta WSDL):

- l'API di Business Flow Manager. Consente alle applicazioni client di interagire con i microflussi e i processi di lunga esecuzione, ad esempio:
  - Creare modelli di processo e istanze di processo
  - Richiedere processi esistenti
  - Eseguire query di un processo mediante il relativo ID

Per un elenco completo delle azioni possibili, fare riferimento a "Sviluppo delle applicazioni per i processi di business" a pagina 63.

- L'API di Human Task Manager. Consente alle applicazioni client di:



- Creare e avviare attività
- Richiedere attività esistenti
- Completare attività
- Eseguire una query di un'attività mediante il relativo ID
- Eseguire una query di una raccolta di attività.

Per un elenco completo delle azioni possibili, fare riferimento a "Sviluppo di applicazioni per attività umane" a pagina 83.

Le applicazioni client possono utilizzare una o entrambe le interfacce dei servizi Web.

## Esempio

Di seguito viene riportato uno schema possibile per un'applicazione client che accede alle API dei servizi Web di Human Task Manager per elaborare un'attività umana partecipante:

1. L'applicazione client emette una chiamata del servizio Web query al WebSphere Process Server richiedendo un elenco di attività partecipanti che l'utente deve eseguire.
2. L'elenco delle attività partecipanti viene restituito in una busta di risposta SOAP/HTTP.
3. L'applicazione client quindi emette una chiamata del servizio claim Web per richiedere una delle attività partecipanti.
4. WebSphere Process Server restituisce il messaggio di input dell'attività.
5. L'applicazione client emette una chiamata del servizio Web complete per completare l'attività con un messaggio di output o di errore.

## Requisiti per processi di business e attività umane

Processi di business e attività umane sviluppati con WebSphere Integration Developer da eseguire su Business Process Choreographer devono essere conformi a regole specifiche accessibili mediante le API dei servizi Web.

I requisiti sono:

1. Le interfacce di processi di business e attività umane devono essere definite mediante lo stile "document/literal wrapped" definito nell'API Java per la specifica RPC basata su XML (JAX-RPC 1.1). Si tratta dello stile predefinito per tutti i processi di business e le attività umane sviluppati con WID.
2. I messaggi di errore generati dai processi di business e dalle attività umane per le operazioni dei servizi Web devono comprendere una singola parte di messaggio WSDL definita con un elemento di schema XML. Ad esempio:

```
<wsdl:part name="myFault" element="myNamespace:myFaultElement"/>
```

### Informazioni correlate

 [Pagina di download API Java per RPC basato su XML \(JAX-RPC\)](#)

 [Quale stile di WSDL è consigliabile utilizzare?](#)

## Sviluppo di applicazioni client

Il processo di sviluppo di applicazioni client è costituito da una serie di fasi.

### Procedure

1. Stabilire quale API dei servizi Web l'applicazione client deve utilizzare: l'API di Business Flow Manager, l'API di Human Task Manager o entrambe.
2. Esportare i file necessari dall'ambiente di WebSphere Process Server. In alternativa, è possibile copiare i file dal CD del client di WebSphere Process Server.
3. Nell'ambiente di sviluppo dell'applicazione client selezionato, generare un *client proxy* utilizzando le risorse esportate.
4. Opzionale: Generare *classi di supporto*. Le classi di supporto sono necessarie se l'applicazione client interagisce direttamente con processi o attività concreti sul server WebSphere. Non sono necessari, tuttavia, se l'applicazione client esegue solo attività generiche, come l'emissione di query.
5. Sviluppare il codice per l'applicazione client.
6. Aggiungere i meccanismi di sicurezza necessari all'applicazione client.

## Copia di risorse

Per consentire la creazione di applicazioni client, un certo numero di risorse deve essere copiato dall'ambiente WebSphere.

Esistono due modi per ottenere le risorse:

- Pubblicare ed esportare le risorse dall'ambiente di WebSphere Process Server.
- Copiare i file dal CD del client di WebSphere Process Server.

### **Pubblicazione ed esportazione di risorse dall'ambiente server**

Per poter sviluppare applicazioni client per l'accesso alle API dei servizi Web, è necessario pubblicare ed esportare un certo numero di risorse dall'ambiente del server WebSphere.

#### **About this task**

Le risorse da esportare sono:

- File WSDL (Web Service Definition Language) che descrivono i tipi di porta e le operazioni che costituiscono le API dei servizi Web.
- File XSD (XML Schema Definition) contenenti definizioni di tipi di dati a cui fanno riferimento servizi e metodi nei file WSDL.
- File WSDL e XSD supplementari che descrivono oggetti business. Gli oggetti business descrivono i processi di business o le attività umane concrete in esecuzione sul server WebSphere. Questi file supplementari sono richiesti solo se l'applicazione client deve interagire direttamente con i processi di business o le attività umane concrete mediante le API dei servizi Web. Non sono necessari se l'applicazione client esegue solo attività generiche, come l'emissione di query.

Una volta pubblicate queste risorse, è necessario copiarle nell'ambiente di programmazione client, dove vengono utilizzate per generare un client proxy e classi di supporto.

#### **Come specificare l'indirizzo dell'endpoint del servizio Web:**

L'indirizzo dell'endpoint del servizio Web è l'URL che un'applicazione client deve specificare per accedere alle API dei servizi Web. L'indirizzo dell'endpoint è scritto nel file WSDL che viene esportato per generare un client proxy per l'applicazione client.

#### **About this task**

L'indirizzo dell'endpoint del servizio Web da utilizzare dipende dalla configurazione del server WebSphere utilizzato:

- Scenario 1. Esiste un unico server WebSphere. L'indirizzo dell'endpoint WebSphere da specificare è il nome host e il numero di porta del server, ad esempio **host1:9080**.
- Scenario 2. Un cluster WebSphere composto da vari server. L'indirizzo dell'endpoint WebSphere da specificare è il nome host e il numero di porta del server su cui si trovano le API dei servizi Web, ad esempio **host2:9081**.
- Scenario 3. Un server Web viene utilizzato come front end. L'indirizzo dell'endpoint WebSphere da specificare è il nome host e il numero di porta del server Web, ad esempio **host:80**.

Per impostazione predefinita, l'indirizzo dell'endpoint del servizio Web utilizza il formato *protocollo://host:porta/root\_contesto/percorso\_fisso*. Dove:

- *protocollo*. Il protocollo di comunicazione da utilizzare tra l'applicazione client e il server WebSphere. Il protocollo predefinito è HTTP. È possibile scegliere di utilizzare il protocollo HTTPS (HTTP su SSL) più sicuro. Si consiglia di utilizzare il protocollo HTTPS.
- *host:porta*. Il nome host e il numero di porta utilizzati per accedere alla macchina su cui si trovano le API dei servizi Web. Questi valori variano in base alla configurazione del server WebSphere, ad esempio, se l'applicazione client deve accedere all'applicazione direttamente o mediante un front end del server Web.
- *root\_contesto*. È possibile scegliere uno dei valori per la root di contesto. Il valore selezionato deve, tuttavia, essere univoco all'interno di ciascuna cella WebSphere. Il valore predefinito utilizza un suffisso "server\_nodo/cluster" che elimina il rischio di conflitti di denominazione.
- Il *percorso\_fisso* è /sca/com/ibm/bpe/api/BFMWS (per l'API di Business Flow Manager) o /sca/com/ibm/task/api/HTMWS (per l'API di Human Task Manager) e non può essere modificato.

L'indirizzo dell'endpoint del servizio Web viene inizialmente specificato quando si configura il contenitore del processo di business o il contenitore dell'attività umana:

### Procedure

1. Accedere alla console di gestione con un ID utente con diritti di amministratore.
2. Scegliere **Applicazioni** → **Moduli SCA**.

**Nota:** È possibile anche selezionare **Applicazioni** → **Applicazioni Enterprise** per visualizzare un elenco di tutte le applicazioni enterprise disponibili.

3. Selezionare **BPEContainer** (per il contenitore del processo di business) o **TaskContainer** (per il contenitore dell'attività umana) dall'elenco di moduli o applicazioni SCA.
4. Scegliere **Fornisci informazioni sull'URL dell'endpoint HTTP** dall'elenco **Ulteriori proprietà**.
5. Selezionare uno dei prefissi predefiniti dall'elenco o immettere un prefisso personalizzato. Utilizzare un prefisso dell'elenco dei prefissi predefiniti se le applicazioni client devono connettersi direttamente al server delle applicazioni su cui si trovano le API dei servizi Web. Altrimenti, specificare un prefisso personalizzato.
6. Fare clic su **Applica** per copiare il prefisso selezionato nel modulo SCA.

7. Fare clic su **OK**. Le informazioni sull'URL vengono salvate nello spazio di lavoro.

## Results

È possibile visualizzare il valore corretto nella console di gestione (ad esempio, per il contenitore del processo di business: **Applicazioni Enterprise** → **BPEContainer** → **Visualizza descrittore di distribuzione**).

Nel file WSDL esportato, l'attributo `location` dell'elemento `soap:address` contiene l'indirizzo dell'endpoint dei servizi Web specificati. Ad esempio:

```
<wsdl:service name="BFMWSservice">
 <wsdl:port name="BFMWSport" binding="this:BFMWSbinding">
 <soap:address location=
 "https://myserver:9080/WebServicesAPIs/sca/com/ibm/bpe/api/BFMWS"/>
 </wsdl:port>
</wsdl:service>
```

### Concetti correlati

“Aggiunta di sicurezza (servizi Web Java)” a pagina 125

È necessario rendere sicure le comunicazioni dei servizi Web implementando i meccanismi di sicurezza nell'applicazione client.

### Attività correlate

“Aggiunta di sicurezza (.NET)” a pagina 134

È possibile rendere sicure le comunicazioni dei servizi Web integrando i meccanismi di sicurezza nell'applicazione client.

## Pubblicazione di file WSDL:

Un file WSDL (Web Service Definition Language) contiene una descrizione dettagliata di tutte le operazioni disponibili con un'API dei servizi Web. File WSDL diversi sono disponibili per le API dei servizi Web di Business Flow Manager e Human Task Manager. È necessario prima pubblicare questi file WSDL, quindi copiarli dall'ambiente WebSphere al proprio ambiente di sviluppo, in cui vengono utilizzati per generare un client proxy.

### Prima di iniziare

Prima di pubblicare i file WSDL, specificare l'indirizzo dell'endpoint dei servizi Web corretto. Tale indirizzo è l'URL che l'applicazione client utilizza per accedere alle API dei servizi Web.

### About this task

È necessario pubblicare i file WSDL una volta.

**Nota:** Se si dispone del CD del client di WebSphere Process Server, è possibile copiare i file direttamente dal CD all'ambiente di programmazione client.

*Pubblicazione del file WSDL del processo di business:*

Utilizzare la console di gestione per pubblicare il file WSDL.

### Procedure

1. Accedere alla console di gestione con un ID utente con diritti di amministratore.
2. Selezionare **Applicazioni** → **Moduli SCA**

**Nota:** È possibile anche selezionare **Applicazioni** → **Applicazioni Enterprise** per visualizzare un elenco di tutte le applicazioni enterprise disponibili.

3. Scegliere l'applicazione **BPEContainer** dall'elenco di moduli o applicazioni SCA.
4. Selezionare **Pubblica file WSDL** dall'elenco di **Ulteriori proprietà**
5. Fare clic sul file zip nell'elenco.
6. Nella finestra Download file visualizzata, fare clic su **Salva**.
7. Passare a una cartella locale e fare clic su **Salva**.

### Results

Il file zip esportato viene denominato BPEContainer\_WSDLFiles.zip. Il file zip contiene un file WSDL che descrive i servizi Web e tutti i file XSD a cui si fa riferimento nel file WSDL.

*Pubblicazione del file WSDL dell'attività umana:*

Utilizzare la console di gestione per pubblicare il file WSDL.

### Procedure

1. Accedere alla console di gestione con un ID utente con diritti di amministratore.
2. Selezionare **Applicazioni** → **Moduli SCA**

**Nota:** È possibile anche selezionare **Applicazioni** → **Applicazioni Enterprise** per visualizzare un elenco di tutte le applicazioni enterprise disponibili.

3. Scegliere l'applicazione **TaskContainer** dall'elenco di moduli o applicazioni SCA.
4. Selezionare **Pubblica file WSDL** dall'elenco di **Ulteriori proprietà**
5. Fare clic sul file zip nell'elenco.
6. Nella finestra Download file visualizzata, fare clic su **Salva**.
7. Passare a una cartella locale e fare clic su **Salva**.

### Results

Il file zip esportato viene denominato TaskContainer\_WSDLFiles.zip. Il file zip contiene un file WSDL che descrive i servizi Web e tutti i file XSD a cui si fa riferimento nel file WSDL.

### Esportazione di oggetti business:

I processi di business e le attività umane hanno interfacce ben definite che consentono l'accesso esterno come servizi Web. Se queste interfacce fanno riferimento a oggetti business, è necessario esportare le definizioni di interfaccia e gli oggetti business nell'ambiente di programmazione client.

### About this task

Questa procedura deve essere ripetuta per ciascun oggetto business con cui l'applicazione client deve interagire.

In WebSphere Process Server, gli oggetti business definiscono il formato dei messaggi di richiesta, risposta ed errore che interagiscono con i processi di business e le attività umane. Questi messaggi possono contenere anche definizioni di tipi di dati complessi.

Ad esempio, per creare e avviare una attività umana, gli elementi di informazione di seguito riportati devono essere trasferiti all'interfaccia dell'attività:

- Il nome del modello di attività
- Lo spazio dei nomi del modello di attività
- Un messaggio di input, contenente dati di business formattati
- Un wrapper per la restituzione del messaggio di risposta
- Un messaggio di errore per la restituzione di errori ed eccezioni

Questi elementi sono inseriti in un singolo oggetto business. Tutte le operazioni dell'interfaccia del servizio Web sono modellate come operazioni del "documento/valore letterale". I parametri di input e output per queste operazioni sono inseriti in documenti del wrapper. Altri oggetti business definiscono i formati dei messaggi di risposta e di errore corrispondenti.

Per creare e avviare il processo di business o la attività umana mediante un servizio Web, gli oggetti del wrapper devono essere resi disponibili per l'applicazione client sul lato client.

A tal scopo, esportare gli oggetti business dall'ambiente WebSphere come file WSDL (Web Service Definition Language) e XSD (XML Schema Definition), importare le definizioni del tipo di dati nell'ambiente di programmazione client, quindi convertirle in classi di supporto per l'utilizzo da parte dell'applicazione client.

### Procedure

1. Lanciare WebSphere Integration Developer Workspace se non è già in esecuzione.
2. Selezionare un modulo Libreria contenente gli oggetti business da esportare. Un modulo Libreria è un file compresso che contiene gli oggetti business necessari.
3. Esportare il modulo Libreria.
4. Copiare i file esportati nell'ambiente di sviluppo dell'applicazione client.

### Esempio

Si supponga che un processo di business presenti la seguente operazione del servizio Web:

```
<wsdl:operation name="updateCustomer">
 <wsdl:input message="tns:updateCustomerRequestMsg"
 name="updateCustomerRequest"/>
 <wsdl:output message="tns:updateCustomerResponseMsg"
 name="updateCustomerResponse"/>
 <wsdl:fault message="tns:updateCustomerFaultMsg"
 name="updateCustomerFault"/>
</wsdl:operation>
```

con i messaggi WSDL definiti come:

```
<wsdl:message name="updateCustomerRequestMsg">
 <wsdl:part element="types:updateCustomer"
 name="updateCustomerParameters"/>
</wsdl:message>
```

```

<wsdl:message name="updateCustomerResponseMsg">
 <wsdl:part element="types:updateCustomerResponse"
 name="updateCustomerResult"/>
</wsdl:message>
<wsdl:message name="updateCustomerFaultMsg">
 <wsdl:part element="types:updateCustomerFault"
 name="updateCustomerFault"/>
</wsdl:message>

```

Gli elementi *concreti* definiti dal cliente `types:updateCustomer`, `types:updateCustomerResponse`, e `types:updateCustomerFault` devono essere trasferiti e ricevuti dalle API dei servizi Web utilizzando i parametri `<xsd:any>` in tutte le operazioni *generiche* (`call`, `sendMessage` ecc.) eseguite dall'applicazione client. Questi elementi definiti dal cliente vengono creati, serializzati e deserializzati nell'applicazione client mediante classi di supporto generate utilizzando i file XSD esportati.

#### Attività correlate

“Creazione di classi di supporto per processi BPEL (.NET)” a pagina 131  
 Alcune operazioni API dei servizi Web richiedono l'utilizzo da parte delle applicazioni client di elementi wrapper in stile "documento/valore letterale". Per generare gli elementi wrapper necessari, le applicazioni client richiedono classi di supporto.

### Utilizzo di file sul CD del client

Come alternativa all'esportazione di risorse dall'ambiente del server WebSphere, è possibile copiare i file necessari per generare un'applicazione client dal CD del client di WebSphere Process Server.

In questo caso, è necessario modificare manualmente l'indirizzo dell'endpoint predefinito dei servizi Web per l'API di Business Flow Manager o l'API di Human Task Manager.

Se l'applicazione client deve accedere a entrambe le API, è necessario modificare l'indirizzo dell'endpoint predefinito per entrambe le API.

#### Copia di file dal CD del client:

I file necessari per accedere alle API dei servizi Web sono disponibili sul CD del client WebSphere Process Server.

#### Procedure

1. Accedere al CD del client e passare alla directory `ProcessChoreographer\client`.
2. Copiare i file necessari nell'ambiente di sviluppo dell'applicazione client.

Per l'API di Business Flow Manager, copiare:

##### **BFMWS.wsdl**

Descrive i servizi Web disponibili nell'API dei servizi Web di Business Flow Manager. Questo file contiene l'indirizzo dell'endpoint.

##### **BFMIF.wsdl**

Descrive i parametri e il tipo di dati di ciascun servizio Web nell'API dei servizi Web di Business Flow Manager.

##### **BFMIF.xsd**

Descrive i tipi di dati utilizzati nell'API dei servizi Web di Business Flow Manager.



**BPCGEN.xsd**

Contiene i tipi di dati comuni alle API dei servizi Web di Business Flow Manager e Human Task Manager.

Per l'API di Human Task Manager, copiare:

**HTMWS.wsdl**

Descrive i servizi Web disponibili nell'API dei servizi Web di Human Task Manager. Questo file contiene l'indirizzo dell'endpoint.

**HTMIF.wsdl**

Descrive i parametri e il tipo di dati di ciascun servizio Web nell'API dei servizi Web di Human Task Manager.

**HTMIF.xsd**

Descrive i tipi di dati utilizzati nell'API dei servizi Web di Human Task Manager.

**BPCGEN.xsd**

Contiene i tipi di dati comuni alle API dei servizi Web di Business Flow Manager e Human Task Manager.

**Nota:** Il file BPCGen.xsd è comune a entrambe le API.

Una volta copiati i file, è necessario modificare manualmente l'indirizzo dell'endpoint delle API dei servizi Web nei file BFMWS.wsdl o HTMWS.wsdl con quello del server delle applicazioni WebSphere su cui si trovano le API dei servizi Web.

**Modifica manuale dell'indirizzo dell'endpoint del servizio Web:**

Se si copiano i file dal CD del client, è necessario modificare l'indirizzo dell'endpoint del servizio Web predefinito specificato nei file WSDL con quello del server su cui si trovano le API dei servizi Web.

**About this task**

È possibile utilizzare la console di gestione per impostare l'indirizzo dell'endpoint del servizio Web prima di esportare i file WSDL. Se, tuttavia, si copiano i file WSDL dal CD del client WebSphere Process Server, è necessario modificare manualmente l'indirizzo predefinito dell'endpoint.

L'indirizzo dell'endpoint del servizio Web da utilizzare dipende dalla configurazione del server WebSphere utilizzato:

- Scenario 1. Esiste un unico server WebSphere. L'indirizzo dell'endpoint WebSphere da specificare è il nome host e il numero di porta del server, ad esempio **host1:9080**.
- Scenario 2. Un cluster WebSphere composto da vari server. L'indirizzo dell'endpoint WebSphere da specificare è il nome host e il numero di porta del server su cui si trovano le API dei servizi Web, ad esempio **host2:9081**.
- Scenario 3. Un server Web viene utilizzato come front end. L'indirizzo dell'endpoint WebSphere da specificare è il nome host e il numero di porta del server Web, ad esempio **host:80**.

*Modifica dell'endpoint dell'API di Business Flow Manager:*



Se si copiano i file dell'API di Business Flow Manager dal CD del client WebSphere Process Server, è necessario modificare manualmente l'indirizzo predefinito dell'endpoint.

### Procedure

1. Passare alla directory contenente i file copiati dal CD del client.
2. Aprire il file BFMWS.wsdl in un editor di testo o in un editor XML.
3. Individuare l'elemento `soap:address` (verso la parte bassa del file).
4. Modificare il valore dell'attributo `location` con l'URL HTTP del server su cui l'API del servizio Web è in esecuzione. Per eseguire questa operazione:
  - a. Se si desidera, sostituire `http` con `https` per utilizzare il protocollo HTTPS più sicuro.
  - b. Sostituire `localhost` con il nome `host` o l'indirizzo IP dell'endpoint del server delle API dei servizi Web.
  - c. Sostituire `9080` con il numero di porta del server delle applicazioni.
  - d. Sostituire `BPEContainer_N1_server1` con il root di contesto dell'applicazione su cui è in esecuzione l'API dei servizi Web. Il contesto predefinito è composto da:
    - `BPEContainer`. Il nome dell'applicazione.
    - `N1`. Il nome del nodo.
    - `server1`. Il nome del server.
  - e. Non modificare la parte fissa dell'URL (`/sca/com/ibm/bpe/api/BFMWS`).

Ad esempio, se l'applicazione è in esecuzione sul server `s1.n1.ibm.com` e il server accetta la richiesta SOAP/HTTP sulla porta `9080`, modificare l'elemento `soap:address` come segue:

```
<soap:address location="http://s1.n1.ibm.com:9080/
BPEContainer_N1_server1/sca/com/ibm/bpe/api/BFMWS"/>
```

### Concetti correlati

"Aggiunta di sicurezza (servizi Web Java)" a pagina 125

È necessario rendere sicure le comunicazioni dei servizi Web implementando i meccanismi di sicurezza nell'applicazione client.

### Attività correlate

"Aggiunta di sicurezza (.NET)" a pagina 134

È possibile rendere sicure le comunicazioni dei servizi Web integrando i meccanismi di sicurezza nell'applicazione client.

### Modifica dell'endpoint dell'API di Human Task Manager:

Se si copiano i file dell'API di Human Task Manager dal CD del client WebSphere Process Server, è necessario modificare manualmente l'indirizzo predefinito dell'endpoint.

### Procedure

1. Passare alla directory contenente i file copiati dal CD del client.
2. Aprire il file HTMWWS.wsdl in un editor di testo o in un editor XML.
3. Individuare l'elemento `soap:address` (verso la parte bassa del file).
4. Modificare il valore dell'attributo `location` con l'indirizzo dell'endpoint corretto. Per eseguire questa operazione:
  - a. Se si desidera, sostituire `http` con `https` per utilizzare il protocollo HTTPS più sicuro.

- b. Sostituire *localhost* con il nome host o l'indirizzo IP dell'endpoint del server delle API dei servizi Web.
- c. Sostituire *9080* con il numero di porta del server delle applicazioni.
- d. Sostituire *HTMContainer\_N1\_server1* con il root di contesto dell'applicazione su cui è in esecuzione l'API dei servizi Web. Il contesto predefinito è composto da:
  - *HTMContainer*. Il nome dell'applicazione.
  - *N1*. Il nome del nodo.
  - *server1*. Il nome del server.
- e. Non modificare la parte fissa dell'URL (*/sca/com/ibm/task/api/HTMWS*) .

Ad esempio, se l'applicazione è in esecuzione sul server **s1.n1.ibm.com** e il server accetta la richiesta SOAP/HTTPS sulla porta **9081**, modificare l'elemento `soap:address` come segue:

```
<soap:address location="https://s1.n1.ibm.com:9081/HTMContainer_N1_server1/sca/com/ibm/task/api/HTMWS"/>
```

#### Concetti correlati

“Aggiunta di sicurezza (servizi Web Java)” a pagina 125

È necessario rendere sicure le comunicazioni dei servizi Web implementando i meccanismi di sicurezza nell'applicazione client.

#### Attività correlate

“Aggiunta di sicurezza (.NET)” a pagina 134

È possibile rendere sicure le comunicazioni dei servizi Web integrando i meccanismi di sicurezza nell'applicazione client.

## Sviluppo di applicazioni client nell'ambiente dei servizi Web Java

È possibile utilizzare qualsiasi ambiente di sviluppo Java compatibile con i servizi Web Java per sviluppare le applicazioni client per le API dei servizi Web.

### Generazione di un client proxy (servizi Web Java)

Le applicazioni client dei servizi Web Java utilizzano un *client proxy* per interagire con le API dei servizi Web.

#### About this task

Un client proxy per i servizi Web Java contiene un certo numero di classi Java Bean che l'applicazione client richiama per eseguire le richieste del servizio Web. Il client proxy gestisce l'insieme di parametri di servizio in messaggi SOAP, invia i messaggi SOAP al servizio Web su HTTP, riceve risposte dal servizio Web e trasferisce i dati restituiti all'applicazione client.

In sostanza, quindi, un client proxy consente ad un'applicazione client di richiamare un servizio Web come se fosse una funzione locale.

**Nota:** È necessario generare un client proxy una sola volta. Tutte le applicazioni client che accedono alle stesse API dei servizi Web possono quindi utilizzare lo stesso client proxy.

Nell'ambiente dei servizi Web IBM, esistono due modi per generare un client proxy:

- Utilizzando ambienti di sviluppo integrati Rational Application Developer o WebSphere Integration Developer.

- Utilizzando lo strumento della riga comandi WSDL2Java.

Altri ambienti di sviluppo dei servizi Web Java in genere includono lo strumento WSDL2Java o funzioni di generazione dell'applicazione client proprietaria.

### Utilizzo di Rational Application Developer per generare un client proxy:

L'ambiente di sviluppo Rational Application Developer integrato consente di generare un client proxy per l'applicazione client.

#### Prima di iniziare

Prima di generare un client proxy, è necessario esportare i file WSDL che descrivono i servizi Web del processo di business o dell'attività umana dall'ambiente WebSphere (o dal CD del client di WebSphere Process Server) e copiare i file nell'ambiente di programmazione client.

#### Procedure

1. Aggiungere al progetto il file WSDL appropriato:
  - Per i processi di business:
    - a. Decomprimere il file esportato BPEContainer\_nomenodo\_nomeserver\_WSDLFiles.zip in una directory temporanea.
    - b. Importare la sottodirectory META-INF dalla sottodirectory decompressa BPEContainer\_nomenodo\_nomeserver.ear/b.jar.
  - Per le attività umane:
    - a. Decomprimere il file esportato TaskContainer\_nomenodo\_nomeserver\_WSDLFiles.zip in una directory temporanea.
    - b. Importare la sottodirectory META-INF dalla directory decompressa TaskContainer\_nomenodo\_nomeserver.ear/h.jar.

Vengono create una nuova directory wsdl e struttura di sottodirectory nel proprio progetto.

2. Modificare le proprietà della procedura guidata del servizio Web:
  - a. In Rational Application Developer, scegliere **Preferenze** → **Servizi Web** → **Generazione codice** → **Runtime IBM WebSphere**.
  - b. Selezionare l'opzione **Genera Java da WSDL utilizzando lo stile nowrapped**.

**Nota:** Se non è possibile selezionare l'opzione **Servizi Web** nel menu **Preferenze**, è necessario per prima cosa abilitare le funzioni richieste come segue: **Finestra** → **Preferenze** → **Workbench** → **Funzioni**. Fare clic su **Sviluppatore Servizi Web** e fare clic su **OK**. Quindi riaprire la finestra **Preferenze** e modificare l'opzione **Generazione Codice**.

3. Selezionare il file BFMWS.WSDL o HTMWS.WSDL ubicati nella nuova directory wsdl.
4. Fare clic con il pulsante destro del mouse e scegliere **Servizi Web** → **Genera client**.

Prima di continuare con i restanti passaggi, assicurarsi che il server sia avviato.

5. Nella finestra **Servizi Web**, fare clic su **Successivo** per accettare tutte le impostazioni predefinite.

6. Nella finestra Selezione Servizi Web, fare clic su **Successivo** per accettare tutte le impostazioni predefinite.
7. Nella finestra Configurazione Ambiente Client:
  - a. Fare clic su **Modifica** e modificare l'opzione Runtime Servizi Web su IBM WebSphere
  - b. Modificare l'opzione Versione J2EE su 1.4.
  - c. Fare clic su **OK**.
  - d. Fare clic su **Avanti**.
8. Questo passaggio è necessario solo se si deve generare un client di Servizi Web che includa sia Business Process che le API Human Task Web Services, in quanto esistono metodi multipli in entrambi i file WSDL.
  - a. Nella finestra di Web Service Proxy, selezionare Definizione associazione personalizzata per il packaging di namespace e quindi fare clic su **OK**.
  - b. Nella finestra Web Service Client per l'associazione tra packaging e namespace, aggiungere il seguente namespace ed effettuare il packaging:  
Per BFMWS.wsdl:

| Namespace                                                                     | Packaging       |
|-------------------------------------------------------------------------------|-----------------|
| http://www.ibm.com/xmlns/prod/websphere/business-process/types/6.0            | com.ibm.sca.bpe |
| http://www.ibm.com/xmlns/prod/websphere/business-process/services/6.0         | com.ibm.sca.bpe |
| http://www.ibm.com/xmlns/prod/websphere/business-process/services/6.0/Binding | com.ibm.sca.bpe |
| http://www.ibm.com/xmlns/prod/websphere/bpc-common/types/6.0                  | com.ibm.sca.bpe |

For HTMWS.wsdl:

| Namespace                                                               | Packaging        |
|-------------------------------------------------------------------------|------------------|
| http://www.ibm.com/xmlns/prod/websphere/human-task/types/6.0            | com.ibm.sca.task |
| http://www.ibm.com/xmlns/prod/websphere/human-task/services/6.0         | com.ibm.sca.task |
| http://www.ibm.com/xmlns/prod/websphere/human-task/services/6.0/Binding | com.ibm.sca.task |
| http://www.ibm.com/xmlns/prod/websphere/bpc-common/types/6.0            | com.ibm.sca.task |

Se viene richiesto di confermare la sovrascrittura, fare clic su **YesToAll**.

9. Fare clic su **Fine**.

## Results

Un client proxy, costituito da un certo numero di proxy, localizzatori e classi di supporto Java, viene generato e aggiunto al progetto. Viene inoltre aggiornato il descrittore di distribuzione.

### Utilizzo di WSDL2Java per generare un client proxy:

WSDL2Java è uno strumento della riga comandi che genera un client proxy. Un client proxy semplifica la programmazione di applicazioni client.

## Prima di iniziare

Prima di generare un client proxy, è necessario esportare i file WSDL che descrivono le API dei servizi Web del processo di business o dell'attività umana dall'ambiente WebSphere (o dal CD del client di WebSphere Process Server) e copiare i file nell'ambiente di programmazione client.

## About this task

### Procedure

1. Utilizzare lo strumento WSDL2Java per generare un client proxy: Immettere:

**wsdl2java** *opzioni WSDLfilepath*

Dove:

- *opzioni* include:

**-noWrappedOperations (-w)**

Disabilita l'individuazione delle operazioni con wrapper. Vengono generati Java bean per messaggi di richiesta e di risposta.

**Nota:** Questo non è il valore predefinito.

**-role (-r)**

Specificare il valore **client** per generare file e file di bind per lo sviluppo client.

**-container (-c)**

Il contenitore client da utilizzare. Gli argomenti validi includono:

**client** Un contenitore client

**ejb** Un contenitore EJB (Enterprise JavaBeans).

**none** Nessun contenitore

**web** Un contenitore Web

**-output (-o)**

La cartella in cui memorizzare i file generati.

Per un elenco completo dei parametri WSDL2Java, utilizzare l'opzione della riga comandi **-help** o fare riferimento alla guida in linea per lo strumento WSDL2Java in WID/RAD.

- *WSDLfilepath* è il percorso e il nome file del file WSDL esportato dall'ambiente WebSphere o copiato dal CD del client.

L'esempio che segue genera un client proxy per l'API dei servizi Web delle attività dell'attività umana:

```
call wsdl2java.bat -r client -c client -noWrappedOperations
-output c:\ws\proxyClient c:\ws\bin\HTMWS.wsdl
```

2. Includere nel progetto i file di classe generati.

## Creazione di classi di supporto per processi BPEL (servizi Web Java)

Gli oggetti business indicati nelle richieste API concrete (ad esempio, sendMessage o call) richiedono l'utilizzo da parte delle applicazioni client di elementi wrapper in stile "documento/valore letterale". Per generare gli elementi wrapper necessari, le applicazioni client richiedono classi di supporto.

## Prima di iniziare

Per creare classi di supporto, il file WSDL dell'API dei servizi Web deve essere stato esportato dall'ambiente di WebSphere Process Server.

### About this task

Le operazioni call() e sendMessage() delle API dei servizi Web consentono l'interazione con i processi BPEL in WebSphere Process Server. Il messaggio di input dell'operazione call() prevede che venga fornito il wrapper del documento/valore letterale del processo.

Esistono varie possibili tecniche per generare classi di supporto per un processo BPEL o un'attività umana, tra cui:

#### 1. Utilizzare l'oggetto SoapElement.

Nell'ambiente Rational Application Developer disponibile in WebSphere Integration Developer, il motore del servizio Web supporta JAX-RPC 1.1. In JAX-RPC 1.1, l'oggetto SoapElement presenta un elemento DOM (Document Object Model), per cui è possibile utilizzare l'API DOM per creare, leggere, caricare e salvare messaggi SOAP.

Si supponga, ad esempio, che il file WSDL contenga il seguente messaggio di input per un processo del flusso di lavoro o un'attività umana:

```
<xsd:element name="operation1">
 <xsd:complexType>
 <xsd:sequence>
 <xsd:element name="input1" nillable="true" type="xsd:string"/>
 </xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

Il file WSDL viene creato quando si sviluppa un modulo di processo o attività umana.

Per creare il messaggio SOAP corrispondente nell'applicazione client mediante l'API DOM:

```
SOAPFactory soapfactoryinstance = SOAPFactory.newInstance();
SOAPElement soapmessage = soapfactoryinstance.createElement
 ("operation1", namespaceprefix, interfaceURI);
SOAPElement inpulement = soapfactoryinstance.createElement("input1");
inpulement.addTextNode(message value);
soapmessage.addChildElement(outpulement);
```

L'esempio di seguito mostra come creare parametri di input per l'operazione sendMessage nell'applicazione client:


```
SendMessage inWsend = new SendMessage();
inWsend.setProcessTemplateName(processtemplatename);
inWsend.setPortType(porttype);
inWsend.setOperation(operationname);
inWsend.set_any(soapmessage);
```

#### 2. Utilizzare la funzione di collegamento personalizzato ai dati di WebSphere.

Questa tecnica è descritta nei seguenti articoli di developerWorks:

- Scelta della tecnologia di associazione personalizzata per i servizi Web
- Sviluppo di servizi Web con SDO EMF per schemi XML complessi

 [Interoperabilità con i modelli e le strategie per servizi Web basati su documentazioni](#)

 [Supporto ai servizi Web per Schema/WSDL\(s\) contenenti tipi di schema JAX-RPC 1.0/1.1 XML opzionali](#)

## Creazione di un'applicazione client (servizi Web Java)

Un'applicazione client invia richieste e riceve risposte dalle API dei servizi Web. Utilizzando un client proxy per gestire la comunicazione e le classi di supporto per formattare tipi di dati complessi, un'applicazione client può richiamare i metodi del servizio Web come se fossero funzioni locali.

### Prima di iniziare

Prima di iniziare a creare un'applicazione client, generare il client proxy e le classi di supporto necessarie.

### About this task

È possibile sviluppare le applicazioni client utilizzando uno strumento di sviluppo compatibile con i servizi Web, ad esempio IBM Rational Application Developer (RAD). È possibile generare qualsiasi tipo di applicazione dei servizi Web per richiamare le API dei servizi Web.

### Procedure

1. Creare un nuovo progetto di applicazione client.
2. Generare il client proxy e aggiungere le classi di supporto Java al progetto.
3. Codificare l'applicazione client.
4. Generare il progetto.
5. Eseguire l'applicazione client.

L'esempio di seguito riportato illustra come utilizzare l'API dei servizi Web di Business Flow Manager.

```
// creare proxy
 BFMIFProxy proxy = new BFMIFProxy();
// preparare i dati di input per l'operazione
 GetProcessTemplate iw = new GetProcessTemplate();
 iw.setIdentifier(your_process_template_name);

// richiamare l'operazione
 GetProcessTemplateResponse oW = proxy.getProcessTemplate(iw);

// elaborare l'output dell'operazione
 ProcessTemplateType ptd = oW.getProcessTemplate();
 System.out.println("getName= " + ptd.getName());
 System.out.println("getPtid= " + ptd.getPtid());
```

### Attività correlate

“Generazione di un client proxy (servizi Web Java)” a pagina 120

Le applicazioni client dei servizi Web Java utilizzano un *client proxy* per interagire con le API dei servizi Web.

“Creazione di classi di supporto per processi BPEL (servizi Web Java)” a pagina 123

Gli oggetti business indicati nelle richieste API concrete (ad esempio, `sendMessage` o `call`) richiedono l'utilizzo da parte delle applicazioni client di elementi wrapper in stile “documento/valore letterale”. Per generare gli elementi wrapper necessari, le applicazioni client richiedono classi di supporto.

## Aggiunta di sicurezza (servizi Web Java)

È necessario rendere sicure le comunicazioni dei servizi Web implementando i meccanismi di sicurezza nell'applicazione client.



WebSphere Application Server attualmente supporta i seguenti meccanismi di sicurezza per le API dei servizi Web:


- Il token nome utente
- LTPA (Lightweight Third Party Authentication)

#### Concetti correlati

Ruoli di autorizzazione per i processi di business

Un ruolo è un insieme di persone che condividono lo stesso livello di autorizzazione. Le azioni che è possibile effettuare sui processi di business dipendono dal ruolo di autorizzazione. Questo può essere un ruolo J2EE o un ruolo basato sull'istanza.

#### Informazioni correlate

 [http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/c6task\\_auth.html](http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.websphere.bpc.610.doc/doc/bpc/c6task_auth.html)

### Implementazione del token del nome utente:

Il meccanismo di sicurezza token nome utente fornisce credenziali relative a nome utente e password.

#### About this task

Con il meccanismo di sicurezza token nome utente, è possibile scegliere di implementare vari *gestori di callback*. In base alla scelta effettuata:

- Viene richiesto di fornire un nome utente e una password ogni volta che viene eseguita l'applicazione client.
- Il nome utente e la password vengono scritti nel descrittore di distribuzione.

In entrambi i casi, il nome utente e la password forniti devono corrispondere a quelli di un ruolo autorizzato nel contenitore del processo di business e dell'attività umana corrispondente.

Il nome utente e la password vengono inseriti nella busta del messaggio di richiesta e vengono visualizzati chiaramente nell'intestazione del messaggio SOAP. Si consiglia tuttavia di configurare l'applicazione client per l'utilizzo del protocollo di comunicazione HTTPS (HTTP su SSL). Tutte le comunicazioni vengono quindi codificate. È possibile selezionare il protocollo di comunicazione HTTPS quando si specifica l'indirizzo URL dell'endpoint del servizio Web.

Per definire un token nome utente:

#### Procedure

1. Creare un token di protezione:
  - a. Aprire l'**Editor di Distribuzione** del proprio modulo
  - b. Fare clic sulla scheda **Estensione WS**.
  - c. Sotto **Riferimenti di servizio**, potrebbero essere elencati i seguenti Riferimenti di servizio Web:
    - service/BFMWSService per processi di business
    - service/HTMWSService per attività umane

Vengono elencati a seconda del fatto che BFMWS.wsdl (per processi di business), HTMWWS.wsdl (per attività umana), o entrambi, siano stati aggiunti quando il client proxy è stato generato.



- d. Per entrambi i riferimenti di servizio:
  - 1) Selezionare uno dei **Riferimenti di servizio**.
  - 2) Espandere la sezione **Configurazione Generatore di Richiesta**.
  - 3) Espandere la sottosezione **Token di Protezione**.
  - 4) Fare clic su **Aggiungi**. Si apre la finestra Token di Protezione.
  - 5) Nel campo **Nome**, digitare un nome per il nuovo token di protezione: **UserNameTokenBFM** o **UserNameTokenHTM**.
  - 6) Nell'elenco a tendina **Tipo Token**, selezionare **Nome utente**. (Il campo **Nome locale** è automaticamente popolato con un valore assunto.)
  - 7) Lasciare vuoto il campo **URI**. Non è necessario un valore URI per un token nome utente.
  - 8) Fare clic su **OK**.
2. Creare un generatore di token:
  - a. Aprire l'**Editor di Distribuzione** del proprio modulo
  - b. Fare clic sulla scheda **Bind WS**
  - c. Sotto **Riferimenti di servizio**, gli stessi Riferimenti di servizio WEB sono elencati come nel precedente passaggio:
    - service/BFMWSService per processi di business
    - service/HTMWSService per attività umane
  - d. Per entrambi i riferimenti di servizio:
    - 1) Selezionare uno dei **Riferimenti di servizio**.
    - 2) Espandere la sezione **Configurazione Bind Generatore Richiesta Protezione**.
    - 3) Espandere la sottosezione **Generatore Token**.
    - 4) Fare clic su **Aggiungi**. Si apre la finestra Generatore Token.
    - 5) Nel campo **Nome**, digitare un nome per il nuovo generatore di token, quale "UserNameTokenGeneratorBFM" o "UserNameTokenGeneratorHTM".
    - 6) Nel campo **Classe generatore token**, assicurarsi che il sia selezionata la seguente classe di generatore di token: **com.ibm.wsspi.wssecurity.token.UsernameTokenGenerator**.
    - 7) Nell'elenco a tendina **Token di protezione**, selezionare il token di protezione appropriato creato precedentemente.
    - 8) Selezionare la casella di spunta **Usa tipo di valore**.
    - 9) Nel campo **Tipo Value**, selezionare **Token Nome utente**. (Il campo **Nome locale** è automaticamente popolato per riflettere la propria scelta di **Token nome utente**.)
    - 10) Nel campo **Gestore richiamata**, digitare o "com.ibm.wsspi.wssecurity.auth.callback.GUIPromptCallbackHandler" (che richiede il nome utente e la password quando si esegue l'applicazione client) o "com.ibm.wsspi.wssecurity.auth.callback.NonPromptCallbackHandler".
    - 11) Se si sceglie **NonPromptCallbackHandler**, è necessario specificare un nome utente e una password validi nel campo corrispondente del descrittore di distribuzione.
    - 12) Fare clic su **OK**.

#### Attività correlate

"Come specificare l'indirizzo dell'endpoint del servizio Web" a pagina 112  
 L'indirizzo dell'endpoint del servizio Web è l'URL che un'applicazione client

deve specificare per accedere alle API dei servizi Web. L'indirizzo dell'endpoint è scritto nel file WSDL che viene esportato per generare un client proxy per l'applicazione client.

### Informazioni correlate



IBM WebSphere Developer Technical Journal: Web services security with WebSphere Application Server V6

### Implementazione del meccanismo di sicurezza LTPA:

Il meccanismo di sicurezza LTPA (Lightweight Third Party Authentication) può essere utilizzato quando l'applicazione client viene eseguita in un contesto di sicurezza stabilito in precedenza.

#### About this task

Il meccanismo di sicurezza LTPA è disponibile solo se l'applicazione client viene eseguita in un ambiente sicuro in cui è già stato stabilito un contesto di sicurezza. Se l'applicazione client, ad esempio, viene eseguita in un contenitore EJB (Enterprise JavaBeans), il client EJB deve registrarsi per poter richiamare l'applicazione client. Un contesto di sicurezza viene quindi stabilito. Se l'applicazione client EJB richiama quindi un servizio Web, il gestore di callback LTPA richiama il token LTPA dal contesto di sicurezza e lo aggiunge al messaggio di richiesta SOAP. Sul server, il token LTPA viene gestito dal meccanismo LTPA.

Per implementare il meccanismo di sicurezza LTPA:

#### Procedure

1. Nell'ambiente Rational Application Developer disponibile in WebSphere Integration Developer, scegliere **Bind WS** → **Configurazione Bind Generatore Richiesta Protezione** → **Generatore token**.
2. Creare un token di protezione:
  - a. Aprire l'**Editor di Distribuzione** del proprio modulo
  - b. Fare clic sulla scheda **Estensione WS**.
  - c. Sotto **Riferimenti di servizio**, potrebbero essere elencati i seguenti **Riferimenti di servizio Web**:
    - service/BFMWSService per processi di business
    - service/HTMWSService per attività umaneVengono elencati a seconda del fatto che BFMWS.wsdl (per processi di business), HTMWS.wsdl (per attività umana), o entrambi, siano stati aggiunti quando il client proxy è stato generato.
  - d. Per entrambi i riferimenti di servizio:
    - 1) Selezionare uno dei **Riferimenti di servizio**.
    - 2) Espandere la sezione **Configurazione Generatore di Richiesta**.
    - 3) Espandere la sottosezione **Token di Protezione**.
    - 4) Fare clic su **Aggiungi**. Si apre la finestra Token di Protezione.
    - 5) Nel campo **Nome**, digitare un nome per il nuovo token di protezione: **LTPATokenBFM** o **LTPATokenHTM**.
    - 6) Nell'elenco a tendina **Tipo Token**, selezionare **LTPAToken**. (I campi **URI** e **Nome locale** sono automaticamente popolati da valori assunti.)
    - 7) Fare clic su **OK**.
3. Creare un generatore di token:

- a. Aprire l'**Editor di Distribuzione** del proprio modulo
- b. Fare clic sulla scheda **Bind WS**
- c. Sotto **Riferimenti di servizio**, gli stessi Riferimenti di servizio WEB sono elencati come nel precedente passaggio:
  - service/BFMWSService per processi di business
  - service/HTMWSService per attività umane
- d. Per entrambi i riferimenti di servizio:
  - 1) Selezionare uno dei **Riferimenti di servizio**.
  - 2) Espandere la sezione **Configurazione Bind Generatore Richiesta Protezione**.
  - 3) Espandere la sottosezione **Generatore Token**.
  - 4) Fare clic su **Aggiungi**. Si apre la finestra Generatore Token.
  - 5) Nel campo **Nome**, digitare un nome per il nuovo generatore di token, quale "LTPATokenGeneratorBFM" o "LTPATokenGeneratorHTM".
  - 6) Nel campo **Classe generatore token**, assicurarsi che il sia selezionata la seguente classe di generatore di token:  
**com.ibm.wsspi.wssecurity.token.LTPATokenGenerator**.
  - 7) Nell'elenco a tendina **Token di protezione**, selezionare il token di protezione appropriato creato precedentemente.
  - 8) Selezionare la casella di spunta **Usa tipo di valore**.
  - 9) Nel campo **Tipo di valore**, selezionare **LTPAToken**. (I campi **URI** e **Nome locale** sono automaticamente popolati per riflettere la propria scelta di **LTPA Token**.)
  - 10) Nel campo **Gestore richiamata**, digitare  
"com.ibm.wsspi.wssecurity.auth.callback.LTPATokenCallbackHandler".
  - 11) Fare clic su **OK**.

## Results

Al runtime, **LTPATokenCallbackHandler** richiama il token LTPA dal contesto di sicurezza esistente e lo aggiunge al messaggio di richiesta SOAP.

## Aggiunta di supporto per le transazioni (servizi Web Java)

Le applicazioni client dei servizi Web Java possono essere configurate per consentire all'elaborazione delle richieste da parte del server di partecipare alle transazioni del client, trasmettendo un contesto di applicazione client come parte della richiesta del servizio. Questo supporto per le transazioni è definito nella specifica WS-AT (Web Services-Atomic Transaction).

### About this task

WebSphere Application Server esegue ciascuna richiesta API dei servizi Web come una transazione diversa. Le applicazioni client possono essere configurate per utilizzare il supporto per le transazioni in uno dei seguenti modi:

- Partecipando alla transazione. L'elaborazione delle richieste da parte del server viene eseguita nel contesto delle transazioni dell'applicazione client. Se il server rileva un problema, quindi, mentre la richiesta API dei servizi Web è in esecuzione e sottoposta al rollback, anche la richiesta dell'applicazione client viene sottoposta al rollback.

- Non utilizzare il supporto per le transazioni. WebSphere Application Server crea una nuova transazione in cui eseguire la richiesta, ma l'elaborazione delle richieste da parte del server non viene eseguita con il contesto delle transazioni dell'applicazione client.

## Sviluppo di applicazioni client nell'ambiente .NET

Microsoft .NET offre un efficace ambiente di sviluppo in cui collegare le applicazioni mediante i servizi Web.

### Generazione di un client proxy (.NET)

Le applicazioni client .NET utilizzano un *client proxy* per interagire con le API dei servizi Web. Un client proxy protegge le applicazioni client dalla complessità del protocollo di messaggistica dei servizi Web.

#### Prima di iniziare

Per creare un client proxy, è necessario prima esportare un certo numero di file WSDL dall'ambiente WebSphere e copiarli nell'ambiente di programmazione client.

**Nota:** Se si dispone del CD del client di WebSphere Process Server, è possibile copiare i file dal CD.

#### About this task

Un client proxy comprende una serie di classi bean C#. Ogni classe contiene tutti i metodi e oggetti relativi a un singolo servizio Web. I metodi del servizio gestiscono l'insieme di parametri in messaggi SOAP completi, inviano i messaggi SOAP al servizio Web su HTTP, ricevono risposte dal servizio Web e gestiscono i dati restituiti.

**Nota:** È necessario generare un client proxy una sola volta. Tutte le applicazioni client che accedono alle API dei servizi Web possono quindi utilizzare lo stesso client proxy.

#### Procedure

1. Utilizzare il comando WSDL per generare un client proxy: Immettere:

```
wSDL opzioni WSDLfilepath
```

Dove:

- *opzioni* include:

##### **/language**

Consente di specificare il linguaggio utilizzato per creare la classe proxy. L'impostazione predefinita è C#. È possibile anche specificare **VB** (Visual Basic), **JS** (JScript), o **VJS** (Visual J#) come argomenti del linguaggio.

##### **/output**

Il nome del file di output, con il suffisso appropriato. Ad esempio, proxy.cs

##### **/protocol**

Il protocollo implementato nella classe proxy. **SOAP** è l'impostazione predefinita.

Per un elenco completo dei parametri **WSDL.exe**, utilizzare l'opzione della riga comandi **/?** o fare riferimento alla guida in linea per lo strumento WSDL in Visual Studio.

- *WSDLfilepath* è il percorso e il nome del file WSDL esportato dall'ambiente WebSphere o copiato dal CD del client.

L'esempio che segue genera un client proxy per l'API dei servizi Web di Human Task Manager:

```
wsdl /language:cs /output:proxycient.cs c:\ws\bin\HTMWS.wsdl
```

2. Compilare il client proxy come un file DLL (Dynamic Link Library).

## Creazione di classi di supporto per processi BPEL (.NET)

Alcune operazioni API dei servizi Web richiedono l'utilizzo da parte delle applicazioni client di elementi wrapper in stile "documento/valore letterale". Per generare gli elementi wrapper necessari, le applicazioni client richiedono classi di supporto.

### Prima di iniziare

Per creare classi di supporto, il file WSDL dell'API dei servizi Web deve essere stato esportato dall'ambiente di WebSphere Process Server.

### About this task

Le operazioni `call()` e `sendMessage()` delle API dei servizi Web determinano l'avvio dei processi BPEL in WebSphere Process Server. Il messaggio di input dell'operazione `call()` prevede che venga fornito il wrapper del documento/valore letterale del processo BPEL. Per generare i bean e le classi necessarie per il processo BPEL, copiare l'elemento `<wsdl:types>` in un nuovo file XSD, quindi utilizzare lo strumento `xsd.exe` per generare classi di supporto.

### Procedure

1. Se questa operazione non è stata ancora eseguita, esportare il file WSDL dell'interfaccia del processo BPEL da WebSphere Integration Developer.
2. Aprire il file WSDL in un editor di testo o in un editor XML.
3. Copiare il contenuto di tutti gli elementi secondari dell'elemento `<wsdl:types>` e incollarli in un nuovo file XSD, skeleton.
4. Eseguire lo strumento `xsd.exe` nel file XSD:

```
call xsd.exe file.xsd /classes /o
```

Dove:

**file.xsd**

Il file di XML Schema Definition da convertire.

**/classes (/c)**

Genera classi di supporto che corrispondono al contenuto del file o dei file XSD specificati.

**/output (/o)**

Specificare la directory di output per i file generati. Se questa directory viene omessa, la directory corrente è la directory predefinita.

Ad esempio:

```
call xsd.exe ProcessCustomer.xsd /classes /output:c:\temp
```

5. Aggiungere il file di classe generato all'applicazione client. Se si utilizza Visual Studio, ad esempio è possibile eseguire questa operazione utilizzando la voce di menu **Project** → **Add Existing Item**.

Se il file `ProcessCustomer.wsdl` contiene quanto segue:

```

<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:bons1="http://com/ibm/bpe/unittest/sca"
 xmlns:tns="http://ProcessTypes/bpel/ProcessCustomer"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 name="ProcessCustomer"
 targetNamespace="http://ProcessTypes/bpel/ProcessCustomer">
 <wsdl:types>
 <xsd:schema targetNamespace="http://ProcessTypes/bpel/ProcessCustomer"
 xmlns:bons1="http://com/ibm/bpe/unittest/sca"
 xmlns:tns="http://ProcessTypes/bpel/ProcessCustomer"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:import namespace="http://com/ibm/bpe/unittest/sca"
 schemaLocation="xsd-includes/http.com.ibm.bpe.unittest.sca.xsd"/>
 <xsd:element name="doit">
 <xsd:complexType>
 <xsd:sequence>
 <xsd:element name="input1" nillable="true" type="bons1:Customer"/>
 </xsd:sequence>
 </xsd:complexType>
 </xsd:element>
 <xsd:element name="doitResponse">
 <xsd:complexType>
 <xsd:sequence>
 <xsd:element name="output1" nillable="true" type="bons1:Customer"/>
 </xsd:sequence>
 </xsd:complexType>
 </xsd:element>
 </xsd:schema>
 </wsdl:types>
 <wsdl:message name="doitRequestMsg">
 <wsdl:part element="tns:doit" name="doitParameters"/>
 </wsdl:message>
 <wsdl:message name="doitResponseMsg">
 <wsdl:part element="tns:doitResponse" name="doitResult"/>
 </wsdl:message>
 <wsdl:portType name="ProcessCustomer">
 <wsdl:operation name="doit">
 <wsdl:input message="tns:doitRequestMsg" name="doitRequest"/>
 <wsdl:output message="tns:doitResponseMsg" name="doitResponse"/>
 </wsdl:operation>
 </wsdl:portType>
</wsdl:definitions>

```

Il file XSD risultante conterrà:

```

<xsd:schema xmlns:bons1="http://com/ibm/bpe/unittest/sca"
 xmlns:tns="http://ProcessTypes/bpel/ProcessCustomer"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://ProcessTypes/bpel/ProcessCustomer">
 <xsd:import namespace="http://com/ibm/bpe/unittest/sca"
 schemaLocation="Customer.xsd"/>
 <xsd:element name="doit">
 <xsd:complexType>
 <xsd:sequence>
 <xsd:element name="input1" type="bons1:Customer" nillable="true"/>
 </xsd:sequence>
 </xsd:complexType>
 </xsd:element>
 <xsd:element name="doitResponse">
 <xsd:complexType>
 <xsd:sequence>
 <xsd:element name="output1" type="bons1:Customer" nillable="true"/>
 </xsd:sequence>
 </xsd:complexType>
 </xsd:element>
</xsd:schema>

```

## Informazioni correlate

 Documentazione Microsoft per XML Schema Definition Tool (XSD.EXE)

## Creazione di un'applicazione client (.NET)

Un'applicazione client invia richieste e riceve risposte dalle API dei servizi Web. Utilizzando un client proxy per gestire le comunicazioni e le classi di supporto per formattare tipi di dati complessi, un'applicazione client può richiamare i metodi del servizio Web come se fossero funzioni locali.

### Prima di iniziare

Prima di iniziare a creare un'applicazione client, generare il client proxy e le classi di supporto necessarie.

### About this task

È possibile sviluppare le applicazioni client .NET utilizzando uno strumento di sviluppo compatibile con .NET, ad esempio Visual Studio .NET. È possibile generare qualsiasi tipo di applicazione .NET per richiamare le API dei servizi Web generiche.

### Procedure

1. Creare un nuovo progetto di applicazione client. Ad esempio, creare una **WinFX Windows Application** in Visual Studio.
2. Nelle opzioni del progetto, aggiungere un riferimento al file DLL del client proxy. Aggiungere tutte le classi di supporto contenenti le definizioni dell'oggetto business per il progetto. In Visual Studio, ad esempio, è possibile eseguire questa operazione utilizzando l'opzione **Project** → **Add existing item**.
3. Creare un oggetto per il client proxy. Ad esempio:

```
HTMClient.HTMReference.HumanTaskManagerComponent1Export_HumanTaskManagerHttpService service =
 new HTMClient.HTMReference.HumanTaskManagerComponent1Export_HumanTaskManagerHttpService();
```

4. Indicare i tipi di dati dell'oggetto business utilizzati nei messaggi da inviare o ricevere dal servizio Web. Ad esempio:

```
HTMClient.HTMReference.TKIID id = new HTMClient.HTMReference.TKIID();
```

```
ClipBG bg = new ClipBG();
Clip clip = new Clip();
```

5. Richiamare funzioni specifiche del servizio Web e specificare i parametri necessari. Ad esempio, per creare e avviare una attività umana:

```
HTMClient.HTMReference.createAndStartTask task =
 new HTMClient.HTMReference.createAndStartTask();
HTMClient.HTMReference.StartTask sTask = new HTMClient.HTMReference.StartTask();
```

```
sTask.taskName = "SimpleTask";
sTask.taskNamespace = "http://myProcess/com/acme/task";
sTask.inputMessage = bg;
task.inputTask = sTask;
```

```
id = service.createAndStartTask(task).outputTask;
```

6. Processi e attività remote vengono identificati con ID persistenti (id nell'esempio al passo precedente). Ad esempio, per richiamare una attività umana creata in precedenza:

```
HTMClient.HTMReference.claimTask claim = new HTMClient.HTMReference.claimTask();
claim.inputTask = id;
```

### Attività correlate



“Generazione di un client proxy (.NET)” a pagina 130

Le applicazioni client .NET utilizzano un *client proxy* per interagire con le API dei servizi Web. Un client proxy protegge le applicazioni client dalla complessità del protocollo di messaggistica dei servizi Web.

“Creazione di classi di supporto per processi BPEL (.NET)” a pagina 131

Alcune operazioni API dei servizi Web richiedono l’utilizzo da parte delle applicazioni client di elementi wrapper in stile “documento/valore letterale”. Per generare gli elementi wrapper necessari, le applicazioni client richiedono classi di supporto.

## Aggiunta di sicurezza (.NET)

È possibile rendere sicure le comunicazioni dei servizi Web integrando i meccanismi di sicurezza nell’applicazione client.

### About this task

Questi meccanismi di sicurezza possono includere token nome utente (nome utente e password) o token di sicurezza binari personalizzati e basati su XML.

### Procedure

1. Scaricare e installare WSE (Web Services Enhancements) 2.0 SP3 per Microsoft .NET. È disponibile in:

<http://www.microsoft.com/downloads/details.aspx?familyid=1ba1f631-c3e7-420a-bc1e-ef18bab66122&displaylang=en>

2. Modificare il codice client del proxy generato come segue.

Modificare:

```
public class Export1_MyMicroflowHttpService : System.Web.Services.Protocols.SoapHttpClientProtocol {
 In:
 public class Export1_MyMicroflowHttpService : Microsoft.Web.Services2.WebServicesClientProtocol {
```

**Nota:** Queste modifiche vengono perdute se si rigenera il client proxy eseguendo lo strumento WSDL.exe.

3. Modificare il codice dell’applicazione client aggiungendo le seguenti righe nella parte superiore del file:

```
using System.Web.Services.Protocols;
using Microsoft.Web.Services2;
using Microsoft.Web.Services2.Security.Tokens;
...
```

4. Aggiungere il codice per implementare il meccanismo di sicurezza desiderato. Ad esempio, il seguente codice aggiunge la protezione di nome utente e password:

```
string user = "U1";
string pwd = "password";
UsernameToken token =
 new UsernameToken(user, pwd, PasswordOption.SendPlainText);

me._proxy.RequestSoapContext.Security.Tokens.Clear();
me._proxy.RequestSoapContext.Security.Tokens.Add(token);
```

## Interrogazione degli oggetti business-process e degli oggetti relativi alle attività

È possibile utilizzare le API dei servizi Web per eseguire una query su oggetti relativi a processi di business e attività nel database di Business Process Choreographer, per richiamare proprietà specifiche di tali oggetti.



## About this task

Il database di Business Process Choreographer memorizza i dati dei modelli e delle istanze (runtime) per la gestione dei processi di business e delle attività.

Mediante le API dei servizi Web, le applicazioni client possono eseguire query per richiamare informazioni dal database su processi di business e attività.

Le applicazioni client possono eseguire una query offline per richiamare una proprietà specifica di un oggetto. Le query utilizzate possono essere spesso salvate. Le query memorizzate possono essere quindi richiamate e utilizzate dall'applicazione client.

## Query sugli oggetti del processo di business e relativi alle attività

Utilizzare l'interfaccia di query delle API dei servizi Web per richiamare informazioni sui processi di business e le attività.

Le applicazioni client utilizzano una sintassi simile a SQL per eseguire query del database.

### Esempio per i servizi Web Java

```
string processTemplateName = "ProcessCustomerLR";
query query1 = new query();
query1.selectClause = "DISTINCT PROCESS_INSTANCE.STARTED, PROCESS_INSTANCE.PIID";
query1.whereClause =
 "PROCESS_INSTANCE.TEMPLATE_NAME = '" + processTemplateName + "'";
query1.orderByClause = "PROCESS_INSTANCE.STARTED";
query1.threshold = null;
query1.timeZone = "UTC"; query1.skipTuples = null;
queryResponse queryResponse1 = proxy.query(query1);
```

Le informazioni richiamate dal database vengono restituite mediante le API dei servizi Web come una *serie di risultati query*.

Ad esempio:

```
QueryResultSetType queryResultSet = queryResponse1.queryResultSet;
if (queryResultSet != null) {
 Console.WriteLine("--> QueryResultSetType");
 Console.WriteLine(" . size= " + queryResultSet.size);
 Console.WriteLine(" . numberColumns= " + queryResultSet.numberColumns);
 string indent = " . ";

 // -- informazioni sulla colonna di query
 QueryColumnInfoType[] queryColumnInfo = queryResultSet.QueryColumnInfo;
 if (queryColumnInfo.Length > 0) {
 Console.WriteLine();
 Console.WriteLine("= . QueryColumnInfoType size= " + queryColumnInfo.Length);
 Console.WriteLine(" | tableName ");
 for (int i = 0; i < queryColumnInfo.Length ; i++) {
 Console.WriteLine(" | " + queryColumnInfo[i].tableName.PadLeft(20));
 }
 Console.WriteLine();
 Console.WriteLine(" | columnName ");
 for (int i = 0; i < queryColumnInfo.Length ; i++) {
 Console.WriteLine(" | " + queryColumnInfo[i].columnName.PadLeft(20));
 }
 Console.WriteLine();
 Console.WriteLine(" | data type ");
 for (int i = 0; i < queryColumnInfo.Length ; i++) {
 QueryColumnInfoType tt = queryColumnInfo[i].type;
```

```

 Console.WriteLine(" | " + tt.ToString());
 }
 Console.WriteLine();
}
else {
 Console.WriteLine("--> queryColumnInfo= <null>");
}

// - valori dei risultati di query
string[][] result = queryResultSet.result;
if (result !=null) {
 Console.WriteLine();
 Console.WriteLine("= . result size= " + result.Length);
 for (int i = 0; i < result.Length; i++) {
 Console.Write(indent + i);
 string[] row = result[i];
 for (int j = 0; j < row.Length; j++) {
 Console.Write(" | " + row[j]);
 }
 Console.WriteLine();
 }
}
else {
 Console.WriteLine("--> result= <null>");
}
}
else {
 Console.WriteLine("--> QueryResultSetType= <null>");
}
}

```

La funzione query restituisce gli oggetti in base all'autorizzazione del chiamante. La serie di risultati della query contiene solo le proprietà degli oggetti che il chiamante è autorizzato a visualizzare.

Le viste del database predefinite sono fornite per eseguire la query delle proprietà dell'oggetto. Per i modelli di processo, la funzione di query dispone della seguente sintassi:

```

ProcessTemplateData[] queryProcessTemplates
 (java.lang.String whereClause,
 java.lang.String orderByClause,
 java.lang.Integer threshold,
 java.util.TimeZone timezone);

```

Per i modelli di processo, la funzione di query dispone della seguente sintassi:

```

TaskTemplate[] queryTaskTemplates
 (java.lang.String whereClause,
 java.lang.String orderByClause,
 java.lang.Integer threshold,
 java.util.TimeZone timezone);

```

Per altri oggetti del processo di business e degli oggetti relativi alle attività, la funzione query ha la seguente sintassi:

```

QueryResultSet query (java.lang.String selectClause,
 java.lang.String whereClause,
 java.lang.String orderByClause,
 java.lang.Integer skipTuples
 java.lang.Integer threshold,
 java.util.TimeZone timezone);

```

L'interfaccia di query contiene anche un metodo queryAll. È possibile utilizzare questo metodo per richiamare tutti i dati rilevanti per un oggetto, ad esempio per scopi di monitoraggio. Il chiamante del metodo queryAll deve disporre di uno dei

seguenti ruoli di Java 2 Platform, Enterprise Edition (J2EE):  
BPESystemAdministrator, BPESystemMonitor, TaskSystemAdministrator o  
TaskSystemMonitor. La verifica dell'autorizzazione, mediante l'utilizzo  
dell'elemento di lavoro corrispondente dell'oggetto, non viene applicata.

### Esempio per .NET

```
ProcessTemplateType[] templates = null;

try {
 queryProcessTemplates iW = new queryProcessTemplates();
 iW.whereClause = "PROCESS_TEMPLATE.STATE=PROCESS_TEMPLATE.STATE.STATE_STARTED";
 iW.orderByClause = null;
 iW.threshold = null;
 iW.timeZone = null;

 Console.WriteLine("--> queryProcessTemplates ... ");
 Console.WriteLine("--> query: WHERE " + iW.whereClause + " ORDER BY " +
 iW.orderByClause + " THRESHOLD " + iW.threshold + " TIMEZONE " + iW.timeZone);

 templates = proxy.queryProcessTemplates(iW);

 if (templates.Length < 1) {
 Console.WriteLine("--> No templates found :-(");
 }
 else {
 for (int i = 0; i < templates.Length ; i++) {
 Console.WriteLine("--> found template with ptid: " + templates[i].ptid);
 Console.WriteLine(" and name: " + templates[i].name);
 /* ... other properties of ProcessTemplateType ... */
 }
 }
}
catch(Exception e) {
 Console.WriteLine("exception= " + e);
}
```

### Parametri di query:

In ciascuna query deve essere specificato un numero di clausole e parametri simili a SQL.

Una query è composta da:

- Clausola Select
- Clausola Where
- Clausola Order-by
- Parametro skip-tuples
- Parametro Threshold
- Parametro Time-zone

### Viste predefinite per le query sugli oggetti processo di business o attività umane

Le viste del database predefinite sono fornite per gli oggetti processo di business e attività umane.

Utilizzare tali viste per eseguire le query dei dati di riferimento per tali oggetti. Quando si utilizzano queste viste, non è necessario aggiungere esplicitamente predicati di unione per le colonne della vista, poiché tali costrutti vengono aggiunti automaticamente. È possibile utilizzare la funzione di query delle API dei servizi Web per eseguire la query di questi dati.

## Gestione delle query memorizzate

Le query memorizzate forniscono una modalità di salvataggio delle query eseguite di frequente. La query memorizzata può essere una query disponibile per tutti gli utenti (query pubblica) o una query che appartiene a un utente specifico (query privata).

### About this task

Una query memorizzata è una query che viene memorizzata nel database ed identificata mediante un nome. Una query memorizzata privata e una query memorizzata pubblica e query memorizzate private di proprietari diversi possono avere lo stesso nome.

È possibile disporre di query memorizzate per gli oggetti del processo di business, oggetti di attività o una combinazione di questi due tipi di oggetti.

Gestione di query memorizzate pubbliche

Le query memorizzate pubbliche vengono create dall'amministratore di sistema. Queste query sono disponibili per tutti gli utenti.

Gestione di query memorizzate private per altri utenti

Le query private possono essere create da qualsiasi utente. Queste query sono disponibili solo per il proprietario della query e l'amministratore di sistema.

Utilizzo delle query memorizzate private

Se non si è un amministratore di sistema, è possibile creare, eseguire ed eliminare le query memorizzate private. È possibile anche utilizzare le query memorizzate pubbliche create dall'amministratore di sistema.

---

## Sviluppo di applicazioni client JMS

È possibile sviluppare applicazioni client che accedono ad applicazioni di processi di business attraverso l'API Java Messaging Service (JMS).

### About this task

## Presentazione di JMS

WebSphere Process Server Versione 6.1 supporta messaggistica asincrona, su interfaccia di programmazione Java Messaging Service (JMS), come metodo di comunicazione.

JMS garantisce ai client Java (applicazioni client o J2EE) un modo comune per creare e ricevere e leggere richieste come messaggi JMS.

JMS è un'interfaccia che supporta messaggistica asincrona che:

- Utilizza messaggistica **point-to-point o produzione/sottoscrizione**. Le strutture di messaggistica possono inviare informazioni ad altre applicazioni senza che queste le richiedano esplicitamente. Le stesse informazioni possono essere inviate a molti sottoscrittori simultaneamente. L'interfaccia JMS di Business Process Choreographer supporta solo messaggistica point-to-point.
- Offre **indipendenza di ritmo**. Le strutture JMS funzionano in maniera asincrona, ma sono anche in grado di simulare una modalità richiesta/risposta sincrona. Ciò permette ai sistemi di provenienza e di destinazione di lavorare simultaneamente senza doversi attendere a vicenda. Questa funzione è estremamente utile per il Business Process Choreographer, in quanto consente di interagire in maniera asincrona con processi di business di esecuzione prolungata.

- Supporta **transazioni**. Le transazioni consentono alle applicazioni client di gestire gruppi di messaggi inviati o ricevuti come un'unica unità atomica. Le transazioni JMS sono eseguite all'interno della transazione del server. Per l'interfaccia JMS di Business Process Choreographer, solitamente viene inviato e ricevuto un unico messaggio per ciascuna transazione.
- **Assicura la consegna delle informazioni**. Le strutture JMS possono gestire i messaggi in modalità transazionale e assicurare la consegna del messaggio (sebbene non sia garantita la tempestività della consegna). Per il Business Process Choreographer, questa funzione affidabile della consegna del messaggi è particolarmente importante in quanto è relativa ai processi di business.
- Assicura **interoperabilità tra strutture eterogenee**. Le applicazioni di provenienza e destinazione possono operare in ambienti eterogenei senza dover gestire problemi di comunicazione e di esecuzione relativi alle rispettive strutture.
- **Rende gli scambi più fluidi**. Il passaggio alla modalità messaggio permette uno scambio di informazioni di maggior qualità.

## Requisiti per processi di business

I processi di business sviluppato con WebSphere Integration Developer per essere eseguiti su devono essere conformi a regole specifiche per essere accessibili mediante le API dei servizi JMS.

I requisiti sono:

1. Le interfacce dei processi di business devono essere definite usando lo stile "document/literal wrapped" definito nella specifica Java API per RPC su piattaforma XML(JAX-RPC 1.1). Si tratta dello stile assunto per tutti i processi di business e attività umane sviluppati con WebSphere Integration Developer.
2. I messaggi di errore generati dai processi di business e dalle attività umane per le operazioni dei servizi Web devono comprendere una singola parte di messaggio WSDL definita con un elemento di schema XML. Ad esempio:
 

```
<wsdl:part name="myFault" element="myNamespace:myFaultElement"/>
```

### Informazioni correlate



Pagina di download API Java per RPC basato su XML (JAX-RPC)



Quale stile di WSDL è consigliabile utilizzare?

## Accesso all'interfaccia JMS

Per inviare e ricevere messaggi attraverso l'interfaccia JMS, un'applicazione deve prima di tutto creare una connessione al BPC.cellname.Bus, creare una sessione, e poi generare produttori e consumatori di messaggi.

### About this task

Il process server accetta messaggi Java Message Service (JMS) che seguono il paradigma point-to-point. Un'applicazione che invia o riceve messaggi JMS deve eseguire le seguenti azioni.

Il seguente esempio suppone che il client JMS sia eseguito in un ambiente gestito (EJB, client applicativo, o Web client container). Se si desidera eseguire il client JMS in un ambiente J2SE, fare riferimento a "IBM Client per JMS su J2SE con IBM WebSphere Application Server" su <http://www-1.ibm.com/support/docview.wss?uid=swg24012804>.

## Procedure

1. Creazione di una connessione al BPC.*cellname*.Bus. Non esiste una di connessione preconfigurata per le richieste di un'applicazione client: un'applicazione client può usare la ReplyConnectionFactory JMS di API o creare la propria factory di connessione, nel cui caso può utilizzare la ricerca Java Naming and Directory Interface (JNDI) per richiamare la factory di connessione. Il nome JNDI-lookup deve essere lo stesso di quello specificato al momento della configurazione della coda di richiesta esterna di Business Process Choreographer. Il seguente esempio suppone che l'applicazione client crea la propria factory di connessione chiamata "jms/clientCF".

```
//Ottenere il contesto iniziale JNDI assunto.
Contesto initialContext = new InitialContext();
```

```
// Ricercare la factory di connessione.
// Creare una factory di connessione che si connetta al bus BPC.
// Darle un nome, per esempio, "jms/clientCF".
// Configurare un alias di autenticazione appropriato.
ConnectionFactory connectionFactory =
 (ConnectionFactory)initialContext.lookup("jms/clientCF");
```

```
// Creare la connessione.
Connection connection = connectionFactory.createConnection();
```

2. Creare una sessione in modo che i produttori e consumatori di messaggi possano essere creati.

```
// Creare una sessione di transazione mediante auto-riconoscimento.
Session session = connection.createSession(true, Session.AUTO_ACKNOWLEDGE);
```

3. Creare un produttore di messaggi per inviare messaggi. Il nome JNDI-lookup deve essere lo stesso di quello specificato al momento della configurazione della coda di richiesta esterna di Business Process Choreographer.

```
// Ricercare la destinazione della coda di entrata di Business Process Choreographer a
// inviare messaggi a.
Queue sendQueue = (Queue) initialContext.lookup("jms/BFMJMSAPIQueue");
```

```
// Creare un produttore di messaggi.
MessageProducer producer = session.createProducer(sendQueue);
```

4. Creare un consumatore di messaggi che riceva le risposte. Il nome JNDI-lookup della destinazione di risposta può specificare una destinazione definita dall'utente, ma può anche specificare la destinazione di risposta assunta (definita da Business Process Choreographer) jms/BFMJMSReplyQueue. In entrambi i casi, la destinazione di risposta deve trovarsi nel BPC.<cellname>.Bus.

```
// Ricercare la destinazione della coda di risposta.
Queue replyQueue = (Queue) initialContext.lookup("jms/BFMJMSReplyQueue");
```

```
// Creare un consumatore di messaggi.
MessageConsumer consumer = session.createConsumer(replyQueue);
```

5. Inviare un messaggio.

```
// Avviare la connessione.
connection.start();
```

```
// Creare un messaggio - cfr. le descrizioni dell'attività per vedere alcuni esempi - e inviarlo.
// Questo metodo è definito altrove ...
String payload = createXMLDocumentForRequest();
TextMessage requestMessage = session.createTextMessage(payload);
```

```
// Impostare intestazione JMS obbligatoria.
// targetFunctionName è il nome dell'operazione di API JMS
// (per esempio, getProcessTemplate, sendMessage)
requestMessage.setStringProperty("TargetFunctionName", targetFunctionName);
```

```

// Impostare la coda di risposta; è obbligatorio se la replyQueue
// non è la coda assunta (come nell'esempio).
requestMessage.setJMSReplyTo(replyQueue);

// Inviare il messaggio.
producer.send(requestMessage);

// Ottenere l'ID del messaggio.
String jmsMessageID = requestMessage.getJMSMessageID();

session.commit();

```

6. Ricevere la risposta.

```

// Ricevere il messaggio di risposta e analizzare la risposta.
TextMessage replyMessage = (TextMessage) consumer.receive();

// Ottenere il payload.
String payload = replyMessage.getText();

session.commit();

```

7. Chiudere la connessione e liberare le risorse.

```

// Pulizia finale; liberare le risorse.
session.close();
connection.close();

```

**Nota:** Non è necessario chiudere la connessione dopo ciascuna transazione. Una volta avviata una connessione, è possibile scambiare qualsiasi numero di messaggi di richiesta e risposta prima che la connessione venga chiusa. L'esempio mostra un semplice caso con una chiamata singola all'interno di un singolo metodo business.

## Struttura di un messaggio JMS Business Process Choreographer

L'intestazione e il corpo di ciascun messaggio JMS devono avere una struttura predefinita.

Un messaggio Java Message Service (JMS) comprende:

- Un'intestazione per l'identificazione del messaggio e instradamento dell'informazione.
- Il corpo (payload) del messaggio che contiene il contenuto.

Business Process Choreographer supporta solo formati di messaggio di testo.

### Intestazione messaggio

JMS permette ai client di accedere a un numero di campi intestazione messaggio.

I seguenti campi intestazione messaggio possono essere impostati da un client JMS Business Process Choreographer:

- **JMSReplyTo**

La destinazione a cui inviare una risposta alla richiesta. Se questo campo non è specificato nel messaggio di richiesta, la risposta viene inviata alla destinazione risposta assunta dell'interfaccia Export (un Export è una versione interfaccia client di un componente di un processo di business). Tale destinazione può essere ottenuta mediante `initialContext.lookup("jms/BFMJMSReplyQueue")`;

- **TargetFunctionName**



Il nome dell'operazione WSDL, per esempio, "queryProcessTemplates". Questo campo deve sempre essere impostato. Va notato che TargetFunctionName specifica l'operazione dell'interfaccia generica del messaggio JMS qui descritta. Ciò non deve essere confuso con le operazioni fornite da processi o attività concrete che possono essere chiamate indirettamente, per esempio, mediante le operazioni **call** o **sendMessage**.

Un client Business Process Choreographer può inoltre accedere ai seguenti campi di intestazione:

- **JMSMessageID**

Identifica in modo univoco un messaggio. Impostato dal provider JMS quando è inviato il messaggio. Se il client imposta il JMSMessageID prima di inviare il messaggio, viene sovrascritto dal provider JMS. Se viene richiesto l'ID del messaggio al fine dell'autenticazione, il client può richiamare il JMSMessageID dopo aver inviato il messaggio.

- **JMSCorrelationID**

Collega i messaggi. Non impostare questo campo. Un messaggio di risposta Business Process Choreographer contiene il JMSMessageID del messaggio di richiesta.

Ciascun messaggio di risposta contiene i seguenti campi di intestazione JMS:

- **IsBusinessException**

"Falso" per messaggi in uscita WSDL, o "vero" per messaggi di errore WSDL.

ServiceRuntimeExceptions non vengono rispediti ad applicazioni client asincrone. Quando si verifica una grave eccezione durante l'esecuzione di un messaggio di richiesta JMS, ne deriva un errore nel runtime, che causa il ripristino della transazione che sta elaborando questo messaggio di richiesta. Il messaggio di richiesta JMS viene successivamente riconsegnato. Se l'errore si verifica inizialmente, durante l'elaborazione del messaggio come parte dell'Export SCA (per esempio, mentre si sta deserializzando il messaggio), vengono effettuati nuovi tentativi fino al numero massimo di consegne fallite specificato dalla destinazione di ricezione dell'Export SCA. Raggiunto il numero massimo di consegne fallite, il messaggio di richiesta viene aggiunto alla destinazione eccezioni del sistema del bus di Business Process Choreographer. Se tuttavia l'errore si verifica durante l'elaborazione corrente della richiesta dal componente SCA di Business Flow Manager il messaggio di richiesta fallito viene gestito dall'infrastruttura di gestione eventi falliti di WebSphere Process Server, ossia è possibile che finisca nel database della gestione eventi falliti se i nuovi tentativi non risolvono la situazione.

### Corpo del messaggio

Il corpo del messaggio JMS è una Stringa contenente un documento XML che rappresenta l'elemento involucro document/literal dell'operazione.

Un semplice esempio di un corpo del messaggio di richiesta valido è:

```
<?xml version="1.0" encoding="UTF-8"?>
<_6:queryProcessTemplates xmlns:_6="http://www.ibm.com/xmlns/prod/
 websphere/business-process/services/6.0">
<whereClause>PROCESS_TEMPLATE.STATE IN (1)</whereClause>
</_6:queryProcessTemplates>
```

## Autorizzazione per rendering JMS

Per autorizzare l'uso dell'interfaccia JMS, le impostazioni di protezione devono essere abilitate in WebSphere Application Server.



Quando viene installato il contenitore del processo di business, il ruolo **JMSAPIUser** deve essere associato ad un ID utente. Questo ID utente è usato per emettere tutte le richieste API JMS. Per esempio, se **JMSAPIUser** è associato a "Utente A", tutte le richieste API JMS per il motore di processo avranno origine dall'"Utente A".

Al ruolo **JMSAPIUser** devono essere assegnate le seguenti autorità:

| Richiesta      | Autorizzazione richiesta                                            |
|----------------|---------------------------------------------------------------------|
| forceTerminate | Amministratore processo                                             |
| sendEvent      | Potenziale proprietario dell'attività o amministratore del processo |

**Nota:** Per tutte le altre richieste, non sono necessarie autorizzazioni speciali.

L'autorità speciale è concessa a una persona avente il ruolo di amministratore del processo di business. Un amministratore del processo di business è un ruolo speciale; si distingue dall'amministratore di processo di un'istanza di processo. Un amministratore del processo di business dispone di tutti i privilegi.

Non è possibile eliminare l'ID utente dell'iniziatore del processo dal proprio registro utente finché l'istanza del processo esiste ancora. Se si procede all'eliminazione, non sarà possibile continuare con questo processo. Si riceve la seguente eccezione nel file di log del sistema:

nessun ID univoco per: <ID utente>

## Panoramica della API JMS

L'interfaccia di messaggio JMS (in seguito denominata "API JMS") permette di sviluppare applicazioni client che in maniera asincrona accedono a processi di business in esecuzione nell'ambiente Business Process Choreographer.

La API JMS permette alle applicazioni client di interagire in maniera asincrona con microflussi e processi di esecuzione prolungata.

La API JMS espone la stessa interfaccia dell'API servizi Web, con le seguenti eccezioni:

- Con la API servizi Web, l'operazione `call` può essere usata solo per chiamare microflussi. Usando la API JMS, tuttavia, l'operazione `call` può essere usata per chiamare sia microflussi che processi di esecuzione prolungata.
- Le operazioni seguenti non sono espone attraverso la API JMS:
  - L'operazione `callAsync` (insieme alle operazioni di richiamata correlate).
  - Le operazioni `completeAndClaimSuccessor` e `getParticipatingTask`

## Esempio - esecuzione di un processo a lunga durata

Per far lavorare un'applicazione client generica con processi di esecuzione prolungata, la sequenza dei passaggi è la seguente:

1. Impostare l'ambiente JMS, come descritto in "Accesso all'interfaccia JMS" a pagina 139.
2. Ottenere un elenco delle definizioni di processo installate:
  - Inviare `queryProcessTemplates`
  - Questo restituisce un elenco di oggetti **ProcessTemplate**.

3. Ottenere un elenco di attività di avvio (ricezione o pick con `createInstance="yes"`):
  - Inviare `getStartActivities`.
  - Questo restituisce un elenco di oggetti **InboundOperationTemplate**.
4. Creare un messaggio di input. Questo è specifico dell'ambiente, e potrebbe richiedere l'uso di risorse predistribuite, specifiche del processo.
5. Creare una versione del processo:
  - Emettere un `sendMessage`.

Con la API JMS, è possibile inoltre usare l'operazione `call` per interagire con operazioni richiesta-risposta di esecuzione prolungata fornite da un processo di business. Questa operazione rispedisce il risultato o l'errore dell'operazione alla destinazione `reply-to` specificata, anche dopo molto tempo. Pertanto, se si usa l'operazione `call`, non è necessario usare le operazioni `query` e `getOutputMessage` per ottenere il messaggio di output o di errore del processo.
6. Facoltativamente, ottenere messaggi di output dalle versioni del processo ripetendo i seguenti passaggi:
  - Emettere `query` per ottenere lo stato concluso della versione del processo.
  - Emettere `getOutputMessage`.
7. Facoltativamente, lavorare con operazioni supplementari esposte dal processo:
  - `getWaitingActivities` o `getActiveEventHandlers` per ottenere un elenco di oggetti **InboundOperationTemplate**.
  - Creare messaggi di input
  - Inviare messaggi con `sendMessage`
8. Facoltativamente, ottenere e impostare proprietà personalizzate definite nel processo o attività contenute con `getCustomProperties` e `setCustomProperties`.
9. Facoltativamente, terminare di lavorare con una versione del processo:
  - Inviare `elimina` e `termina` per terminare di lavorare con il processo di esecuzione prolungata.

## Sviluppo di applicazioni JMS

Le applicazioni client JMS devono essere sviluppate in Java utilizzando l'ambiente Java 2 Enterprise Edition (J2EE).

### About this task

Le applicazioni client JMS scambiano messaggi di richiesta e di risposta con API JMS. Per creare un messaggio di richiesta, l'applicazione client riempie il corpo del messaggio `TextMessage` JMS con un elemento XML che rappresenta l'involucro `document/literal` dell'operazione corrispondente.

### Copia di risorse

È possibile copiare una serie di risorse dall'ambiente WebSphere per poter creare applicazioni client JMS.

L'uso di tali risorse è obbligatorio solo se viene usato `BOXMLSerializer` per creare il corpo del messaggio JMS.

Esistono due modi per ottenere le risorse:

- Pubblicare ed esportare le risorse dall'ambiente di WebSphere Process Server.

Per WebSphere Process Server 6.1, tutte le risorse client devono essere ricercate nella directory *root\_installazione*\ProcessChoreographer\client. Per la API JMS, queste risorse sono:

BFMIF.wsdl  
BFMIF.xsd  
BPCGen.xsd

- Copiare i file dal CD del client di WebSphere Process Server.

### **Pubblicazione di risorse dall'ambiente del server:**

Per aiutare a sviluppare le applicazioni client che accedono alla API JMS, è possibile pubblicare un numero di risorse dall'ambiente del server WebSphere.

#### **About this task**

Per WebSphere Process Server 6.1, tutte le risorse client devono essere ricercate nella directory *was\_home*\ProcessChoreographer\client. Per la API JMS, queste risorse sono:

BFMIF.wsdl  
BFMIF.xsd  
BPCGen.xsd

Una volta pubblicate queste risorse, copiarle nel proprio ambiente di programmazione client.

#### **Copia di file dal CD del client:**

I file necessari per accedere alla API dei servizi JMS sono disponibili sul CD del client WebSphere Process Server.

#### **Procedure**

1. Accedere al CD del client e passare alla directory ProcessChoreographer\client.
2. Copiare i file necessari nell'ambiente di sviluppo dell'applicazione client  
Per WebSphere Process Server 6.1, tutte le risorse client devono essere ricercate nella directory \ProcessChoreographer\client. Per la API JMS, queste risorse sono:

BFMIF.wsdl  
BFMIF.xsd  
BPCGen.xsd

### **Controllo del messaggio di risposta per le eccezioni di business**

Le applicazioni client JMS devono controllare l'intestazione del messaggio di tutti i messaggi di risposta per le eccezioni di business.

#### **About this task**

Un'applicazione client JMS deve per prima cosa controllare la proprietà **IsBusinessException** nell'intestazione del messaggio di risposta.

Ad esempio:

```

// ricezione del messaggio di risposta
Message receivedMessage = ((JmsProxy) getToBeInvokedUponObject()).receiveMessage();
String strResponse = ((TextMessage) receivedMessage).getText();

se (receivedMessage.getStringProperty("IsBusinessException") {
 // strResponse è un errore business
 // ogni api può terminare w/a processFaultMsg
 // la chiamata api anche w/a businessFaultMsg
}
else {
 // strResponse è il messaggio di output
}

```

---

## Sviluppo delle applicazioni Web per i processi di business e attività umane utilizzando i componenti JSF

Business Process Choreographer fornisce vari componenti JSF (JavaServer Faces). È possibile estendere ed integrare questi componenti per aggiungere le funzioni processo di business e attività umane alle applicazioni Web.

### About this task

È possibile utilizzare WebSphere Integration Developer per creare la propria applicazione Web.

### Procedure

1. Creare un progetto dinamico e modificare le proprietà delle funzioni del progetto Web per inserire i componenti di base JSF.  
Per ulteriori informazioni sulla creazione di un progetto Web, visitare l'Infocenter di WebSphere Integration Developer.
2. Aggiungere l'archivio di Business Process Choreographer Explorer Java (file JAR).

Aggiungere i seguenti file alla directory WEB-INF/lib del progetto:

- bpcclientcore.jar
- bfmclientmodel.jar
- htmclientmodel.jar
- bpcjsfcomponents.jar

Se si sta distribuendo la propria applicazione Web su un server remoto, si consiglia di aggiungere i seguenti file. Questi file sono necessari per accedere remotamente agli API Business Process Choreographer.

- bpe137650.jar
- task137650.jar

In WebSphere Process Server, tutti questi file si trovano nella seguente directory:

- Sui sistemi Windows: *root\_installazione*\ProcessChoreographer\client
- Sui sistemi UNIX, Linux, e i5/OS: *root\_installazione*/ProcessChoreographer/client

3. Aggiungere i riferimenti EJB necessari al descrittore di distribuzione dell'applicazione Web, file web.xml.

```

<ejb-ref id="EjbRef_1">
 <ejb-ref-name>ejb/BusinessProcessHome</ejb-ref-name>
 <ejb-ref-type>Session</ejb-ref-type>
 <home>com.ibm.bpe.api.BusinessFlowManagerHome</home>
 <remote>com.ibm.bpe.api.BusinessFlowManager</remote>

```

```

</ejb-ref>
<ejb-ref id="EjbRef_2">
 <ejb-ref-name>ejb/HumanTaskManagerEJB</ejb-ref-name>
 <ejb-ref-type>Session</ejb-ref-type>
 <home>com.ibm.task.api.HumanTaskManagerHome</home>
 <remote>com.ibm.task.api.HumanTaskManager</remote>
</ejb-ref>
<ejb-local-ref id="EjbLocalRef_1">
 <ejb-ref-name>ejb/LocalBusinessProcessHome</ejb-ref-name>
 <ejb-ref-type>Session</ejb-ref-type>
 <local-home>com.ibm.bpe.api.LocalBusinessFlowManagerHome</local-home>
 <local>com.ibm.bpe.api.LocalBusinessFlowManager</local>
</ejb-local-ref>
<ejb-local-ref id="EjbLocalRef_2">
 <ejb-ref-name>ejb/LocalHumanTaskManagerEJB</ejb-ref-name>
 <ejb-ref-type>Session</ejb-ref-type>
 <local-home>com.ibm.task.api.LocalHumanTaskManagerHome</local-home>
 <local>com.ibm.task.api.LocalHumanTaskManager</local>
</ejb-local-ref>

```

4. Aggiungere i componenti JSF di Business Process Choreographer Explorer all'applicazione JSF.

a. Aggiungere i riferimenti delle librerie dei tag necessari per le applicazioni ai file JavaServer Pages (JSP). In genere, sono necessarie le librerie di tag HTML e JSF e la libreria di tag richiesta dai componenti JSF.

- <%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
- <%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
- <%@ taglib uri="http://com.ibm.bpe.jsf/taglib" prefix="bpe" %>

b. Aggiungere un tag <f:view> al corpo della pagina JSP ed un tag <h:form> al tag <f:view>.

c. Aggiungere i componenti JSF ai file JSP.

In base all'applicazione, aggiungere il componente Elenco, il componente Dettagli e il componente CommandBar o il componente Messaggio ai file JSP. È possibile aggiungere più istanze di ciascun componente.

d. Configurare i bean gestiti nel file di configurazione JSF.

Per impostazione predefinita, il file di configurazione si trova nel file faces-config.xml. Questo file si trova nella directory WEB-INF dell'applicazione Web.

In base al componente da aggiungere al file JSP, è inoltre necessario aggiungere i riferimenti alla query e altri oggetti del wrapper al file di configurazione JSF. Per garantire una corretta soluzione degli errori, è inoltre necessario definire sia un bean di errore e una destinazione di navigazione per la pagina di errori nel file di configurazione JSF.

```

<faces-config>
...
<managed-bean>
 <managed-bean-name>BPCErrors</managed-bean-name>
 <managed-bean-class>com.ibm.bpc.clientcore.util.ErrorBeanImpl
 </managed-bean-class>
 <managed-bean-scope>session</managed-bean-scope>
</managed-bean>

...
<navigation-rule>
...
<navigation-case>
<descrizione>
La pagina di errori generale.
</descrizione>
<from-outcome>error</from-outcome>

```

```

<to-view-id>/Error.jsp</to-view-id>
</navigation-case>
...
</navigation-rule>
</faces-config>

```

In situazioni di errore che lanciano la pagina di errori, l'eccezione viene impostata sul bean di errore.

- e. Implementare il codice personalizzato necessario per supportare i componenti JSF.
5. Distribuire l'applicazione.

Se si sta distribuendo l'applicazione in un ambiente di distribuzione in rete, modificare i nomi della risorsa di destinazione Java Naming and Directory Interface (JNDI) in valori in cui gli API di Business Flow Manager e Human Task Manager possano essere trovati nella cella.

- Se i contenitori dei processi di business sono configurati su un altro server nella stessa cella gestita, i nomi hanno la struttura che segue:  
`cell/nodes/nodename/servers/servername/com/ibm/bpe/api/BusinessManagerHome`  
`cell/nodes/nodename/servers/servername/com/ibm/task/api/HumanTaskManagerHome`
- Se i contenitori dei processi di business sono configurati su un cluster nella stessa cella, i nomi hanno la struttura che segue:  
`cell/clusters/clustername/com/ibm/bpe/api/BusinessFlowManagerHome`  
`cell/clusters/clustername/com/ibm/task/api/HumanTaskManagerHome`

Associare i riferimenti EJB ai nomi JNDI o aggiungere manualmente i riferimenti al file `ibm-web-bnd.xml`.

La tabella di seguito riportata elenca i bind di riferimento e le relative associazioni predefinite.

Tabella 33. Associazione dei bind di riferimento ai nomi JNDI

| Bind di riferimento                       | Nome JNDI                                            | Commenti                |
|-------------------------------------------|------------------------------------------------------|-------------------------|
| <code>ejb/BusinessProcessHome</code>      | <code>com/ibm/bpe/api/BusinessFlowManagerHome</code> | Bean di sessione remota |
| <code>ejb/LocalBusinessProcessHome</code> | <code>com/ibm/bpe/api/BusinessFlowManagerHome</code> | Bean di sessione locale |
| <code>ejb/HumanTaskManagerEJB</code>      | <code>com/ibm/task/api/HumanTaskManagerHome</code>   | Bean di sessione remota |
| <code>ejb/LocalHumanTaskManagerEJB</code> | <code>com/ibm/task/api/HumanTaskManagerHome</code>   | Bean di sessione locale |

## Results

L'applicazione Web distribuita contiene le funzioni fornite dai componenti di Business Process Choreographer Explorer.

## Operazioni successive

Se si stanno usando JSP personalizzati per i messaggi di processo e di attività, è necessario associare i moduli Web usati per distribuire i JSP agli stessi server a cui viene associato il client JSF personalizzato.

### Concetti correlati

"Errore di gestione dei componenti JSF" a pagina 150

I componenti JSF (JavaServer Faces) sfruttano un bean associato predefinito, `BPCError`, per la gestione degli errori. In situazioni di errore che eseguono il trigger della pagina di errore, l'eccezione viene impostata sul bean di errore.

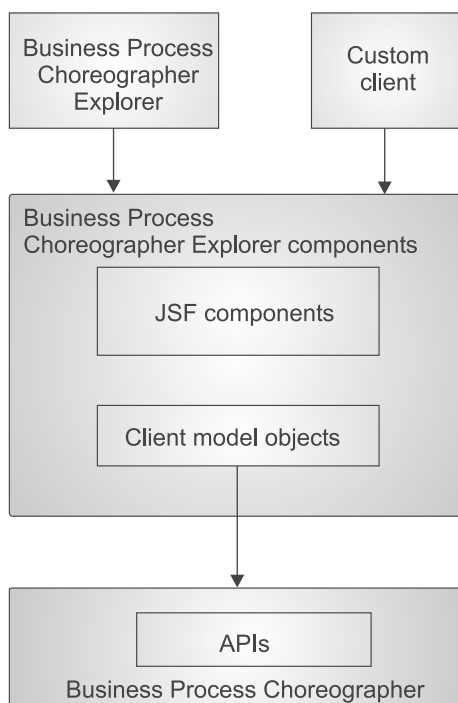
### Attività correlate

“Accedere all’interfaccia remota del bean di sessione” a pagina 22  
Un’applicazione client EJB accede all’interfaccia remota del bean di sessione attraverso l’interfaccia home remota del bean.

## Componenti di Business Process Choreographer Explorer

I componenti di Business Process Choreographer Explorer sono una serie di elementi riutilizzabili e configurabili basati sulla tecnologia JSF (JavaServer Faces). È possibile integrare tali elementi nelle applicazioni Web. Tali applicazioni possono accedere alle applicazioni di processi di business e attività umane installate.

I componenti sono costituiti da una serie di componenti JSF e una serie di oggetti del modello del client. La relazione dei componenti con Business Process Choreographer, Business Process Choreographer Explorer e altri client personalizzati viene illustrata nella seguente figura.



### Componenti JSF

I componenti di Business Process Choreographer Explorer comprendono i seguenti componenti JSF. Integrare questi componenti JSF nei file JSP (JavaServer Pages) quando si creano le applicazioni Web per le attività con i processi di business e le attività umane.

- **Componente Elenco**  
Il componente Elenco visualizza un elenco di oggetti dell’applicazione in una tabella, ad esempio, attività, operazioni, istanze di processo, modelli di processo, elementi di lavoro o escalation. Questo componente dispone di un gestore elenco associato.
- **Componente Dettagli**  
Il componente Dettagli visualizza le proprietà delle attività, degli elementi di lavoro, delle operazioni, delle istanze di processo e dei modelli di processo. Questo componente dispone di un gestore dettagli associato.
- **Componente CommandBar**

Il componente `CommandBar` visualizza una barra con pulsanti. Questi pulsanti rappresentano i comandi che funzionano su un oggetto in una vista dettagli o sugli oggetti selezionati in un elenco. Questi oggetti sono forniti da un gestore elenco o un gestore dettagli.

- **Componente Messaggio**

Il componente `Messaggio` visualizza un messaggio che può contenere un SDO (Service Data Object) o un tipo semplice.

## Oggetti di modello del client

Gli oggetti di modello del client vengono utilizzati con i componenti JSF. Gli oggetti implementano alcune delle interfacce dell'API di Business Process Choreographer sottostante e avviluppiano l'oggetto di origine. Gli oggetti del modello del client forniscono il supporto in lingua nazionale per le etichette e utilità di conversione per alcune proprietà.

## Errore di gestione dei componenti JSF

I componenti JSF (JavaServer Faces) sfruttano un bean associato predefinito, `BPCError`, per la gestione degli errori. In situazioni di errore che eseguono il trigger della pagina di errore, l'eccezione viene impostata sul bean di errore.

Questo bean implementa l'interfaccia `com.ibm.bpc.clientcore.util.ErrorBean`. La pagina di errore viene visualizzata nelle seguenti situazioni:

- Se si verifica un errore durante l'esecuzione di una query definita per un gestore eventi e l'errore viene generato come errore `ClientException` mediante il metodo `execute` di un comando
- Se un errore `ClientException` viene generato dal metodo `execute` di un comando e questo errore non è un errore `ErrorsInCommandException` né implementa l'interfaccia `CommandBarMessage`
- Se viene visualizzato un messaggio di errore nel componente e si segue il collegamento ipertestuale per il messaggio

È disponibile un'implementazione predefinita dell'interfaccia `com.ibm.bpc.clientcore.util.ErrorBeanImpl`.

L'interfaccia viene definita nel modo seguente:

```
public interface ErrorBean {

 public void setException(Exception ex);

 /*
 * Questo richiamo del metodo setter consente di inoltrare
 * le impostazioni internazionali e l'eccezione. Ciò consente ai metodi
 * getMessage di restituire le stringhe localizzate
 */
 public void setException(Exception ex, Locale locale);

 public Exception getException();
 public String getStack();
 public String getNestedExceptionMessage();
 public String getNestedExceptionStack();
 public String getRootExceptionMessage();
 public String getRootExceptionStack();

 /*
 * Questo metodo restituisce il messaggio di eccezione
 * concatenato in modo ricorsivo con i messaggi di tutte
```



```

 * le eccezioni nidificate.
 */
 public String getAllExceptionMessages();

 /*
 * Questo metodo restituisce l'accodamento dell'eccezione
 * concatenato in modo ricorsivo agli accodamenti di tutte
 * le eccezioni nidificate.
 */
 public String getAllExceptionStacks();
}

```

### Concetti correlati

“Errore di gestione del componente Elenco” a pagina 156  
 Quando si utilizza il componente Elenco per visualizzare elenchi nell'applicazione JSF, è possibile approfittare delle funzioni di gestione errori fornite dalla classe `com.ibm.bpe.jsf.handler.BPCListHandler`.

## Convertitori ed etichette assunti per oggetti del modello de client

Gli oggetti del modello di client implementano le interfacce corrispondenti dell'API di Business Process Choreographer.

Il componente Elenco e il componente Dettagli operano su qualsiasi bean. È possibile visualizzare tutte le proprietà di un bean. Tuttavia, se si desidera impostare i convertitori e le etichette usati per le proprietà di un bean, è necessario usare il tag `column` per il componente Elenco, o il tag `property` per il componente Dettagli. Invece di impostare convertitori ed etichette, è possibile definire convertitori ed etichette assunti per le proprietà definendo i seguenti metodi statici. È possibile definire i seguenti metodi statici:

```

static public String getLabel(String property,Locale locale);
static public com.ibm.bpc.clientcore.converter.SimpleConverter
 getConverter(String property);

```

La seguente tabella mostra gli oggetti del modello del client che implementano le corrispondenti classi API di Business Flow Manager e Human Task Manager e forniscono etichette e convertitori assunti per le loro proprietà. Questo insieme di interfacce fornisce etichette sensibili locali e utilità di conversione per una serie di proprietà. La seguente tabella illustra la associazione delle interfacce di Business Process Choreographer agli oggetti del modello client corrispondenti.

Tabella 34. Modo in cui le interfacce di Business Process Choreographer sono associate agli oggetti del modello di client

| Interfaccia di Business Process Choreographer            | Classe dell'oggetto di modello del client                             |
|----------------------------------------------------------|-----------------------------------------------------------------------|
| <code>com.ibm.bpe.api.ActivityInstanceData</code>        | <code>com.ibm.bpe.clientmodel.bean.ActivityInstanceBean</code>        |
| <code>com.ibm.bpe.api.ActivityServiceTemplateData</code> | <code>com.ibm.bpe.clientmodel.bean.ActivityServiceTemplateBean</code> |
| <code>com.ibm.bpe.api.ProcessInstanceData</code>         | <code>com.ibm.bpe.clientmodel.bean.ProcessInstanceBean</code>         |
| <code>com.ibm.bpe.api.ProcessTemplateData</code>         | <code>com.ibm.bpe.clientmodel.bean.ProcessTemplateBean</code>         |
| <code>com.ibm.task.api.Escalation</code>                 | <code>com.ibm.task.clientmodel.bean.EscalationBean</code>             |
| <code>com.ibm.task.api.Task</code>                       | <code>com.ibm.task.clientmodel.bean.TaskInstanceBean</code>           |
| <code>com.ibm.task.api.TaskTemplate</code>               | <code>com.ibm.task.clientmodel.bean.TaskTemplateBean</code>           |

## Aggiunta del componente Elenco ad un'applicazione JSF

Utilizzare il componente Elenco di Business Process Choreographer Explorer per visualizzare un elenco di oggetti modello del client, ad esempio, istanze di processi di business o istanze di attività.

### Procedure

1. Aggiungere il componente Elenco al file JSP (JavaServer Pages).

Aggiungere il tag `bpe:list` al tag `h:form`. Il tag `bpe:list` deve comprendere un attributo di modello. Aggiungere i tag `bpe:column` al tag `bpe:list` per aggiungere le proprietà degli oggetti che devono essere visualizzate in ciascuna delle righe nell'elenco.

L'esempio di seguito riportato illustra il modo in cui aggiungere un componente Elenco per visualizzare le istanze dell'attività.

```
<h:form>
```

```
 <bpe:list model="#{TaskPool}">
 <bpe:column name="name" action="taskInstanceDetails" />
 <bpe:column name="state" />
 <bpe:column name="kind" />
 <bpe:column name="owner" />
 <bpe:column name="originator" />
 </bpe:list>
```

```
</h:form>
```

L'attributo del modello fa riferimento ad un bean gestito, `TaskPool`. Il bean gestito fornisce l'elenco di oggetti Java sui quali viene iterato l'elenco ed in seguito viene visualizzato in singole righe.

2. Configurare il bean gestito di riferimento nel tag `bpe:list`.

Per il componente Elenco, questo bean gestito deve essere un'istanza della classe `com.ibm.bpe.jsf.handler.BPCListHandler`.

L'esempio di seguito riportato illustra il modo in cui aggiungere il bean gestito `TaskPool` al file di configurazione.

```
<managed-bean>
<managed-bean-name>TaskPool</managed-bean-name>
<managed-bean-class>com.ibm.bpe.jsf.handler.BPCListHandler</managed-bean-class>
<managed-bean-scope>session</managed-bean-scope>
 <managed-property>
 <property-name>query</property-name>
 <value>#{TaskPoolQuery}</value>
 </managed-property>
 <managed-property>
 <property-name>type</property-name>
 <value>com.ibm.task.clientmodel.bean.TaskInstanceBean</value>
 </managed-property>
</managed-bean>

<managed-bean>
<managed-bean-name>TaskPoolQuery</managed-bean-name>
<managed-bean-class>sample.TaskPoolQuery</managed-bean-class>
<managed-bean-scope>session</managed-bean-scope>
 <managed-property>
 <property-name>type</property-name>
 <value>com.ibm.task.clientmodel.bean.TaskInstanceBean</value>
 </managed-property>
</managed-bean>

<managed-bean>
<managed-bean-name>htmConnection</managed-bean-name>
<managed-bean-class>com.ibm.task.clientmodel.HTMConnection</managed-bean-class>
<managed-bean-scope>application</managed-bean-scope>
```

```

 <managed-property>
 <property-name>jndiName</property-name>
 <value>java:comp/env/ejb/LocalHumanTaskManagerEJB</value>
 </managed-property>
 </managed-bean>

```

L'esempio illustra che TaskPool dispone di due proprietà configurabili: query e tipo. Il valore della proprietà di query fa riferimento ad un altro bean gestito, TaskPoolQuery. Il valore della proprietà del tipo specifica la classe del bean, le cui proprietà vengono illustrate nella colonna dell'elenco visualizzato. L'istanza della query associata può disporre anche di un tipo di proprietà. Se è specificato un tipo di proprietà, deve essere uguale al tipo specificato per il gestore eventi.

È possibile aggiungere qualsiasi tipo di logica di query all'applicazione JSF fintanto che il risultato della query può essere rappresentato come un elenco di bean strongly-typed. Per esempio, TaskPoolQuery viene implementata mediante un elenco di oggetti com.ibm.task.clientmodel.bean.TaskInstanceBean.

### 3. Aggiungere il codice personalizzato per il bean gestito riferito al gestore elenco.

L'esempio di seguito riportato illustra il modo in cui aggiungere il codice personalizzato per il bean gestito TaskPool.

```

public class TaskPoolQuery implements Query {

 public List execute throws ClientException {

 // Esaminare il file faces-config per un bean gestito "htmConnection".
 //
 FacesContext ctx = FacesContext.getCurrentInstance();
 Application app = ctx.getApplication();
 ValueBinding htmVb = app.createValueBinding("#{htmConnection}");
 htmConnection = (HTMConnection) htmVb.getValue(ctx);
 HumanTaskManagerService taskService =
 htmConnection.getHumanTaskManagerService();

 // Quindi richiamare il metodo della query corrente sul servizio Human Task Manager.
 //
 QueryResultSet queryResult = taskService.query(
 "DISTINCT TASK.TKIID, TASK.NAME, TASK.KIND, TASK.STATE, TASK.TYPE,"
 + "TASK.STARTED, TASK.ACTIVATED, TASK.DUE, TASK.EXPIRES, TASK.PRIORITY" ,
 "TASK.KIND IN(101,102,105) AND TASK.STATE IN(2)
 AND WORK_ITEM.REASON IN (1)",
 (String)null,
 (Integer)null,
 (TimeZone)null);
 List applicationObjects = transformToTaskList (queryResult);
 return applicationObjects ;
 }

 private List transformToTaskList(QueryResultSet result) {

 ArrayList array = null;
 int entries = result.size();
 array = new ArrayList(entries);

 // Trasforma ciascuna riga di QueryResultSet in bean dell'istanza di un'attività.
 for (int i = 0; i < entries; i++) {
 result.next();
 array.add(new TaskInstanceBean(result, connection));
 }
 return array ;
 }
}

```

Il bean `TaskPoolQuery` esegue una query delle proprietà degli oggetti Java. Questo bean deve implementare l'interfaccia `com.ibm.bpc.clientcore.Query`. Quando il gestore elenco aggiorna il relativo contenuto, richiama il metodo `execute` della query. Il richiamo restituisce un elenco di oggetti Java. Il metodo `getType` deve restituire il nome della classe degli oggetti Java restituiti.

## Results

L'applicazione JSF contiene ora una pagina JavaServer che visualizza le proprietà dell'elenco di oggetti richiesti, ad esempio, lo stato, il tipo, il proprietario e l'originatore delle istanze di attività disponibili.

### Concetti correlati

"Informazioni sul fuso orario specifiche per l'utente" a pagina 155  
I componenti JavaServer Faces (JSF) forniscono un'utilità per gestire informazioni sul fuso orario specifiche per l'utente nel componente Elenco.

### Riferimenti correlati

"Componente Elenco: definizioni di tag" a pagina 157  
Il componente Elenco di Business Process Choreographer Explorer visualizza un elenco di oggetti in una tabella, ad esempio attività, istanze di processo, modelli di processo, elementi di lavoro ed escalation.

## Come vengono elaborati gli elenchi

Ciascuna istanza del componente Elenco è associata ad un'istanza della classe `com.ibm.bpe.jsf.handler.BPCListHandler`.

Questo gestore elenchi tiene traccia degli elementi selezionati nell'elenco associato e fornisce un meccanismo di notifica per associare le voci dell'elenco alle pagine dei dettagli per i differenti tipi di elementi. Il gestore elenchi è associato al componente Elenco mediante l'attributo **model** del tag `bpe:list`.

Il meccanismo di notifica del gestore elenchi viene implementato utilizzando l'interfaccia `com.ibm.bpe.jsf.handler.ItemListener`. È possibile registrare le implementazioni di questa interfaccia nel file di configurazione dell'applicazione JSF (JavaServer Faces).

La notifica è avviata quando si fa clic su un link nell'elenco. I link vengono resi per tutte le colonne per cui è impostato l'attributo **action**. Il valore dell'attributo **azione** è una destinazione di navigazione JSF, o un metodo di azione JSF che restituisce una destinazione di navigazione JSF.

La classe `BPCListHandler` fornisce anche un metodo `refreshList`. È possibile utilizzare questo metodo nei bind del metodo JSF per implementare un controllo dell'interfaccia utente per eseguire di nuovo la query.

## Implementazioni della query

È possibile utilizzare il gestore elenchi per visualizzare tutti i tipi di oggetti e le relative proprietà. Il contenuto dell'elenco che viene visualizzato dipende dall'elenco degli oggetti che viene restituito dall'implementazione dell'interfaccia `com.ibm.bpc.clientcore.Query` configurata per il gestore elenco. È possibile impostare la query in modo automatico utilizzando il metodo `setQuery` della classe `BPCListHandler` oppure è possibile configurarla nei file di configurazione JSF dell'applicazione.

È possibile eseguire le query non solo con le API di Business Process Choreographer, ma anche con altre fonti di informazioni accessibili dall'applicazione, ad esempio, un sistema di gestione del contenuto o un database. Il solo requisito è che il risultato della query viene restituito come `java.util.List` degli oggetti dal metodo `execute`.

Il tipo di oggetti restituiti deve garantire che siano disponibili i metodi `getter` appropriati per tutte le proprietà visualizzate nelle colonne dell'elenco per cui è definita la query. Per assicurare che il tipo di oggetto restituito corrisponda alle definizioni dell'elenco, è possibile impostare il valore della proprietà del tipo nell'istanza `BPCListHandler` definita nel file di configurazione `faces` con il nome classe completo degli oggetti restituiti. È possibile restituire questo nome nella chiamata `getType` dell'implementazione della query. Al runtime, il gestore elenchi verifica che i tipi di oggetto sono conformi alle definizioni.

Per associare i messaggi di errore a voci specifiche in un elenco, gli oggetti restituiti dalla query devono implementare un metodo con la firma `public Object getID()`.

### Convertitori ed etichette assunti

Gli elementi restituiti da una query devono essere bean e la loro classe deve corrispondere alla classe specificata come tipo nella definizione della classe `BPCListHandler` o interfaccia `com.ibm.bpc.clientcore.Query`. Inoltre, il componente Elenco controlla se la classe di elementi o una sovraclassa realizza i seguenti metodi:

```
static public String getLabel(String property,Locale locale);
static public com.ibm.bpc.clientcore.converter.SimpleConverter
 getConverter(String property);
```

Se questi metodi sono definiti per i bean, il componente Elenco utilizza l'etichetta come etichetta assunta per l'elenco e il `SimpleConverter` come convertitore assunto per la proprietà. È possibile sovrascrivere queste impostazioni con gli attributi **etichetta** e **converterID** del tag `bpe:list`. Per ulteriori informazioni, cfr. Javadoc per l'interfaccia `SimpleConverter` e la classe `ColumnTag`.

### Informazioni sul fuso orario specifiche per l'utente

I componenti `JavaServer Faces (JSF)` forniscono un'utilità per gestire informazioni sul fuso orario specifiche per l'utente nel componente Elenco.

La classe `BPCListHandler` utilizza l'interfaccia `com.ibm.bpc.clientcore.util.User` per ottenere informazioni sul fuso orario e sulle impostazioni internazionali per ciascun utente. Il componente Elenco prevede l'implementazione dell'interfaccia da configurare con l'**utente** come nome del bean gestito nel file di configurazione `JSF (JavaServer Faces)`. Se questa voce manca dal file di configurazione, viene restituito il fuso orario in cui `WebSphere Process Server` è in esecuzione.

L'interfaccia `com.ibm.bpc.clientcore.util.User` viene definita come segue:

```
public interface User {

 /**
 * Le impostazioni internazionali utilizzate dal client dell'utente.
 * @return Locale.
 */
 public Locale getLocale();
 /**
 * Il fuso orario utilizzato dal client dell'utente.
 * @return TimeZone.
```

```

 */
 public TimeZone getTimeZone();

 /**
 * Il nome dell'utente.
 * @return name of the user.
 */
 public String getName();
}

```

## Errore di gestione del componente Elenco

Quando si utilizza il componente Elenco per visualizzare elenchi nell'applicazione JSF, è possibile approfittare delle funzioni di gestione errori fornite dalla classe `com.ibm.bpe.jsf.handler.BPCListHandler`.

## Errori che si verificano quando vengono eseguite le query o i comandi

Se si verifica un errore durante l'esecuzione di una query, la classe `BPCListHandler` distingue tra gli errori causati dai privilegi di accesso insufficienti ed altre eccezioni. Per rilevare gli errori dovuti a privilegi di accesso insufficienti, il parametro **rootCause** di `ClientException` restituito dal metodo `execute` della query deve essere un'eccezione `com.ibm.bpe.api.EngineNotAuthorizedException` o `com.ibm.task.api.NotAuthorizedException`. Il componente Elenco visualizza il messaggio di errore invece del risultato della query.

Se l'errore non è causato da privilegi di accesso insufficienti, la classe `BPCListHandler` inoltra l'oggetto dell'eccezione all'implementazione dell'interfaccia `com.ibm.bpc.clientcore.util.ErrorBean` definita dalla chiave `BPCError` nel file di configurazione dell'applicazione JSF. Quando viene impostata l'eccezione, viene richiamata la destinazione di navigazione dell'errore.

## Errori che si verificano quando si effettuano attività con gli elementi visualizzati in un elenco

La classe `BPCListHandler` implementa l'interfaccia `com.ibm.bpe.jsf.handler.ErrorHandler`. Pertanto, possono essere fornite informazioni su tali errori con il parametro di associazione di tipo `java.util.Map` nel metodo `setErrors`. Questa associazione contiene gli identificativi come chiavi e le eccezioni come valori. Gli identificativi devono essere valori restituiti dal metodo `getID` dell'oggetto che ha causato l'errore. Se l'associazione è impostata e ogni ID corrisponde a ciascun elemento visualizzato nell'elenco, il gestore elenchi aggiunge automaticamente all'elenco una colonna contenente il messaggio di errore.

Per evitare i messaggi di errore obsoleti nell'elenco, azzerare l'associazione degli errori. Nelle seguenti situazioni, l'associazione viene azzerata automaticamente:

- Viene richiamata la classe `BPCListHandler` del metodo `refreshList`.
- Viene impostata una nuova query nella classe `BPCListHandler`.
- Il componente `CommandBar` è utilizzato per eseguire il trigger delle azioni sugli elementi dell'elenco. Il componente `CommandBar` utilizza questo meccanismo come uno dei metodi per la gestione degli errori.

### Concetti correlati

“Errore di gestione dei componenti JSF” a pagina 150

I componenti JSF (JavaServer Faces) sfruttano un bean associato predefinito, `BPCError`, per la gestione degli errori. In situazioni di errore che eseguono il trigger della pagina di errore, l'eccezione viene impostata sul bean di errore.

## Componente Elenco: definizioni di tag

Il componente Elenco di Business Process Choreographer Explorer visualizza un elenco di oggetti in una tabella, ad esempio attività, istanze di processo, modelli di processo, elementi di lavoro ed escalation.

Il componente Elenco è costituito da tag del componente JSF: `bpe:list` e `bpe:column`. Il tag `bpe:column` è un elemento secondario del tag `bpe:list`.

### Classe del componente

`com.ibm.bpe.jsf.component.ListComponent`

### Sintassi di esempio

```
<bpe:list model="#{ProcessTemplateList}">
 rows="20"
 styleClass="list"
 headerStyleClass="listHeader"
 rowClasses="normal">

 <bpe:column name="name" action="processTemplateDetails"/>
 <bpe:column name="validFromTime"/>
 <bpe:column name="executionMode" label="Execution mode"/>
 <bpe:column name="state" converterID="my.state.converter"/>
 <bpe:column name="autoDelete"/>
 <bpe:column name="description"/>

</bpe:list>
```

### Attributi del tag

Il corpo del tag `bpe:list` può contenere solo tag `bpe:column`. Al rendering della tabella, il componente Elenco itera l'elenco di oggetti applicativi ed esegue il rendering di tutte le colonne per ciascuno degli oggetti.

Tabella 35. Attributi `bpe:list`

Attributo	Richiesto	Descrizione
<code>buttonStyleClass</code>	no	La classe di stile CSS (cascading style sheet) per il rendering dei pulsanti nell'area delle note a piè di pagina.
<code>cellStyleClass</code>	no	La classe di stile CSS per visualizzare le singole celle della tabella.
<code>checkbox</code>	no	Determina se visualizzare o meno la casella di spunta per la selezione di più voci. L'attributo presenta un valore <code>true</code> o <code>false</code> . Se il valore è impostato su <code>true</code> , viene eseguito il rendering della colonna della casella di spunta.
<code>headerStyleClass</code>	no	La classe di stile CSS per visualizzare l'intestazione della tabella.
<code>model</code>	sì	Un bind del valore per un bean gestito della classe <code>com.ibm.bpe.jsf.handler.BPCListHandler</code> .



Tabella 35. Attributi `bpe:list` (Continua)

Attributo	Richiesto	Descrizione
<code>rows</code>	no	Il numero di righe visualizzate in una pagina. Se il numero di elementi supera il numero di righe, i pulsanti di paginazione vengono visualizzati alla fine della tabella. Le espressioni di valore non sono supportate per questo attributo.
<code>rowClasses</code>	no	La classe di stile CSS per visualizzare le righe della tabella.
<code>selectAll</code>	no	Se questo attributo è impostato su <code>true</code> , tutte le voci dell'elenco sono selezionate per assunto.
<code>styleClass</code>	no	La classe di stile CSS per il rendering della tabella contenente titoli, righe e pulsanti di paginazione.

Tabella 36. Attributi `bpe:column`

Attributo	Richiesto	Descrizione
<code>action</code>	no	Se questo attributo viene specificato, viene eseguito il rendering di un link nella colonna. Quando si fa clic sul link, viene lanciato un metodo di azione JavaServer Faces o la destinazione di navigazione the Faces. Un metodo di azione JavaServer Faces ha la seguente firma: <code>String method()</code> .
<code>converterID</code>	no	L'ID del convertitore Faces usato per convertire il valore della proprietà. Se questo attributo non è impostato, viene usato qualsiasi ID del convertitore Faces fornito dal modello per questa proprietà.
<code>label</code>	no	Un'espressione letterale o di bind del valore usata come etichetta per l'intestazione della colonna o della cella della riga dell'intestazione della tabella. Se questo attributo non è impostato, viene usata un'etichetta fornita dal modello per questa proprietà.
<code>name</code>	sì	Il nome della proprietà visualizzato nella colonna.

## Aggiunta del componente Dettagli ad un'applicazione JSF

Utilizzare il componente Dettagli di Business Process Choreographer Explorer per visualizzare le proprietà di attività, elementi di lavoro, attività, istanze di processo e modelli di processo.

### Procedure

1. Aggiungere il componente Dettagli al file JSP (JavaServer Pages).  
 Aggiungere il tag `bpe:details` al tag `<h:form>`. Il tag `bpe:details` deve contenere un attributo di **modello**. È possibile aggiungere le proprietà al componente Dettagli con il tag `bpe:property`.



L'esempio di seguito riportato illustra il modo in cui aggiungere un componente Dettagli per visualizzare alcune delle proprietà per un'istanza dell'attività.

```
<h:form>

 <bpe:details model="#{TaskInstanceDetails}">
 <bpe:property name="displayName" />
 <bpe:property name="owner" />
 <bpe:property name="kind" />
 <bpe:property name="state" />
 <bpe:property name="escalated" />
 <bpe:property name="suspended" />
 <bpe:property name="originator" />
 <bpe:property name="activationTime" />
 <bpe:property name="expirationTime" />
 </bpe:details>

</h:form>
```

L'attributo di **modello** si riferisce a un bean gestito, TaskInstanceDetails. Il bean fornisce le proprietà dell'oggetto Java.

## 2. Configurare il bean gestito riferito al tag bpe:details.

Per il componente Dettagli, questo bean gestito deve essere un'istanza della classe com.ibm.bpe.jsf.handler.BPCDetailsHandler. Questa classe del gestore avviluppa un oggetto Java e ne visualizza le proprietà pubbliche al componente dei dettagli.

L'esempio di seguito riportato illustra il modo in cui aggiungere il bean gestito TaskInstanceDetails al file di configurazione.

```
<managed-bean>
 <managed-bean-name>TaskInstanceDetails</managed-bean-name>
 <managed-bean-class>com.ibm.bpe.jsf.handler.BPCDetailsHandler</managed-bean-class>
 <managed-bean-scope>session</managed-bean-scope>
 <managed-property>
 <property-name>type</property-name>
 <value>com.ibm.task.clientmodel.bean.TaskInstanceBean</value>
 </managed-property>
</managed-bean>
```

L'esempio illustra che il bean TaskInstanceDetails dispone di una proprietà type configurabile. Il valore della proprietà type specifica la classe del bean (com.ibm.task.clientmodel.bean.TaskInstanceBean), le cui proprietà sono illustrate nelle righe dei dettagli visualizzati. La classe di bean può essere qualsiasi classe JavaBeans. Se il bean fornisce convertitori ed etichette di proprietà assunti, convertitori ed etichette sono utilizzati per il rendering allo stesso modo del componente Elenco.

## Results

L'applicazione JSF contiene una pagina JSP (JavaServer page) che visualizza i dettagli dell'oggetto specificato, ad esempio i dettagli di un'istanza dell'attività.

### Riferimenti correlati

“Componente Dettagli: definizioni di tag”

Il componenteDettagli di Business Process Choreographer Explorer visualizza le proprietà di attività, elementi di lavoro, attività, istanze di processo e modelli di processo.

## Componente Dettagli: definizioni di tag

Il componenteDettagli di Business Process Choreographer Explorer visualizza le proprietà di attività, elementi di lavoro, attività, istanze di processo e modelli di processo.

Il componente Dettagli è costituito dai tag del componente JSF: `bpe:details` e `bpe:property`. Il tag `bpe:property` è un elemento secondario del tag `bpe:details`.

## Classe del componente

`com.ibm.bpe.jsf.component.DetailsComponent`

## Sintassi di esempio

```
<bpe:details model="#{MyActivityDetails}">
 <bpe:property name="name"/>
 <bpe:property name="owner"/>
 <bpe:property name="activated"/>
</bpe:details>

<bpe:details model="#{MyActivityDetails}" style="style" styleClass="cssStyle">
 style="style"
 styleClass="cssStyle"
</bpe:details>
```

## Attributi del tag

Utilizzare i tag `bpe:property` per specificare la serie secondaria di attributi illustrati ed il relativo ordine di visualizzazione. Se il tag dei dettagli non contiene alcun tag di attributo, vengono visualizzati tutti gli attributi disponibili nell'oggetto del modello.

Tabella 37. Attributi `bpe:details`

Attributo	Richiesto	Descrizione
<code>columnClasses</code>	no	Un elenco di classi di stile CSS (cascading style sheet), separate da virgola, per il rendering di colonne.
<code>id</code>	no	L'ID JavaServer Faces del componente.
<code>model</code>	sì	Un bind del valore per un bean gestito della classe <code>com.ibm.bpe.jsf.handler.BPCDetailsHandler</code> .
<code>rowClasses</code>	no	Un elenco di classi di stile CSS, separati da virgola, per il rendering delle righe.
<code>styleClass</code>	no	La classe CSS usata per il rendering dell'elemento HTML.

Tabella 38. Attributi `bpe:property`

Attributo	Richiesto	Descrizione
<code>converterID</code>	no	L'ID utilizzato per registrare l'utilità di conversione nel file di configurazione JSF (JavaServer Faces).
<code>label</code>	no	L'etichetta della proprietà. Se questo attributo non è impostato, viene fornita un'etichetta predefinita dalla classe di modello del client.
<code>name</code>	sì	Il nome della proprietà da visualizzare. Questo nome deve corrispondere ad una proprietà denominata come definita nella classe del modello del client corrispondente.

## Aggiunta del componente CommandBar ad un'applicazione JSF

Utilizzare il componente CommandBar di Business Process Choreographer Explorer per visualizzare una barra con i pulsanti. Questi pulsanti rappresentano i comandi che funzionano sulla vista dettagli di un oggetto o degli oggetti selezionati in un elenco.

### About this task

Facendo clic su un pulsante nell'interfaccia utente, il comando corrispondente viene eseguito sugli oggetti selezionati. È possibile aggiungere ed estendere il componente CommandBar nell'applicazione JSF di cui si dispone.

### Procedure

1. Aggiungere il componente CommandBar al file JSP (JavaServer Pages).

Aggiungere il tag `bpe:commandbar` al tag `<h:form>`. Il tag `bpe:commandbar` deve contenere un attributo di modello.

L'esempio di seguito riportato illustra il modo in cui aggiungere un componente CommandBar che fornisce comandi di aggiornamento e richiesta per un elenco di istanze dell'attività.

```
<h:form>

 <bpe:commandbar model="#{TaskInstanceList}">
 <bpe:command commandID="Refresh" >
 action="#{TaskInstanceList.refreshList}"
 label="Refresh"/>

 <bpe:command commandID="MyClaimCommand" >
 label="Claim" >
 commandClass="<customcode>" />
 </bpe:commandbar>

</h:form>
```

L'attributo **model** fa riferimento ad un bean gestito. Questo bean deve implementare l'interfaccia `ItemProvider` e definire gli oggetti Java selezionati. Il comando `CommandBar` viene generalmente utilizzato con il componente `Elenco` o il componente `Dettagli` nello stesso file JSP. In genere, il modello specificato nel tag è lo stesso del modello specificato nel componente `Elenco` o `Dettagli` nella stessa pagina. Pertanto, per un componente `Elenco`, ad esempio il comando agisce sulle voci selezionate nell'elenco.

In questo esempio, l'attributo **model** fa riferimento al bean gestito `TaskInstanceList`. Questo bean fornisce gli oggetti selezionati nell'elenco di istanze dell'attività. Il bean deve implementare l'interfaccia `ItemProvider`. Questa interfaccia viene implementata dalle classi `BPCListHandler` e `BPCDetailsHandler`.

2. Opzionale: Configurare il bean gestito che fa riferimento al tag `bpe:commandbar`. Se l'attributo **model** di `CommandBar` fa riferimento ad un bean gestito che è già configurato, ad esempio, per un gestore elenco o dettagli, non viene richiesta un'ulteriore configurazione. Se si modifica la configurazione di tali gestori oppure si utilizza un bean gestito diverso, aggiungere un bean gestito che implementa l'interfaccia `ItemProvider` per il file di configurazione JSF.
3. Aggiungere il codice che implementa i comandi personalizzati per l'applicazione JSF.

Il seguente frammento di codice illustra come scrivere una classe di comandi che implementa l'interfaccia `Command`. Questa classe di comandi (`MyClaimCommand`) fa riferimento al tag `bpe:command` nel file JSP.

```
public class MyClaimCommand implements Command {

 public String execute(List selectedObjects) throws ClientException {
 if(selectedObjects != null && selectedObjects.size() > 0) {
 try {
 // Determinare HumanTaskManagerService da un bean HTMConnection.
 // Configurare il bean in faces-config.xml per un facile accesso
 // all'applicazione JSF.
 FacesContext ctx = FacesContext.getCurrentInstance();
 ValueBinding vb =
 ctx.getApplication().createValueBinding("{htmConnection}");
 HTMConnection htmConnection = (HTMConnection) htmVB.getValue(ctx);
 HumanTaskManagerService htm =
 htmConnection.getHumanTaskManagerService();

 Iterator iter = selectedObjects.iterator() ;
 while(iter.hasNext()) {
 try {
 TaskInstanceBean task = (TaskInstanceBean) iter.next() ;
 TKIID tiid = task.getID() ;

 htm.claim(tiid) ;
 task.setState(new Integer(TaskInstanceBean.STATE_CLAIMED)) ;

 }
 catch(Exception e) {
 ; // Errore durante l'iterazione o la richiesta dell'istanza di attività.
 // Ignorare per una migliore comprensione dell'esempio.
 }
 }
 }
 catch(Exception e) {
 ; // Errore di configurazione o di comunicazione.
 // Ignorare per una migliore comprensione dell'esempio
 }
 }
 return null;
 }

 // Implementazioni predefinite
 public boolean isMultiSelectEnabled() { return false; }
 public boolean[] isApplicable(List itemsOnList) {return null; }
 public void setContext(Object targetModel) {}; // Not used here
}
```

Il comando viene elaborato come segue:

- a. Un comando viene richiamato quando un utente fa clic sul pulsante corrispondente nella barra dei comandi. Il componente `CommandBar` richiama gli elementi selezionati dal provider dell'elemento specificato nell'attributo **model** e inoltra l'elenco degli oggetti selezionati al metodo `execute` dell'istanza `commandClass`.
- b. L'attributo **commandClass** fa riferimento all'implementazione di un comando personalizzato che implementa l'interfaccia dei comandi. Il comando deve implementare il metodo `public String execute(List selectedObjects) throws ClientException`. Il comando restituisce un output utilizzato per determinare la regola di navigazione successiva per l'applicazione JSF.
- c. Una volta completato il comando, il componente `CommandBar` valuta l'attributo **action**. L'attributo **action** può essere una stringa statica o un bind del metodo per un metodo dell'azione JSF con la firma `public String`

Method(). Utilizzare l'attributo **action** per sostituire l'output di una classe di comandi o specificare esplicitamente un output per le regole di navigazione. L'attributo **action** non viene elaborato se il comando genera un'eccezione diversa da `ErrorsInCommandException`.

- d. Se l'attributo **commandClass** non ha una classe di comandi specificata, l'azione viene invocata immediatamente. Per esempio, per il comando aggiorna nell'esempio, invece di un comando viene invocata l'espressione del valore JSF `#{TaskInstanceList.refreshList}`.

## Results

L'applicazione JSF ora contiene una JavaServer page che implementa una barra dei comandi personalizzata.

### Riferimenti correlati

"Componente CommandBar: definizioni di tag"

Il componente CommandBar di Business Process Choreographer Explorer visualizza una barra con i pulsanti. Questi pulsanti funzionano sull'oggetto in una vista dettagli o sugli oggetti selezionati in un elenco.

## Come vengono elaborati i comandi

Utilizzare il componente CommandBar per aggiungere pulsanti di azione all'applicazione. Il componente crea i pulsanti per le azioni nell'interfaccia utente e gestisce gli eventi che vengono creati quando si fa clic su un pulsante.

Questi pulsanti attivano funzioni che agiscono sugli oggetti restituiti da un'interfaccia `com.ibm.bpe.jsf.handler.ItemProvider`, ad esempio la classe `BPCListHandler` o la classe `BPCDetailsHandler`. Il componente CommandBar utilizza il provider di elementi definito dal valore dell'attributo **model** nel tag `bpe:commandbar`.

Quando si fa clic su un pulsante nella sezione della barra dei comandi dell'interfaccia utente dell'applicazione, l'evento associato viene gestito dal componente CommandBar nel modo seguente.

1. Il componente CommandBar identifica l'implementazione dell'interfaccia `com.ibm.bpc.clientcore.Command` specificata per il pulsante che ha generato l'evento.
2. Se il modello associato al componente CommandBar implementa l'interfaccia `com.ibm.bpe.jsf.handler.ErrorHandler`, viene richiamato il metodo `clearErrorMap` per rimuovere i messaggi di errore dagli eventi precedenti.
3. Viene richiamato il metodo `getSelectedItems` dell'interfaccia `ItemProvider`. L'elenco di elementi che viene restituito è inoltrato al metodo `execute` del comando che viene a sua volta richiamato.
4. Il componente CommandBar determina la destinazione di navigazione JSF (JavaServer Faces). Se l'attributo **action** non è specificato nel tag `bpe:commandbar`, il valore restituito del metodo `execute` specifica la destinazione di navigazione. Se l'attributo **action** è impostato su un bind del metodo JSF, la stringa restituita dal metodo è interpretata come destinazione di navigazione. L'attributo **action** può, inoltre, specificare una destinazione di navigazione esplicita.

## Componente CommandBar: definizioni di tag

Il componente CommandBar di Business Process Choreographer Explorer visualizza una barra con i pulsanti. Questi pulsanti funzionano sull'oggetto in una vista dettagli o sugli oggetti selezionati in un elenco.

Il componente CommandBar è costituito da tag del componente JSF: `bpe:command` e `bpe:commandbar`. Il tag `bpe:command` è un elemento secondario del tag `bpe:commandbar`.

## Classe del componente

`com.ibm.bpe.jsf.component.CommandBarComponent`

## Sintassi di esempio

```
<bpe:commandbar model="#{TaskInstanceList}">

 <bpe:command
 commandID="Work on"
 label="Work on..."
 commandClass="com.ibm.bpc.explorer.command.WorkOnTaskCommand"
 context="#{TaskInstanceDetailsBean}" />

 <bpe:command
 commandID="Cancel"
 label="Cancel"
 commandClass="com.ibm.task.clientmodel.command.CancelClaimTaskCommand"
 context="#{TaskInstanceList}" />

</bpe:commandbar>
```

## Attributi del tag

Tabella 39. Attributi `bpe:commandbar`

Attributo	Richiesto	Descrizione
<code>buttonStyleClass</code>	no	La classe di stile CSS (cascading style sheet) usata per il rendering dei pulsanti nella barra dei comandi.
<code>id</code>	no	L'ID JavaServer Faces del componente.
<code>model</code>	sì	Un'espressione di bind del valore per un bean gestito che implementa l'interfaccia <code>ItemProvider</code> . Questo bean gestito è in genere la classe <code>com.ibm.bpe.jsf.handler.BPCListHandler</code> o <code>com.ibm.bpe.jsf.handler.BPCDetailsHandler</code> utilizzata dal componente Elenco o Dettagli nello stesso file JSP (JavaServer Pages) del componente <code>CommandBar</code> .
<code>styleClass</code>	no	La classe di stile CSS usata per il rendering della barra dei comandi.

Tabella 40. Attributi `bpe:command`

Attributo	Richiesto	Descrizione
<code>action</code>	no	Un metodo di azione JavaServer Faces o la destinazione di navigazione JavaServer Faces che deve essere lanciato dal pulsante dei comandi. La destinazione di navigazione restituita dall'azione sovrascrive tutte le altre regole di navigazione. L'azione viene chiamata quando non viene lanciata un'eccezione o quando viene lanciata un'eccezione <code>ErrorsInCommandException</code> dal comando.

Tabella 40. Attributi `bpe:command` (Continua)

Attributo	Richiesto	Descrizione
<code>commandClass</code>	no	Il nome della classe di comandi. Se si seleziona il pulsante dei comandi, un'istanza della classe viene creata dal componente <code>CommandBar</code> ed eseguita.
<code>commandID</code>	sì	L'ID del comando.
<code>context</code>	no	Un oggetto che fornisce contesto per comandi specificati mediante l'attributo <code>commandClass</code> . L'oggetto di contesto viene recuperato quando si accede per la prima volta alla barra dei comandi.
<code>immediato</code>	no	Specifica quando viene lanciato il comando. Se il valore di questo attributo è <code>true</code> , il comando è lanciato prima che venga elaborata l'input della pagina. Il valore assunto è <code>false</code> .
<code>label</code>	sì	L'etichetta del pulsante che viene visualizzato nella barra dei comandi.
<code>rendering</code>	no	Determina se un pulsante è soggetto a rendering. Il valore dell'attributo può essere un valore Booleano o un'espressione di valore.
<code>styleClass</code>	no	La classe di stile CSS usata per il rendering del pulsante. Questo stile sostituisce lo stile del pulsante definito per la barra dei comandi.

## Aggiunta del componente Messaggio ad un'applicazione JSF

Utilizzare il componente Messaggio di Business Process Choreographer Explorer per eseguire il rendering degli oggetti di dati e dei tipi primitivi in un'applicazione JSF (JavaServer Faces).

### About this task

Se il messaggio è di tipo primitivo, vengono visualizzati un'etichetta e un campo di input. Se il tipo di messaggio è un oggetto dati, il componente attraversa l'oggetto e visualizza gli elementi all'interno di esso.

### Procedure

1. Aggiungere il componente Messaggio al file JSP (JavaServer Pages).

Aggiunge il tag `bpe:form` al tag `<h:form>`. Il tag `bpe:form` deve comprendere un attributo `model`.

L'esempio di seguito riportato illustra come aggiungere un componente Messaggio.

```
<h:form>

 <h:outputText value="Input Message" />
 <bpe:form model="#{MyHandler.inputMessage}" readOnly="true" />

 <h:outputText value="Output Message" />
 <bpe:form model="#{MyHandler.outputMessage}" />

</h:form>
```

L'attributo **model** del componente Messaggio fa riferimento ad un oggetto `com.ibm.bpc.clientcore.MessageWrapper`. Questo oggetto del wrapper avvolge un oggetto SDO (Service Data Object) o un tipo primitivo Java, ad esempio `int` o `boolean`. In questo esempio, il messaggio viene fornito da una proprietà del bean gestito `MyHandler`.

## 2. Configurare il bean gestito riferito al tag `bpe:form`.

L'esempio di seguito riportato illustra il modo in cui aggiungere il bean gestito `MyHandler` al file di configurazione.

```
<managed-bean>
<managed-bean-name>MyHandler</managed-bean-name>
<managed-bean-class>com.ibm.bpe.sample.jsf.MyHandler</managed-bean-class>
<managed-bean-scope>session</managed-bean-scope>

 <managed-property>
 <property-name>type</property-name>
 <value>com.ibm.task.clientmodel.bean.TaskInstanceBean</value>
 </managed-property>

</managed-bean>
```

## 3. Aggiungere il codice personalizzato all'applicazione JSF.

L'esempio di seguito riportato illustra il modo in cui implementare i messaggi di input o di output.

```
public class MyHandler implements ItemListener {

 private TaskInstanceBean taskBean;
 private MessageWrapper inputMessage, outputMessage

 /* Metodo del listener, ad esempio quando un'istanza di attività era stata
 selezionata in un gestore elenco.
 * Verificare che il gestore sia registrato in faces-config.xml o manualmente.
 */
 public void itemChanged(Object item) {
 if(item instanceof TaskInstanceBean) {
 taskBean = (TaskInstanceBean) item ;
 }
 }

 /* Ottenere il wrapper del messaggio di input
 */
 public MessageWrapper getInputMessage() {
 try{
 inputMessage = taskBean.getInputMessageWrapper() ;
 }
 catch(Exception e) {
 ; //...ignora gli errori per semplicità
 }
 return inputMessage;
 }

 /* Ottenere il wrapper del messaggio di output
 */
 public MessageWrapper getOutputMessage() {
 // Richiamare il messaggio dal bean. Se non è presente alcun messaggio, crearne
 // uno se l'attività è stata richiesta dall'utente. Verificare che solo i
 // potenziali proprietari o proprietari possono modificare il messaggio di output.
 try{
 outputMessage = taskBean.getOutputMessageWrapper();
 if(outputMessage == null
 && taskBean.getState() == TaskInstanceBean.STATE_CLAIMED) {
 HumanTaskManagerService htm = getHumanTaskManagerService();
 outputMessage = new MessageWrapperImpl();
 outputMessage.setMessage(
 htm.createOutputMessage(taskBean.getID()).getObject()
);
 }
 }
 }
}
```



```

);
 }
}
catch(Exception e) {
 ; //...ignora gli errori per semplicità
}
return outputMessage
}
}

```

Il bean gestito MyHandler implementa l'interfaccia `com.ibm.jsf.handler.ItemListener`, in modo che possa registrarsi come un listener dell'elemento ai gestori Elenchi. Se l'utente fa clic su un elemento dell'elenco, viene inviata una notifica al bean MyHandler nel relativo metodo `itemChanged( Object item )` sull'elemento selezionato. Il gestore verifica il tipo di elemento, quindi memorizza un riferimento all'oggetto `TaskInstanceBean` associato. Per utilizzare questa interfaccia, aggiungere una voce all'elenco `itemListener` nel gestore elenco appropriato nel file `faces-config.xml`.

Il bean MyHandler fornisce i metodi `getInputMessage` e `getOutputMessage`. Entrambi i metodi restituiscono un oggetto `MessageWrapper`. I metodi delegano i richiami al bean di istanza dell'attività di riferimento. Se il bean dell'istanza dell'attività restituisce null, ad esempio perché non è impostato un messaggio, il gestore crea e memorizza un nuovo messaggio vuoto. Il componente Messaggio visualizza i messaggi forniti dal bean MyHandler.

## Results

L'applicazione JSF contiene ora una JSP (JavaServer page) che può visualizzare gli oggetti di dati ed i tipi primitivi.

### Riferimenti correlati

“Componente Messaggio: definizioni di tag”

Il componente Messaggio di Business Process Choreographer Explorer visualizza gli oggetti `commonj.sdo.DataObject` ed i tipi primitivi, come ad esempio valori interi e stringhe, in un'applicazione JSF (JavaServer Faces).

## Componente Messaggio: definizioni di tag

Il componente Messaggio di Business Process Choreographer Explorer visualizza gli oggetti `commonj.sdo.DataObject` ed i tipi primitivi, come ad esempio valori interi e stringhe, in un'applicazione JSF (JavaServer Faces).

Il componente Message è costituito dal tag del componente JSF: `bpe:form`.

## Classe del componente

`com.ibm.bpe.jsf.component.MessageComponent`

## Sintassi di esempio

```

<bpe:form model="#{TaskInstanceDetailsBean.inputMessageWrapper}"
 simplification="true" readOnly="true"
 styleClass4table="messageData"
 styleClass4output="messageDataOutput">
</bpe:form>

```

## Attributi del tag

Tabella 41. Attributi bpe:form

Attributo	Richiesto	Descrizione
id	no	L'ID JavaServer Faces del componente.
model	sì	Un'espressione di bind del valore che fa riferimento ad un oggetto <code>commonj.sdo.DataObject</code> o <code>com.ibm.bpc.clientcore.MessageWrapper</code> .
readOnly	no	Se questo attributo viene impostato su <code>true</code> , viene visualizzato un modulo in sola lettura. Per impostazione predefinita, questo attributo è impostato su <code>false</code> .
simplification	no	Se questo attributo è impostato su <code>true</code> , vengono visualizzate le proprietà che contengono tipi semplici e hanno una cardinalità zero o uno. Per assunto, questo attributo è impostato su <code>true</code> .
style4validinput	no	Lo stile CSS (cascading style sheet) per visualizzare gli input validi.
style4invalidinput	no	Lo stile CSS per visualizzare l'input non valido.
styleClass4invalidInput	no	Il nome della classe di stile CSS per il rendering dell'input non è valido.
styleClass4output	no	Il nome della classe dello stile CSS per visualizzare gli elementi di output.
styleClass4table	no	Il nome della classe dello stile di tabella CSS per visualizzare le tabelle del componente del messaggio.
styleClass4validInput	no	Il nome della classe di stile CSS per il rendering dell'input è valido.

---

## Sviluppo di pagine JSP per messaggi di attività e processo

L'interfaccia di Business Process Choreographer Explorer fornisce moduli input e output assunte per visualizzare e immettere dati business. È possibile utilizzare le pagine JSP per fornire moduli input output personalizzati.

### About this task

Per includere le pagine JavaServer Pages (JSP) definite dall'utente nel client Web è necessario specificarle quando si modella una attività umana in WebSphere Integration Developer. Per esempio, è possibile fornire le pagine JSP per un'attività specifica e i relativi messaggi di input e output, e per uno specifico ruolo utente o tutti i ruoli utente. Al runtime, le pagine JSP definite dall'utente sono incluse nell'interfaccia utente per visualizzare dati output e raccogliere dati input.

I moduli personalizzati non sono pagine Web autonome; sono frammenti HTML che Business Process Choreographer Explorer integra in un modulo HTML, per esempio, frammenti per tutte le etichette e campi input di un messaggio.

Quando si fa clic su un pulsante della pagina che contiene i moduli personalizzati, i dati input sono immessi e convalidati in Business Process Choreographer

Explorer. La convalida è basata sul tipo di proprietà fornite e l'impostazione internazionale usata nel browser. Se i dati input non possono essere convalidati, la stessa pagina viene mostrata nuovamente e vengono fornite informazioni relative agli errori di convalida nell'attributo di richiesta `messageValidationErrors`. Le informazioni sono fornite come associazione contenente la XML Path Expression (XPath) delle proprietà che non sono valide per le eccezioni di convalida verificatesi.

Per aggiungere moduli personalizzati a Business Process Choreographer Explorer, effettuare i seguenti passaggi utilizzando WebSphere Integration Developer.

### Procedure

1. Creare i moduli personalizzati.

Le pagine JSP definite dall'utente per i moduli input e output usati nell'interfaccia Web richiedono l'accesso ai dati di messaggio. Usare i codici `snippetJava` in una JSP o il linguaggio di esecuzione JSP per accedere ai dati di messaggio. I dati nei moduli sono accessibili attraverso il contesto di richiesta.

2. Assegnazione delle pagine JSP a un'attività.

Aprire la attività umana nell'editor delle attività umane. Nelle impostazioni client, specificare l'ubicazione delle pagine JSP definite dall'utente e il ruolo a cui il modulo personalizzato fa riferimento, per esempio, amministratore. Le impostazioni client per Business Process Choreographer Explorer sono archiviate nel modello dell'attività. Al runtime queste impostazioni sono richiamate con il modello dell'attività.

3. Packaging le pagine JSP definite dall'utente in un archivio Web (file WAR).

È possibile includere il file WAR nell'enterprise archive con il modulo che contiene le attività o distribuire il file WAR separatamente. Se le JSP sono distribuite separatamente rendere le JSP disponibili sul server dove il Business Process Choreographer Explorer o il client personalizzato è distribuito.

Se si stanno usando JSP personalizzati per i messaggi di processo e di attività, è necessario associare i moduli Web usati per distribuire i JSP agli stessi server a cui viene associato il client JSF personalizzato.

### Results

I moduli personalizzati sono resi in Business Process Choreographer Explorer al runtime.

## Frammenti JSP definiti dall'utente

I frammenti JavaServer Pages (JSP) definiti dall'utente sono inseriti in un tag in formato HTML. Al runtime, Business Process Choreographer Explorer include questi frammenti nella pagina resa.

Il frammento JSP definito dall'utente per il messaggio di input è inserito prima del frammento JSP per il messaggio di output.

```
<html....>
...
<form...>
 Input JSP (visualizza messaggio di input dell'attività)

 Output JSP (visualizza messaggio di output dell'attività)

</form>
...
</html>
```

Poiché i frammenti JSP definiti dall'utente sono inseriti in un tag in formato HTML, è possibile aggiungere elementi di input. Il nome dell'elemento di input deve corrispondere all'espressione XML Path Language (XPath) dell'elemento dati. È importante prefissare il nome dell'elemento di input con il valore di prefisso fornito:

```
<input id="address"
 tipo="text"
 nome="{prefisso}/selectPromotionalGiftResponse/address"
 valore="{messageMap['/selectPromotionalGiftResponse/address']}"
 dimensione="60"
 allineamento="sinistra" />
```

Il valore del prefisso è fornito come attributo a richiesta. L'attributo assicura che il nome di input sia unico nel formato di inclusione. Il prefisso è generato da Business Process Choreographer Explorer e non va cambiato:

```
Prefisso stringa = (Stringa)request.getAttribute("prefisso");
```

L'elemento prefisso è impostato solo se il messaggio può essere modificato nel contesto fornito. È possibile visualizzare i dati di output in modi diversi a seconda dello stato della attività umana. Per esempio, se l'attività si trova nello stato richiesto, i dati di output possono essere modificati. Tuttavia, se l'attività è conclusa, i dati possono solo essere visualizzati. Nel proprio frammento JSP, è possibile verificare se esiste l'elemento prefisso e di conseguenza rendere il messaggio. La seguente istruzione JSTL mostra come verificare se l'elemento prefisso è impostato.

```
...
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
...
<c:choose>
 <c:when test="{not empty prefix}">
 <!--Read/write mode-->
 </c:when>
 <c:otherwise>
 <!--Read-only mode-->
 </c:otherwise>
</c:choose>
```

---

## Creazione di plug-in per personalizzare la funzionalità delle attività umane

Business Process Choreographer fornisce un'infrastruttura di gestione eventi per gli eventi che si verificano durante l'elaborazione delle attività umane. Vengono inoltre forniti dei plug-in per adattare la funzionalità alle proprie necessità. È possibile utilizzare SPI (service provider interface) per creare plug-in personalizzati per la gestione di eventi e l'elaborazione di query dello staff.

### About this task

È possibile creare plug-in per gli eventi API delle attività umane e per gli eventi di notifica di escalation. È inoltre possibile creare un plug-in che elabora i risultati restituiti dalla risoluzione manuale. Ad esempio, nei periodi di picco, è possibile che si desideri aggiungere utenti all'elenco dei risultati per bilanciare il carico di lavoro.

È possibile registrare i plug-in su livelli diversi, per tutte le attività ad un livello globale, per le attività di un componente dell'applicazione, per tutte le attività associate a un modello di attività o per una singola istanza di attività.

## Creazione di gestori eventi API

Un evento API si verifica quando un metodo API modifica una attività umana. Utilizzare l'SPI (service provider interface) del plug-in del gestore eventi API per creare plug-in per la gestione degli eventi di attività inviati dall'API o eventi interni equivalenti agli eventi API.

### About this task

Per creare un gestore eventi API, completare la seguente procedura:

### Procedure

1. Scrivere una classe che implementi l'interfaccia `APIEventHandlerPlugin2` o estenda la classe di implementazione `APIEventHandler`. Questa classe può richiamare i metodi di altre classi.
  - Se si utilizza l'interfaccia `APIEventHandlerPlugin2`, è necessario implementare tutti i metodi dell'interfaccia `APIEventHandlerPlugin2` e dell'interfaccia `APIEventHandlerPlugin`.
  - Se si estende la classe di implementazione SPI, sovrascrivere i metodi necessari.

Questa classe viene eseguita nel contesto di un'applicazione Java 2 Enterprise Edition (J2EE) Enterprise JavaBeans (EJB). Verificare che questa classe e le classi di supporto corrispondenti seguano la specifica EJB.

**Suggerimento:** Se si desidera richiamare l'interfaccia `HumanTaskManagerService` da questa classe, non richiamare un metodo che aggiorni l'attività che ha prodotto l'evento. Questa azione genera un deadlock del database.

2. Inserire la classe di plug-in e le relative classi di supporto in un file JAR. Se le classi di supporto vengono utilizzate da varie applicazioni J2EE, è possibile comprimere queste classi in un file JAR diverso, che viene registrato come una libreria condivisa.
3. Creare un file di configurazione del provider dei servizi per il plug-in nella directory `META-INF/services/` del file JAR. Il file di configurazione fornisce il meccanismo per identificare e caricare il plug-in. Questo file è conforme alla specifica dell'interfaccia del provider del servizio Java 2.
  - a. Creare un file con il nome `com.ibm.task.spi.nome_plug-inAPIEventHandlerPlugin`, dove `nome_plug-in` è il nome del plug-in. Ad esempio, se il plug-in si chiama `Customer` e implementa l'interfaccia `com.ibm.task.spi.APIEventHandlerPlugin`, il nome del file di configurazione è `com.ibm.task.spi.CustomerAPIEventHandlerPlugin`.
  - b. Nella prima riga del file, che non è una riga di commento né una riga vuota, specificare il nome completo della classe di plug-in creata al passo 1. Ad esempio, se la classe di plug-in è denominata `MyAPIEventHandler` e si trova nel package `com.customer.plugins`, la prima riga del file di configurazione deve contenere la seguente voce:  
`com.customer.plugins.MyAPIEventHandler`.

### Results

Si avrà un file JAR installabile contenente un plug-in che gestisce gli eventi API e un file di configurazione del provider dei servizi che può essere utilizzato per caricare il plug-in.

**Suggerimento:** Per registrare gestori eventi API e gestori eventi di notifica è disponibile un'unica proprietà `eventHandlerName`. Se si desidera utilizzare un gestore eventi API e un gestore eventi di notifica, le implementazioni del plug-in devono avere lo stesso nome, per esempio `Client` come nome del gestore eventi per l'implementazione SPI.

È possibile implementare entrambi i plug-in usando un'unica classe, o due classi separate. In entrambi i casi, è necessario creare due file nella directory `META-INF/services/` del file JAR, per esempio, `com.ibm.task.spi.CustomerNotificationEventHandlerPlugin` e `com.ibm.task.spi.CustomerAPIEventHandlerPlugin`.

Inserire l'implementazione del plug-in e le classi helper in un unico file JAR.

### Operazioni successive

È necessario ora installare e registrare il plug-in in modo che risulti disponibile per il contenitore dell'attività umana al runtime. È possibile registrare i gestori eventi API con un'istanza dell'attività, un modello di attività o un componente dell'applicazione.

### Gestori eventi API

Gli eventi API si verificano quando un'attività umana viene modificata o cambia lo stato. Per gestire gli eventi API, il gestore eventi viene richiamato subito prima della modifica dell'attività (metodo pre-evento) e subito prima della restituzione della chiamata API (metodo post-evento).

Se il metodo pre-evento genera un'eccezione `ApplicationVetoException`, l'azione API non viene eseguita, l'eccezione viene restituita al chiamante dell'API e viene eseguito il rollback della transazione associata all'evento. Se il metodo pre-evento è stato attivato da un evento interno e viene generata un'eccezione `ApplicationVetoException`, l'evento interno, ad esempio una richiesta automatica, non viene eseguito, ma un'eccezione non viene restituita all'applicazione client. In questo caso, un messaggio informativo viene scritto nel file `SystemOut.log`. Se il metodo API genera un'eccezione durante l'elaborazione, l'eccezione viene acquisita e trasferita al metodo post-evento. L'eccezione viene trasferita di nuovo al chiamante dopo la restituzione del metodo post-evento.

Ai metodi pre-evento vengono applicate le seguenti regole:

- I metodi pre-evento ricevono i parametri del metodo API associato o dell'evento interno.
- I metodi pre-evento possono generare un'eccezione `ApplicationVetoException` per evitare la continuazione dell'elaborazione.

Ai metodi post-evento vengono applicate le seguenti regole:

- I metodi post-evento ricevono i parametri forniti alla chiamata API e il valore di restituzione. Se un'eccezione viene generata dall'implementazione del metodo API, anche il metodo post-evento riceve l'eccezione.
- I metodi post-evento non possono modificare i valori di restituzione.
- I metodi post-evento non possono generare eccezioni, le eccezioni di runtime vengono registrate ma ignorate.

Per implementare i gestori eventi API, è possibile utilizzare l'interfaccia `APIEventHandlerPlugin2` che estende l'interfaccia `APIEventHandlerPlugin` o estendere la classe di implementazione SPI `com.ibm.task.spi.APIEventHandler` predefinita. Se il gestore eventi eredita dalla classe di implementazione predefinita, implementa sempre la versione più recente dell'SPI. Se si passa a una versione più recente di Business Process Choreographer, sono necessarie poche modifiche, se si desidera utilizzare i nuovi metodi SPI.

Se si dispone di un gestore eventi di notifica e di un gestore eventi API, i gestori devono avere lo stesso nome, in quanto è possibile registrare un solo nome di gestore eventi.

## Creazione di gestori eventi di notifica

Gli eventi di notifica vengono prodotti quando si verifica l'escalation delle attività umane. Business Process Choreographer fornisce funzioni per la gestione delle escalation, ad esempio la creazione di elementi di lavoro dell'escalation o l'invio di e-mail. È possibile creare gestori eventi di notifica per personalizzare il modo in cui vengono gestite le escalation.

### About this task

Per implementare i gestori eventi di notifica, è possibile utilizzare l'interfaccia `NotificationEventHandlerPlugin` o estendere la classe di implementazione SPI (service provider interface) `com.ibm.task.spi.NotificationEventHandler` predefinita.

Per creare un gestore eventi di notifica, completare la seguente procedura:

### Procedure

1. Scrivere una classe che implementi l'interfaccia `NotificationEventHandlerPlugin` o estenda la classe di implementazione `NotificationEventHandler`. Questa classe può richiamare i metodi di altre classi.

Se si utilizza l'interfaccia `NotificationEventHandlerPlugin`, è necessario implementare tutti i metodi dell'interfaccia. Se si estende la classe di implementazione SPI, sovrascrivere i metodi necessari.

Questa classe viene eseguita nel contesto di un'applicazione Java 2 Enterprise Edition (J2EE) Enterprise JavaBeans (EJB). Verificare che questa classe e le classi di supporto corrispondenti seguano la specifica EJB.

Il plug-in viene richiamato con l'autorizzazione del ruolo `EscalationUser`. Questo ruolo viene definito quando si configura il contenitore dell'attività umana.

**Suggerimento:** Se si desidera richiamare l'interfaccia `HumanTaskManagerService` da questa classe, non richiamare un metodo che aggiorni l'attività o l'escalation che ha prodotto l'evento. Questa azione genera un deadlock del database.

2. Inserire la classe di plug-in e le relative classi di supporto in un file JAR.  
Se le classi di supporto vengono utilizzate da varie applicazioni J2EE, è possibile comprimere queste classi in un file JAR diverso, che viene registrato come una libreria condivisa.
3. Creare un file di configurazione del provider dei servizi per il plug-in nella directory `META-INF/services/` del file JAR.



Il file di configurazione fornisce il meccanismo per identificare e caricare il plug-in. Questo file è conforme alla specifica dell'interfaccia del provider del servizio Java 2.

- a. Creare un file con il nome `com.ibm.task.spi.nome_plug-inNotificationEventHandlerPlugin`, dove *nome\_plug-in* è il nome del plug-in.  
Ad esempio, se il plug-in si chiama `HelpDeskRequest` (nome del gestore eventi) e implementa l'interfaccia `com.ibm.task.spi.NotificationEventHandlerPlugin`, il nome del file di configurazione è `com.ibm.task.spi.HelpDeskRequestNotificationEventHandlerPlugin`.
- b. Nella prima riga del file, che non è una riga di commento né una riga vuota, specificare il nome completo della classe di plug-in creata al passo 1.  
Ad esempio, se la classe di plug-in è denominata `MyEventHandler` e si trova nel package `com.customer.plugins`, la prima riga del file di configurazione deve contenere la seguente voce: `com.customer.plugins.MyEventHandler`.

## Results

Si avrà un file JAR installabile contenente un plug-in che gestisce gli eventi di notifica e un file di configurazione del provider dei servizi che può essere utilizzato per caricare il plug-in. È possibile registrare i gestori eventi API con un'istanza dell'attività, un modello di attività o un componente dell'applicazione.

**Suggerimento:** Per registrare gestori eventi API e gestori eventi di notifica è disponibile un'unica proprietà `eventHandlerName`. Se si desidera utilizzare un gestore eventi API e un gestore eventi di notifica, le implementazioni del plug-in devono avere lo stesso nome, per esempio `Client` come nome del gestore eventi per l'implementazione SPI.

È possibile implementare entrambi i plug-in usando un'unica classe, o due classi separate. In entrambi i casi, è necessario creare due file nella directory `META-INF/services/` del file JAR, per esempio, `com.ibm.task.spi.CustomerNotificationEventHandlerPlugin` e `com.ibm.task.spi.CustomerAPIEventHandlerPlugin`.

Inserire l'implementazione del plug-in e le classi helper in un unico file JAR.

## Operazioni successive

È necessario ora installare e registrare il plug-in in modo che risulti disponibile per il contenitore dell'attività umana al runtime. È possibile registrare i gestori eventi di notifica con un'istanza dell'attività, un modello di attività o un componente dell'applicazione.

## Creazione di plug-in per la post-elaborazione dei risultati delle query persone

La risoluzione dello staff restituisce un elenco degli utenti a cui è assegnato uno specifico ruolo, ad esempio, il potenziale proprietario di un'attività. È possibile creare un plug-in per modificare i risultati delle query manuali restituiti da risoluzione manuale. Per migliorare il bilanciamento del carico di lavoro, ad esempio, si potrebbe disporre di un plug-in che rimuove degli utenti dal risultato di una query che già presenta un carico di lavoro elevato.



## About this task

È possibile avere un solo plug-in di post-elaborazione; questo significa che il plug-in deve gestire i risultati delle query persone da tutte le attività. Il plug-in può aggiungere o rimuovere utenti o modificare informazioni su utenti o gruppi. È possibile anche modificare il tipo di risultato, ad esempio, da un elenco di utenti a un gruppo o a tutti gli utenti.

Poiché il plug-in viene eseguito dopo il completamento della risoluzione manuale, tutte le regole per preservare riservatezza o sicurezza sono già state applicate. Il plug-in riceve informazioni su utenti che sono stati rimossi durante la risoluzione manuale (nella chiave di associazione HTM\_REMOVED\_USERS). È necessario verificare che il plug-in utilizzi queste informazioni di contesto per preservare le regole di confidenzialità o sicurezza di cui si dispone.

Per implementare la post-elaborazione dei risultati delle query persone, è possibile usare l'interfaccia `StaffQueryResultPostProcessorPlugin`. L'interfaccia presenta metodi per la modifica dei risultati delle query per attività, escalation, modelli di attività e componenti di applicazioni.

Completare la seguente procedura per creare un plug-in per post-elaborare i risultati delle query persone.

### Procedure

1. Scrivere una classe che implementa l'interfaccia `StaffQueryResultPostProcessorPlugin`.

È necessario implementare tutti i metodi dell'interfaccia. Questa classe può richiamare metodi di altre classi.

Questa classe viene eseguita nel contesto di un'applicazione Java 2 Enterprise Edition (J2EE) Enterprise JavaBeans (EJB). Verificare che questa classe e le classi di supporto corrispondenti seguano la specifica EJB.

**Suggerimento:** Se si desidera richiamare l'interfaccia `HumanTaskManagerService` da questa classe, non richiamare un metodo che aggiorni l'attività che ha prodotto l'evento. Questa azione genera un deadlock del database.

L'esempio che segue mostra come modificare il ruolo di editor di un'attività chiamata `SpecialTask`.

```
public StaffQueryResult processStaffQueryResult
 (StaffQueryResult originalStaffQueryResult,
 Task task,
 int role,
 Map context)
{
 StaffQueryResult newStaffQueryResult = originalStaffQueryResult;
 StaffQueryResultFactory staffResultFactory =
 StaffQueryResultFactory.newInstance();
 if (role == com.ibm.task.api.WorkItem.REASON_EDITOR &&
 task.getName() != null &&
 task.getName().equals("SpecialTask"))
 {
 UserData user = staffResultFactory.newUserData
 ("SuperEditor",
 new Locale("en-US"),
 "SuperEditor@company.com");
 ArrayList userList = new ArrayList();
 userList.add(user);
 }
}
```

```

 newStaffQueryResult = staffResultFactory.newStaffQueryResult(userList);
 }
 return(newStaffQueryResult);
}

```

2. Inserire la classe di plug-in e le relative classi di supporto in un file JAR.  
Se le classi di supporto vengono utilizzate da varie applicazioni J2EE, è possibile comprimere queste classi in un file JAR diverso, che viene registrato come una libreria condivisa.
3. Creare un file di configurazione del provider dei servizi per il plug-in nella directory META-INF/services/ del file JAR.  
Il file di configurazione fornisce il meccanismo per identificare e caricare il plug-in. Questo file è conforme alla specifica dell'interfaccia del provider del servizio Java 2.
  - a. Creare un file con il nome `com.ibm.task.spi.plug-in_nameStaffQueryResultPostProcessorPlugin`, in cui *plug-in\_name* è il nome del plug-in.  
Per esempio, se il plug-in si chiama `MyHandler` e implementa l'interfaccia `com.ibm.task.spi.StaffQueryResultPostProcessorPlugin`, il nome del file di configurazione è `com.ibm.task.spi.MyHandlerStaffQueryResultPostProcessorPlugin`.
  - b. Nella prima riga del file, che non è una riga di commento né una riga vuota, specificare il nome completo della classe di plug-in creata al passo 1.  
Ad esempio, se la classe di plug-in è denominata `StaffPostProcessor` e si trova nel package `com.customer.plugins`, la prima riga del file di configurazione deve contenere la seguente voce:  
`com.customer.plugins.StaffPostProcessor`. Si ha un file JAR installabile che contiene un plug-in che post-elabora i risultati delle query persone e un file di configurazione del provider del servizio che può essere usato per caricare il plug-in.
4. Installare il plug-in.  
È possibile avere un solo plug-in di post-elaborazione per i risultati delle query persone. È necessario installare il plug-in come una libreria condivisa.
5. Registrare il plug-in.
  - a. Nella console di gestione, andare alla pagina Proprietà personalizzate di Human Task Manager (**Server delle applicazioni** → *nome\_server* → **Contentitore delle attività umane** → **Proprietà personalizzate**).
  - b. Aggiungere una proprietà personalizzata con il nome **Staff.PostProcessorPlugin**, e un valore del nome assegnato al plug-in, `MyHandler` in questo esempio.

## Installazione di plug-in

Per utilizzare un plug-in, è necessario installarlo per potervi accedere dal contenitore delle attività.

### About this task

Il modo in cui si installa il plug-in dipende dal fatto se il plug-in deve essere usato da una sola applicazione Java 2 Enterprise Edition (J2EE), o da diverse applicazioni.

Per installare un plug-in, completare uno dei seguenti passi:

- Installare un plug-in per l'utilizzo da parte di una singola applicazione J2EE.

Aggiungere il file JAR del plug-in al file EAR dell'applicazione. Nell'editor del descrittore di distribuzione in WebSphere Integration Developer, installare il file JAR per il plug-in come file JAR di utilità del progetto per l'applicazione J2EE del modulo principale enterprise JavaBeans (EJB).

- Installare un plug-in per l'utilizzo da parte di varie applicazioni J2EE.  
Inserire il file JAR in una libreria condivisa WebSphere Application Server e associare la libreria con le applicazioni che richiedono accesso al plug-in. Per rendere il file JAR disponibile in un ambiente di distribuzione di rete, distribuire il file JAR su ciascun server manualmente, quindi installare la libreria condivisa una volta per ciascuna cella.

### Operazioni successive

È possibile ora registrare il plug-in.

## Registrazione di plug-in

È possibile registrare i plug-in su vari livelli nella gerarchia di risorse del contenitore di attività. Ad esempio, per tutte le attività ad un livello globale, per le attività di un componente dell'applicazione, per tutte le attività associate a un modello di attività o per una singola istanza di attività.

### About this task

Quando si registrano più plug-in, è supportata la funzionalità relativa all'ambito. Ciò significa che un plug-in registrato ad un livello più basso della gerarchia delle risorse del contenitore delle attività, ad esempio l'istanza di un'attività, viene utilizzata al posto del plug-in registrato ad un livello più elevato, ad esempio il modello di attività o il componente dell'applicazione. La funzionalità relativa all'ambito è supportata per tutti i livelli della gerarchia. Il contenitore delle attività utilizza il plug-in registrato al livello più basso della gerarchia.

È possibile registrare un plug-in in uno dei seguenti modi:

- Registrare il plug-in nel modello di attività.  
Nell'editor di attività in WebSphere Integration Developer nella pagina dei Dettagli dell'area delle proprietà per l'attività, specificare il nome del gestore eventi nel campo **Nome gestore eventi**.
- Registrare il plug-in per le attività ad-hoc o i modelli di attività creati al runtime.  
Utilizzare il metodo `setEventHandlerName` della classe `TTask` per registrare il nome del gestore eventi.
- Modificare il gestore eventi registrato per un'istanza dell'attività al runtime.  
Utilizzare il metodo `update(Task task)` per utilizzare un gestore eventi diverso per un'istanza dell'attività al runtime. Per aggiornare questa proprietà, il chiamante deve avere l'autorizzazione di amministratore delle attività.
- Registrare il plug-in ad un livello globale.  
Nella console di gestione, nella pagina Proprietà personalizzate per il contenitore delle attività umane, definire una proprietà personalizzata per il plug-in. Il valore della proprietà personalizzata corrisponde al nome del plug-in.



---

## **Parte 2. Distribuzione delle applicazioni**



---

## Capitolo 3. Panoramica di preparazione e installazione dei moduli

L'installazione dei moduli (denominata anche distribuzione) attiva i moduli in un ambiente di test o in un ambiente di produzione. Questa panoramica descrive brevemente gli ambienti di produzione e di test ed alcuni passi relativi all'installazione dei moduli.

**Nota:** Il processo per le applicazioni di installazione in un ambiente del prodotto è simile al processo descritto nell'argomento "Sviluppo e distribuzione delle applicazioni" nel centro informazioni di WebSphere Application Server Network Deployment, Versione 6. Se non si conoscono tali argomenti, esaminarli prima di continuare.

Prima di installare un modulo in un ambiente di produzione, verificare sempre le modifiche in un ambiente di test. Per installare i moduli in un ambiente di test, utilizzare WebSphere Integration Developer (per ulteriori informazioni, consultare l'Infocenter di WebSphere Integration Developer). Per installare i moduli in un ambiente di produzione, utilizzare WebSphere Process Server.

Questo argomento descrive i concetti e le attività necessarie per preparare ed installare i moduli in un ambiente di produzione. Gli altri argomenti descrivono i file in cui si trovano gli oggetti che il modulo utilizza e consentono di spostare il modulo dall'ambiente di test nell'ambiente di produzione. È importante comprendere tali file e il relativo contenuto, in modo da essere sicuri di aver installato correttamente i moduli.

---

### Panoramica delle librerie e dei file JAR

I moduli spesso utilizzano le risorse che si trovano nelle librerie. Le risorse e le librerie sono contenute nei file JAR (Java) identificati durante la distribuzione di un modulo.

Durante lo sviluppo di un modulo, è possibile identificare alcune risorse o componenti che possono essere utilizzati dalle varie parti del modulo. Tali risorse o componenti possono essere oggetti creati durante lo sviluppo del modulo o oggetti già esistenti che si trovano in una libreria che è già distribuita sul server. Questo argomento descrive le librerie e i file necessari per l'installazione di un'applicazione.

#### Cos'è una libreria?

Una libreria contiene oggetti o risorse utilizzati da più moduli all'interno di WebSphere Integration Developer. Le risorse possono trovarsi in file JAR, RAR o WAR. Alcune di tali risorse comprendono:

- Descrittori di servizi Web o interfacce (file con un'estensione .wsdl)
- Definizioni di schema XML di oggetti business (file con estensione .xsd)
- Associazioni degli oggetti business (file con estensione .map)
- Definizioni di relazioni e ruoli (file con estensione .rel e .rol)

Quando un modulo necessita di una risorsa, il server ricerca la risorsa dal classpath EAR e la carica, se non è già caricata, in memoria. Da questo punto in

poi, qualunque richiesta di risorse utilizza la copia fino a quando non viene sostituita. La Figura 5 illustra il modo in cui un'applicazione contiene i componenti e le relative librerie.

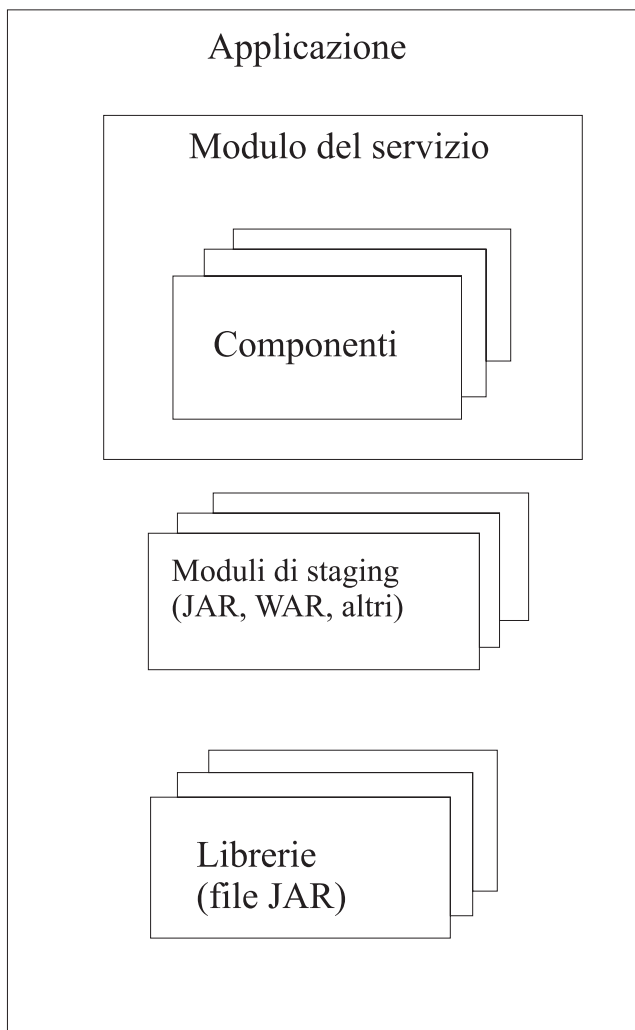


Figura 5. Relazione tra modulo, componente e libreria

### **Cosa sono i file JAR, RAR e WAR?**

È presente un numero di file contenenti i componenti di un modulo. Questi file vengono descritti nel dettaglio nella specifica J2EE (Java Platform Enterprise Edition). È possibile trovare i dettagli relativi ai file JAR nella relativa specifica JAR.

In WebSphere Process Server, un file JAR contiene, inoltre, un'applicazione, ovvero la versione assemblata del modulo con tutti i riferimenti e le interfacce di supporto per altri componenti del servizio utilizzati dal modulo. Per installare completamente l'applicazione, è necessario questo file JAR, eventuali altre librerie come ad esempio i file JAR, file WAR (Web services archive), file RAR (resource archive), file JAR EJB (Enterprise Java Beans) delle librerie di staging, o eventuali altri archivi, ed è necessario creare un file EAR installabile utilizzando il comando `serviceDeploy`.



## Convenzioni di denominazione per i moduli di staging

All'interno della libreria sono presenti dei requisiti per i nomi dei moduli di staging. Questi nomi sono univoci per un determinato modulo. Denominare eventuali altri moduli richiesti per distribuire l'applicazione, in modo che non si verifichino servizi con i nomi del modulo di staging. Per un modulo denominato *myService*, i nomi del modulo di staging sono:

- *myServiceApp*
- *myServiceEJB*
- *myServiceEJBClient*
- *myServiceWeb*

**Nota:** Il comando `serviceDeploy` crea solo il modulo di staging Web *myService* se il servizio comprende un servizio di tipo porta WSDL.

## Considerazioni relative all'utilizzo delle librerie

L'utilizzo delle librerie fornisce una coerenza degli oggetti business e di elaborazione tra i moduli, poiché ciascun modulo di richiamo dispone della relativa copia di un determinato componente. Per impedire le incoerenze e gli errori, è importante assicurarsi che le modifiche al componente e agli oggetti business utilizzati dai moduli di richiamo siano coordinati con tutti i moduli di richiamo. Aggiornare i moduli di richiamo come segue:

1. Copia del modulo e copia più aggiornata delle librerie sul server di produzione
2. Creazione del file EAR installabile utilizzando il comando `serviceDeploy`
3. Arresto dell'applicazione in esecuzione contenente il modulo di richiamo e relativa reinstallazione
4. Riavvio dell'applicazione contenente il modulo di richiamo

### Riferimenti correlati



Comando `serviceDeploy`

Usare il comando `serviceDeploy` per eseguire il packaging dei moduli compatibili SCA (Service Component Architecture) come applicazioni Java installabili su un server. Questo comando è utile per eseguire installazioni batch mediante `wsadmin`.

---

## Panoramica del file EAR

Un file EAR è una parte fondamentale nello sviluppo di un'applicazione del servizio ad un server di produzione.

Un file EAR (enterprise archive) è un file compresso che contiene le librerie, i bean enterprise e i file JAR richiesti dall'applicazione per la distribuzione.

Creare un file JAR quando si esportano i moduli dell'applicazione da WebSphere Integration Developer. Utilizzare il file JAR ed eventuali altre librerie della risorsa o oggetti come input per il processo di installazione. Il comando `serviceDeploy` crea un file EAR dai file di input contenenti le descrizioni del componente e il codice Java incluso nell'applicazione.

### Riferimenti correlati



Comando `serviceDeploy`

Usare il comando `serviceDeploy` per eseguire il packaging dei moduli compatibili SCA (Service Component Architecture) come applicazioni Java installabili su un server. Questo comando è utile per eseguire installazioni batch mediante `wsadmin`.

---

## Preparazione alla distribuzione su un server

Una volta sviluppato e eseguito il test di un modulo, è necessario esportarlo da un sistema di test ad un ambiente di produzione per la distribuzione. Per installare un'applicazione, è necessario anche conoscerne il percorso durante l'esportazione del modulo e di eventuali librerie richieste dal modulo stesso.

### Prima di iniziare

Prima di iniziare questa attività, è necessario aver distribuito e verificato i moduli su un server di test e aver risolto gli eventuali problemi generici e di prestazioni.

### About this task

Questa attività verifica che tutte le parti necessarie di un'applicazione siano disponibili e che si trovino nei file corretti per portare al server di produzione.

**Nota:** Inoltre, è possibile esportare un file EAR (enterprise archive) da WebSphere Integration Developer, quindi installare il file direttamente in WebSphere Process Server.

**Importante:** Se i servizi all'interno di un componente utilizzano un database, installare l'applicazione su un server direttamente connesso al database.

### Procedure

1. Cercare la cartella contenente i componenti per il modulo che si sta per distribuire.

La cartella del componente deve essere denominata *modulo-nome* con un file denominato *module.module*, il modulo di base.

2. Verificare che tutti i componenti contenuti nel modulo siano cartelle secondarie sotto la cartella del modulo.

Per semplicità di utilizzo, la cartella secondaria simile a *module/component*.

3. Verificare che tutti i file che comprendono ciascun componente siano contenuti nella cartella secondaria del componente appropriata e dispongano di un nome simile a *componente-file-nome.component*.

I file del componente contengono le definizioni per ciascun singolo componente all'interno del modulo.

4. Verificare che tutti i componenti e risorse siano nelle cartelle secondarie del componente che li richiede.

Questo passo assicura che i riferimenti alle risorse richieste da un componente siano disponibili. I nomi dei componenti non devono essere in conflitto con i nomi utilizzati dal comando `serviceDeploy` per i moduli di staging. Consultare Convenzioni di denominazione per i moduli di staging.

5. Verificare che il file di riferimento *module.references*, esista nella cartella del modulo del passo 1.

Il file di riferimento definisce i riferimenti e le interfacce all'interno del modulo.

6. Verificare che un file relativo ai collegamenti *module.wires* esista nella cartella del componente.

Il file relativo ai collegamenti completa la connessione tra i riferimenti e le interfacce all'interno del modulo.

7. Verificare che un file manifest, *module.manifest*, esista nella cartella del componente.

Il manifest elenca il modulo e tutti i componenti che comprendono il modulo. Inoltre, contiene un'istruzione del classpath, in modo che il comando `serviceDeploy` possa rilevare altri moduli necessari per quel modulo.

8. Creare un file compresso o un file JAR del modulo come input al comando `serviceDeploy` che verrà utilizzato per preparare il modulo all'installazione sul server di produzione.


## Esempio della struttura di una cartella per il modulo MyValue prima della distribuzione

L'esempio di seguito riportato illustra la struttura di directory per il modulo `MyValueModule`, costituito dai componenti `MyValue`, `CustomerInfo` e `StockQuote`.

```
MyValueModule
 MyValueModule.manifest
 MyValueModule.references
 MyValueModule.wiring
 MyValueClient.jsp
process/myvalue
 MyValue.component
 MyValue.java
 MyValueImpl.java
service/customerinfo
 CustomerInfo.component
 CustomerInfo.java
 Customer.java
 CustomerInfoImpl.java
service/stockquote
 StockQuote.component
 StockQuote.java
 StockQuoteAsynch.java
 StockQuoteCallback.java
 StockQuoteImpl.java
```

Installare il modulo nei sistemi di produzione come descritto nell'argomento [Installazione di un modulo su un server di produzione](#) o un server di produzione.

### Riferimenti correlati

 [Comando serviceDeploy](#)

Usare il comando `serviceDeploy` per eseguire il packaging dei moduli compatibili SCA (Service Component Architecture) come applicazioni Java installabili su un server. Questo comando è utile per eseguire installazioni batch mediante `wsadmin`.

---

## Considerazioni sull'installazione delle applicazioni di servizio sui cluster

L'installazione di un'applicazione del servizio in un cluster consente all'amministratore di scegliere il posizionamento degli ulteriori requisiti. È importante considerare quanto segue se si installano le applicazioni del servizio in un cluster.

I cluster possono fornire molti vantaggi all'ambiente di elaborazione, mediante un'economia di scala che consente di bilanciare il carico di lavoro delle richieste tra i server e fornire un livello di disponibilità per i client delle applicazioni. Considerare quanto di seguito riportato prima di installare un'applicazione contenente i servizi in un cluster:

- Gli utenti dell'applicazione richiederanno l'alimentazione di elaborazione e la disponibilità fornita dall'organizzazione in cluster?  
In caso affermativo, l'organizzazione in cluster è la soluzione corretta. L'organizzazione in cluster aumenta la disponibilità e la capacità delle applicazioni.
- Il cluster è preparato correttamente per le applicazioni del servizio?  
È necessario configurare correttamente il cluster prima di installare ed avviare la prima applicazione contenente un servizio. Se non si riesce a configurare correttamente un cluster, le applicazioni non riusciranno ad elaborare correttamente le richieste.
- Il cluster dispone di un backup?  
È necessario installare l'applicazione anche sul cluster di backup.

---

## Capitolo 4. Installazione di un modulo su un server di produzione

Questo argomento descrive i passi per distribuire un'applicazione da un server di test ad un ambiente di produzione.

### Prima di iniziare

Prima di distribuire un'applicazione del servizio ad un server di produzione, assemblare e verificare l'applicazione in un server di test. Dopo il test, esportare i relativi file, come descritto in *Preparazione alla distribuzione su un server* nel PDF Sviluppo e distribuzione dei moduli e portare tali file al sistema di produzione da distribuire. Consultare i centri informazioni per WebSphere Integration Developer e WebSphere Application Server Network Deployment per ulteriori informazioni.

### Procedure

1. Copiare il modulo e gli altri file sul server di produzione.  
I moduli e le risorse (file EAR, JAR, RAR e WAR) necessari per l'applicazione vengono spostati nell'ambiente di produzione.
2. Eseguire il comando `serviceDeploy` per creare un file EAR che può essere installato.  
Questo passo definisce il modulo al server per la preparazione all'installazione dell'applicazione nel server di produzione.
  - a. Ricercare il file JAR contenente il modulo da distribuire.
  - b. Immettere il comando utilizzando il file JAR dal passo precedente come input.
3. Installare il file EAR dal passo 2. Il modo di installazione delle applicazioni dipende dall'installazione effettuata su un server autonomo o un server in una cella.

**Nota:** È possibile utilizzare la console di gestione o uno script per installare l'applicazione. Consultare il centro informazioni WebSphere Application Server per ulteriori informazioni.

4. Salvare la configurazione. Il modulo è ora installato come un'applicazione.
5. Avviare l'applicazione.


### Results

L'applicazione è ora attiva e il lavoro dovrebbe passare attraverso il modulo.

### Operazioni successive

Monitorare l'applicazione per assicurarsi che il server stia elaborando correttamente le richieste.

#### Riferimenti correlati

 Comando `serviceDeploy`

Usare il comando `serviceDeploy` per eseguire il packaging dei moduli compatibili SCA (Service Component Architecture) come applicazioni Java installabili su un server. Questo comando è utile per eseguire installazioni batch mediante `wsadmin`.

---

## Creazione di un file EAR che può essere installato utilizzando `serviceDeploy`

Per installare un'applicazione nell'ambiente di produzione, copiare i file nel server di produzione e creare un file EAR installabile.

### Prima di iniziare

Prima di avviare quest'attività, è necessario disporre di un file JAR contenente il modulo e i servizi che si stanno distribuendo al server. Consultare Preparazione alla distribuzione su un server per maggiori informazioni.

### About this task

Il comando `serviceDeploy` richiama un file JAR, eventuali file EAR, JAR, RAR, WAR e ZIP dipendenti e crea un file EAR che è possibile installare su un server.

### Procedure

1. Cercare il file JAR contenente il modulo da distribuire.
2. Immettere il comando utilizzando il file JAR dal passo precedente come input. Questo passo crea un file EAR sul primo server.

**Nota:** Eseguire i passi indicati in una console di gestione.

3. Selezionare il file EAR da installare nella console di gestione del server.
4. Fare clic su **Salva** per installare il file EAR.

### Riferimenti correlati

 Comando `serviceDeploy`

Usare il comando `serviceDeploy` per eseguire il packaging dei moduli compatibili SCA (Service Component Architecture) come applicazioni Java installabili su un server. Questo comando è utile per eseguire installazioni batch mediante `wsadmin`.

---

## Distribuzione di applicazioni mediante le attività Apache Ant

Questo argomento descrive come utilizzare le attività Apache Ant per automatizzare la distribuzione delle applicazioni in WebSphere Process Server. Utilizzando le attività Apache Ant, è possibile definire la distribuzione di più applicazioni ed eseguirle in modalità non presidiata su un server.

### Prima di iniziare

Questa attività presuppone quanto segue:

- Le applicazioni che stanno per essere distribuite sono già state distribuite e verificate.
- Le applicazioni devono essere installate sullo stesso server.
- È necessario avere alcune conoscenze sulle attività Apache Ant.
- È necessaria la comprensione del processo di distribuzione.

Le informazioni sullo sviluppo e la verifica delle applicazioni si trovano nell'Infocenter WebSphere Integration Developer.

La porzione di riferimento del centro informazioni per WebSphere Application Server Network Deployment contiene una sezione sulle interfacce di programmazione dell'applicazione. Le attività Apache Ant sono descritte nel package `com.ibm.websphere.ant.tasks`. In relazione a questo argomento, le attività di interesse sono `ServiceDeploy` e `InstallApplication`.

### About this task

Se è necessario installare più applicazioni simultaneamente, sviluppare un'attività Apache Ant prima della distribuzione. L'attività Apache Ant può quindi distribuire ed installare le applicazioni sui server senza coinvolgimento da parte dell'utente nel processo.

### Procedure

1. Identificare le applicazioni da distribuire.
2. Creare un file JAR per ciascuna applicazione.
3. Copiare i file JAR ai server di destinazione.
4. Creare un'attività Apache Ant per eseguire il comando `ServiceDeploy` per creare il file EAR per ogni server.
5. Creare un'attività Apache Ant per eseguire il comando `InstallApplication` per ogni file EAR dal passo 4 sui server applicabili.
6. Eseguire l'attività Apache Ant `ServiceDeploy` per creare il file EAR per le applicazioni.
7. Eseguire l'attività Apache Ant `InstallApplication` per installare i file EAR dal passo 6.

### Results

Le applicazioni vengono distribuite correttamente sui server di destinazione.

## Esempio di distribuzione di un'applicazione in modo non presidiato

Questo esempio illustra un'attività Apache Ant contenuta in un file `myBuildScript.xml`.

```
<?xml version="1.0">
<project name="OwnTaskExample" default="main" basedir=".">
 <taskdef name="servicedeploy"
 classname="com.ibm.websphere.ant.tasks.ServiceDeployTask" />
 <target name="main" depends="main2">
 <servicedeploy scaModule="c:/synctest/SyncTargetJAR"
 ignoreErrors="true"
 outputApplication="c:/synctest/SyncTargetEAREAR"
 workingDirectory="c:/synctest"
 noJ2eeDeploy="true"
 cleanStagingModules="true"/>
 </target>
</project>
```

Questa istruzione mostra come richiamare un'attività Apache Ant.


```
${WAS}/bin/ws_ant -f myBuildScript.xml
```

**Suggerimento:** È possibile distribuire più applicazioni in modo non presidiato aggiungendo ulteriori istruzioni del progetto al file.

### **Operazioni successive**

Utilizzare la console di gestione per verificare che le applicazioni al momento installate siano avviate e che stiano elaborando correttamente il flusso di lavoro.

#### **Riferimenti correlati**

 Comando `serviceDeploy`

Usare il comando `serviceDeploy` per eseguire il packaging dei moduli compatibili SCA (Service Component Architecture) come applicazioni Java installabili su un server. Questo comando è utile per eseguire installazioni batch mediante `wsadmin`.



---

## Capitolo 5. Installazione di applicazioni di processo di business e attività umana

È possibile distribuire moduli Service Component Architecture (SCA) che contengono processi di business o attività umane, o entrambi, a destinazioni di distribuzione. Una destinazione di distribuzione può essere un server o un cluster.

### Prima di iniziare

Verificare che Business Flow Manager, Human Task Manager, o entrambi siano installati e configurati per ciascun server applicativo o cluster su cui si desidera installare l'applicazione.

### About this task

È possibile installare le applicazioni di processo di business e le applicazioni di attività dalla console di gestione, dalla riga comandi o eseguendo uno script di gestione.

### Results

Dopo che un'applicazione di processo di business o attività umane viene installata, tutti i modelli di processo di business e attività umane sono messi in stato di avvio. È possibile creare istanze di processo e di attività da questi modelli.

### Operazioni successive

Prima di creare le istanze del processo o le istanze dell'attività, è necessario avviare l'applicazione.

#### Concetti correlati

“Distribuzione di processi di business e attività umane” a pagina 192  
Quando WebSphere Integration Developer o la distribuzione del servizio generano il codice di distribuzione per il processo, o attività, ciascun componente del processo o dell'attività è associata in un enterprise bean di sessione. Tutto il codice di distribuzione viene compresso in un file enterprise application (EAR). Inoltre, per ciascun processo, una classe Java che rappresenta il codice Java nel processo è generata e integrata nel file EAR durante l'installazione dell'enterprise application. Per ogni nuova versione di un modello che deve essere distribuito deve essere effettuato il packaging in una nuova enterprise application.

“Installazione di applicazioni di processi di business e attività umane in un ambiente di distribuzione in rete” a pagina 192

Quando vengono installati i modelli di processo o di attività umane in un ambiente di distribuzione in rete, le seguenti azioni sono eseguite automaticamente dall'installazione dell'applicazione.

---

## Installazione di applicazioni di processi di business e attività umane in un ambiente di distribuzione in rete

Quando vengono installati i modelli di processo o di attività umane in un ambiente di distribuzione in rete, le seguenti azioni sono eseguite automaticamente dall'installazione dell'applicazione.

L'applicazione viene installata in modo asincrono attraverso diversi passaggi. Ciascun passaggio deve essere completato con successo prima di poter iniziare il successivo.

1. L'installazione dell'applicazione inizia sul gestore distribuzione.

Durante questo passaggio, i modelli di processo di business e di attività umane sono configurati nel repository di configurazione di WebSphere. L'applicazione viene inoltre convalidata. Se si verificano errori, sono riportati nel file System.out, nel file System.err, o come voci FFDC sul gestore distribuzione.

2. L'installazione dell'applicazione continua sull'agent del nodo.

Durante questo passaggio, viene attivata l'installazione dell'applicazione su un'istanza del server applicativo. Questa istanza del server applicativo può essere, o essere parte della destinazione di distribuzione. Se la destinazione di distribuzione è un cluster con multipli membri cluster l'istanza del server viene scelta arbitrariamente dai membri cluster di questo cluster. Se si verificano errori durante questo passaggio, sono riportati nel file SystemOut.log, nel file SystemErr.log, o come voci FFDC sull'agent del nodo.

3. L'applicazione viene eseguita sull'istanza del server.

Durante questo passaggio, i modelli di processo e umani sono distribuiti al database di Business Process Choreographer nella destinazione di distribuzione. Se si verificano degli errori, questi vengono riportati nel file System.out, nel file SystemErr.log, o come voci FFDC su questa istanza del server.

### Attività correlate

Capitolo 5, "Installazione di applicazioni di processo di business e attività umana", a pagina 191

È possibile distribuire moduli Service Component Architecture (SCA) che contengono processi di business o attività umane, o entrambi, a destinazioni di distribuzione. Una destinazione di distribuzione può essere un server o un cluster.

---

## Distribuzione di processi di business e attività umane

Quando WebSphere Integration Developer o la distribuzione del servizio generano il codice di distribuzione per il processo, o attività, ciascun componente del processo o dell'attività è associata in un enterprise bean di sessione. Tutto il codice di distribuzione viene compresso in un file enterprise application (EAR). Inoltre, per ciascun processo, una classe Java che rappresenta il codice Java nel processo è generata e integrata nel file EAR durante l'installazione dell'enterprise application. Per ogni nuova versione di un modello che deve essere distribuito deve essere effettuato il packaging in una nuova enterprise application.

Quando viene installata un'enterprise application che contiene processi di business o attività umane, questi vengono appropriatamente archiviati come modelli di processo di business o modelli di attività umane nel database Business Process Choreographer. I modelli appena installati sono, per impostazione predefinita, in

stato avviato. Tuttavia, l'applicazione enterprise appena installata è in stato arrestato. Ciascuna applicazione enterprise installata può essere avviata e arrestata singolarmente.

È possibile distribuire varie versioni diverse di un modello di processo o di attività, ciascuna in un'applicazione enterprise diversa. Quando viene installata una nuova enterprise application, la versione del modello installata si determina come segue:

- Se il nome del modello e del namespace di destinazione non esiste già, viene installato un nuovo modello
- Se il nome del modello e il namespace di destinazione sono uguali a quelli di un modello esistente, ma la data di inizio validità è diversa, viene installata una nuova versione di un modello esistente

**Nota:** Il nome del modello deriva dal nome del componente e non dal processo di business o attività umane.

Se non si specifica una data Valido da, la data viene determinata nel modo seguente:

- Se si utilizza WebSphere Integration Developer, la data di inizio validità è la data in cui l'attività umana o il processo di business sono stati modellati.
- Se si utilizza la distribuzione del servizio, la data di inizio di validità è la data in cui il comando serviceDeploy è stato eseguito. Solo le attività di collaborazione hanno la data in cui l'applicazione è stata installata come data di inizio validità.

#### **Attività correlate**

Capitolo 5, "Installazione di applicazioni di processo di business e attività umana", a pagina 191

È possibile distribuire moduli Service Component Architecture (SCA) che contengono processi di business o attività umane, o entrambi, a destinazioni di distribuzione. Una destinazione di distribuzione può essere un server o un cluster.

---

## **Installazione di applicazioni di processo di business e attività umane in modalità interattiva**

È possibile installare un'applicazione interattivamente al runtime mediante lo strumento wsadmin e lo script installInteractive. È possibile utilizzare questo script per modificare le impostazioni che non possono essere modificate, se si utilizza la console di gestione per installare l'applicazione

#### **About this task**

Per installare interattivamente le applicazioni del processo di business, completare la seguente procedura:

#### **Procedure**

1. Avviare lo strumento wsadmin.  
Nella directory *root\_profilo/bin*, immettere *wsadmin*.
2. Installare l'applicazione.  
Al prompt della riga comandi, immettere il seguente comando:  
`$AdminApp installInteractive application.ear`

dove *application.ear* è il nome completo del file di archivio enterprise contenente l'applicazione del processo. Vengono proposte varie attività in cui è possibile modificare i valori per l'applicazione.

3. Salvare le modifiche alla configurazione.

Al prompt della riga comandi, immettere il seguente comando:

```
$AdminConfig save
```

Per trasferire gli aggiornamenti al repository di configurazione principale, è necessario salvare le modifiche. Se il processo di script viene terminato e le modifiche non sono state salvate, le modifiche vengono eliminate.

## Configurazione dell'origine dati dell'applicazione del processo e delle impostazioni di riferimento definite

Potrebbe essere necessario configurare le applicazioni del processo che eseguono istruzioni SQL per la specifica infrastruttura del database. Queste istruzioni SQL possono derivare dalle attività del servizio informazioni o possono essere istruzioni eseguite durante l'installazione del processo o l'avvio di un'istanza.

### About this task

Quando si installa l'applicazione, è possibile specificare i seguenti tipi di origini dati:

- Origini dati per eseguire istruzioni SQL durante l'installazione del processo
- Origini dati per eseguire istruzioni SQL durante l'avvio di un'istanza del processo
- Origini dati per eseguire attività di frammenti SQL

L'origine dati richiesta per eseguire un'attività di frammenti SQL è definita in una variabile BPEL di tipo `tDataSource`. Lo schema di database e i nomi di tabelle richiesti da un'attività di frammenti SQL sono definiti nelle variabili BPEL di tipo `tSetReference`. È possibile configurare i valori iniziali di entrambe le variabili.

È possibile utilizzare lo strumento `wsadmin` per specificare le origini dati.

### Procedure

1. Installare l'applicazione del processo in modo interattivo mediante lo strumento `wsadmin`.
2. Analizzare le varie attività, fino a individuare le attività per l'aggiornamento delle origini dati e dei riferimenti impostati.

Configurare queste impostazioni in base all'ambiente utilizzato. Nell'esempio che segue sono riportate le impostazioni che è possibile modificare per ciascuna di queste attività.

3. Salvare le modifiche.

### Esempio: Aggiornamento delle origini dati e dei riferimenti impostati mediante lo strumento `wsadmin`

Nell'attività **Aggiornamento delle origini dati**, è possibile modificare i valori delle origini dati per i valori e le istruzioni delle variabili iniziali utilizzati durante l'installazione del processo o l'avvio del processo. Nell'attività **Aggiornamento dei riferimenti impostati**, è possibile configurare le impostazioni correlate allo schema di database e ai nomi di tabelle.

```

Attività [24]: Aggiornamento delle origini dati

//Modificare i valori delle origini dati per i valori delle variabili iniziali all'avvio del processo

Nome processo: Test
// Il nome del modello di processo
Avvio del processo o installazione: Avvio del processo
// Indica se il valore specificato viene valutato
//all'avvio del processo o al momento dell'installazione
Istruzione o variabile: Variabile
// Indica che una variabile di un'origine dati deve essere modificata
Nome origine dati: MyDataSource
// Il nome della variabile
Nome JNDI:[jdbc/sample]:jdbc/newName
// Imposta il nome JNDI su jdbc/newName
Attività [25]: Aggiornamento dei riferimenti impostati

// Modificare i valori di riferimento impostati utilizzati come valori per le variabili BPEL

Nome processo: Test
// Il nome del modello di processo
Variabile: SetRef
// Il nome della variabile BPEL
Nome JNDI:[jdbc/sample]:jdbc/newName
// Imposta il nome JNDI dell'origine dati del riferimento impostato su jdbc/newName
Nome schema: [IISAMPLE]
// Il nome dello schema di database
Prefisso schema: []:
// Il prefisso del nome dello schema.
// Questa impostazione si applica solo se viene generato il nome dello schema.
Nome tabella: [SETREFTAB]: NEWTABLE
// Imposta il nome della tabella di database su NEWTABLE
Prefisso tabella: []:
// Il prefisso del nome della tabella.
// Questa impostazione si applica solo se viene generato il nome del prefisso.

```

---

## Disinstallazione delle applicazioni di processi di business e attività umane utilizzando la console di gestione

È possibile utilizzare la console di gestione per disinstallare le applicazioni che contengono processi di business o attività umane.

### Prima di iniziare

Per disinstallare un'applicazione che contiene processi di business o attività umane, sono necessari i seguenti requisiti:

- Se l'applicazione è installata su un server autonomo, il server deve essere in esecuzione e avere accesso al database di Business Process Choreographer.
- Se l'applicazione è installata su un cluster, devono essere in esecuzione il gestore distribuzione e almeno un membro cluster. Il membro cluster ha accesso al database di Business Process Choreographer.
- Se l'applicazione è installata su server gestito, il gestore distribuzione e questo server devono essere in esecuzione. Il server deve avere accesso al database di Business Process Choreographer.
- Tutti i modelli di processo di business e attività umane che appartengono all'applicazione devono essere in stato di arresto.
- Non ci sono istanze di modelli di processo di business o attività umane presenti in qualunque stato.
-

Per ambienti server autonomi utilizzati come ambienti di sviluppo e verifica unità il server può essere configurato per essere eseguito in modalità di sviluppo. Questa configurazione non richiede che i modelli siano arrestati e che non siano presenti istanze. Tuttavia, questa configurazione non è valida per ambienti di produzione.

### About this task

Per disinstallare un'applicazione enterprise che include processi di business o attività umane, eseguire le azioni di seguito riportate:

### Procedure

1. Arrestare tutti i modelli di attività e di processo dell'applicazione.

Questa operazione impedisce la creazione di istanze di processo e di attività.

- a. Fare clic su **Applicazioni** → **Moduli SCA** nel riquadro di navigazione della console di gestione.
- b. Selezionare il modulo che contiene i modelli che si desidera arrestare.
- c. In Ulteriori proprietà, fare clic su **Business Process** o **Attività umane** o su entrambi se appropriato.
- d. Selezionare tutti i modelli di processo e di attività facendo clic sulla relativa casella di spunta.
- e. Fare clic su **Arresta**.

Ripetere i passi per tutti i moduli EJB contenenti i modelli di processo di business o attività umana.

2. Verificare che il database, almeno un server dell'applicazione per ciascun cluster e il server autonomo siano in esecuzione dove l'applicazione viene distribuita.

In un ambiente network deployment, è necessario che siano in esecuzione il gestore distribuzione, tutti i server delle applicazioni autonomi gestiti ed almeno un server delle applicazioni per ogni cluster in cui è installata l'applicazione.

3. Verificare che l'applicazione non presenti istanze di processo di business o di attività umana.

Se necessario, un amministratore può utilizzare Business Process Choreographer Explorer per eliminare eventuali istanze di attività o di processo.

4. Arresto e disinstallazione dell'applicazione:

- a. Far clic su **Applicazioni** → **Applicazioni Enterprise** nel riquadro di navigazione della console di gestione.
- b. Selezionare l'applicazione che si desidera disinstallare e fare clic su **Arresta**. Questo passo non riesce se le istanze di processo o le istanze di attività esistono ancora nell'applicazione.
- c. Selezionare di nuovo l'applicazione che si desidera disinstallare, quindi fare clic su **Disinstalla**.
- d. Fare clic su **Salva** per salvare le modifiche.

### Results

L'applicazione viene disinstallata.

---

## Disinstallazione delle applicazioni di processi di business e attività umane utilizzando i comandi di gestione

I comandi di gestione forniscono un'alternativa alla console di gestione per la disinstallazione delle applicazioni che contengono processi di business o attività umane.

### Prima di iniziare

Per disinstallare un'applicazione che contiene processi di business o attività umane, sono necessari i seguenti requisiti:

- Se l'applicazione è installata su un server autonomo, il server deve essere in esecuzione e avere accesso al database di Business Process Choreographer.
- Se l'applicazione è installata su un cluster, devono essere in esecuzione il gestore distribuzione e almeno un membro cluster. Il membro cluster ha accesso al database di Business Process Choreographer.
- Se l'applicazione è installata su server gestito, il gestore distribuzione e questo server devono essere in esecuzione. Il server deve avere accesso al database di Business Process Choreographer.
- Tutti i modelli di processo di business e attività umane che appartengono all'applicazione devono essere in stato di arresto.
- Non ci sono istanze di modelli di processo di business o attività umane presenti in qualunque stato.
- 

Per ambienti server autonomi utilizzati come ambienti di sviluppo e verifica unità il server può essere configurato per essere eseguito in modalità di sviluppo. Questa configurazione non richiede che i modelli siano arrestati e che non siano presenti istanze. Tuttavia, questa configurazione non è valida per ambienti di produzione.

Inoltre, se è abilitata la sicurezza globale, verificare che l'ID utente disponga dell'autorizzazione di operatore.

Verificare che il processo del server cui si connette il client di gestione sia in esecuzione. Per assicurarsi che il client di gestione si colleghi automaticamente al processo del server, non utilizzare l'opzione `-conntype NONE` come opzione del comando.

### About this task

I passi di seguito riportati descrivono il modo in cui utilizzare lo script `bpcTemplates.jacl` per disinstallare le applicazioni che contengono modelli di processi di business o attività umane. È necessario arrestare un modello prima di poter disinstallare l'applicazione cui appartiene. È possibile utilizzare lo script `bpcTemplates.jacl` per arrestare e disinstallare i modelli in una fase.

Prima di disinstallare le applicazioni, è possibile eliminare le istanze di processo o di attività associate ai modelli nelle applicazioni, ad esempio utilizzando Business Process Choreographer Explorer. È possibile inoltre utilizzare l'opzione `-force` con lo script `bpcTemplates.jacl` per eliminare le istanze associate ai modelli, arrestare i modelli e disinstallarli in un'unica fase.

**Avvertenza:**

Poiché l'opzione **-force** elimina tutti dati delle istanze di processo e di attività, è opportuno utilizzarla con molta attenzione.

**Procedure**

1. Passare alla directory degli esempi di Business Process Choreographer.

Su piattaforme Windows, digitare:

```
cd root_installazione\ProcessChoreographer\admin
```

Su piattaforme Linux, UNIX, and i5/OS, digitare:

```
cd root_installazione/ProcessChoreographer/admin
```

2. Arrestare i modelli, quindi disinstallare le applicazioni corrispondenti.

Su piattaforme Windows, digitare:

```
root_installazione\bin\wsadmin -f bpcTemplates.jacl
 [-user user_name]
 [-password user_password]
 -uninstall application_name
 [-force]
```

Su piattaforme Linux, UNIX, and i5/OS, digitare:

```
root_installazione/bin/wsadmin -f bpcTemplates.jacl
 [-user user_name]
 [-password user_password]
 -uninstall application_name
 [-force]
```

Dove:

*user\_name*

Se è abilitata la sicurezza globale, fornire un ID utente per l'autenticazione.

*user\_password*

Se è abilitata la sicurezza globale, fornire una password utente per l'autenticazione.

*application\_name*

Se è abilitata la sicurezza globale, fornire una password utente per l'autenticazione.

**Results**

L'applicazione viene disinstallata.



---

## Capitolo 6. Installazione adattatori

Gli adattatori consentono alla propria applicazione di comunicare con altri componenti del sistema di informazioni enterprise.

Il processo utilizzato per l'installazione degli adattatori è descritto in Configurazione e utilizzo degli adattatori, al centro informazioni WebSphere Integration Developer.



---

## Capitolo 7. Installazione delle applicazioni EIS

Un modulo dell'applicazione EIS può essere distribuito su una piattaforma J2EE. La distribuzione risulta in un'applicazione, compressa come file EAR distribuito sul server. Tutte le risorse J2EE vengono create, l'applicazione viene configurata e pronta per essere eseguita.

### About this task

La distribuzione sulla piattaforma J2EE crea le seguenti risorse J2EE:

Tabella 42. Associazione di bind con risorse J2EE

Bind nel modulo SCA	Risorse J2EE generate	Risorse J2EE create
Importazione EIS	Riferimenti alle risorse generati nell'EJB di sessione del modulo.	ConnectionFactory
Esportazione EIS	I bean basati sul messaggio, generati o distribuiti, in base all'interfaccia di listener supportata da Resource Adapter.	ActivationSpec
Importazione JMS	L'MDB (Message Driven Bean) fornito dal runtime viene distribuito, i riferimenti alle risorse vengono generati nell'EJB di sessione del modulo. Si noti che MDB viene creato solo se l'importazione dispone di una destinazione di ricezione.	<ul style="list-style-type: none"><li>• ConnectionFactory</li><li>• ActivationSpec</li><li>• Destinations</li></ul>
Esportazione JMS	L'MDB (Message Driven Bean) fornito dal runtime viene distribuito, i riferimenti alle risorse generati nell'EJB di sessione del modulo	<ul style="list-style-type: none"><li>• ActivationSpec</li><li>• ConnectionFactory</li><li>• Destinations</li></ul>

Quando l'importazione o l'esportazione definisce una risorsa come ConnectionFactory, il riferimento della risorsa viene generato in un descrittore di distribuzione dell'EJB della sessione senza stato del modulo. Inoltre, il bind appropriato viene generato nel file di bind EJB. Il nome cui il riferimento della risorsa è legato, è il valore dell'attributo di destinazione, se presente, o il nome di ricerca JNDI assegnato alla risorsa, in base al nome del modulo e al nome di importazione.

Al momento della distribuzione, l'implementazione colloca il bean di sessione del modulo e lo utilizza per la ricerca delle risorse.

Durante la distribuzione dell'applicazione al server, l'attività di installazione EIS verifica l'esistenza della risorsa dell'elemento di bind. Se non esiste, e il file SCDDL specifica almeno una proprietà, la risorsa verrà creata e configurata dall'attività di installazione EIS. Se la risorsa non esiste, non venga eseguita alcuna azione. Si presuppone che la risorsa viene creata prima dell'esecuzione dell'applicazione.

Quando viene distribuita l'importazione JMS con una destinazione di ricezione, viene distribuito un bean basato sul messaggio (MDB). Esso rimane in ascolto delle risposte alle richieste inviate. L'MDB è associato (ascolta) alla destinazione inviata con la richiesta nel campo di intestazione JMSreplyTo del messaggio JMS. Quando viene recapitato il messaggio di risposta, l'MDB utilizza il relativo ID di correlazione per richiamare le informazioni di callback memorizzate nella Destinazione di callback, quindi richiama l'oggetto di callback.

L'attività di installazione crea la ConnectionFactory e tre destinazioni dalle informazioni nel file di importazione. Inoltre, crea ActivationSpec per abilitare l'MDB di runtime per ascoltare le risposte sulla Destinazione di ricezione. Le proprietà di ActivationSpec derivano dalle proprietà Destination/ConnectionFactory. Se il provider JMS è un SIBus Resource Adapter, vengono create le destinazioni SIBus corrispondenti alla destinazione JMS.

Quando viene distribuita l'esportazione JMS, viene distribuito anche un MDB (Message Driven Bean) (non lo stesso di quello distribuito per l'importazione JMS). Ascolta le richieste in entrata sulla destinazione di ricezione, quindi invia le richieste da elaborare da parte di SCA. L'attività di installazione crea la serie di risorse simile a quella per l'importazione JMS, un ActivationSpec, ConnectionFactory utilizzati per l'invio di una risposta e due destinazioni. Tutte le proprietà di tali risorse vengono specificate nel file di esportazione. Se il provider JMS è un SIBus Resource Adapter, vengono create le destinazioni SIBus corrispondenti alla destinazione JMS.

---

## Distribuzione di un modulo dell'applicazione EIS esistente sulla piattaforma J2SE

Il modulo EIS può essere distribuito alle piattaforme J2SE, tuttavia sarà supportata solo l'importazione EIS.

### Prima di iniziare

È necessario creare un modulo dell'applicazione EIS con un bind di importazione JMS nell'ambiente di sviluppo WebSphere Integration Development prima di iniziare questa attività.

### About this task

Un modulo dell'applicazione EIS dovrebbe essere fornito con un bind di importazione JMS quando si desidera accedere ai sistemi EIS in modo asincrono mediante l'utilizzo delle code messaggi.

La distribuzione alla piattaforma J2SE è la sola istanza in cui l'implementazione di bind può essere eseguita in modo non gestito. Il bind JMS richiede supporto JNDI asincrono non fornito dall'architettura del componente del servizio di base o da J2SE. L'architettura del connettore J2EE non supporta una comunicazione in entrata non gestita, eliminando così l'esportazione EIS.

Quando viene distribuito il modulo dell'applicazione EIS con l'importazione EIS in J2SE, oltre alle dipendenze del modulo, il WebSphere Adapter utilizzato dall'importazione deve essere specificato come dipendenza, nel manifest o in qualunque altra forma supportata da SCA.

## Distribuzione di un modulo dell'applicazione EIS sulla piattaforma J2EE

La distribuzione del modulo EIS nella piattaforma J2EE risulta in un'applicazione, compressa come file EAR, distribuita sul server. Tutte le risorse J2EE vengono create, l'applicazione viene configurata e pronta per essere eseguita.

### Prima di iniziare

È necessario creare un modulo EIS con un bind di importazione JMS nell'ambiente di sviluppo WebSphere Integration Development prima di iniziare questa attività.

### About this task

La distribuzione sulla piattaforma J2EE crea le seguenti risorse J2EE:

Tabella 43. Associazione di bind con risorse J2EE

Bind nel modulo SCA	Risorse J2EE generate	Risorse J2EE create
Importazione EIS	Riferimenti alle risorse generati nell'EJB di sessione del modulo.	ConnectionFactory
Esportazione EIS	I bean basati sul messaggio, generati o distribuiti, in base all'interfaccia di listener supportata da Resource Adapter.	ActivationSpec
Importazione JMS	L'MDB (Message Driven Bean) fornito dal runtime viene distribuito, i riferimenti alle risorse vengono generati nell'EJB di sessione del modulo. Si noti che MDB viene creato solo se l'importazione dispone di una destinazione di ricezione.	<ul style="list-style-type: none"><li>• ConnectionFactory</li><li>• ActivationSpec</li><li>• Destinations</li></ul>
Esportazione JMS	L'MDB (Message Driven Bean) fornito dal runtime viene distribuito, i riferimenti alle risorse generati nell'EJB di sessione del modulo	<ul style="list-style-type: none"><li>• ActivationSpec</li><li>• ConnectionFactory</li><li>• Destinations</li></ul>

Quando l'importazione o l'esportazione definisce una risorsa come ConnectionFactory, il riferimento della risorsa viene generato in un descrittore di distribuzione dell'EJB della sessione senza stato del modulo. Inoltre, il bind appropriato viene generato nel file di bind EJB. Il nome cui il riferimento della risorsa è legato, è il valore dell'attributo di destinazione, se presente, o il nome di ricerca JNDI assegnato alla risorsa, in base al nome del modulo e al nome di importazione.

Al momento della distribuzione, l'implementazione colloca il bean di sessione del modulo e lo utilizza per la ricerca delle risorse.

Durante la distribuzione dell'applicazione al server, l'attività di installazione EIS verifica l'esistenza della risorsa dell'elemento di bind. Se non esiste, e il file SCDL specifica almeno una proprietà, la risorsa verrà creata e configurata dall'attività di

installazione EIS. Se la risorsa non esiste, non venga eseguita alcuna azione. Si presuppone che la risorsa viene creata prima dell'esecuzione dell'applicazione.

Quando viene distribuita l'importazione JMS con una destinazione di ricezione, viene distribuito un bean basato sul messaggio (MDB). Esso rimane in ascolto delle risposte alle richieste inviate. L'MDB è associato (ascolta) alla destinazione inviata con la richiesta nel campo di intestazione JMSreplyTo del messaggio JMS. Quando viene recapitato il messaggio di risposta, l'MDB utilizza il relativo ID di correlazione per richiamare le informazioni di callback memorizzate nella Destinazione di callback, quindi richiama l'oggetto di callback.

L'attività di installazione crea la ConnectionFactory e tre destinazioni dalle informazioni nel file di importazione. Inoltre, crea ActivationSpec per abilitare l'MDB di runtime per ascoltare le risposte sulla Destinazione di ricezione. Le proprietà di ActivationSpec derivano dalle proprietà Destination/ConnectionFactory. Se il provider JMS è un SIBus Resource Adapter, vengono create le destinazioni SIBus corrispondenti alla destinazione JMS.

Quando viene distribuita l'esportazione JMS, viene distribuito anche un MDB (Message Driven Bean) (non lo stesso di quello distribuito per l'importazione JMS). Ascolta le richieste in entrata sulla destinazione di ricezione, quindi invia le richieste da elaborare da parte di SCA. L'attività di installazione crea la serie di risorse simile a quella per l'importazione JMS, un ActivationSpec, ConnectionFactory utilizzati per l'invio di una risposta e due destinazioni. Tutte le proprietà di tali risorse vengono specificate nel file di esportazione. Se il provider JMS è un SIBus Resource Adapter, vengono create le destinazioni SIBus corrispondenti alla destinazione JMS.

---

## Capitolo 8. Risoluzione dei problemi relativi ad una distribuzione non riuscita

Questo argomento descrive i passi da effettuare per determinare la causa di un problema durante la distribuzione di un'applicazione. Inoltre, presenta alcune possibili soluzioni.

### Prima di iniziare

Questo argomento presuppone quanto segue:

- Si dispone di una conoscenza di base del debug di un modulo.
- La registrazione è attiva durante la distribuzione del modulo.

### About this task

L'attività di risoluzione dei problemi relativi alla distribuzione inizia dopo aver ricevuto notifica di un errore. Esistono vari sintomi di distribuzione non riuscita che è necessario esaminare prima di effettuare qualche azione.

### Procedure

1. Determinare se l'installazione dell'applicazione ha avuto esito negativo.

Esaminare il file SystemOut.log per i messaggi che specificano la causa dell'errore. Alcuni dei motivi per cui un'applicazione potrebbe non essere installata comprendono quanto di seguito riportato:

- Si sta tentando di installare un'applicazione su più server nella stessa cella Network Deployment.
- Un'applicazione dispone dello stesso nome del modulo esistente nel Network Deployment in cui si sta installando l'applicazione.
- Si sta tentando di distribuire i moduli J2EE all'interno di un file EAR su diversi server di destinazione.

**Importante:** Se l'installazione non è riuscita e l'applicazione contiene dei servizi, è necessario rimuovere eventuali destinazioni SIBus o specifiche di attivazione J2C create prima dell'errore e poi tentare di installare di nuovo l'applicazione. Il modo più semplice per rimuovere queste risorse è fare clic su **Salva > Elimina tutto** dopo l'errore. Se inavvertitamente vengono salvate le modifiche, è necessario rimuovere manualmente le destinazioni SIBus e le specifiche di attivazione J2C (consultare Eliminazione delle destinazioni SIBus ed Eliminazione delle specifiche di attivazione J2C nella sezione Gestione).

2. Se l'applicazione è stata installata correttamente, esaminarla per determinare se l'avvio è riuscito.

Se l'avvio dell'applicazione non è riuscito, l'errore si è verificato quando il server ha tentato di inizializzare le risorse per l'applicazione.

- a. Esaminare il file SystemOut.log per i messaggi con le istruzioni sul modo in cui procedere.
- b. Determinare se le risorse richieste dall'applicazione sono disponibili e/o sono state avviate correttamente.

Le risorse non avviate impediscono l'esecuzione di un'applicazione. Questo protegge dalla perdita di informazioni. I motivi per cui una risorsa non viene avviata comprendono:

- Bind specificati in modo non corretto
  - Risorse non configurate correttamente
  - Risorse non comprese nel file RAR (resource archive)
  - Risorse Web non comprese nel file WAR (Web services archive)
- c. Determinare se manca qualche componente.
- Il motivo per cui manca un componente è un file EAR creato in modo non corretto. Assicurarsi che tutti i componenti richiesti dal modulo siano nelle cartelle corrette sul sistema di test su cui viene creato il file di archivio (JAR) Java. “Preparazione alla distribuzione su un server” contiene maggiori informazioni al riguardo.
3. Esaminare l’applicazione per controllare se le informazioni passano attraverso l’applicazione.
- Anche un’applicazione in esecuzione può non riuscire ad elaborare le informazioni. I motivi per cui ciò avviene sono simili a quelli illustrati al passo 2b a pagina 205.
- a. Determinare se l’applicazione utilizza servizi contenuti in un’altra applicazione. Assicurarsi che l’altra applicazione sia installata e sia stata avviata correttamente.
- b. Determinare se i bind di importazione ed esportazione per tutte le unità contenute in altre applicazioni utilizzate dall’applicazione in errore sono configurati correttamente. Utilizzare la console di gestione per esaminare e correggere i bind.
4. Risolvere il problema e riavviare l’applicazione.

---

## Eliminazione delle specifiche dell’attivazione J2C

Il sistema crea le specifiche per le applicazioni J2C durante l’installazione di un’applicazione che contiene i servizi. Talvolta, è necessario eliminare tali specifiche prima di reinstallare l’applicazione.

### Prima di iniziare

Se si sta eliminando la specifica a causa di un’installazione di un’applicazione non riuscita, assicurarsi che il modulo nel nome JNDI (Java Naming and Directory Interface) corrisponda al nome del modulo per cui si è verificato l’errore di installazione. La seconda parte del nome JNDI è il nome del modulo che ha implementato la destinazione. Ad esempio in `sca/SimpleBOCrsmA/ActivationSpec`, **SimpleBOCrsmA** è il nome del modulo.

**Ruolo di sicurezza necessario per questa attività:** Quando sono abilitate le autorizzazioni basate su sicurezza e ruolo, è necessario aver eseguito il login come amministratore o configuratore per eseguire questa attività.

### About this task

Eliminare le specifiche dell’attivazione J2C se inavvertitamente è stata salvata una configurazione dopo aver installato un’applicazione contenente i servizi e che non richiede le specifiche.

### Procedure

1. Ricercare la specifica dell’attivazione da eliminare.  
Le specifiche sono contenute nel pannello di Resource adapter. Passare a questo pannello facendo clic su **Risorse > Resource adapter**.



- a. Ricercare **Platform Messaging Component SPI Resource Adapter**.  
Per trovare questo adattatore, è necessario trovarsi nell'ambito del **nodo** di un server autonomo o nell'ambito di un **server** in un ambiente di distribuzione.
2. Visualizzare le specifiche di attivazione J2C associate a Platform Messaging Component SPI Resource Adapter.  
Fare clic sul nome del Resource adapter, quindi viene visualizzato il pannello successivo con le specifiche associate.
3. Eliminare tutte le specifiche con un **Nome JNDI** corrispondente al nome del modulo che si sta eliminando.
  - a. Selezionare la casella di spunta accanto alle specifiche appropriate.
  - b. Fare clic su **Elimina**.

### Results

Il sistema rimuove le specifiche selezionate dalla visualizzazione.

### Operazioni successive

Salvare le modifiche.

---

## Eliminazione delle destinazioni SIBus

Le destinazioni SIBus sono connessioni che rendono i servizi disponibili alle applicazioni. Talvolta, è necessario rimuovere le destinazioni.

### Prima di iniziare

Se si sta eliminando la destinazione a causa dell'installazione di un'applicazione non riuscita, assicurarsi che il modulo nel nome della destinazione corrisponda al nome del modulo la cui installazione ha causato l'errore. La seconda parte della destinazione è il nome del modulo che ha implementato la destinazione. Ad esempio, in `sca/SimpleBOCrsmA/component/test/sca/cros/simple/cust/` Customer, **SimpleBOCrsmA** è il nome del modulo.

**Ruolo di sicurezza necessario per questa attività:** Quando sono abilitate le autorizzazioni basate su sicurezza e ruolo, è necessario aver eseguito il login come amministratore o configuratore per eseguire questa attività.

### About this task

Eliminare le destinazioni SIBus se inavvertitamente è stata salvata una configurazione dopo l'installazione di un'applicazione contenente servizi o le destinazioni non sono più necessarie.

**Nota:** Questa attività elimina la destinazione solo dal bus del sistema SCA. È necessario rimuovere le voci dal bus dell'applicazione anche prima di reinstallare un'applicazione contenente i servizi (consultare l'argomento Eliminazione delle specifiche di attivazione J2C nella sezione Gestione dell'Infocenter).

### Procedure

1. Collegarsi alla console di gestione.
2. Visualizzare le destinazioni sul bus del sistema SCA.  
Passare al pannello facendo clic su **Integrazione servizi > Bus**

3. Selezionare le destinazioni del bus del sistema SCA.  
Nella visualizzazione, fare clic su **SCA.SYSTEM.cellname.Bus**, dove *cellname* è il nome della cella contenente il modulo con le destinazioni che si stanno eliminando.
4. Eliminare le destinazioni che contengono il nome del modulo che corrisponde al modulo che si sta rimuovendo.
  - a. Fare clic sulla casella di spunta accanto alle destinazioni pertinenti.
  - b. Fare clic su **Elimina**.

### **Results**

Il pannello visualizza solo le destinazioni restanti.

### **Operazioni successive**

Eliminare le specifiche di attivazione J2C relative al modulo che ha creato queste destinazioni.

---

## Parte 3. Appendici



---

## Informazioni particolari

Queste informazioni sono state sviluppate per i prodotti ed i servizi offerti negli Stati Uniti d'America.

In altri paesi, l'IBM potrebbe non offrire tali prodotti, servizi o funzioni illustrati in questo documento. Per le informazioni sui prodotti ed i servizi disponibili al momento nella propria area, rivolgersi al rivenditore IBM locale. Qualsiasi riferimento ad un prodotto, programma o servizio IBM non implica o intende dichiarare che solo quel prodotto, programma o servizio IBM può essere utilizzato. Qualsiasi prodotto funzionalmente equivalente al prodotto, programma o servizio che non violi alcun diritto di proprietà intellettuale IBM può essere utilizzato. Tuttavia, è responsabilità dell'utente valutare e verificare il funzionamento di eventuali prodotti, programmi o servizi non IBM.

IBM può avere brevetti o richieste di brevetti in corso relativi a quanto trattato nella presente pubblicazione. La fornitura di questo documento non implica la concessione di alcuna licenza su di essi. È possibile inviare per iscritto richieste di licenze a:

*IBM Director of Licensing  
IBM Corporation  
Schoenaicher Str. 220  
D-7030 Boeblingen  
U.S.A.*

Per richieste di licenze relative ad informazioni double-byte (DBCS), contattare il Dipartimento di Proprietà Intellettuale IBM nel proprio paese o inviare richieste per iscritto a:

*IBM World Trade Asia Corporation Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106-0032, Japan*

**Il seguente paragrafo non si applica al Regno Unito e a tutti i paesi in cui tali disposizioni sono in contrasto con le leggi nazionali:** L'INTERNATIONAL BUSINESS MACHINES CORPORATION FORNISCE QUESTA PUBBLICAZIONE "NELLO STATO IN CUI SI TROVA" SENZA ALCUNA GARANZIA, ESPLICITA O IMPLICITA, IVI INCLUSE, MA NON LIMITATE, GARANZIE DI IDONEITÀ E COMMERCIALIZZABILITÀ AD UNO SCOPO PARTICOLARE. Alcuni stati non consentono la rinuncia a garanzie esplicite o implicite in determinate transazioni, quindi, la presente dichiarazione potrebbe non essere a voi applicabile.

Questa pubblicazione potrebbe contenere imprecisioni tecniche o errori tipografici. Le correzioni relative saranno incluse nelle nuove edizioni della pubblicazione. L'IBM si riserva il diritto di apportare modifiche al prodotto o al programma descritto in questa pubblicazione in qualsiasi momento e senza alcun preavviso.

Tutti i riferimenti a siti Web non IBM contenuti in questo documento sono forniti solo per consultazione. I materiali contenuti in questi siti Web non fanno parte di questo prodotto e il loro utilizzo è a discrezione dell'utente.

IBM può utilizzare o distribuire qualsiasi informazione fornita dall'utente nel modo più appropriato senza incorrere in alcuna obbligazione.

I licenziatari di questo programma che desiderano avere informazioni allo scopo di abilitare: (i) lo scambio di informazioni tra i programmi creati indipendentemente e gli altri programmi (incluso il presente) e (ii) il reciproco utilizzo di informazioni che sono state scambiate, dovrebbero contattare:

IBM Corporation  
577 Airport Blvd., Suite 800  
Burlingame, CA 94010  
Deutschland

Queste informazioni possono essere rese disponibili sulla base di condizioni contrattuali appropriate, in alcuni casi dietro pagamento di un canone.

Il programma concesso in licenza in questo documento e tutto il materiale su licenza ad esso relativo sono forniti dalla IBM nel rispetto di termini dell'IBM Customer Agreement, IBM International Program License Agreement o qualunque altro accordo equivalente.

Tutti i dati relativi alle prestazioni contenuti in questa pubblicazione sono stati determinati in ambiente controllato. Pertanto, i risultati ottenuti in ambienti operativi diversi possono variare in modo considerevole. Alcune misure potrebbero essere state fatte su sistemi di livelli di sviluppo per cui non si garantisce che queste saranno uguali su tutti i sistemi disponibili. Inoltre, alcuni dati potrebbero essere stati ricavati mediante deduzione. I risultati possono quindi variare. Gli utenti di questa pubblicazione devono verificare che i dati siano applicabili al loro specifico ambiente.

Le informazioni relative a prodotti non IBM sono state ottenute dai fornitori di tali prodotti. L'IBM non ha verificato tali prodotti e, pertanto, non può garantirne l'accuratezza delle prestazioni o la compatibilità o comunque qualunque reclamo relativo a prodotti non IBM. Eventuali commenti relativi alle prestazioni dei prodotti non IBM devono essere indirizzati ai fornitori di tali prodotti.

Tutte le dichiarazioni relative al futuro orientamento e le intenzioni dell'IBM sono soggette a modifica o senza alcun preavviso e rappresentano solo scopi e obiettivi.

Queste informazioni contengono esempi di dati e report utilizzati quotidianamente nelle operazioni di business. Per meglio illustrarli, tali esempi contengono nomi di persone, società, marchi e prodotti. Tutti i nomi contenuti nel manuale sono fittizi e ogni riferimento a nomi ed indirizzi reali è puramente casuale.

#### LICENZA DEL COPYRIGHT:

Queste informazioni contengono esempi di programmi applicativi in lingua di origine, che illustrano le tecniche di programmazione su diverse piattaforme operative. È possibile copiare, modificare e distribuire questi esempi di programmi sotto qualsiasi forma senza alcun pagamento alla IBM, allo scopo di sviluppare, utilizzare, commercializzare o distribuire i programmi applicativi in modo conforme alle API (Application Programming Interface) a seconda della piattaforma operativa per cui gli esempi dei programmi sono stati scritti. Questi esempi non sono stati testati approfonditamente tenendo conto di tutte le condizioni possibili. L'IBM, pertanto non può garantire l'affidabilità, la funzionalità e l'accuratezza delle prestazioni di tali programmi.

Ciascuna copia o qualsiasi porzione di questi programmi di esempio o qualsiasi lavoro derivato, deve includere il seguente avviso di copyright: (c) (nome della società) (anno). Parti di questo codice derivano dai programmi di esempio IBM Corp. (c) Copyright IBM Corp. \_inserire anno o anni\_. Tutti i diritti riservati.

Se si visualizza la bozza di stampa di tali informazioni, le fotografie e le illustrazioni a colori potrebbero non essere visualizzate.

## Informazioni sull'interfaccia di programmazione

Le informazioni sull'interfaccia di programmazione, se fornite, hanno lo scopo di supportare l'utente nella creazione del software di applicazione utilizzando questo programma.

Le interfacce di programmazione di utilizzo generico consentono all'utente di scrivere il software dell'applicazione che ottiene i servizi di questi strumenti del programma.

Tuttavia, queste informazioni possono contenere anche diagnosi, modifiche ed informazioni sull'ottimizzazione. Le informazioni sulle diagnosi, le modifiche e sull'ottimizzazione vengono fornite come supporto per l'esecuzione del debug del software applicativo.

**Avvertenza:** Non utilizzare queste informazioni sulle diagnosi, sulle modifiche e sull'ottimizzazione come interfaccia di programmazione, poiché tali informazioni sono soggette a cambiamenti.

## Marchi e marchi di servizio

IBM, il logo IBM, developerWorks, WebSphere, e z/OS sono marchi registrati di International Business Machines Corporation negli Stati Uniti, in altri paesi o in entrambi.

Adobe è un marchio registrato di Adobe Systems Incorporated negli Stati Uniti e/o in altri paesi.

Java e tutti i marchi basati su Java sono marchi della Sun Microsystems, Inc. negli Stati Uniti e/o negli altri paesi.

Altri nomi di società, prodotti o servizi potrebbero essere marchi o marchi di servizio di terzi.

Questo prodotto include software sviluppato da Eclipse Project (<http://www.eclipse.org>).



IBM WebSphere Process Server for Multiplatforms, Versione 6.1.0









Stampato in Italia