WebSphere® Process Server for Multiplatforms

Version 6.1.0

Module entwickeln und implementieren

WebSphere® Process Server for Multiplatforms

Version 6.1.0

Module entwickeln und implementieren

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen unter "Bemerkungen" am Ende dieses Dokuments gelesen werden.

März 2008

Diese Ausgabe gilt für Version 6, Release 1, Modifikation 0 von WebSphere Process Server for Multiplatforms (Produktnummer 5724-L01) und alle nachfolgenden Releases und Modifikationen, sofern in neuen Ausgaben keine anderen Angaben gemacht werden.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM WebSphere Process Server for Multiplatforms Version 6.1.0, Developing and Deploying Modules,
herausgegeben von International Business Machines Corporation, USA

© Copyright International Business Machines Corporation 2005, 2008
© Copyright IBM Deutschland GmbH 2008

Informationen, die nur für bestimmte Länder Gültigkeit haben und für Deutschland, Österreich und die Schweiz nicht zutreffen, wurden in dieser Veröffentlichung im Originaltext übernommen.

Möglicherweise sind nicht alle in dieser Übersetzung aufgeführten Produkte in Deutschland angekündigt und verfügbar; vor Entscheidungen empfiehlt sich der Kontakt mit der zuständigen IBM Geschäftsstelle.

Änderung des Textes bleibt vorbehalten.

Herausgegeben von:
SW TSC Germany
Kst. 2877
März 2008

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellen

Teil 1. Anwendungen entwickeln

Kapitel 1. Übersicht über die Modulentwicklung

Ein Modul ist eine grundlegende Verteilungseinheit für eine WebSphere Process Server-Anwendung. Ein Modul enthält mindestens eine Komponentenbibliothek und Bereitstellungsmodule, die von der Anwendung verwendet werden. Eine Komponente kann auf andere Servicekomponenten verweisen. Beim Entwickeln von Modulen muss sichergestellt werden, dass die für die Anwendung erforderlichen Komponenten, Bereitstellungsmodule und Bibliotheken (Sammlungen von Artefakten, auf die das Modul verweist) auf dem Produktionsserver verfügbar sind.

WebSphere Integration Developer ist das Haupttool zum Entwickeln von Modulen für die Implementierung in WebSphere Process Server. Sie können Module zwar auch in anderen Umgebungen entwickeln, aber die Verwendung von WebSphere Integration Developer ist unbedingt zu empfehlen.

WebSphere Process Server unterstützt zwei Servicemodultypen: Module für Business-Services und Mediationsmodule. Ein Modul für Business-Services implementiert die Logik eines Prozesses. Ein Mediationsmodul ermöglicht die Kommunikation zwischen Anwendungen durch die Umsetzung des Serviceaufrufs in ein Format, das vom Ziel verstanden wird, durch Übergeben der Anforderung an das Ziel und durch Zurückgeben des Ergebnisses an den Ersteller.

In den folgenden Abschnitten wird das Implementieren und Aktualisieren von Modulen in WebSphere Process Server beschrieben.

Übersicht über Komponenten

Eine Komponente ist der Grundbaustein zum Einbinden wiederverwendbarer Geschäftslogik. Eine Servicekomponente wird Schnittstellen, Referenzen und Implementierungen zugeordnet. Die Schnittstelle definiert eine Vereinbarung zwischen einer Servicekomponente und einer aufrufenden Komponente. In WebSphere Process Server kann ein Servicemodul eine Servicekomponente für die Verwendung durch andere Module exportieren oder eine zu verwendende Servicekomponente importieren. Zum Aufrufen einer Servicekomponente verweist ein aufrufendes Modul auf die Schnittstelle zu der Servicekomponente. Die Referenzen auf die Schnittstellen werden durch Konfigurieren der Referenzen vom aufrufenden Modul an die entsprechenden Schnittstellen aufgelöst.

Zum Entwickeln eines Moduls sind folgende Aktivitäten erforderlich:

1. Definieren von Schnittstellen für die Komponenten in dem Modul.
2. Definieren, Ändern oder Bearbeiten der von Servicekomponenten verwendeten Business-Objekte.
3. Definieren oder Ändern von Servicekomponenten über die entsprechenden Schnittstellen.

   Anmerkung: (Eine Servicekomponente wird durch ihre Schnittstelle definiert.)
4. (Optional) Exportieren oder Importieren von Servicekomponenten.

5. Erstellen einer EAR-Datei zum Installieren eines Moduls, das Komponenten verwendet. Die Datei wird entweder mit der EAR-Exportfunktion in WebSphere Integration Developer oder dem Befehl serviceDeploy zum Erstellen einer EAR-Datei zum Installieren eines Servicemoduls, das Servicekomponenten verwendet, erstellt.

Entwicklungstypen

WebSphere Process Server stellt ein Komponentenprogrammiermodell bereit, um ein serviceorientiertes Programmierkonzept zu ermöglichen. Bei der Verwendung dieses Modells exportiert ein Anbieter Schnittstellen einer Servicekomponente, damit ein Anwender diese Schnittstellen importieren und die Servicekomponente wie eine lokale Komponente verwenden kann. Ein Entwickler verwendet entweder stark typisierte oder dynamisch typisierte Schnittstellen, um die Servicekomponente zu implementieren oder aufzurufen. Die Schnittstellen und die dazugehörigen Methoden werden im Abschnitt über Referenzen in diesem Information Center beschrieben.

Nach dem Installieren von Servicemodulen auf Ihren Servern können Sie mit der Administrationskonsole die Zielkomponente für eine Referenz aus einer Anwendung ändern. Das neue Ziel muss denselben Business-Objekttyp akzeptieren und dieselbe Operation ausführen, die die Referenz aus der Anwendung anfordert.

Hinweise zur Servicekomponentenentwicklung

Stellen Sie sich beim Entwickeln einer Servicekomponente die folgenden Fragen:

- Wird diese Servicekomponente exportiert und von einem anderen Modul verwendet?

  Wenn ja, stellen Sie sicher, dass auch ein anderes Modul die von Ihnen für die Komponente definierte Schnittstelle verwenden kann.
- Dauert die Ausführung dieser Servicekomponente relativ lange?

  Wenn ja, sollten Sie in Erwägung ziehen, eine asynchrone Schnittstelle für die Servicekomponente zu implementieren.
- Ist es von Vorteil, die Servicekomponente zu dezentralisieren?

  Wenn ja, ziehen Sie in Erwägung, eine Kopie der Servicekomponente in einem Servicemodul bereitzustellen, das auf einem Server-Cluster implementiert wird, um die Vorteile der Parallelverarbeitung zu nutzen.
- Ist für Ihre Anwendung eine Mischung aus ein- und zweiphasigen COMMIT-Ressourcen erforderlich?

  Wenn ja, stellen Sie sicher, dass Sie die Unterstützung des letzten Teilnehmers für die Anwendung aktivieren.

  Anmerkung: Wenn Sie Ihre Anwendung mit WebSphere Integration Developer oder die installierbare EAR-Datei mit dem Befehl serviceDeploy erstellen, aktivieren diese Tools automatisch die Unterstützung für die Anwendung. Informationen finden Sie unter „1-PC- und 2-PC-Ressourcen in einer Transaktion verwenden” im Information Center für WebSphere Application Server Network Deployment.

Servicemodule entwickeln

Eine Servicekomponente muss in einem Servicemodul enthalten sein. Das Entwickeln von Servicemodulen, die Servicekomponenten enthalten sollen, ist der Schlüssel zum Bereitstellen von Services für andere Module.

Bei dieser Task wird angenommen, dass die Analyse der Voraussetzungen zeigt, dass es vorteilhaft ist, eine Servicekomponente für die Verwendung durch andere Module zu implementieren.

Nach dem Analysieren der Voraussetzungen stellen Sie möglicherweise fest, dass das Bereitstellen und Verwenden von Servicekomponenten eine effiziente Methode zum Verarbeiten von Informationen ist. Wenn Sie zu der Auffassung gelangen, dass wiederverwendbare Servicekomponenten in Ihrer Umgebung von Vorteil sind, erstellen Sie ein Servicemodul, das die Servicekomponenten enthalten soll.

1. Identifizieren Sie Servicekomponenten, die von anderen Modulen verwendet werden können.

   Fahren Sie nach dem Identifizieren der Servicekomponenten mit Servicekomponenten entwickeln fort.
2. Identifizieren Sie Servicekomponenten in einer Anwendung, die Servicekomponenten in anderen Servicemodulen verwenden kann.

   Fahren Sie nach dem Identifizieren der Servicekomponenten und der Zielkomponenten mit Komponenten aufrufen fort.
3. Stellen Sie Verbindungen zwischen den Clientkomponenten und den Zielkomponenten her.

Servicekomponenten entwickeln

Entwickeln Sie Servicekomponenten, um wiederverwendbare Logik für mehrere Anwendungen auf Ihrem Server bereitzustellen.

In dieser Task wird vorausgesetzt, dass Sie bereits Verarbeitungsabläufe entwickelt und identifiziert haben, die für mehrere Module hilfreich sind.

Eine Servicekomponente kann von mehreren Modulen verwendet werden. Durch Exportieren wird eine Servicekomponente für andere Module verfügbar gemacht, die über eine Schnittstelle auf die Servicekomponente verweisen. In dieser Task wird beschrieben, wie eine Servicekomponente erstellt wird, die auch von anderen Modulen verwendet werden kann.

Anmerkung: Eine einzige Servicekomponente kann mehrere Schnittstellen enthalten.

1. Definieren Sie das Datenobjekt zum Verschieben von Daten zwischen dem aufrufenden Element und der Servicekomponente.

   Das Datenobjekt und der dazugehörige Typ sind Teil der Schnittstelle zwischen den aufrufenden Elementen und der Servicekomponente.
2. Definieren Sie eine Schnittstelle, über die aufrufende Elemente auf die Servicekomponente verweisen können.

   Diese Schnittstellendefinition benennt die Servicekomponente und listet alle in der Servicekomponente verfügbaren Methoden auf.
3. Entwickeln Sie die Klasse, die die Implementierung definiert.
   - Wenn es um eine Komponente mit langer Laufzeit (asynchrone Komponente) geht, fahren Sie mit dem Schritt 4 auf Seite 6 fort.

- Wenn es nicht um eine Komponente mit langer Laufzeit (asynchrone Komponente) geht, fahren Sie mit dem Schritt 5 fort.

4. Entwickeln Sie eine asynchrone Implementierung.

   Wichtig: In einer asynchronen Komponente darf das Merkmal joinsTransaction nicht auf `true` gesetzt sein.
   a. Definieren Sie die Schnittstelle, die die synchrone Servicekomponente repräsentiert.
   b. Definieren Sie die Implementierung der Servicekomponente.
   c. Fahren Sie mit Schritt 6 fort.
5. Entwickeln Sie eine synchrone Implementierung.
   a. Definieren Sie die Schnittstelle, die die synchrone Servicekomponente repräsentiert.
   b. Definieren Sie die Implementierung der Servicekomponente.
6. Speichern Sie die Komponentenschnittstellen und -implementierungen in Dateien mit der Erweiterung .java.
7. Packen Sie das Servicemodul und die erforderlichen Ressourcen in eine JAR-Datei.

   In dem Artikel „Modul auf einem Produktionsserver implementieren" in diesem Information Center finden Sie eine Beschreibung der Schritte 7 bis 9.
8. Führen Sie den Befehl serviceDeploy aus, um eine installierbare EAR-Datei mit der Anwendung zu erstellen.
9. Installieren Sie die Anwendung auf dem Serverknoten.
10. Optional: Konfigurieren Sie die Verbindungen zwischen den aufrufenden Elementen und der entsprechenden Servicekomponente, wenn eine Servicekomponente in einem anderen Servicemodul aufgerufen wird.

    Im Abschnitt über „Verwaltung" in diesem Information Center wird das Konfigurieren der Verbindungen beschrieben.

Beispiele für das Entwickeln von Komponenten

Dieses Beispiel zeigt eine synchrone Servicekomponente, die eine einzige Methode (CustomerInfo) implementiert. Im ersten Abschnitt wird die Schnittstelle zu der Servicekomponente definiert, die eine Methode getCustomerInfo implementiert.

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

Der folgende Codeabschnitt implementiert die Servicekomponente.

```
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;
 }
}
```

In diesem Beispiel wird eine asynchrone Servicekomponente entwickelt. Im ersten Codeabschnitt wird die Schnittstelle zu der Servicekomponente definiert, die eine Methode getQuote implementiert.

```
public interface StockQuote {

 public float getQuote(String symbol);
}
```

Der folgende Abschnitt ist die Implementierung der StockQuote zugeordneten Klasse.

```
public class StockQuoteImpl implements StockQuote {

 public float getQuote(String symbol) {


     return 100.0f;
 }
}
```

Der nächste Codeabschnitt implementiert die asynchrone Schnittstelle StockQuote-Async.

```
public interface StockQuoteAsync {

 // Verzögerte Antwort
 public Ticket getQuoteAsync(String symbol);
 public float getQuoteResponse(Ticket ticket, long timeout);

 // Rückruf
 public Ticket getQuoteAsync(String symbol, StockQuoteCallback callback);
}
```

Dieser Abschnitt ist die Schnittstelle StockQuoteCallback, die die Methode onGet-QuoteResponse definiert.

```
public interface StockQuoteCallback {

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

Rufen Sie den Service auf.

Komponenten aufrufen

Komponenten mit Modulen können Komponenten auf jedem beliebigen Knoten eines WebSphere Process Server-Clusters verwenden.

Vergewissern Sie sich vor dem Aufrufen einer Komponente, dass das Modul, in dem die Komponente enthalten ist, auf WebSphere Process Server installiert ist.

Komponenten können jede beliebige in einem WebSphere Process Server-Cluster verfügbare Servicekomponente verwenden, wenn der Name der Komponente verwendet und der von der Komponente erwartete Datentyp übergeben wird. Zum Aufrufen einer Komponente in dieser Umgebung muss die erforderliche Komponente lokalisiert und anschließend die Referenz auf die erforderliche Komponente erstellt werden.

Anmerkung: Eine Komponente in einem Modul kann eine Komponente in demselben Modul aufrufen; dies wird auch Intramodulaufruf genannt. Exportieren Sie

zum Implementieren von externen Aufrufen (modulübergreifende Aufrufe) die Schnittstelle in die bereitstellende Komponente, und importieren Sie die Schnittstelle in die aufrufende Komponente.

Wichtig: Wenn Sie eine Komponente aufrufen, die sich auf einem anderen Server als dem befindet, auf dem das aufrufende Modul ausgeführt wird, müssen Sie weitere Konfigurationen für die Server durchführen. Die erforderlichen Konfigurationen sind davon abhängig, ob die Komponente asynchron oder synchron aufgerufen wird. Die Vorgehensweise zum Konfigurieren der Anwendungsserver in diesem Fall wird in den zugehörigen Tasks beschrieben.

1. Stellen Sie fest, welche Komponenten für das aufrufende Modul erforderlich sind.

   Notieren Sie den Namen der Schnittstelle innerhalb einer Komponente und den für die Schnittstelle erforderlichen Datentyp.
2. Definieren Sie ein Datenobjekt.

   Die Eingabe oder Rückgabe kann auch aus einer Java-Klasse bestehen, aber ein Servicedatenobjekt ist dafür am besten geeignet.
3. Lokalisieren Sie die Komponente.
   a. Verwenden Sie die Klasse ServiceManager, um die für das aufrufende Modul verfügbaren Verweise zu ermitteln.
   b. Verwenden Sie die Methode locateService(), um die Komponente zu finden.

      Je nach Komponente kann die Schnittstelle einen WSDL-Porttyp (WSDL = Web Service Descriptor Language) aufweisen oder eine Java-Schnittstelle sein.
4. Rufen Sie die Komponente synchron oder asynchron auf.

   Sie können die Komponente über eine Java-Schnittstelle aufrufen oder dynamisch mit der Methode invoke().
5. Verarbeiten der Rückgabedaten.

   Möglicherweise generiert die Komponente eine Ausnahmebedingung, d. h. der Client muss diese Situation verarbeiten können.

Beispiel für das Aufrufen einer Komponente

Das folgende Beispiel erstellt eine Klasse ServiceManager.

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

Im folgenden Beispiel wird die Klasse ServiceManager verwendet, um eine Liste der Komponenten aus einer Datei abzurufen, die die Komponentenverweise enthält.

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

Der folgende Code lokalisiert eine Komponente, von der die Java-Schnittstelle StockQuote implementiert wird.

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

Der folgende Code lokalisiert eine Komponente, von der eine Java-Schnittstelle oder eine WSDL-Porttypschnittstelle implementiert wird. Das aufrufende Modul verwendet die Schnittstelle Service für Interaktionen mit der Komponente.

Tipp: Wenn die Komponente eine Java-Schnittstelle implementiert, kann sie über die Schnittstelle oder mit der Methode invoke() aufgerufen werden.

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

Das folgende Beispiel zeigt den Code für MyValue, der eine andere Komponente aufruft.

```
public class MyValueImpl implements MyValue {

 public float myValue throws MyValueException {

  ServiceManager serviceManager = new ServiceManager();

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

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

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

         // Aufrufen
       StockQuoteAsync sQuote =
      (StockQuoteAsync)serviceManager.locateService("stockQuote");
       Ticket ticket =  sQuote.getQuoteAsync(customer.getSymbol());
    // ... Andere Aktion ausführen ...
       quote =  sQuote.getQuoteResponse(ticket, Service.WAIT);

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

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

Konfigurieren Sie die Verbindungen zwischen den Verweisen des aufrufenden Moduls und den Komponentenschnittstellen.

Komponente dynamisch aufrufen

Wenn ein Modul eine Komponente aufruft, die über eine WSDL-Porttypschnittstelle (WSDL = Web Service Descriptor Language) verfügt, muss die Komponente dynamisch mit der Methode invoke() aufgerufen werden.

Bei dieser Task wird vorausgesetzt, dass eine aufrufende Komponente eine andere Komponente dynamisch aufruft.

Bei einer WSDL-Porttypschnittstelle muss eine aufrufende Komponente die Methode invoke() verwenden, um die gewünschte Komponente aufzurufen. Ein aufrufendes Modul kann auf diese Weise auch eine Komponente aufrufen, die über eine Java-Schnittstelle verfügt.

1. Stellen Sie fest, welches Modul die erforderliche Komponente enthält.
2. Stellen Sie fest, welche Matrix für die Komponente erforderlich ist.

   Drei Typen von Eingabematrix kommen in Frage:
   - Primitive Java-Typen oder -Arrays in Großschreibung
   - Gewöhnliche Java-Klassen oder Arrays der Klassen
   - Servicedatenobjekte (SDOs)

3. Definieren Sie eine Matrix für die Antwort der Komponente.
   Die Antwortmatrix kann vom gleichen Typ sein wie die Eingabematrix.
4. Rufen Sie die erforderliche Komponente mit der Methode invoke() auf, und übergeben Sie das Matrixobjekt an die Komponente.
5. Verarbeiten Sie das Ergebnis.

Beispiele für das dynamische Aufrufen einer Komponente

Im folgenden Beispiel ruft ein Modul mit der Methode invoke() eine Komponente auf, die Primitive Java-Datentypen in Großschreibung verwendet.

```
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);
```

Im folgenden Beispiel wird die Methode invoke mit einer WSDL-Porttyp-schnittstelle als Ziel aufgerufen.

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

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

 DataObject wrapBo = factory.createByElement
  ("http://MultiCallWSServerOne/wsdl/ServerOneInf", "methodOne");
   wrapBo.set("input1", dob); //wrapBo kapselt alle Parameter von methodOne
   wrapBo.set("input2", "XXXX");
   wrapBo.set("input3", "yyyy");

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

Übersicht über die Isolation von Modulen und Zielen

Bei der Entwicklung von Modulen stellen Sie in der Regel fest, dass bestimmte Services von mehreren Modulen genutzt werden können. Eine solche Wiederverwendung von Services verringert den Entwicklungszyklus und die Kosten. Wenn Sie über einen Service verfügen, der von vielen Modulen verwendet wird, sollten Sie die aufrufenden Module vom Ziel isolieren, damit bei einem Upgrade des Ziels der Wechsel auf den neuen Service für das aufrufende Modul transparent ist. Das vorliegende Thema stellt dem einfachen Aufrufmodell das isolierte Aufrufmodell gegenüber und enthält ein Beispiel für die Nützlichkeit einer Isolierung. Das spezielle Beispiel, anhand dessen die Isolierung beschrieben wird, stellt jedoch nicht die einzige Methode dar, mit der Module von Zielen isoliert werden können.

Einfaches Aufrufmodell

Beim Entwickeln eines Moduls können Sie Services verwenden, die in anderen Modulen vorhanden sind. Hierzu importieren Sie den Service in das Modul und rufen anschließend diesen Service auf. Der importierte Service ist mit dem durch das andere Modul exportierten Service entweder in WebSphere Integration Developer oder durch die Bindung des Services in der Administrationskonsole „verbunden". Dieses Modell ist in der Abbildung Einfaches Aufrufmodell dargestellt.

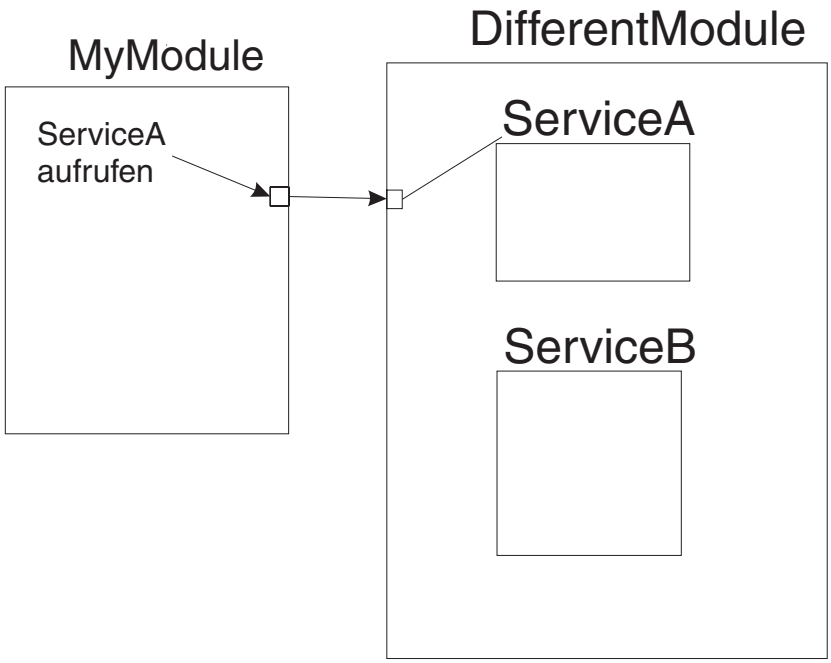


Abbildung 1. Einfaches Aufrufmodell

Isoliertes Aufrufmodell

Um das Ziel eines Aufrufs zu ändern, ohne hierbei die aufrufenden Module zu stoppen, können Sie die aufrufenden Module vom Ziel des Aufrufs isolieren. Hierdurch können die Module die Verarbeitung fortsetzen, während Sie das Ziel ändern. Dies liegt daran, dass Sie nicht das Modul selbst, sondern das nachgelagerte Ziel ändern. Die Abbildung Beispiel für die Isolation von Anwendungen macht deutlich, wie Sie mit der Isolation das Ziel ändern können, ohne den Status des aufrufenden Moduls zu beeinflussen.

Beispiel für die Isolation von Anwendungen

Bei Verwendung des einfachen Aufrufmodells würden mehrere Module, die denselben Service aufrufen, in etwa der Abbildung Aufruf eines Services durch mehrere Anwendungen entsprechen. MODA, MODB und MODC rufen den Service CalculateFinalCost auf.

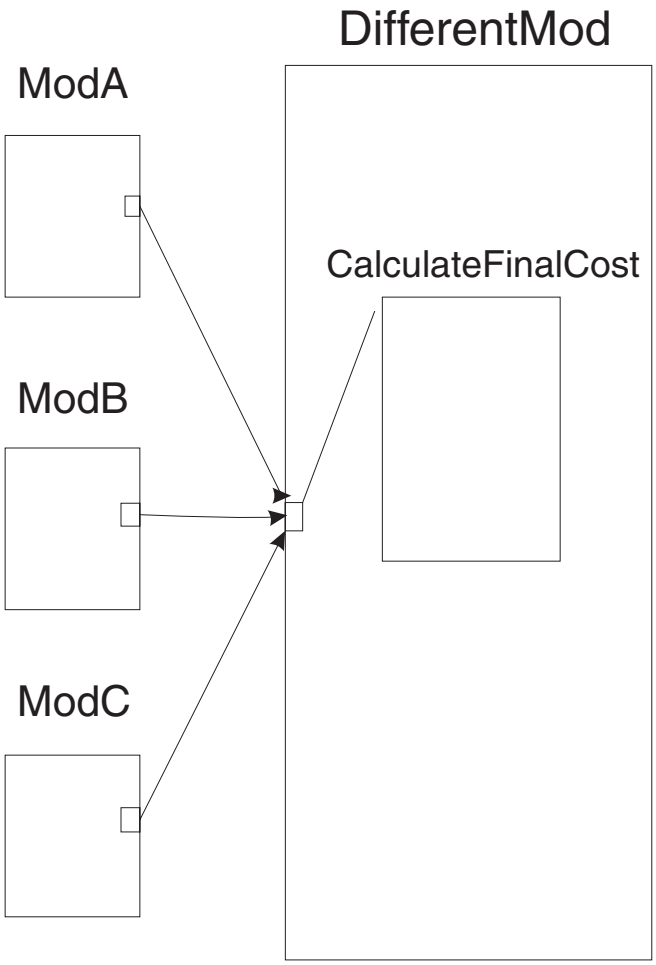


Abbildung 2. Aufruf eines Services durch mehrere Anwendungen

Der von CalculateFinalCost bereitgestellte Service muss aktualisiert werden, damit neue Kosten in allen Modulen wiedergegeben werden, die diesen Service verwenden. Das Entwicklerteam erstellt und testet den neuen Service UpdatedCalculateFinal, um die Änderungen zu integrieren. Danach kann der neue Service in der Produktion eingesetzt werden. Ohne eine Isolation müssten Sie alle Module, die den Service CalculateFinalCost aufrufen, so aktualisieren, dass der Service UpdateCalculateFinal aufgerufen wird. Bei einem Einsatz der Isolation müssen Sie lediglich die Bindung ändern, die das Puffermodul mit dem Ziel verbindet.

Anmerkung: Wenn Sie den Service auf diese Weise ändern, können Sie weiterhin den ursprünglichen Service für andere Module bereitstellen, die ihn möglicherweise benötigen.

Bei Verwendung der Isolation erstellen Sie zwischen den Anwendungen und dem Ziel ein Puffermodul (siehe Isoliertes Aufrufmodell mit Aufruf von UpdateCalculateFinal).

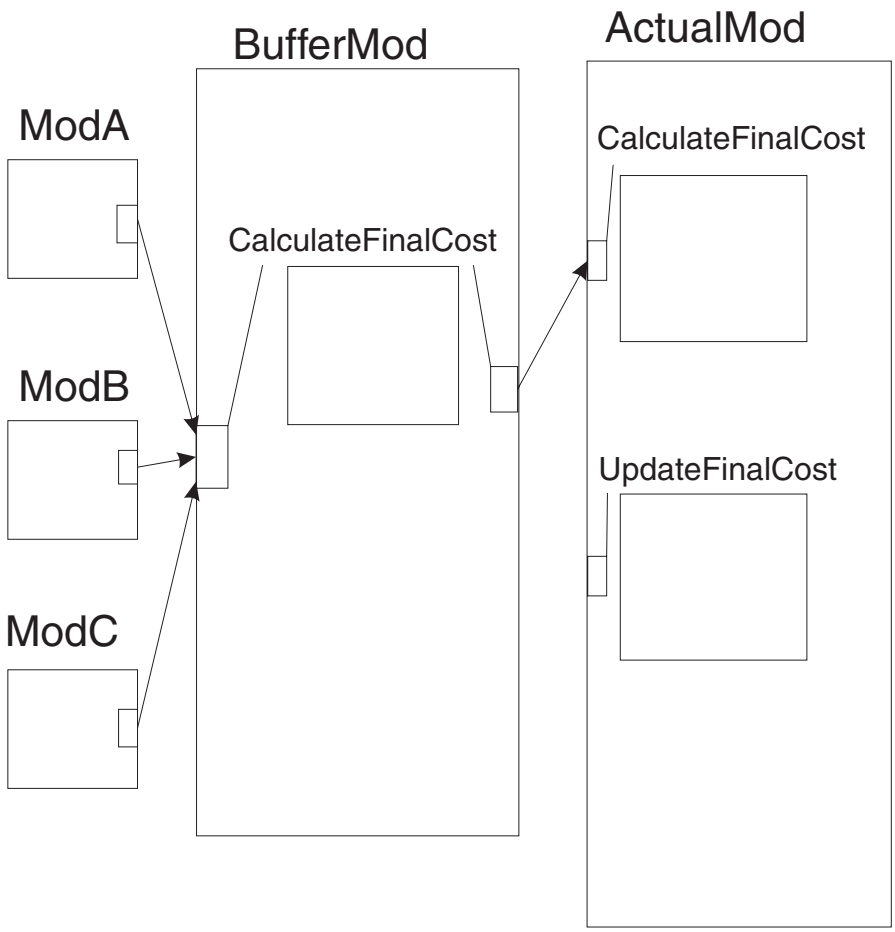


Abbildung 3. Isoliertes Aufrufmodell mit Aufruf von UpdateCalculateFinal

Bei diesem Modell werden die aufrufenden Module nicht geändert, und Sie müssen lediglich die Bindung vom Puffermodulimport an das Ziel ändern (siehe Isoliertes Aufrufmodell mit Aufruf von UpdatedCalculateFinal).

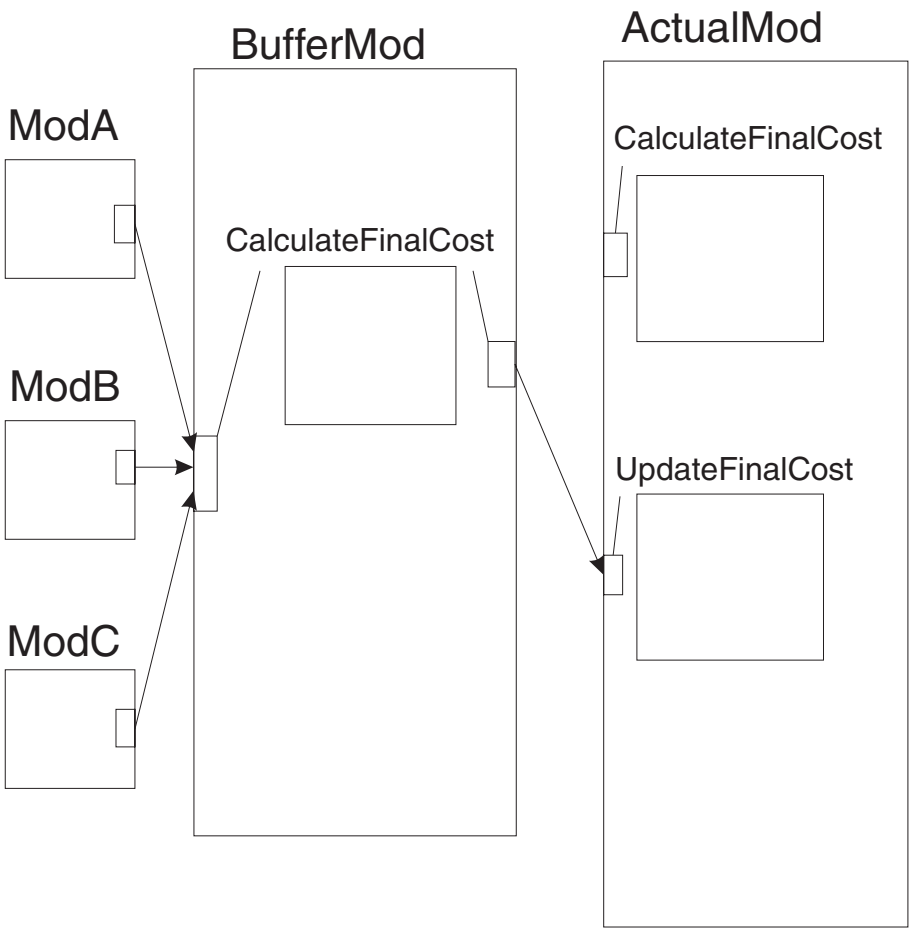


Abbildung 4. Isoliertes Aufrufmodell mit Aufruf von UpdatedCalculateFinal

Falls das Puffermodul das Ziel synchron aufruft, stammen die Ergebnisse, die an die Ursprungsanwendung zurückgegeben werden, vom neuen Ziel, sobald das Puffermodul (ein Mediationsmodul oder ein Service für Business-Module) erneut gestartet wurde. Ruft das Puffermodul das Ziel asynchron auf, stammen die Ergebnisse, die an die Ursprungsanwendung zurückgegeben werden, beim nächsten Aufruf vom neuen Ziel.

Zugehörige Tasks

Ziele ändern
Durch das Ändern des Ziels einer Referenz können Anwendungen flexibler die Vorteile des Fortschritts für Komponenten nutzen, da hierfür keine erneute Kompilierung und Neuinstallation der Anwendung notwendig ist.

HTTP-Bindungen

Die HTTP-Bindung wurde für die Bereitstellung der SCA-Konnektivität für HTTP konzipiert. Dadurch können vorhandene oder neu entwickelte HTTP-Anwendungen in SCA-Umgebungen Architecture (SOA) eingebunden werden.

Außerdem kann ein Netz aus SCA-Laufzeitumgebungen in einer vorhandenen HTTP-Infrastruktur kommunizieren.

Die HTTP-Bindung stellt verschiedene HTTP-Funktionen bereit:

- Nachrichten werden Mediationskomponenten in einer Art und Weise präsentiert, bei der das HTTP-Format und Nachrichtenheaderinformationen bewahrt bleiben.

HTTP-Anwendungsprogrammierer, Benutzer und Administratoren sehen sich somit einer vertrauteren Anzeige gegenüber.

- Ein vorhandenes Datenbindungsframework wurde für HTTP-Konventionen erweitert und stellt eine Zuordnung zwischen SCA-Nachrichten und HTTP-Nachrichtenheadern und -Hauptteilen bereit.
- Importe und Exporte können zur Unterstützung einer Reihe von allgemeinen HTTP-Funktionen konfiguriert werden.
- Bei der Installation eines SCA-Moduls mit HTTP-Importen oder -Exporten wird die Laufzeitumgebung entsprechend konfiguriert, damit eine Verbindung zu HTTP möglich ist.

Ausführliche Anweisungen zur Erstellung von HTTP-Importen und -Exporten finden Sie im Information Center von unter **WebSphere Integration Developer** > **Developing integration applications** > **HTTP data binding**.

Zugehörige Tasks

HTTP-Bindungen anzeigen
Nach der Implementierung einer Anwendung können Sie die HTTP-Bindungen auf ihre Richtigkeit hin überprüfen.

HTTP-Exportbindungen ändern
Mit der Administrationskonsole können Sie die Konfiguration von HTTP-Exportbindungen ändern; dabei ist es nicht erforderlich, die ursprüngliche Quelle zu ändern und anschließend die Anwendung erneut zu implementieren.

HTTP-Importbindungen ändern
Mit der Administrationskonsole können Sie die Konfiguration von HTTP-Importbindungen ändern; dabei ist es nicht erforderlich, die ursprüngliche Quelle zu ändern und anschließend die Anwendung erneut zu implementieren.

Generierte SCA-Implementierung überschreiben

Gelegentlich entspricht die Konvertierung, die das System zwischen einem Java-Code und einem SDO erstellt, möglicherweise nicht Ihren Anforderungen. Mit dieser Prozedur können Sie die standardmäßige SCA-Klassenimplementierung durch ihre eigene ersetzen.

Stellen Sie sicher, dass Sie die Java-in-WSDL-Typ-Konvertierung entweder mit WebSphere Integration Developer oder mit dem Befehl genMapper generiert haben.

Sie können eine generierte Komponente überschreiben, die einen Java-Typ einem WSDL-Typ zuordnet; hierfür müssen Sie den generierten Code durch Code ersetzen, der Ihren Anforderungen entspricht. Ziehen Sie die Verwendung Ihrer eigenen Zuordnung in betracht, wenn Sie Ihre eigenen Java-Klassen definiert haben. Nehmen Sie die Änderungen mithilfe dieser Vorgehensweise vor.

1. Suchen Sie nach der generierten Komponente. Die Komponente hat den Namen *java-klasse*Mapper.component.
2. Bearbeiten Sie die Komponente mit einem Texteditor.
3. Setzen Sie den generierten Code auf Kommentar, und geben Sie Ihre eigene Methode an.

   Ändern Sie nicht den Dateinamen mit der Komponentenimplementierung.

Im Folgenden sehen Sie ein Beispiel für eine zu ersetzende generierte Komponente:

```
private DataObject javatodata_setAccount_output(Object myAccount) {

  // Sie können diesen Code für eine benutzerdefinierte Zuordnung überschreiben.
  // Setzen Sie diesen Code auf Kommentar, und schreiben Sie einen angepassten Code.

  // Sie können auch den an das Konvertierungstool übergebenen Java-Typ ändern,
  // den das Konvertierungstool zu erstellen versucht.

 return SDOJavaObjectMediator.java2Data(myAccount);

}
```

Kopieren Sie die Komponente und weitere Dateien in das Verzeichnis, in dem sich das übergeordneten Modul befindet, und verbinden Sie entweder die Komponente in WebSphere Integration Developer, oder generieren Sie eine EAR-Datei; verwenden Sie dabei den Befehl serviceDeploy.

Zugehörige Konzepte

„Laufzeitregeln für die Konvertierung von Java in Service Data Objects" auf Seite 17
Wenn Sie generierten Code ordnungsgemäß überschreiben möchten oder mögliche Laufzeitausnahmenbedingungen ermitteln möchten, die sich auf Konvertierungen von Java in Service Data Object (SDO) beziehen, müssen Sie mit den entsprechenden Regeln vertraut sein. Die überwiegende Zahl der Konvertierungen ist unkompliziert; es gibt allerdings einige komplexe Fälle, in denen die Laufzeit die beste Möglichkeit bei der Konvertierung des generierten Codes bereitstellt.

Zugehörige Verweise

Konvertierung von Java in XML
Mithilfe vordefinierter Regeln generiert das System XML auf der Basis von Java-Typen.

Befehl 'genMapper'
Verwenden Sie den Befehl 'genMapper', um eine Komponente zu erstellen, die eine SCA-Referenz mit einer Java-Schnittstelle verbindet.

Konvertierung eines SDO in Java überschreiben

Gelegentlich entspricht die Konvertierung, die das System zwischen einem Service Data Object (SDO) und einem Java-Typobjekt erstellt, möglicherweise nicht Ihren Anforderungen. Mit dieser Prozedur können Sie die Standardimplementierung durch ihre eigene ersetzen.

Stellen Sie sicher, dass Sie die WSDL-in-Java-Typ-Konvertierung entweder mit WebSphere Integration Developer oder mit dem Befehl genMapper generiert haben.

Sie können eine generierte Komponente überschreiben, die einen WSDL-Typ einem Java-Typ zuordnet; hierfür müssen Sie den generierten Code durch Code ersetzen, der Ihren Anforderungen entspricht. Ziehen Sie die Verwendung Ihrer eigenen Zuordnung in betracht, wenn Sie Ihre eigenen Java-Klassen definiert haben. Nehmen Sie die Änderungen mithilfe dieser Vorgehensweise vor.

1. Suchen Sie nach der generierten Komponente. Die Komponente hat den Namen *java-klasse*Mapper.component.
2. Bearbeiten Sie die Komponente mit einem Texteditor.

3. Setzen Sie den generierten Code auf Kommentar, und geben Sie Ihre eigene Methode an.

   Ändern Sie nicht den Dateinamen mit der Komponentenimplementierung.

Im Folgenden sehen Sie ein Beispiel für eine zu ersetzende generierte Komponente:

```
 private Object datatojava_get_customerAcct(DataObject myCustomerID,
    String integer)
{

  // Sie können diesen Code für eine benutzerdefinierte Zuordnung überschreiben.
  // Setzen Sie diesen Code auf Kommentar, und schreiben Sie einen angepassten Code.

  // Sie können auch den an das Konvertierungstool übergebenen Java-Typ ändern,
  // den das Konvertierungstool zu erstellen versucht.


  return SDOJavaObjectMediator.data2Java(customerID, integer) ;

}
```

Kopieren Sie die Komponente und weitere Dateien in das Verzeichnis, in dem sich das übergeordneten Modul befindet, und verbinden Sie entweder die Komponente in WebSphere Integration Developer, oder generieren Sie eine EAR-Datei; verwenden Sie dabei den Befehl serviceDeploy.

Zugehörige Konzepte

„Laufzeitregeln für die Konvertierung von Java in Service Data Objects"
Wenn Sie generierten Code ordnungsgemäß überschreiben möchten oder mögliche Laufzeitausnahmenbedingungen ermitteln möchten, die sich auf Konvertierungen von Java in Service Data Object (SDO) beziehen, müssen Sie mit den entsprechenden Regeln vertraut sein. Die überwiegende Zahl der Konvertierungen ist unkompliziert; es gibt allerdings einige komplexe Fälle, in denen die Laufzeit die beste Möglichkeit bei der Konvertierung des generierten Codes bereitstellt.

Zugehörige Verweise

Konvertierung von Java in XML
Mithilfe vordefinierter Regeln generiert das System XML auf der Basis von Java-Typen.

Befehl 'genMapper'
Verwenden Sie den Befehl 'genMapper', um eine Komponente zu erstellen, die eine SCA-Referenz mit einer Java-Schnittstelle verbindet.

Laufzeitregeln für die Konvertierung von Java in Service Data Objects

Wenn Sie generierten Code ordnungsgemäß überschreiben möchten oder mögliche Laufzeitausnahmenbedingungen ermitteln möchten, die sich auf Konvertierungen von Java in Service Data Object (SDO) beziehen, müssen Sie mit den entsprechenden Regeln vertraut sein. Die überwiegende Zahl der Konvertierungen ist unkompliziert; es gibt allerdings einige komplexe Fälle, in denen die Laufzeit die beste Möglichkeit bei der Konvertierung des generierten Codes bereitstellt.

Basistypen und -klassen

In der Laufzeit wird eine unkomplizierte Konvertierung von Service Data Objects in Java-Basistypen und -klassen und umgekehrt durchgeführt. Folgende Basistypen und -klassen sind gültig:

- Char oder java.lang.Character

- Boolean
- Java.lang.Boolean
- Byte oder java.lang.Byte
- Short oder java.lang.Short
- Int oder java.lang.Integer
- Long oder java.lang.Long
- Float oder java.lang.Float
- Double oder 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[]

Benutzerdefinierte Java-Klassen und -Arrays

Bei der Konvertierung einer Java-Klasse oder eines -Arrays in ein SDO, erstellt die Laufzeit ein Datenobjekt, das über eine URI verfügt, die durch die Umkehrung des Paketnames des Java-Typs generiert wird und die einen Typ aufweist, der dem Namen der Java-Klasse entspricht. So wird die Java-Klasse com.ibm.xsd.Customer beispielsweise in ein SDO und die URI http://xsd.ibm.com mit dem Typ 'Customer' konvertiert. Die Laufzeit überprüft dann den Inhalt der Java-Klassenmember und ordnet Merkmalen die Werte im SDO zu.

Bei der Konvertierung eines SDOs in einen Java-Typ generiert die Laufzeit den Paketnamen durch Umkehrung der URI, und der Name des Typs entspricht SDO-Typen. Beispiel: Das Datenobjekt mit dem Typ 'Customer' und der URI http://xsd.ibm.com generiert eine Instanz des Java-Pakets com.ibm.xsd.Customer. Bei der Laufzeit werden anschließend Werte aus den Merkmalen der SDO extrahiert; diese Merkmale werden Feldern in der Instanz der Java-Klasse zugeordnet.

Wenn es sich bei der Java-Klasse um eine benutzerdefinierte Schnittstelle handelt, müssen Sie den generierten Code überschreiben und eine konkrete Klasse angeben, die bei der Laufzeit instanziiert werden kann. Wenn bei der Laufzeit keine konkrete Klasse erstellt werden kann, kommt es zu einer Ausnahmebedingung.

Java.lang.Object

Wenn der Java-Typ 'java.lang.Object' lautet, lautet der generierte Typ 'xsd:anyType'. Ein Modul kann diese Schnittstelle mit einem beliebigen SDO aufrufen. Bei der Laufzeit wird versucht, eine konkrete Klasse so zu instanziieren wie benutzerdefinierte Java-Klassen und -Arrays instanziiert werden, wenn diese Klasse bei der Laufzeit gefunden wird. Ansonsten wird das SDO bei der Laufzeit an die Java-Schnittstelle übergeben.

Auch wenn die Methode den Typ 'java.lang.Object' zurückgibt, wird die SDO-Konvertierung bei der Laufzeit nur durchgeführt, wenn die Methode einen konkreten Typ zurückgibt. Bei der Laufzeit wird eine der für die Konvertierung benutzerdefinierter Java-Klassen und -Arrays in SDOs ähnliche Konvertierung verwendet, die im nächsten Abschnitt beschrieben wird.

Bei der Konvertierung einer Java-Klasse oder eines -Arrays in ein SDO, erstellt die Laufzeit ein Datenobjekt, das über eine URI verfügt, die durch die Umkehrung des

Paketnames des Java-Typs generiert wird und die einen Typ aufweist, der dem Namen der Java-Klasse entspricht. So wird die Java-Klasse com.ibm.xsd.Customer beispielsweise in ein SDO und die URI http://xsd.ibm.com mit dem Typ 'Customer' konvertiert. Die Laufzeit überprüft dann den Inhalt der Java-Klassenmember und ordnet Merkmalen die Werte im SDO zu.

In beiden Fällen tritt eine Ausnahmebedingung auf, wenn die Konvertierung bei der Laufzeit nicht abgeschlossen werden kann.

Java.util-Containerklassen

Bei der Konvertierung in eine konkrete Java-Containerklasse wie z. B. Vector, HashMap, HashSet u. Ä. wird bei der Laufzeit die entsprechende Containerklasse instanziiert. Bei der Laufzeit wird eine der für die benutzerdefinierten Java-Klassen und -Arrays ähnliche Methode zum Füllen der Containerklasse verwendet. Wenn bei der Laufzeit keine konkrete Java-Klasse gefunden werden kann, wird die Containerklasse in der Laufzeit mit dem SDO gefüllt.

Bei der Konvertierung konkreter Java-Containerklassen in SDOs werden bei der Laufzeit die in der Referenz zur Konvertierung von Java in XML angezeigten generierten Schemas verwendet.

Java.util-Schnittstellen

Für bestimmte Containerschnittstellen im java.util-Paket instanziiert die Laufzeit die folgenden konkreten Klassen:

Tabelle 1. WSDL-Typ in Java-Klasse - Konvertierung

Schnittstelle	Konkrete Standardklassen
Objektgruppe	HashSet
Zuordnung	HashMap
Liste	ArrayList
Gruppe	HashSet

Zugehörige Tasks

„Generierte SCA-Implementierung überschreiben" auf Seite 15
Gelegentlich entspricht die Konvertierung, die das System zwischen einem Java-Code und einem SDO erstellt, möglicherweise nicht Ihren Anforderungen. Mit dieser Prozedur können Sie die standardmäßige SCA-Klassenimplementierung durch ihre eigene ersetzen.

„Konvertierung eines SDO in Java überschreiben" auf Seite 16
Gelegentlich entspricht die Konvertierung, die das System zwischen einem Service Data Object (SDO) und einem Java-Typobjekt erstellt, möglicherweise nicht Ihren Anforderungen. Mit dieser Prozedur können Sie die Standardimplementierung durch ihre eigene ersetzen.

Zugehörige Verweise

Konvertierung von Java in XML
Mithilfe vordefinierter Regeln generiert das System XML auf der Basis von Java-Typen.

Befehl 'genMapper'
Verwenden Sie den Befehl 'genMapper', um eine Komponente zu erstellen, die eine SCA-Referenz mit einer Java-Schnittstelle verbindet.

Kapitel 2. Clientanwendungen für Business-Prozesse und Tasks entwickeln

Mit einem Modellierungstool können Sie Business-Prozesse und Tasks erstellen und implementieren. Zur Laufzeit finden Interaktionen mit diesen Prozessen und Tasks statt (z. B. wird ein Prozess gestartet oder es werden Tasks beansprucht und abgeschlossen). Für Interaktionen mit Prozessen und Tasks können Sie Business Process Choreographer Explorer verwenden oder die Business Process Choreographer-APIs, um benutzerdefinierte Anwendungen für diese Interaktionen zu entwickeln.

Bei diesen Clients kann es sich um EJB-Clients (EJB = Enterprise JavaBeans), um Web-Service-Clients oder um Web-Clients handeln, die die JSF-Komponenten (JSF = JavaServer Faces) von Business Process Choreographer Explorer nutzen. Business Process Choreographer stellt EJB-APIs und Schnittstellen für Web-Services bereit, mit denen Sie diese Clients entwickeln können. Auf die EJB-API kann mit jeder Java-Anwendung zugegriffen werden (auch mit einer anderen EJB-Anwendung). Auf die Schnittstellen für Web-Services kann von Java-Umgebungen oder Microsoft .Net-Umgebungen zugegriffen werden.

EJB-Clientanwendungen für Business-Prozesse und Benutzertasks entwickeln

Die EJB-APIs stellen eine Reihe generischer Methoden für die Entwicklung von EJB-Clientanwendungen zum Arbeiten mit Business-Prozessen und Benutzertasks zur Verfügung, die auf einem WebSphere Process Server installiert sind.

Mit diesen EJB-APIs (EJB = Enterprise JavaBeans) können Sie Clientanwendungen für die folgenden Aufgaben erstellen:

- Lebenszyklus von Prozessen und Tasks vom Start bis zum Löschen nach ihrer Beendigung verwalten
- Aktivitäten und Prozesse reparieren
- Workload verwalten und auf Mitglieder einer Arbeitsgruppe verteilen

Die EJB-APIs werden in Form von zwei statusfreien Session Enterprise-Beans zur Verfügung gestellt:

- Die Schnittstelle BusinessFlowManagerService stellt Methoden für Business-Prozessanwendungen bereit
- Die Schnittstelle HumanTaskManagerService stellt Methoden für taskbezogene Anwendungen bereit

Weitere Informationen zu den EJB-APIs enthält das Javadoc im Paket com.ibm.bpe.api und im Paket com.ibm.task.api.

Die folgenden Schritte geben eine Übersicht über die erforderlichen Aktionen zum Entwickeln einer EJB-Clientanwendung.

1. Legen Sie fest, welche Funktionen die Anwendung zur Verfügung stellen soll.
2. Legen Sie fest, welche der Session-Beans Sie verwenden möchten.

   Je nach den Szenarien, die Sie mit Ihrer Anwendung implementieren möchten, können Sie eine oder beide Session-Beans verwenden.

© IBM Corporation 2005, 2006

3. Legen Sie fest, welche Berechtigungen Benutzer der Anwendung benötigen.

   Die Benutzer Ihrer Anwendung müssen dazu berechtigt werden, die Methoden aufzurufen, die Sie in Ihre Anwendung integrieren, und die Objekte und Attribute dieser Objekte anzuzeigen, die diese Methoden zurückgeben. Wenn eine Instanz der entsprechenden Session-Bean erstellt wird, ordnet WebSphere Application Server der Instanz einen Kontext zu. Der Kontext enthält Informationen zur Principal-ID des Aufrufenden, zur Gruppenzugehörigkeitsliste und zu Aufgabenbereichen. Anhand dieser Informationen wird die Berechtigung des Aufrufenden bei jedem Aufruf überprüft.

   Das Javadoc enthält Berechtigungsinformationen für alle einzelnen Methoden.
4. Legen Sie fest, wie die Anwendung realisiert werden soll.

   Die EJB-APIs können lokal oder über Remotezugriff aufgerufen werden.
5. Entwickeln Sie die Anwendung.
   a. Greifen Sie auf die EJB-API zu.
   b. Verwenden Sie die EJB-API für Interaktionen mit Prozessen oder Tasks.
      - Fragen Sie die Daten ab.
      - Arbeiten Sie mit den Daten.

Auf die EJB-APIs zugreifen

Die Enterprise JavaBeans-APIs (EJB-APIs) werden in Form von zwei statusfreien Session Enterprise-Beans zur Verfügung gestellt. Business-Prozessanwendungen und Taskanwendungen greifen auf die entsprechende Session Enterprise-Bean über die Home-Schnittstelle der Bean zu.

Die Schnittstelle BusinessFlowManagerService stellt die Methoden für Business-Prozessanwendungen bereit. Die Schnittstelle HumanTaskManagerService stellt die Methoden für taskbezogene Anwendungen bereit. Die Anwendung kann eine beliebige Java-Anwendung sein sowie eine andere EJB-Anwendung (EJB = Enterprise JavaBeans).

Auf die ferne Schnittstelle der Session-Bean zugreifen

Eine EJB-Clientanwendung greift auf die ferne Schnittstelle der Session-Bean über die ferne Home-Schnittstelle der Bean zu.

Die Session-Bean kann die BusinessFlowManager-Session-Bean für Prozessanwendungen oder die HumanTaskManager-Session-Bean für Taskanwendungen sein.

1. Fügen Sie einen Verweis auf die ferne Schnittstelle der Session-Bean zum Deskriptor für die Anwendungsimplementierung hinzu. Fügen Sie den Verweis einer der folgenden Dateien hinzu:
   - Der Datei application-client.xml für eine J2EE-Clientanwendung (Java 2 Platform, Enterprise Edition)
   - Der Datei web.xml für eine Webanwendung
   - Der Datei ejb-jar.xml für eine EJB-Anwendung (EJB = Enterprise JavaBeans)

   Im folgenden Beispiel wird der Verweis auf die ferne Home-Schnittstelle für Prozessanwendungen gezeigt:

```
<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>
```

Im folgenden Beispiel wird der Verweis auf die ferne Home-Schnittstelle für Taskanwendungen gezeigt:

```
<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>
```

Wenn Sie mit WebSphere Integration Developer den EJB-Verweis dem Implementierungsdeskriptor hinzufügen, wird die Bindung für den EJB-Verweis beim Implementieren der Anwendung automatisch erstellt. Weitere Informationen zum Hinzufügen von EJB-Verweisen enthält die Dokumentation von WebSphere Integration Developer.

2. Fügen Sie die generierten Elemente in Ihr Anwendungspaket ein.

   Wenn Ihre Anwendung auf einer anderen Java Virtual Machine (JVM) ausgeführt wird als diejenige, auf der die Anwendung BPEContainer oder die Anwendung TaskContainer ausgeführt wird, führen Sie die folgenden Aktionen aus:

   a. Fügen Sie bei Prozessanwendungen die Datei *<installationsstammverzeichnis>*/ProcessChoreographer/client/bpe137650.jar zur EAR-Datei (Enterprise Archive) Ihrer Anwendung hinzu.
   b. Fügen Sie bei Taskanwendungen die Datei *<installationsstammverzeichnis>*/ProcessChoreographer/client/task137650.jar zur EAR-Datei Ihrer Anwendung hinzu.
   c. Legen Sie den Parameter **Classpath** in der Manifestdatei des Anwendungsmoduls fest, um die JAR-Datei einzubinden.

      Das Anwendungsmodul kann eine J2EE-Anwendung, eine Webanwendung oder eine EJB-Anwendung sein.
   d. Wenn Sie in Ihrem Business-Prozess oder in Ihrer Benutzertask komplexe Datentypen verwenden und Ihr Client nicht in einer EJB- oder Webanwendung ausgeführt wird, fügen Sie die entsprechenden XSD- oder WSDL-Dateien zur EAR-Datei Ihrer Anwendung hinzu.

3. Suchen Sie nach der fernen Home-Schnittstelle der Session-Bean über JNDI (Java Naming and Directory Interface).

   Das folgende Beispiel zeigt diesen Schritt für eine Prozessanwendung:

```
// Ursprünglichen JNDI-Standardkontext abfragen
InitialContext initialContext = new InitialContext();

// Ferne Home-Schnittstelle der Bean BusinessFlowManager suchen
Object result =
       initialContext.lookup("java:comp/env/ejb/BusinessFlowManagerHome");

// Suchergebnis in den korrekten Typ umwandeln
   BusinessFlowManagerHome processHome =
           (BusinessFlowManagerHome)javax.rmi.PortableRemoteObject.narrow
           (result,BusinessFlowManagerHome.class);
```

   Die Home-Schnittstelle der Session-Bean enthält eine Erstellungsmethode für EJB-Objekte. Die Methode gibt die ferne Schnittstelle der Session-Bean zurück.

4. Greifen Sie auf die ferne Schnittstelle der Session-Bean zu.

   Das folgende Beispiel zeigt diesen Schritt für eine Prozessanwendung:

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

   Die Zugriffsmöglichkeit auf die Session-Bean bedeutet nicht unbedingt, dass der Aufrufende alle von der Bean bereitgestellten Aktionen ausführen darf. Der Aufrufende muss zusätzlich für diese Aktionen berechtigt sein. Beim Erstellen

einer Instanz der Session-Bean wird der Instanz ein Kontext zugeordnet. Der Kontext enthält die Principal-ID des Aufrufenden und die Gruppenzugehörigkeitsliste und gibt an, ob der Aufrufende einem der J2EE-Aufgabenbereiche von Business Process Choreographer angehört. Anhand des Kontexts wird die Berechtigung des Aufrufenden bei jedem Aufruf überprüft, auch wenn die globale Sicherheit nicht konfiguriert ist. Wenn die globale Sicherheit nicht konfiguriert ist, hat die Principal-ID des Aufrufenden den Wert `UNAUTHENTICATED`.

5. Rufen Sie die von der Serviceschnittstelle bereitgestellten Business-Funktionen auf.

   Das folgende Beispiel zeigt diesen Schritt für eine Prozessanwendung:

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

   Aufrufe von Anwendungen werden als Transaktionen ausgeführt. Eine Transaktion wird auf eine der folgenden Arten eingerichtet und beendet:
   - Automatisch durch WebSphere Application Server (im Implementierungsdeskriptor ist TX_REQUIRED angegeben)
   - Explizit durch die Anwendung; Sie können Anwendungsaufrufe in einer einzigen Transaktion bündeln:

     ```
     // Benutzertransaktionsschnittstelle abrufen
     UserTransaction transaction=
            (UserTransaction)initialContext.lookup("jta/usertransaction");

     // Eine Transaktion beginnen
     transaction.begin();

     // Anwendungsaufrufe ...

     // Bei erfolgreicher Rückgabe die Transaktion festschreiben
     transaction.commit();
     ```

   Tipp: Um Datenbanksperrenkonflikte zu vermeiden, führen Sie keine Anweisungen ähnlich den folgenden parallel aus:

   ```
   // Benutzertransaktionsschnittstelle abrufen
   UserTransaction transaction=
          (UserTransaction)initialContext.lookup("jta/usertransaction");

   transaction.begin();

   //Aktivitätsinstanz lesen
   process.getActivityInstance(aiid);
   //Aktivitätsinstanz anfordern
   process.claim(aiid);

   transaction.commit();
   ```

   Mit der Methode getActivityInstance und weiteren Leseoperationen wird eine Lesesperre gesetzt. In diesem Beispiel wird für die Lesesperre der Aktivitätsinstanz ein Upgrade auf eine Aktualisierungssperre der Aktivitätsinstanz durchgeführt. Dies kann zu einer Datenbanksperre führen, wenn diese Transaktionen gleichzeitig ausgeführt werden.

Beispiel

Das folgende Beispiel zeigt, wie die Schritte 3 bis 5 für eine Taskanwendung aussehen können.

```
// Ursprünglichen JNDI-Standardkontext abfragen
InitialContext initialContext = new InitialContext();

// Ferne Home-Schnittstelle der Bean HumanTaskManager suchen
```

```
Object result =
         initialContext.lookup("java:comp/env/ejb/HumanTaskManagerHome");

// Suchergebnis in den korrekten Typ umwandeln
HumanTaskManagerHome taskHome =
           (HumanTaskManagerHome)javax.rmi.PortableRemoteObject.narrow
           (result,HumanTaskManagerHome.class);

...
// Zugriff auf die ferne Schnittstelle der Session-Bean
HumanTaskManager task = taskHome.create();

...
// Aufrufen der von der Serviceschnittstelle bereitgestellten Business-Funktionen
task.callTask(tkiid,input);
```

Auf die lokale Schnittstelle der Session-Bean zugreifen

Eine EJB-Clientanwendung greift auf die lokale Schnittstelle der Session-Bean über die lokale Home-Schnittstelle der Bean zu.

Die Session-Bean kann die BusinessFlowManager-Session-Bean für Prozessanwendungen oder die HumanTaskManager-Session-Bean für Benutzertaskanwendungen sein.

1. Fügen Sie einen Verweis auf die lokale Schnittstelle der Session-Bean zum Deskriptor für die Anwendungsimplementierung hinzu. Fügen Sie den Verweis einer der folgenden Dateien hinzu:
   - Der Datei application-client.xml für eine J2EE-Clientanwendung (Java 2 Platform, Enterprise Edition)
   - Der Datei web.xml für eine Webanwendung
   - Der Datei ejb-jar.xml für eine EJB-Anwendung (EJB = Enterprise JavaBeans)

   Im folgenden Beispiel wird der Verweis auf die lokale Home-Schnittstelle für Prozessanwendungen gezeigt:

   ```
   <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>
   ```

   Im folgenden Beispiel wird der Verweis auf die lokale Home-Schnittstelle für Taskanwendungen gezeigt:

   ```
   <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>
   ```

   Wenn Sie mit WebSphere Integration Developer den EJB-Verweis dem Implementierungsdeskriptor hinzufügen, wird die Bindung für den EJB-Verweis beim Implementieren der Anwendung automatisch erstellt. Weitere Informationen zum Hinzufügen von EJB-Verweisen enthält die Dokumentation von WebSphere Integration Developer.

2. Suchen Sie nach der lokalen Home-Schnittstelle der Session-Bean über JNDI (Java Naming and Directory Interface).

   Das folgende Beispiel zeigt diesen Schritt für eine Prozessanwendung:

   ```
   // Ursprünglichen JNDI-Standardkontext abfragen
   InitialContext initialContext = new InitialContext();

   // Lokale Home-Schnittstelle der BusinessFlowManager-Bean suchen
   ```

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

Die lokale Home-Schnittstelle der Session-Bean enthält eine Erstellungsmethode für EJB-Objekte. Die Methode gibt die lokale Schnittstelle der Session-Bean zurück.

3. Greifen Sie auf die lokale Schnittstelle der Session-Bean zu.

   Das folgende Beispiel zeigt diesen Schritt für eine Prozessanwendung:

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

   Die Zugriffsmöglichkeit auf die Session-Bean bedeutet nicht unbedingt, dass der Aufrufende alle von der Bean bereitgestellten Aktionen ausführen darf. Der Aufrufende muss zusätzlich für diese Aktionen berechtigt sein. Beim Erstellen einer Instanz der Session-Bean wird der Instanz ein Kontext zugeordnet. Der Kontext enthält die Principal-ID des Aufrufenden und die Gruppenzugehörigkeitsliste und gibt an, ob der Aufrufende einem der J2EE-Aufgabenbereiche von Business Process Choreographer angehört. Anhand des Kontexts wird die Berechtigung des Aufrufenden bei jedem Aufruf überprüft, auch wenn die globale Sicherheit nicht konfiguriert ist. Wenn die globale Sicherheit nicht konfiguriert ist, hat die Principal-ID des Aufrufenden den Wert `UNAUTHENTICATED`.

4. Rufen Sie die von der Serviceschnittstelle bereitgestellten Business-Funktionen auf.

   Das folgende Beispiel zeigt diesen Schritt für eine Prozessanwendung:

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

   Aufrufe von Anwendungen werden als Transaktionen ausgeführt. Eine Transaktion wird auf eine der folgenden Arten eingerichtet und beendet:
   - Automatisch durch WebSphere Application Server (im Implementierungsdeskriptor ist TX_REQUIRED angegeben)
   - Explizit durch die Anwendung; Sie können Anwendungsaufrufe in einer einzigen Transaktion bündeln:

     ```
     // Benutzertransaktionsschnittstelle abrufen
     UserTransaction transaction=
            (UserTransaction)initialContext.lookup("jta/usertransaction");

     // Eine Transaktion beginnen
     transaction.begin();

     // Anwendungsaufrufe ...

     // Bei erfolgreicher Rückgabe die Transaktion festschreiben
     transaction.commit();
     ```

     Tipp: Um Datenbank-Deadlocks zu vermeiden, führen Sie keine Anweisungen ähnlich den folgenden parallel aus:

     ```
     // Benutzertransaktionsschnittstelle abrufen
     UserTransaction transaction=
            (UserTransaction)initialContext.lookup("jta/usertransaction");

     transaction.begin();

     //Aktivitätsinstanz lesen
     process.getActivityInstance(aiid);
     //Aktivitätsinstanz anfordern
     process.claim(aiid);

     transaction.commit();
     ```

Mit der Methode getActivityInstance und weiteren Leseoperationen wird eine Lesesperre gesetzt. In diesem Beispiel wird für die Lesesperre der Aktivitätsinstanz ein Upgrade auf eine Aktualisierungssperre der Aktivitätsinstanz durchgeführt. Dies kann zu einer Datenbanksperre führen, wenn diese Transaktionen gleichzeitig ausgeführt werden.

Beispiel

Das folgende Beispiel zeigt, wie die Schritte 2 bis 4 für eine Taskanwendung aussehen können.

```
// Ursprünglichen JNDI-Standardkontext abfragen
InitialContext initialContext = new InitialContext();

// Lokale Home-Schnittstelle der HumanTaskManager-Bean suchen
LocalHumanTaskManagerHome taskHome =
        (LocalHumanTaskManagerHome)initialContext.lookup
        ("java:comp/env/ejb/LocalHumanTaskManagerHome");

...
// Zugriff auf die lokale Schnittstelle der Session-Bean
LocalHumanTaskManager task = taskHome.create();

...
// Aufrufen der von der Serviceschnittstelle bereitgestellten Business-Funktionen
task.callTask(tkiid,input);
```

Abfragen für Business-Prozessobjekte und taskbezogene Objekte

Die Clientanwendungen arbeiten mit Business-Prozessobjekten und taskbezogenen Objekten. Sie können Business-Prozessobjekte und taskbezogene Objekte in der Datenbank abfragen, um spezifische Merkmale dieser Objekte abzurufen.

Beim Konfigurieren von Business Process Choreographer wird sowohl dem Business-Prozesscontainer als auch dem Taskcontainer eine relationale Datenbank zugeordnet. Die Datenbank speicher alle Daten der Schablone (Modelldaten) und der Instanz (Laufzeitdaten)zum Verwalten von Business-Prozessen und Tasks. Zum Abfragen dieser Daten wird eine SQL-ähnliche Syntax verwendet.

Durch eine Einzelabfrage können Sie ein bestimmtes Merkmal eines Objekts abrufen. Sie können Abfragen auch speichern, um sie mehrmals zu verwenden, und Sie können gespeicherte Abfragen in Ihre Anwendung einfügen.

Abfragen für Business-Prozessobjekte und taskbezogene Objekte

Verwenden Sie die Methode query oder queryAll der Service-API zum Abrufen gespeicherter Informationen zu Business-Prozessen und Tasks.

Die Methode query kann von allen Benutzern aufgerufen werden; sie gibt die Merkmale der Objekte zurück, für die Arbeitselemente vorhanden sind. Die Methode queryAll hingegen kann nur von Benutzern aufgerufen werden, die über die folgenden J2EE-Aufgabenbereiche verfügen: BPESystemAdministrator, TaskSystemAdministrator, BPESystemMonitor bzw. TaskSystemMonitor. Diese Methode gibt die Merkmale aller Objekte zurück, die in der Datenbank gespeichert sind.

Alle API-Abfragen sind SQL-Abfragen zugeordnet. Das Format der resultierenden SQL-Abfrage ist von den folgenden Aspekten abhängig:

- Ob die Abfrage von einer Person mit einem der J2EE-Aufgabenbereichen aufgerufen wurde.
- Von den abgefragten Objekten. Zum Abfragen der Objektmerkmale werden vordefinierte Datenbankansichten bereitgestellt.
- Von der Insertion einer FROM-Klausel, von Joinbedingungen und benutzerspezifischen Bedingungen für die Zugriffssteuerung.

Sie können in Abfragen sowohl benutzerdefinierte Merkmale als auch Variablenmerkmale angeben. Wenn Sie In Ihrer Abfrage mehrere benutzerdefinierte Merkmale oder Variablenmerkmale angeben, führt dies zu rückbezüglichen Verknüpfungen in der betreffenden Datenbanktabelle. Je nach verwendetem Datenbanksystem können diese Aufrufe von `query()` Auswirkungen auf die Leistung haben.

Zum Speichern von Abfragen in der Business Process Choreographer-Datenbank können Sie auch die Methode createStoredQuery verwenden. Die Abfragekriterien werden beim Definieren der gespeicherten Abfrage angegeben. Die Kriterien werden dynamisch angewendet, wenn die gespeicherte Abfrage ausgeführt wird, d. h. die Daten werden während der Laufzeit zusammengestellt. Wenn die gespeicherte Abfrage Parameter enthält, werden diese beim Ausführen der Abfrage aufgelöst.

Weitere Informationen zu den Business Process Choreographer-APIs finden Sie im Javadoc des Pakets com.ibm.bpe.api für prozessbezogene Methoden und im Paket com.ibm.task.api für taskbezogene Methoden.

Syntax der API-Methode query:

Die Syntax der Business Process Choreographer-API-Abfragen ist der Syntax von SQL-Abfragen ähnlich. Eine Abfrage kann eine SELECT-Klausel, eine WHERE-Klausel, eine Sortierklausel, einen Parameter 'skip-tuples', einen Grenzwertparameter sowie einen Zeitzonenparameter umfassen.

Die Syntax der Abfrage hängt vom Objekttyp ab. Die folgende Tabelle zeigt die Syntax für jeden der verschiedenen Objekttypen.

Tabelle 2.

Objekt	Syntax
Prozessschablone	`ProcessTemplateData[] queryProcessTemplates (java.lang.String whereClause, java.lang.String orderByClause, java.lang.Integer threshold, java.util.TimeZone timezone);`
Taskschablone	`TaskTemplate[] queryTaskTemplates (java.lang.String whereClause, java.lang.String orderByClause, java.lang.Integer threshold, java.util.TimeZone timezone);`
Zugehörige Daten für Business-Prozesse und Tasks	`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);`

SELECT-Klausel:

Die SELECT-Klausel in der Abfragefunktion gibt an, welche Objektmerkmale eine Abfrage zurückgeben soll.

Die SELECT-Klausel beschreibt das Abfrageergebnis. Sie gibt eine Liste mit den Namen der Objektmerkmale (Ergebnisspalten) an, die zurückgegeben werden sollen. Die Syntax dieser Klausel ist ähnlich wie die Syntax der SQL-Klausel SELECT. Trennen Sie die einzelnen Teile der Klausel durch Kommata von einander. In jedem Teil der Klausel muss eine Spalte aus einer der vordefinierten Ansichten angegeben werden. Die Spalten müssen vollständig mit dem Anzeigenamen und dem Spaltennamen angegeben werden. Die in dem Objekt QueryResultSet zurückgegebenen Spalten haben die gleiche Reihenfolge wie die in der SELECT-Klausel angegebenen Spalten. Die SELECT-Klausel unterstützt keine SQL-Aggregationsfunktionen wie AVG(), SUM(), MIN() oder MAX().

Fügen Sie zum Auswählen der Merkmale von mehreren Name/Wert-Paaren (z. B. benutzerdefinierte Merkmale und Merkmale von Variablen, die abgefragt werden können) einen aus einer Ziffer bestehenden Zähler zum Namen der Ansicht hinzu. Dieser Zähler kann Werte von 1 bis 9 annehmen.

Beispiele für SELECT-Klauseln

- ″WORK_ITEM.OBJECT_TYPE, WORK_ITEM.REASON″

  Ruft die Objekttypen der zugeordneten Objekte und den Grund der Zuordnung der Arbeitselemente ab.
- ″DISTINCT WORK_ITEM.OBJECT_ID″

  Ruft alle Objekt-IDs (ohne Duplikate) ab, für die der Aufrufende über ein Arbeitselement verfügt.
- ″ACTIVITY.TEMPLATE_NAME, WORK_ITEM.REASON″

  Ruft die Namen der Aktivitäten ab, für die der Aufrufende über Arbeitselemente verfügt, und die Gründe für ihre Zuordnung.
- ″ACTIVITY.STATE, PROCESS_INSTANCE.STARTER″

  Ruft die Statuszustände der Aktivitäten und die Starter der ihnen zugeordneten Prozessinstanzen ab.
- ″DISTINCT TASK.TKIID, TASK.NAME″

  Ruft alle Task-IDs und Tasknamen (ohne Duplikate) ab, für die der Aufrufende über ein Arbeitselement verfügt.
- ″TASK_CPROP1.STRING_VALUE, TASK_CPROP2.STRING_VALUE″

  Ruft die Werte der benutzerdefinierten Merkmale ab, die im weiteren Verlauf der WHERE-Klausel angegeben sind.
- ″QUERY_PROPERTY1.STRING_VALUE, QUERY_PROPERTY2.INT_VALUE

  Ruft die Merkmalswerte von Variablen ab, die abgefragt werden können. Diese Komponenten sind im weiteren Verlauf der WHERE-Klausel angegeben.
- ″COUNT(DISTINCT TASK.TKIID)″

  Zählt die Anzahl der Arbeitselemente für eindeutige Tasks, die der WHERE-Klausel entsprechen.

WHERE-Klausel:

Die WHERE-Klausel in der Abfragefunktion beschreibt die Filterbedingungen, die auf die Abfragedomäne angewendet werden.

Die Syntax der WHERE-Klausel ist ähnlich wie die Syntax einer SQL-Klausel WHERE. Sie müssen keine SQL-Klausel FROM und keine Verknüpfungsprädikate zur API-WHERE-Klausel hinzufügen. Diese Konstrukte werden beim Ausführen der Abfrage automatisch hinzugefügt. Zum Anwenden von Filterbedingungen müssen Sie für die WHERE-Klausel `null` angeben.

Die Syntax der WHERE-Klausel unterstützt Folgendes:

- Schlüsselwörter: AND, OR, NOT
- Vergleichsoperatoren: =, <=, <, <>, >,>=, LIKE

  Der Operator LIKE unterstützt die für die abgefragte Datenbank definierten Platzhalterzeichen.
- Gruppenverarbeitung: IN

Außerdem gelten die folgenden Regeln:

- Objekt-ID-Konstanten sind als `ID('string-rep-of-oid')` anzugeben.
- Binärkonstanten sind als `BIN('UTF-8 string')` anzugeben.
- Anstelle von Ganzzahlaufzählungen sind symbolische Konstanten zu verwenden. Geben Sie beispielsweise anstelle eines Ausdrucks für Aktivitätsstatus (`ACTIVITY.STATE=2`) `ACTIVITY.STATE=ACTIVITY.STATE.STATE_READY` an.
- Wenn der Wert des Merkmals in der Vergleichsanweisung einfache Anführungszeichen (') enthält, machen Sie daraus doppelte Anführungszeichen; Beispiel: `"TASK_CPROP.STRING_VALUE='d''automatisation'"`.
- Merkmale von mehreren Name/Wert-Paaren (z. B. benutzerdefinierte Merkmale) sind durch Hinzufügen eines einstelligen Suffixes zum Ansichtsnamen zu referenzieren. Beispiel: `"TASK_CPROP1.NAME='prop1' AND "TASK_CPROP2.NAME='prop2'"`
- Zeitmarkenkonstanten sind als `TS('jjjj-mm-ttThh:mm:ss')` anzugeben. Das aktuelle Datum ist durch die Zeitmarke `CURRENT_DATE` anzugeben.

  In der Zeitmarke ist mindestens ein Datums- oder Zeitwert anzugeben:
  - Wenn Sie nur ein Datum angeben, wird der Zeitwert auf Null gesetzt.
  - Wenn Sie nur eine Zeit angeben, wird das Datum auf das aktuelle Datum gesetzt.
  - Wenn Sie ein Datum angeben, muss das Jahr aus vier Ziffern bestehen, die Werte für Monat und Tag sind optional. Fehlende Werte für Monat und Datum werden auf 01 gesetzt. Beispiel: `TS('2003')` ist identisch mit `TS('2003-01-01T00:00:00')`.
  - Wenn Sie eine Zeit angeben, wird dieser Wert im 24-Stunden-Format angegeben. Beispiel: Ist das aktuelle Datum der 1. Januar 2003, ist `TS('T16:04')` oder `TS('16:04')` identisch mit `TS('2003-01-01T16:04:00')`.

Beispiele für WHERE-Klauseln

- Objekt-ID mit vorhandener ID vergleichen

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

  Dieser Typ der WHERE-Klausel wird normalerweise mit einer vorhandenen Objekt-ID aus einem vorangegangenen Aufruf dynamisch erstellt. Ist diese Objekt-ID in einer Variablen *wiid1* gespeichert, kann die Klausel wie folgt konstruiert werden:

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

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

- Symbolische Konstanten verwenden

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

- Boolesche Werte 'wahr' und 'falsch' verwenden

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

- Benutzerdefinierte Merkmale verwenden

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

Sortierklausel:

Die Sortierklausel in der Abfragefunktion gibt die Sortierkriterien für das Abfrageergebnis an.

Sie können eine Liste von Spalten der Ansichten angeben, nach denen das Ergebnis sortiert wird. Diese Spalten müssen vollständig durch den Namen der Ansicht und der Spalte qualifiziert werden. Es hat sich bewährt, Spalten anzugeben, die sich in der SELECT-Klausel befinden.

Die Syntax der Sortierklausel ist ähnlich wie die Syntax der SQL-Sortierklausel. Trennen Sie die einzelnen Teile der Klausel durch Kommata von einander. Sie können auch `ASC` für eine Sortierung der Spalten in aufsteigender Reihenfolge und `DESC` für eine Sortierung der Spalten in absteigender Reihenfolge angeben. Wenn das Abfrageergebnis nicht sortiert werden soll, geben Sie in der Sortierklausel `null` an.

Sortierbedingungen werden auf dem Server angewendet, d. h. beim Sortieren werden die länderspezifischen Angaben des Servers verwendet. Wenn Sie mehrere Spalten angeben, wird das Abfrageergebnis zuerst nach den Werten der ersten Spalte sortiert, dann nach den Werten der zweiten Spalte usw. Es ist nicht möglich, die Spalten in der Sortierklausel nach Position anzugeben; dies ist nur bei einer SQL-Abfrage möglich.

Beispiele für Sortierklauseln

- ″PROCESS_TEMPLATE.NAME″

  Sortiert das Abfrageergebnis alphabetisch nach dem Namen der Prozessschablone.

- ″PROCESS_INSTANCE.CREATED, PROCESS_INSTANCE.NAME DESC″

  Sortiert das Abfrageergebnis nach dem Erstellungsdatum und (für ein bestimmtes Datum) alphabetisch nach dem Namen der Prozessschablone in absteigender Reihenfolge.

- ″ACTIVITY.OWNER, ACTIVITY.TEMPLATE_NAME, ACTIVITY.STATE″

  Sortiert das Abfrageergebnis zuerst nach dem Eigner der Aktivität, dann nach dem Namen der Aktivitätsschablone und zuletzt nach dem Status der Aktivität.

Parameter 'skip-tuples':

Der Parameter 'skip-tuples' gibt an, wie viele Tupel vom Anfang des Abfrageergebnisses ausgehend ignoriert und nicht an den Aufrufenden in den Abfrageergebnissen zurückgegeben werden sollen.

Verwenden Sie diesen Parameter zusammen mit dem Grenzwertparameter, um in einer Clientanwendung das Auslagern zu implementieren (z. B. um zunächst die ersten 20 Elemente abzurufen, danach die nächsten 20 Elemente usw.).

Wenn dieser Parameter auf `null` gesetzt und kein Grenzwertparameter definiert ist, werden alle in Frage kommenden Tupel zurückgegeben.

Beispiel für einen Parameter 'skip-tuples'

- new Integer(5)

  Gibt an, dass die ersten fünf in Frage kommenden Tupel nicht zurückgegeben werden sollen.

Grenzwertparameter:

Der Grenzwertparameter in der Abfragefunktion begrenzt die Anzahl der Objekte, die im Abfrageergebnis vom Server an den Client zurückgegeben werden.

Da Abfrageergebnisse in Produktionsszenarios Tausende oder gar Millionen von Einträgen enthalten können, empfiehlt es sich, immer einen Grenzwert anzugeben. Der Grenzwertparameter kann zum Beispiel in einer grafischen Benutzerschnittstelle nützlich sein, in der jeweils nur eine geringe Anzahl von Elementen angezeigt werden soll. Wenn der Grenzwertparameter entsprechend festgelegt wird, ist die Datenbankabfrage schneller, und es müssen weniger Daten vom Server zum Client übertragen werden.

Wenn dieser Parameter auf `null` gesetzt ist und der Parameter 'skip-tuples' nicht gesetzt ist, werden alle in Frage kommenden Objekte zurückgegeben.

Beispiel für einen Grenzwertparameter

- new Integer(50)

  Gibt an, dass 50 in Frage kommende Tupel zurückgegeben werden sollen.

Zeitzonenparameter:

Der Zeitzonenparameter in der Abfragefunktion definiert die Zeitzone für Zeitmarkenkonstanten in der Abfrage.

Der Client, von dem die Abfrage gestartet wird, und der Server, der die Abfrage verarbeitet, können sich in verschiedenen Zeitzonen befinden. Mit dem Zeitzonenparameter können Sie die Zeitzone für Zeitmarkenkonstanten angeben, die in der WHERE-Klausel verwendet werden (z. B. um Ortszeiten anzugeben). Die im Abfrageergebnis enthaltenen Zeitangaben weisen die in der Abfrage angegebene Zeitzone auf.

Wenn der Parameter auf `null` gesetzt ist, wird UTC (Coordinated Universal Time) als Zeitzone für die Zeitmarkenkonstanten verwendet.

Beispiele für Zeitzonenparameter

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

  Gibt Objekt-IDs für Aktivitäten zurück, die nach 17:40 Ortszeit am 1. Januar 2005 gestartet wurden.
- ```
  process.query("ACTIVITY.AIID",
                "ACTIVITY.STARTED > TS('2005-01-01T17:40')",
                 (String)null, (Integer)null, (TimeZone)null);
  ```

Gibt Objekt-IDs für Aktivitäten zurück, die nach 17:40 UTC am 1. Januar 2005 gestartet wurden. Diese Zeitangabe liegt beispielsweise in Eastern Standard Time 6 Stunden früher.

Parameter in gespeicherten Abfragen:

Eine gespeicherte Abfrage ist eine Abfrage, die in der Datenbank gespeichert und mit einem Namen versehen ist. Die dazugehörigen Tupel werden beim Ausführen der Abfrage dynamisch zusammengesetzt. Um gespeicherte Abfragen wiederverwendbar zu machen, können Sie Parameter in der Abfragedefinition verwenden, die während der Laufzeit aufgelöst werden.

Angenommen, Sie haben benutzerdefinierte Merkmale zum Speichern von Kundennamen definiert. Sie können nun Abfragen definieren, um die Tasks abzurufen, die einem bestimmten Kunden zugeordnet sind (z. B. der Firma ACME Co.). Zum Abfragen dieser Informationen kann die Klausel where in Ihrer Abfrage ungefähr so aussehen:

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

Damit diese Abfrage auch zum Suchen nach dem Kunden (BCME Ltd) verwendet werden kann, können Sie Parameter für die Werte des benutzerdefinierten Merkmals verwenden. Wenn Sie Parameter zu der Taskabfrage hinzufügen, kann sie ungefähr wie im folgenden Beispiel aussehen:

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

Der Parameter `@param1` wird während der Laufzeit über die Parameterliste aufgelöst, die an die Methode query übergeben wird. Für die Verwendung von Parametern in Abfragen gelten folgende Regeln:

- Parameter können nur in der Klausel where verwendet werden.
- Parameter sind Zeichenfolgen.
- Parameter werden zur Laufzeit durch Zeichenfolgen ersetzt. Wenn Sonderzeichen erforderlich sind, müssen diese in der Klausel where angegeben oder zur Laufzeit als Bestandteil des Parameters übergeben werden.
- Parameternamen bestehen aus der Zeichenfolge `@param`, an die eine ganze Zahl angehängt ist. Die kleinste Zahl 1 verweist auf den ersten Eintrag in der Parameterliste, die zur Laufzeit an die Abfrage-API übergeben wird.
- Ein Parameter kann innerhalb einer Klausel where mehrmals verwendet werden. Alle Vorkommen des Parameters werden durch denselben Wert ersetzt.

Zugehörige Tasks

„Gespeicherte Abfragen verwalten” auf Seite 60
Gespeicherte Abfragen bieten eine Möglichkeit zum Aufbewahren häufig verwendeter Abfragen. Die gespeicherte Abfrage kann eine für alle Benutzer verfügbare Abfrage sein (öffentliche Abfrage) oder eine Abfrage, die einem bestimmten Benutzer zugeordnet ist (private Abfrage).

Abfrageergebnisse:

Ein Abfrageergebnis enthält die Ergebnisse einer Abfrage.

Die Elemente der Ergebnisliste sind Merkmale der Objekte, die der vom Aufrufenden angegebenen Where-Klausel entsprechen und die der Aufrufende anzeigen darf. Für den relativen Zugriff auf Listenelemente können Sie die API-Methode next ('Nächstes') verwenden, für den absoluten Zugriff die Methoden first und last ('Erstes' und 'Letztes'). Da der implizite Cursor eines Abfrageergebnisses zunächst vor dem ersten Element positioniert wird, müssen Sie zum Lesen eines Elements die Methode 'Erstes' oder 'Nächstes' aufrufen. Mit der Methode 'Größe' können Sie die Anzahl der Elemente in der Ergebnisliste ermitteln.

Ein Element des Abfrageergebnisses besteht aus den ausgewählten Attributen von Arbeitselementen und den zugehörigen Referenzobjekten wie Aktivitätsinstanzen und Prozessinstanzen. Das erste Attribut (Spalte) eines QueryResultSet-Elements gibt den Wert des ersten Attributs an, das in der Klausel SELECT der Abfrageanforderung angegeben ist. Das zweite Attribut (Spalte) eines QueryResultSet-Elements gibt den Wert des zweiten Attributs an, das in der Klausel SELECT der Abfrageanforderung angegeben ist, usw.

Die Werte der Attribute können Sie durch Aufrufen einer Methode abfragen, die mit dem Attributtyp kompatibel ist, und durch Angeben des entsprechenden Spaltenindexes. Die Nummerierung der Spaltenindizes beginnt mit 1.

Attributtyp	**Methode**
Zeichenfolge	getString
Objektkennung	getOID
Zeitmarke	getTimestamp getString getTimestampAsLong
Ganzzahl	getInteger getShort getLong getString getBoolean
Boolesch	getBoolean getShort getInteger getLong getString
Byte[]	getBinary

Beispiel:

Die folgende Abfrage wird ausgeführt:

```
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);
```

Das zurückgegebene Abfrageergebnis enthält vier Spalten:
- Spalte 1 ist eine Zeitmarke
- Spalte 2 ist eine Zeichenfolge
- Spalte 3 ist eine Objekt-ID
- Spalte 4 ist eine Ganzzahl

Mit den folgenden Methoden können Sie die Attributwerte abrufen:

```
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);
}
```

Sie können die Anzeigenamen des Ergebnisses beispielsweise als Überschriften für eine gedruckte Tabelle verwenden. Diese Namen sind die Spaltennamen der Ansicht oder der durch die Klausel AS der Abfrage definierte Name. Mit der folgenden Methode können Sie die Anzeigenamen in dem Beispiel abrufen:

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

Benutzerspezifische Zugriffsbedingungen:

Benutzerspezifische Zugriffsbedingungen werden hinzugefügt, wenn die SQL-Anweisung SELECT aus der API-Abfrage generiert wird. Diese Bedingungen garantieren, dass nur die Objekte an den Aufrufenden zurückgegeben werden, die der durch den Aufrufenden angegebenen Bedingung entsprechen und für die der Aufrufende berechtigt ist.

Welche Zugriffsbedingung hinzugefügt wird, ist davon abhängig, ob der Benutzer ein Systemadministrator ist.

Von Benutzern ohne Systemadministratorberechtigung aufgerufene Abfragen

Die generierte SQL-Klausel WHERE kombiniert die API-WHERE-Klausel mit einer Bedingung für die Zugriffssteuerung, die für den Benutzer spezifisch ist. Mit der Abfrage werden nur die Objekte abgerufen, auf die der Benutzer zugreifen darf, d. h., nur die Objekte, für die der Benutzer ein Arbeitselement aufweisen kann. Ein Arbeitselement stellt die Zuordnung eines Benutzers oder einer Benutzergruppe zu einem Berechtigungsaufgabenbereich eines Business-Objekts wie z. B. einer Task oder einem Prozess dar. Wenn beispielsweise der Benutzer Karl Schmidt ein Mitglied des Aufgabebereichs der potenziellen Eigner einer angegebenen Task ist, gibt es ein Arbeitselementobjekt, das diese Beziehung repräsentiert.

Wenn beispielsweise ein Benutzer, der kein Systemadministrator ist, Tasks abfragt, wird der Klausel WHERE die folgende Zugriffsbedingung hinzugefügt, wenn keine Gruppenarbeitselemente aktiviert sind:

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

Wenn also Karl Schmidt eine Liste der Tasks abrufen möchte, für die er der potenzielle Eigner ist, sieht die API-WHERE-Klausel ähnlich wie folgt aus:

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

Diese API-WHERE-Klauseln resultieren in der SQL-Anweisung in der folgenden Zugriffsbedingung:

```
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 = true)
AND WI.REASON = 1
```

Dies bedeutet außerdem, dass wenn Karl Schmidt die Aktivitäten und Tasks anzeigen möchte, für die er ein Prozessleser oder ein Prozessadministrator ist und für die er über kein Arbeitselement verfügt, muss der SELECT- oder der WHERE-Klausel bzw. der Sortierklausel der Abfrage ein Merkmal der Ansicht PROCESS_INSTANCE hinzugefügt werden, z. B. PROCESS_INSTANCE.PIID.

Wenn Gruppenarbeitselemente aktiviert sind, wird der Klausel WHERE eine zusätzliche Zugriffsbedingung hinzugefügt, mit der ein Benutzer auf Objekte zugreifen kann, auf die auch die Gruppe Zugriff hat.

Von Systemadministratoren aufgerufene Abfragen

Systemadministratoren können die Methode query zum Abrufen von Objekten aufrufen, die über zugeordnete Arbeitselemente verfügen. In diesem Fall wird der generierten SQL-Abfrage eine Verknüpfung (Join) mit der Ansicht WORK_ITEM hinzugefügt; es wird allerdings keine Bedingung für die Zugriffssteuerung für WORK_ITEM.OWNER_ID hinzugefügt.

In diesem Fall enthält die SQL-Abfrage für Tasks Folgendes:

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

queryAll-Abfragen

Dieser Typ von Abfrage kann nur von Systemadministratoren oder Systemmonitoren aufgerufen werden. Der Ansicht WORK_ITEM werden weder Bedingungen für die Zugriffssteuerung noch eine Verknüpfung (Join) hinzugefügt. Dieser Typ von Abfrage gibt alle Datei für sämtliche Objekte zurück.

Beispiele für die Methoden query und queryAll:

In diesen Beispielen wird die Syntax verschiedener herkömmlicher API-Abfragen und der entsprechenden SQL-Anweisungen aufgezeigt, die bei der Verarbeitung der Abfrage generiert werden.

Beispiel: Tasks im Status 'Bereit' abfragen:

In diesem Beispiel wird die Verwendung der Methode query zum Abrufen von Tasks, mit denen der angemeldete Benutzer arbeiten kann.

Karl Schmidt möchte eine Liste der Tasks abrufen, die ihm zugeordnet wurden. Damit ein Benutzer mit einer Task arbeiten kann, muss die Task den Status 'Bereit' aufweisen. Der angemeldete Benutzer muss für die Task außerdem über ein Arbeitselement für einen möglichen Eigner verfügen. Das folgende Codefragment zeigt den Methodenaufruf query für diese Abfrage:

```
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 )
```

Die folgenden Aktionen werden ausgeführt, wenn die SQL-Anweisung SELECT generiert wird:

- Der WHERE-Klausel wird eine Bedingung für die Zugriffssteuerung hinzugefügt. Bei diesem Beispiel wird angenommen, dass Arbeitselemente für Gruppen nicht aktiviert sind.
- Konstanten wie z. B. TASK.STATE.STATE_READY werden, ihren numerischen Werten entsprechend ersetzt.
- Es werden eine Klausel FROM sowie Bedingungen des Typs 'join' hinzugefügt.

Das folgende Codefragment zeigt die SQL-Anweisung, die auf der Basis der API-Abfrage generiert wird:

```
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
   AND  ( WI.OWNER_ID = 'KarlSchmidt' OR WI.OWNER_ID = null AND  WI.EVERYBODY = true )
```

Um die API-Abfrage auf Tasks für einen bestimmten Prozess zu beschränken, z. B. sampleProcess, muss die Abfrage wie folgt aussehen:

```
query( "DISTINCT TASK.TKIID",
           "PROCESS_TEMPLATE.NAME = 'Musterprozess' 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 )
```

Beispiel: Tasks im Status 'Angefordert' abfragen:

In diesem Beispiel wird die Verwendung der Methode query zum Abrufen von Tasks, die der angemeldete Benutzer angefordert hat.

Der Benutzer Karl Schmidt möchte nach Tasks suchen, die er angefordert hat und die sich immer noch im Status 'Angefordert' befinden. Die Bedingung, mit der angegeben wird, dass Karl Schmidt die Tasks angefordert hat, lautet TASK.OWNER = 'KarlSchmidt'. Das folgende Codefragment zeigt den Methodenaufruf query für die Abfrage:

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

Das folgende Codefragment zeigt die SQL-Anweisung, die auf der Basis der API-Abfrage generiert wird:

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

Wenn eine Task angefordert wurde, werden für den Eigner der Task Arbeitselemente erstellt. Eine Alternative zur Erstellung der Abfrage für Karl Schmidts angeforderte Tasks ist das Hinzufügen der folgenden Bedingung zur Abfrage anstelle der Verwendung von TASK.OWNER = 'KarlSchmidt':

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

Die Abfrage sieht dann wie im folgenden Codefragment dargestellt aus:

```
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 )
```

Die folgenden Aktionen werden ausgeführt, wenn die SQL-Anweisung SELECT generiert wird:

- Der WHERE-Klausel wird eine Bedingung für die Zugriffssteuerung hinzugefügt. Bei diesem Beispiel wird angenommen, dass Arbeitselemente für Gruppen nicht aktiviert sind.
- Konstanten wie z. B. `TASK.STATE.STATE_READY` werden, ihren numerischen Werten entsprechend ersetzt.
- Es werden eine Klausel FROM sowie Bedingungen des Typs 'join' hinzugefügt.

Das folgende Codefragment zeigt die SQL-Anweisung, die auf der Basis der API-Abfrage generiert wird:

```
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 = 'KarlSchmidt' OR WI.OWNER_ID = null AND WI.EVERYBODY = true )
```

Karl nimmt frei, und seine Teamleiterin Anne Grant möchte seine momentane Arbeitsauslastung prüfen. Anne verfügt über Berechtigungen für Systemadministratoren. Bei der von ihr aufgerufenen Abfrage handelt es sich um dieselbe Abfrage, die Karl aufgerufen hat. Die generierte SQL-Anweisung jedoch unterscheidet sich, da Anne Administrator ist. Im folgenden Codefragment sehen Sie die generierte SQL-Anweisung:

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

Da Anne ein Administrator ist, wird der Klausel WHERE keine Bedingung zur Zugriffssteuerung hinzugefügt.

Beispiel: Eskalationen abfragen:

In diesem Beispiel wird die Verwendung der Methode query zum Abrufen von Eskalationen für den angemeldeten Benutzer dargestellt.

Wenn eine Task eskaliert wird, wird ein Arbeitselement für Eskalationsempfänger erstellt. Der Benutzer Mary Jones möchte eine Liste der Tasks anzeigen, die für sie eskaliert wurden. Das folgende Codefragment zeigt den Methodenaufruf query für die Abfrage:

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

Die folgenden Aktionen werden ausgeführt, wenn die SQL-Anweisung SELECT generiert wird:

- Der WHERE-Klausel wird eine Bedingung für die Zugriffssteuerung hinzugefügt. Bei diesem Beispiel wird angenommen, dass Arbeitselemente für Gruppen nicht aktiviert sind.

- Konstanten wie z. B. TASK.STATE.STATE_READY werden, ihren numerischen Werten entsprechend ersetzt.
- Es werden eine Klausel FROM sowie Bedingungen des Typs 'join' hinzugefügt.

Das folgende Codefragment zeigt die SQL-Anweisung, die auf der Basis der API-Abfrage generiert wird:

```
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 )
```

Beispiel: Verwendung der Methode queryAll:

In diesem Beispiel wird die Verwendung der Methode queryAll zum Abrufen aller Aktivitäten dargestellt, die zu einer Prozessschablone gehören.

Die Methode queryAll ist nur für Benutzer mit Systemadministrator oder Systemmonitorberechtigungen verfügbar. Das folgende Codefragment zeigt den Methodenaufruf queryAll für die Abfrage, um alle Aktivitäten abzurufen, die zu der Prozessschablone sampleProcess gehören:

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

Das folgende Codefragment zeigt die SQL-Abfrage, die auf der Basis der API-Abfrage generiert wird:

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

Da der Aufruf von einem Administrator getätigt wird, wird der generierten SQL-Anweisung keine Bedingung zur Zugriffssteuerung hinzugefügt. Es wird auch keine Bedingung 'join' mit der Ansicht WORK_ITEM hinzugefügt. Dies bedeutet, dass bei der Abfrage alle Aktivitäten für die Prozessschablone abgerufen werden, einschließlich den Aktivitäten ohne Arbeitselemente.

Beispiel: Abfragemerkmale in eine Abfrage einbeziehen:

In diesem Beispiel wird die Verwendung der Methode query zum Abrufen von Tasks dargestellt, die zu einem Business-Prozess gehören. Der Prozess verfügt über Abfragemerkmale, die Sie in die Suchoperation einbeziehen können.

Sie möchten beispielsweise nach allen Benutzertasks im Status 'Bereit' suchen, die zu einem Business-Prozess gehören. Der Prozess weist ein Abfragemerkmal (**customerID**) mit dem Wert CID_12345 und einem Namespace auf. Das folgende Codefragment zeigt den Methodenaufruf query für die Abfrage:

```
query ( " DISTINCT TASK.TKIID, TASK_TEMPL.NAME, TASK.STATE,
                   PROCESS_INSTANCE.NAME",
          " QUERY_PROPERTY.NAME = 'Kunden-ID' 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 ",
        (String)null, (String)null, (Integer)null, (TimeZone)null );
```

Wenn Sie nun der Abfrage ein zweites Abfragemerkmal mit einem angegebenen Namespace hinzufügen möchten, z. B. **Priority**, sieht der Methodenaufruf query für die Abfrage wie folgt aus:

```
query ( " DISTINCT TASK.TKIID, TASK_TEMPL.NAME, TASK.STATE,
                  PROCESS_INSTANCE.NAME",
        " QUERY_PROPERTY1.NAME = 'Kunden-ID' AND " +
        " QUERY_PROPERTY1.STRING_VALUE = 'CID_12345' AND " +
        " QUERY_PROPERTY1.NAMESPACE =
               'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/' AND " +
        " QUERY_PROPERTY2.NAME = 'Priorität' 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 ",
        (String)null, (String)null, (Integer)null, (TimeZone)null );
```

Wenn Sie der Abfrage mehr als ein Abfragemerkmal hinzufügen, müssen Sie alle Merkmale, die Sie hinzufügen, wie im Codefragment dargestellt nummerieren. Die Abfrage von angepassten Merkmalen wirkt sich jedoch auf die Leistung aus; die Leistung verschlechtert sich mit der Anzahl der angepassten Merkmale in der Abfrage.

Beispiel: Angepasste Merkmale in eine Abfrage einbeziehen:

In diesem Beispiel wird die Verwendung der Methode query zum Abrufen von Tasks dargestellt, die über angepasste Merkmale verfügen.

Sie möchten beispielsweise nach allen Benutzertasks im Status 'Bereit' suchen, die über ein angepasstes Merkmal (**customerID**) verfügen und den Wert `CID_12345` aufweisen. Das folgende Codefragment zeigt den Methodenaufruf query für die Abfrage:

```
query ( " DISTINCT TASK.TKIID ",
        " TASK_CPROP.NAME = 'Kunden-ID' 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 ",
        (String)null, (String)null, (Integer)null, (TimeZone)null );
```

Wenn Sie nun die Tasks und ihre angepassten Merkmale abfragen möchten, sieht der Methodenaufruf query für die Abfrage wie folgt aus:

```
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 ",
        (String)null, (String)null, (Integer)null, (TimeZone)null );
```

Das folgende Codefragment zeigt die SQL-Anweisung, die auf der Basis dieser API-Abfrage generiert wird:

```
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 = 'KarlSchmidt' OR WI.OWNER_ID IS NULL AND WI.EVERYBODY = 1 )
```

In dieser SQL-Anweisung ist zwischen der Ansicht TASK und der Ansicht TASK_CPROP ein Outer Join enthalten. Dies bedeutet, dass Tasks, die der Klausel WHERE entsprechen, auch abgerufen werden, wenn sie keine angepassten Merkmale aufweisen.

Vordefinierte Ansichten zum Abfragen von Business-Prozess- und Benutzertaskobjekten:

Für Business-Prozess- und Benutzertaskobjekte werden vordefinierte Datenbankansichten bereitgestellt. Verwenden Sie diese Ansichten zum Abfragen von Referenzdaten für diese Objekte.

Bei Verwendung der vordefinierten Ansichten müssen Sie keine Verknüpfungsprädikate für Ansichtsspalten explizit hinzufügen. Diese Konstrukte werden automatisch hinzugefügt. Sie können die generische Abfragefunktion der Service-API (BusinessFlowManagerService oder HumanTaskManagerService) verwenden, um diese Daten abzufragen. Außerdem können Sie die entsprechende Methode der API HumanTaskManagerDelegate oder die vordefinierten Abfragen Ihrer Implementierungen der Schnittstelle ExecutableQuery verwenden.

Anmerkung: Die Ansichten enthalten möglicherweise nicht beschriebene Spalten. Diese Spalten dienen nur der internen Verwendung.

Ansicht ACTIVITY:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen von Aktivitäten.

Tabelle 3. Spalten in der Ansicht ACTIVITY

Spaltenname	Typ	Kommentare
PIID	ID	Die ID der Prozessinstanz
AIID	ID	Die ID der Aktivitätsinstanz
PTID	ID	Die ID der Prozessschablone
ATID	ID	Die ID der Aktivitätsschablone

Tabelle 3. Spalten in der Ansicht ACTIVITY (Forts.)

Spaltenname	Typ	Kommentare
KIND	Ganzzahl	Die Art der Aktivität; gültige Werte sind: 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	Zeitmarke	Die Zeit, zu der die Aktivität beendet wird
ACTIVATED	Zeitmarke	Die Zeit, zu der die Aktivität aktiviert wird
FIRST_ACTIVATED	Zeitmarke	Die Zeit, zu der die Aktivität zum ersten Mal aktiviert wurde
STARTED	Zeitmarke	Die Zeit, zu der die Aktivität gestartet wird
STATE	Ganzzahl	Der Status der Aktivität; gültige Werte sind: 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	Zeichenfolge	Teilnehmer-ID des Eigners

Tabelle 3. Spalten in der Ansicht ACTIVITY (Forts.)

Spaltenname	Typ	Kommentare
DESCRIPTION	Zeichenfolge	Wenn die Beschreibung der Aktivitätsschablone Platzhalter enthält, enthält diese Spalte die Beschreibung der Aktivitätsschablone mit aufgelösten Platzhaltern
TEMPLATE_NAME	Zeichenfolge	Name der zugeordneten Aktivitätsschablone
TEMPLATE_DESCR	Zeichenfolge	Beschreibung der zugeordneten Aktivitätsschablone
BUSINESS_RELEVANCE	Boolesch	Gibt an, ob die Aktivität geschäftsrelevant ist gültige Werte sind: **TRUE** Die Aktivität ist geschäftsrelevant. Sie können den Aktivitätsstatus in Business Process Choreographer Explorer anzeigen. **FALSE** Die Aktivität ist nicht geschäftsrelevant.
EXPIRES	Zeitmarke	Datum und Uhrzeit des Zeitpunkts, an dem die Taskinstanz abläuft. Datum und Uhrzeit des Eintretens dieses Ereignisses, wenn die Aktivität abgelaufen ist.

Ansicht ACTIVITY_ATTRIBUTE:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen benutzerdefinierter Merkmale für Aktivitäten.

Tabelle 4. Spalten in der Ansicht ACTIVITY_ATTRIBUTE

Spaltenname	Typ	Kommentare
AIID	ID	Die ID der Aktivitätsinstanz, die über ein benutzerdefiniertes Merkmal verfügt
NAME	Zeichenfolge	Der Name des benutzerdefinierten Merkmals
VALUE	Zeichenfolge	Der Wert des benutzerdefinierten Merkmals

Ansicht ACTIVITY_SERVICE:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen von Aktivitätsservices.

Tabelle 5. Spalten in der Ansicht ACTIVITY_SERVICE

Spaltenname	Typ	Kommentare
EIID	ID	Die ID der Ereignisinstanz
AIID	ID	Die ID der Aktivitätsinstanz, die auf das Ereignis wartet

Tabelle 5. Spalten in der Ansicht ACTIVITY_SERVICE (Forts.)

Spaltenname	Typ	Kommentare
PIID	ID	Die ID der Prozessinstanz, die das Ereignis enthält
VTID	ID	Die ID der Serviceschablone, die das Ereignis beschreibt
PORT_TYPE	Zeichenfolge	Der Name des Porttyps
NAME_SPACE_URI	Zeichenfolge	Der URI des Namespace
OPERATION	Zeichenfolge	Der Operationsname des Service

Ansicht APPLICATION_COMP:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen der Anwendungskomponenten-ID und der Standardeinstellungen für Tasks.

Tabelle 6. Spalten in der Ansicht APPLICATION_COMP

Spaltenname	Typ	Kommentare
ACOID	Zeichenfolge	Die ID der Anwendungskomponente
BUSINESS_ RELEVANCE	Boolesch	Die Standardtaskrichtlinie für die Geschäftsrelevanz der Komponente (der Wert kann durch eine Definition in der Taskschablone oder Task überschrieben werden, und das Attribut wirkt sich auf die Prüfprotokollierung aus); gültige Werte sind: **TRUE** Die Task ist geschäftsrelevant und wird protokolliert **FALSE** Die Task ist nicht geschäftsrelevant und wird nicht protokolliert
NAME	Zeichenfolge	Der Name der Anwendungskomponente
SUPPORT_ AUTOCLAIM	Boolesch	Die Standardrichtlinie für automatische Beanspruchung der Komponente (ist dieses Attribut auf TRUE gesetzt, kann die Task automatisch beansprucht werden, wenn der potenzielle Eigner ein einzelner Benutzer ist); dieser Wert kann durch eine Definition in der Taskschablone oder Task überschrieben werden
SUPPORT_CLAIM_ SUSP	Boolesch	Die Standardeinstellung der Komponente, die festlegt, ob ausgesetzte Tasks beansprucht werden können (ist dieses Attribut auf TRUE gesetzt, können ausgesetzte Tasks beansprucht werden); der Wert kann durch eine Definition in der Taskschablone oder Task überschrieben werden.
SUPPORT_ DELEGATION	Boolesch	Die Standardrichtlinie für Taskdelegierung der Komponente. Ist dieses Attribut auf TRUE gesetzt, können die Arbeitselementzuordnungen für die Task geändert werden. Dies bedeutet, dass die Arbeitselemente erstellt, gelöscht oder übertragen werden können.

Tabelle 6. Spalten in der Ansicht APPLICATION_COMP (Forts.)

Spaltenname	Typ	Kommentare
SUPPORT_ FOLLOW_ON	Boolesch	Die Standardrichtlinie für Folgetask der Komponente. Ist dieses Attribut auf TRUE gesetzt, können Folgetasks für Tasks erstellt werden. Der Wert kann durch eine Definition in der Taskschablone oder Task überschrieben werden.
SUPPORT_ SUB_TASK	Boolesch	Die Standardrichtlinie für Subtask der Komponente. Ist dieses Attribut auf TRUE gesetzt, können Subtasks für Tasks erstellt werden. Der Wert kann durch eine Definition in der Taskschablone oder Task überschreiben werden.

Ansicht ESCALATION:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen von Daten für Eskalationen.

Tabelle 7. Spalten in der Ansicht ESCALATION

Spaltenname	Typ	Kommentare
ESIID	Zeichenfolge	Die ID der Eskalationsinstanz
ACTION	Ganzzahl	Die von der Eskalation ausgelöste Aktion; gültige Werte sind: **ACTION_CREATE_WORK_ITEM (1)** Erstellt ein Arbeitselement für jeden Eskalationsempfänger **ACTION_SEND_EMAIL (2)** Sendet eine E-Mail an jeden Eskalationsempfänger **ACTION_CREATE_EVENT (3)** Erstellt und publiziert ein Ereignis
ACTIVATION_STATE	Ganzzahl	Eine Eskalationsinstanz wird erstellt, wenn die entsprechende Task einen der folgenden Statuszustände erreicht: **ACTIVATION_STATE_READY (2)** Gibt an, dass die Benutzertask oder teilnehmende Task beansprucht werden kann **ACTIVATION_STATE_RUNNING (3)** Gibt an, dass die ursprüngliche Task gestartet und aktiv ist **ACTIVATION_STATE_CLAIMED (8)** Gibt an, dass die Task beansprucht ist **ACTIVATION_STATE_WAITING_ FOR_SUBTASK (20)** Gibt an, dass die Task auf die Beendigung von Subtasks wartet.
ACTIVATION_TIME	Zeitmarke	Die Zeit, zu der die Eskalation aktiviert wird

Tabelle 7. Spalten in der Ansicht ESCALATION (Forts.)

Spaltenname	Typ	Kommentare
AT_LEAST_ EXP_STATE	Ganzzahl	Der von der Eskalation erwartete Taskstatus (wenn eine Zeitlimitüberschreitung eintritt, wird der Taskstatus mit dem Wert dieses Attributs verglichen); gültige Werte sind: **AT_LEAST_EXPECTED_STATE_CLAIMED (8)** Gibt an, dass die Task beansprucht ist **AT_LEAST_EXPECTED_STATE_ENDED (20)** Gibt an, dass sich die Task in einem Endstatus befindet (FINISHED, FAILED, TERMINATED oder EXPIRED) **AT_LEAST_EXPECTED_STATE_ SUBTASKS_COMPLETED (21)** Gibt an, dass alle Subtasks der Task beendet sind.
ESTID	Zeichenfolge	Die ID der entsprechenden Eskalationsschablone
FIRST_ESIID	Zeichenfolge	Die ID der ersten Eskalation in der Kette
INCREASE_PRIORITY	Ganzzahl	Gibt an, wie die Priorität der Task erhöht wird; gültige Werte sind: **INCREASE_PRIORITY_NO (1)** Die Taskpriorität wird nicht erhöht **INCREASE_PRIORITY_ONCE (2)** Die Taskpriorität wird um Eins erhöht **INCREASE_PRIORITY_REPEATED (3)** Die Taskpriorität wird bei jeder Wiederholung der Eskalation um Eins erhöht
NAME	Zeichenfolge	Der Name der Eskalation
STATE	Ganzzahl	Der Status der Eskalation; gültige Werte sind: STATE_INACTIVE (1) STATE_WAITING (2) STATE_ESCALATED (3) STATE_SUPERFLUOUS (4)
TKIID	Zeichenfolge	Die ID der Taskinstanz, zu der die Eskalation gehört

Ansicht ESCALATION_CPROP:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen der benutzerdefinierten Merkmale für Eskalationen.

Tabelle 8. Spalten in der Ansicht ESCALATION_CPROP

Spaltenname	Typ	Kommentare
ESIID	Zeichenfolge	Die ID der Eskalation
NAME	Zeichenfolge	Der Name des Merkmals
DATA_TYPE	Zeichenfolge	Der Typ der Klasse für benutzerdefinierte Merkmale, die keine Zeichenfolgen sind
STRING_VALUE	Zeichenfolge	Der Wert für benutzerdefinierte Merkmale des Typs 'Zeichenfolge'

Ansicht ESCALATION_DESC:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen mehrsprachiger beschreibender Daten für Eskalationen.

Tabelle 9. Spalten in der Ansicht ESCALATION_DESC

Spaltenname	Typ	Kommentare
ESIID	Zeichenfolge	Die ID der Eskalation
LOCALE	Zeichenfolge	Der Name der länderspezifischen Angaben, die der Beschreibung oder dem Anzeigenamen zugeordnet sind
DESCRIPTION	Zeichenfolge	Eine Beschreibung der Taskschablone
DISPLAY_NAME	Zeichenfolge	Der beschreibende Name der Eskalation

Ansicht ESC_TEMPL:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen von Daten für Eskalationsschablonen.

Tabelle 10. Spalten in der Ansicht ESC_TEMPL

Spaltenname	Typ	Kommentare
ESTID	Zeichenfolge	Die ID der Eskalationsschablone
ACTION	Ganzzahl	Die von der Eskalation ausgelöste Aktion; gültige Werte sind: **ACTION_CREATE_WORK_ITEM (1)** Erstellt ein Arbeitselement für jeden Eskalationsempfänger **ACTION_SEND_EMAIL (2)** Sendet eine E-Mail an jeden Eskalationsempfänger **ACTION_CREATE_EVENT (3)** Erstellt und publiziert ein Ereignis
ACTIVATION_STATE	Ganzzahl	Eine Eskalationsinstanz wird erstellt, wenn die entsprechende Task einen der folgenden Statuszustände erreicht: **ACTIVATION_STATE_READY (2)** Gibt an, dass die Benutzertask oder teilnehmende Task beansprucht werden kann **ACTIVATION_STATE_RUNNING (3)** Gibt an, dass die ursprüngliche Task gestartet und aktiv ist **ACTIVATION_STATE_CLAIMED (8)** Gibt an, dass die Task beansprucht ist **ACTIVATION_STATE_WAITING_ FOR_SUBTASK (20)** Gibt an, dass die Task auf die Beendigung von Subtasks wartet.

Tabelle 10. Spalten in der Ansicht ESC_TEMPL (Forts.)

Spaltenname	Typ	Kommentare
AT_LEAST_ EXP_STATE	Ganzzahl	Der von der Eskalation erwartete Taskstatus (wenn eine Zeitlimitüberschreitung eintritt, wird der Taskstatus mit dem Wert dieses Attributs verglichen); gültige Werte sind: **AT_LEAST_EXPECTED_STATE_CLAIMED (8)** Gibt an, dass die Task beansprucht ist **AT_LEAST_EXPECTED_STATE_ENDED (20)** Gibt an, dass sich die Task in einem Endstatus befindet (FINISHED, FAILED, TERMINATED oder EXPIRED) **AT_LEAST_EXPECTED_STATE_ SUBTASKS_COMPLETED (21)** Gibt an, dass alle Subtasks der Task beendet sind.
CONTAINMENT_ CTX_ID	Zeichenfolge	Wenn die Eskalationsschablone zu einer integrierten Taskschablone gehört, ist der Einschlusskontext die Prozessschablone. Wenn der Kontext für Eskalationsschablonen zu einer eigenständigen Taskschablone gehört, ist der Einschlusskontext die Taskschablone.
FIRST_ESTID	Zeichenfolge	Die ID der ersten Eskalationsschablone in einer Kette von Eskalationsschablonen
INCREASE_PRIORITY	Ganzzahl	Gibt an, wie die Priorität der Task erhöht wird; gültige Werte sind: **INCREASE_PRIORITY_NO (1)** Die Taskpriorität wird nicht erhöht **INCREASE_PRIORITY_ONCE (2)** Die Taskpriorität wird um Eins erhöht **INCREASE_PRIORITY_REPEATED (3)** Die Taskpriorität wird bei jeder Wiederholung der Eskalation um Eins erhöht
NAME	Zeichenfolge	Der Name der Eskalationsschablone
PREVIOUS_ESTID	Zeichenfolge	Die ID der vorherigen Eskalationsschablone in einer Kette von Eskalationsschablonen
TKTID	Zeichenfolge	Die ID der Taskschablone, zu der die Eskalationsschablone gehört

Ansicht ESC_TEMPL_CPROP:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen der benutzerdefinierten Merkmale für Eskalationsschablonen.

Tabelle 11. Spalten in der Ansicht ESC_TEMPL_CPROP

Spaltenname	Typ	Kommentare
ESTID	Zeichenfolge	Die ID der Eskalationsschablone
NAME	Zeichenfolge	Der Name des Merkmals
TKTID	Zeichenfolge	Die ID der Taskschablone, zu der die Eskalationsschablone gehört

Tabelle 11. Spalten in der Ansicht ESC_TEMPL_CPROP (Forts.)

Spaltenname	Typ	Kommentare
DATA_TYPE	Zeichenfolge	Der Typ der Klasse für benutzerdefinierte Merkmale, die keine Zeichenfolgen sind
VALUE	Zeichenfolge	Der Wert für benutzerdefinierte Merkmale des Typs 'Zeichenfolge'

Ansicht ESC_TEMPL_DESC:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen mehrsprachiger beschreibender Daten für Eskalationsschablonen.

Tabelle 12. Spalten in der Ansicht ESC_TEMPL_DESC

Spaltenname	Typ	Kommentare
ESTID	Zeichenfolge	Die ID der Eskalationsschablone
LOCALE	Zeichenfolge	Der Name der länderspezifischen Angaben, die der Beschreibung oder dem Anzeigenamen zugeordnet sind
TKTID	Zeichenfolge	Die ID der Taskschablone, zu der die Eskalationsschablone gehört
DESCRIPTION	Zeichenfolge	Eine Beschreibung der Taskschablone
DISPLAY_NAME	Zeichenfolge	Der beschreibende Name der Eskalation

Ansicht PROCESS_ATTRIBUTE:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen benutzerdefinierter Merkmale für Prozesse.

Tabelle 13. Spalten in der Ansicht PROCESS_ATTRIBUTE

Spaltenname	Typ	Kommentare
PIID	ID	Die ID der Prozessinstanz, die über ein benutzerdefiniertes Merkmal verfügt
NAME	Zeichenfolge	Der Name des benutzerdefinierten Merkmals
VALUE	Zeichenfolge	Der Wert des benutzerdefinierten Merkmals

Ansicht PROCESS_INSTANCE:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen von Prozessinstanzen.

Tabelle 14. Spalten in der Ansicht PROCESS_INSTANCE

Spaltenname	Typ	Kommentare
PTID	ID	Die ID der Prozessschablone
PIID	ID	Die ID der Prozessinstanz
NAME	Zeichenfolge	Der Name der Prozessinstanz

Tabelle 14. Spalten in der Ansicht PROCESS_INSTANCE (Forts.)

Spaltenname	Typ	Kommentare
STATE	Ganzzahl	Der Status der Prozessinstanz; gültige Werte sind: 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	Zeitmarke	Die Zeit, zu der die Prozessinstanz erstellt wird
STARTED	Zeitmarke	Die Zeit, zu der die Prozessinstanz gestartet wird
COMPLETED	Zeitmarke	Die Zeit, zu der die Prozessinstanz beendet wurde
PARENT_PIID	ID	Die ID der übergeordneten Prozessinstanz
PARENT_NAME	Zeichenfolge	Der Name der übergeordneten Prozessinstanz
TOP_LEVEL_PIID	ID	Die Prozessinstanz-ID der Prozessinstanz der höchsten Ebene (gibt es keine Prozessinstanz der höchsten Ebene, ist dies die Prozessinstanz-ID der aktuellen Prozessinstanz)
TOP_LEVEL_NAME	Zeichenfolge	Der Name der Prozessinstanz der höchsten Ebene (gibt es keine Prozessinstanz der höchsten Ebene, ist dies der Name der aktuellen Prozessinstanz)
STARTER	Zeichenfolge	Die Teilnehmer-ID des Starters der Prozessinstanz
DESCRIPTION	Zeichenfolge	Wenn die Beschreibung der Prozessschablone Platzhalter enthält, enthält diese Spalte die Beschreibung der Prozessschablone mit aufgelösten Platzhaltern
TEMPLATE_NAME	Zeichenfolge	Der Name der zugeordneten Prozessschablone
TEMPLATE_DESCR	Zeichenfolge	Die Beschreibung der zugeordneten Prozessschablone
RESUMES	Zeitmarke	Die Zeit, zu der die Prozessinstanz automatisch fortgesetzt werden soll

Ansicht PROCESS_TEMPLATE:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen von Prozessschablonen.

Tabelle 15. Spalten in der Ansicht PROCESS_TEMPLATE

Spaltenname	Typ	Kommentare
PTID	ID	Die ID der Prozessschablone
NAME	Zeichenfolge	Der Name der Prozessschablone
VALID_FROM	Zeitmarke	Die Zeit, ab der eine Instanz der Prozessschablone erstellt werden kann
TARGET_NAMESPACE	Zeichenfolge	Der Zielnamespace der Prozessschablone

Tabelle 15. Spalten in der Ansicht PROCESS_TEMPLATE (Forts.)

Spaltenname	Typ	Kommentare
APPLICATION_NAME	Zeichenfolge	Der Name der Unternehmensanwendung, zu der die Prozessschablone gehört
VERSION	Zeichenfolge	Benutzerdefinierte Versionsangabe
CREATED	Zeitmarke	Die Zeit, zu der die Prozessschablone in der Datenbank erstellt wird
STATE	Ganzzahl	Gibt an, ob die Prozessschablone zum Erstellen von Prozessinstanzen verfügbar ist; gültige Werte sind: STATE_STARTED (1) STATE_STOPPED (2)
EXECUTION_MODE	Ganzzahl	Gibt an, wie Prozessinstanzen ausgeführt werden können, die von dieser Prozessschablone abgeleitet wurden; gültige Werte sind: EXECUTION_MODE_MICROFLOW (1) EXECUTION_MODE_LONG_RUNNING (2)
DESCRIPTION	Zeichenfolge	Die Beschreibung der Prozessschablone
COMP_SPHERE	Ganzzahl	Gibt das Kompensationsverhalten von Mikroprozessinstanzen in der Prozessschablone an (es wird ein vorhandener Kompensationsrahmen verwendet oder ein neuer erstellt); gültige Werte sind: COMP_SPHERE_REQUIRED (2) COMP_SPHERE_SUPPORTS (4)
DISPLAY_NAME	Zeichenfolge	Der beschreibende Name des Prozesses

Ansicht QUERY_PROPERTY:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen von Variablen auf Prozessebene.

Tabelle 16. Spalten in der Ansicht QUERY_PROPERTY

Spaltenname	Typ	Kommentare
PIID	ID	Die ID der Prozessinstanz
VARIABLE_NAME	Zeichenfolge	Der Name der Variablen auf Prozessebene.
NAME	Zeichenfolge	Der Name des Abfragemerkmals.
NAMESPACE	Zeichenfolge	Der Namespace für das Abfragemerkmal.
GENERIC_VALUE	Zeichenfolge	Eine Zeichenfolgedarstellung für Merkmalstypen, die auf keinen der definierten Typen STRING_VALUE, NUMBER_VALUE, DECIMAL_VALUE und TIMESTAMP_VALUE abgestimmt werden können.

Tabelle 16. Spalten in der Ansicht QUERY_PROPERTY (Forts.)

Spaltenname	Typ	Kommentare
STRING_VALUE	Zeichenfolge	Wenn ein Merkmalstyp auf einen Zeichenfolgetyp abgestimmt wird, ist dies der Wert der Zeichenfolge.
NUMBER_VALUE	Ganzzahl	Wenn ein Merkmalstyp auf einen ganzzahligen Typ abgestimmt wird, ist dies der Wert der ganzen Zahl.
DECIMAL_VALUE	Dezimalzahl	Wenn ein Merkmalstyp auf einen Gleitkommatyp abgestimmt wird, ist dies der Wert der Dezimalzahl.
TIMESTAMP_VALUE	Zeitmarke	Wenn ein Merkmalstyp auf eine Zeitmarke abgestimmt wird, ist dies der Wert der Zeitmarke.

Ansicht TASK:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen von Taskobjekten.

Tabelle 17. Spalten in der Ansicht TASK

Spaltenname	Typ	Kommentare
TKIID	ID	Die ID der Taskinstanz
ACTIVATED	Zeitmarke	Die Zeit, zu der die Task aktiviert wurde
APPLIC_ DEFAULTS_ID	ID	Die ID der Anwendungskomponente, die die Standardwerte für die Task angibt
APPLIC_NAME	Zeichenfolge	Der Name der Unternehmensanwendung, zu der die Task gehört
BUSINESS_ RELEVANCE	Boolesch	Gibt an, ob die Task geschäftsrelevant ist (das Attribut wirkt sich auf die Prüfprotokollierung aus); gültige Werte sind: **TRUE** Die Task ist geschäftsrelevant und wird protokolliert **FALSE** Die Task ist nicht geschäftsrelevant und wird nicht protokolliert
COMPLETED	Zeitmarke	Die Zeit, zu der die Task beendet wurde
CONTAINMENT_ CTX_ID	ID	Die ID des Einschlusskontexts für diese Task (dieses Attribut bestimmt den Lebenszyklus der Task); beim Löschen des Einschlusskontexts einer Task wird auch die Task gelöscht
CTX_ AUTHORIZATION	Ganzzahl	Ermöglicht dem Taskeigner den Zugriff auf den Taskkontext; gültige Werte sind: **AUTH_NONE** Keine Zugriffsberechtigung für das zugeordnete Kontextobjekt **AUTH_READER** Zum Bearbeiten des zugeordneten Kontextobjekts ist Leseberechtigung erforderlich (z. B. Lesen der Merkmale einer Prozessinstanz)

Tabelle 17. Spalten in der Ansicht TASK (Forts.)

Spaltenname	Typ	Kommentare
DUE	Zeitmarke	Die Zeit, zu der die Task fällig ist
EXPIRES	Zeitmarke	Das Datum, an dem die Task abläuft
FIRST_ACTIVATED	Zeitmarke	Die Zeit, zu der die Task zum ersten Mal aktiviert wurde
FOLLOW_ON_TKIID	ID	Die ID der Instanz der Folgetask
HIERARCHY_ POSITION	Ganzzahl	gültige Werte sind: **HIERARCHY_POSITION_TOP_TASK (0)** Die oberste Task in der Taskhierarchie **HIERARCHY_POSITION_SUB_TASK (1)** Die Task ist eine Subtask in der Taskhierarchie **HIERARCHY_POSITION_FOLLOW_ON_TASK (2)** Die Task ist eine Folgetask in der Taskhierarchie
IS_AD_HOC	Boolesch	Gibt an, ob diese Task während der Laufzeit dynamisch erstellt oder von einer Taskschablone erstellt wurde
IS_ESCALATED	Boolesch	Gibt an, ob eine Eskalation dieser Task aufgetreten ist
IS_INLINE	Boolesch	Gibt an, ob diese Task eine integrierte Task in einem Business-Prozess ist
IS_WAIT_FOR_ SUB_TK	Boolesch	Gibt an, ob die übergeordnete Task darauf wartet, dass eine Subtask einen Endestatus erreicht
KIND	Ganzzahl	Die Art der Task; gültige Werte sind: **KIND_HUMAN (101)** Gibt an, dass es sich bei der Task um eine *Collaboration-Task* handelt, die von einem Benutzer erstellt und bearbeitet wird **KIND_WPC_STAFF_ACTIVITY (102)** Gibt an, dass es sich bei der Task um eine Benutzertask handelt, die eine Staff-Aktivität eines Business-Prozesses von WebSphere Business Integration Server Foundation Version 5 ist **KIND_ORIGINATING (103)** Gibt an, dass es sich bei der Task um eine *Aufruftask* handelt, die Computereingaben von Benutzern unterstützt, d. h. Benutzer können Services erstellen, einleiten und starten **KIND_PARTICIPATING (105)** Gibt an, dass es bei der Task um eine *unerledigte Task* handelt, die Computereingaben von Benutzern unterstützt, d. h. Benutzer können Services implementieren **KIND_ADMINISTRATIVE (106)** Gibt an, dass die Task eine Verwaltungstask ist
LAST_MODIFIED	Zeitmarke	Die Zeit, zu der die Task zuletzt geändert wurde

Tabelle 17. Spalten in der Ansicht TASK (Forts.)

Spaltenname	Typ	Kommentare
LAST_STATE_ CHANGE	Zeitmarke	Die Zeit, zu der der Taskstatus zuletzt geändert wurde
NAME	Zeichenfolge	Der Name der Task
NAME_SPACE	Zeichenfolge	Der zum Kategorisieren der Task verwendete Namensbereich
ORIGINATOR	Zeichenfolge	Die Teilnehmer-ID des Erstellers der Task
OWNER	Zeichenfolge	Die Teilnehmer-ID des Eigners der Task
PARENT_ CONTEXT_ID	Zeichenfolge	Der übergeordnete Kontext für diese Task (dieses Attribut ist ein Schlüssel zum entsprechenden Kontext in der aufrufenden Anwendungskomponente); der übergeordnete Kontext wird durch die Anwendungskomponente festgelegt, von der die Task erstellt wird
PRIORITY	Ganzzahl	Die Priorität der Task
RESUMES	Zeitmarke	Die Zeit, zu der die Task automatisch fortgesetzt werden soll
STARTED	Zeitmarke	Die Zeit, zu der die Task gestartet wurde (STATE_RUNNING, STATE_CLAIMED)
STARTER	Zeichenfolge	Die Teilnehmer-ID des Taskstarters
STATE	Ganzzahl	Der Status der Task; gültige Werte sind: **STATE_READY (2)** Gibt an, dass die Task beansprucht werden kann **STATE_RUNNING (3)** Gibt an, dass die Task gestartet und aktiv ist **STATE_FINISHED (5)** Gibt an, dass die Task erfolgreich fertig gestellt wurde **STATE_FAILED (6)** Gibt an, dass die Task nicht erfolgreich fertig gestellt wurde **STATE_TERMINATED (7)** Gibt an, dass die Task aufgrund einer externen oder internen Anforderung beendet wurde **STATE_CLAIMED (8)** Gibt an, dass die Task beansprucht ist **STATE_EXPIRED (12)** Gibt an, dass die Task beendet wurde, weil ihre angegebene Dauer überschritten wurde **STATE_FORWARDED (101)** Gibt an, dass die Task beendet ist und über eine Folgetask verfügt
SUPPORT_ AUTO-CLAIM	Boolesch	Gibt an, ob diese Task automatisch beansprucht wird, wenn Sie einem einzelnen Benutzer zugeordnet ist

Tabelle 17. Spalten in der Ansicht TASK (Forts.)

Spaltenname	Typ	Kommentare
SUPPORT_CLAIM_ SUSP	Boolesch	Gibt an, ob diese Task beansprucht werden kann, wenn sie ausgesetzt ist
SUPPORT_ DELEGA-TION	Boolesch	Gibt an, ob diese Task das Delegieren von Arbeit durch Erstellen, Löschen oder Übertragen von Arbeitselementen unterstützt
SUPPORT_ FOLLOW_ON	Boolesch	Gibt an, ob diese Task das Erstellen von Folgetasks unterstützt
SUPPORT_SUB_TASK	Boolesch	Gibt an, ob diese Task das Erstellen von Subtasks unterstützt
SUSPENDED	Boolesch	Gibt an, ob die Task ausgesetzt ist
TKTID	ID	Die ID der Taskschablone
TOP_TKIID	ID	Die ID der obersten übergeordneten Taskinstanz, wenn dies eine Subtask ist
TYPE	Zeichenfolge	Der zum Kategorisieren der Task verwendete Typ

Ansicht TASK_CPROP:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen der benutzerdefinierten Merkmale für Taskobjekte.

Tabelle 18. Spalten in der Ansicht TASK_CPROP

Spaltenname	Typ	Kommentare
TKIID	Zeichenfolge	Die ID der Taskinstanz
NAME	Zeichenfolge	Der Name des Merkmals
DATA_TYPE	Zeichenfolge	Der Typ der Klasse für benutzerdefinierte Merkmale, die keine Zeichenfolgen sind
STRING_VALUE	Zeichenfolge	Der Wert für benutzerdefinierte Merkmale des Typs 'Zeichenfolge'

Ansicht TASK_DESC:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen mehrsprachiger beschreibender Daten für Taskobjekte.

Tabelle 19. Spalten in der Ansicht TASK_DESC

Spaltenname	Typ	Kommentare
TKIID	Zeichenfolge	Die ID der Taskinstanz
LOCALE	Zeichenfolge	Der Name der länderspezifischen Angaben, die der Beschreibung oder dem Anzeigenamen zugeordnet sind
DESCRIPTION	Zeichenfolge	Eine Beschreibung der Task
DISPLAY_NAME	Zeichenfolge	Der beschreibende Name der Task

Ansicht TASK_TEMPL:

Diese vordefinierte Datenbankansicht enthält Daten, die Sie zum Erstellen von Taskinstanzen verwenden können.

Tabelle 20. Spalten in der Ansicht TASK_TEMPL

Spaltenname	Typ	Kommentare
TKTID	Zeichenfolge	Die ID der Taskschablone
VALID_FROM	Zeitmarke	Die Zeit, ab der die Taskschablone für die Instanzerstellung zur Verfügung steht
APPLIC_ DEFAULTS_ID	Zeichenfolge	Die ID der Anwendungskomponente, die die Standardwerte für die Taskschablone angibt
APPLIC_NAME	Zeichenfolge	Der Name der Unternehmensanwendung, zu der die Taskschablone gehört
BUSINESS_ RELEVANCE	Boolesch	Gibt an, ob die Taskschablone geschäftsrelevant ist (das Attribut wirkt sich auf die Prüfprotokollierung aus); gültige Werte sind: **TRUE** Die Task ist geschäftsrelevant und wird protokolliert **FALSE** Die Task ist nicht geschäftsrelevant und wird nicht protokolliert
CONTAINMENT_ CTX_ID	ID	Der Einschlusskontext für diese Taskschablone (dieses Attribut bestimmt den Lebenszyklus der Taskschablone); beim Löschen eines Einschlusskontexts wird auch die Taskschablone gelöscht
CTX_ AUTHORIZATION	Ganzzahl	Ermöglicht dem Taskeigner den Zugriff auf den Taskkontext; gültige Werte sind: **AUTH_NONE** Keine Zugriffsberechtigung für das zugeordnete Kontextobjekt **AUTH_READER** Zum Bearbeiten des zugeordneten Kontextobjekts ist Leseberechtigung erforderlich (z. B. Lesen der Merkmale einer Prozessinstanz)
DEFINITION_NAME	Zeichenfolge	Der Name der Taskschablonendefinition in der TEL-Datei (TEL = Task Execution Language).
DEFINITION_NS	Zeichenfolge	Der Namespace der Taskschablonendefinition in der TEL-Datei.
IS_AD_HOC	Boolesch	Gibt an, ob diese Taskschablone während der Laufzeit dynamisch erstellt wurde, oder dass die Task als Teil einer EAR-Datei implementiert wurde
IS_INLINE	Boolesch	Gibt an, ob diese Taskschablone als Task in einem Business-Prozess modelliert wird

Tabelle 20. Spalten in der Ansicht TASK_TEMPL (Forts.)

Spaltenname	Typ	Kommentare
KIND	Ganzzahl	Die Art der Tasks, die von dieser Taskschablone abgeleitet werden; gültige Werte sind: **KIND_HUMAN (101)** Gibt an, dass es sich bei der Task um eine *Collaboration-Task* handelt, die von einem Benutzer erstellt und bearbeitet wird **KIND_ORIGINATING (103)** Gibt an, dass es sich bei der Task um eine *Aufruftask* handelt, die Computereingaben von Benutzern unterstützt, d. h. Benutzer können Services erstellen, einleiten und starten **KIND_PARTICIPATING (105)** Gibt an, dass es bei der Task um eine *unerledigte Task* handelt, die Computereingaben von Benutzern unterstützt, d. h. Benutzer können Services implementieren **KIND_ADMINISTRATIVE (106)** Gibt an, dass die Task eine Verwaltungstask ist
NAME	Zeichenfolge	Der Name der Taskschablone
NAMESPACE	Zeichenfolge	Der zum Kategorisieren der Taskschablone verwendete Namensbereich
PRIORITY	Ganzzahl	Die Priorität der Taskschablone
STATE	Ganzzahl	Der Status der Taskschablone; gültige Werte sind: **STATE_STARTED (1)** Gibt an, dass die Taskschablone zum Erstellen von Taskinstanzen zur Verfügung steht **STATE_STOPPED (2)** Gibt an, dass die Taskschablone gestoppt ist (von einer Taskschablone mit diesem Status können keine Taskinstanzen erstellt werden)
SUPPORT_ AUTO-CLAIM	Boolesch	Gibt an, ob die von dieser Taskschablone abgeleiteten Tasks automatisch beansprucht werden können, wenn sie einem einzelnen Benutzer zugeordnet sind
SUPPORT_CLAIM_ SUSP	Boolesch	Gibt an, ob die von dieser Taskschablone abgeleiteten Tasks beansprucht werden können, wenn sie ausgesetzt sind
SUPPORT_ DELEGA-TION	Boolesch	Gibt an, ob die von dieser Taskschablone abgeleiteten Tasks das Delegieren von Arbeit durch Erstellen, Löschen und Übertragen von Arbeitselementen unterstützen
SUPPORT_ FOLLOW_ON	Boolesch	Gibt an, ob die Taskschablone das Erstellen von Folgetasks unterstützt
SUPPORT_SUB_TASK	Boolesch	Gibt an, ob die Taskschablone das Erstellen von Subtasks unterstützt
TYPE	Zeichenfolge	Der zum Kategorisieren der Taskschablone verwendete Typ

Ansicht TASK_TEMPL_CPROP:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen der benutzerdefinierten Merkmale für Taskschablonen.

Tabelle 21. Spalten in der Ansicht TASK_TEMPL_CPROP

Spaltenname	Typ	Kommentare
TKTID	Zeichenfolge	Die ID der Taskschablone
NAME	Zeichenfolge	Der Name des Merkmals
DATA_TYPE	Zeichenfolge	Der Typ der Klasse für benutzerdefinierte Merkmale, die keine Zeichenfolgen sind.
STRING_VALUE	Zeichenfolge	Der Wert für benutzerdefinierte Merkmale des Typs 'Zeichenfolge'

Ansicht TASK_TEMPL_DESC:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen mehrsprachiger beschreibender Daten für Taskschablonenobjekte.

Tabelle 22. Spalten in der Ansicht TASK_TEMPL_DESC

Spaltenname	Typ	Kommentare
TKTID	Zeichenfolge	Die ID der Taskschablone
LOCALE	Zeichenfolge	Der Name der länderspezifischen Angaben, die der Beschreibung oder dem Anzeigenamen zugeordnet sind
DESCRIPTION	Zeichenfolge	Eine Beschreibung der Taskschablone
DISPLAY_NAME	Zeichenfolge	Der beschreibende Name der Taskschablone

Ansicht WORK_ITEM:

Verwenden Sie diese vordefinierte Datenbankansicht zum Abfragen von Arbeitselementen und Berechtigungsdaten für Prozesse, Tasks und Eskalationen.

Tabelle 23. Spalten in der Ansicht WORK_ITEM

Spaltenname	Typ	Kommentare
WIID	ID	Die ID des Arbeitselements
OWNER_ID	Zeichenfolge	Die Teilnehmer-ID des Eigners
GROUP_NAME	Zeichenfolge	Der Name der zugeordneten Gruppenarbeitsliste
EVERYBODY	Boolesch	Gibt an, ob jeder Benutzer der Eigner dieses Arbeitselements ist

Tabelle 23. Spalten in der Ansicht WORK_ITEM (Forts.)

Spaltenname	Typ	Kommentare
OBJECT_TYPE	Ganzzahl	Der Typ des zugeordneten Objekts; gültige Werte sind: **OBJECT_TYPE_ACTIVITY (1)** Gibt an, dass das Arbeitselement für eine Aktivität erstellt wurde **OBJECT_TYPE_PROCESS_INSTANCE (3)** Gibt an, dass das Arbeitselement für eine Prozessinstanz erstellt wurde **OBJECT_TYPE_TASK_INSTANCE (5)** Gibt an, dass das Arbeitselement für eine Task erstellt wurde **OBJECT_TYPE_TASK_TEMPLATE (6)** Gibt an, dass das Arbeitselement für eine Taskschablone erstellt wurde **OBJECT_TYPE_ESCALATION_ INSTANCE (7)** Gibt an, dass das Arbeitselement für eine Eskalationsinstanz erstellt wurde **OBJECT_TYPE_APPLICATION_ COMPONENT (9)** Gibt an, dass das Arbeitselement für eine Anwendungskomponente erstellt wurde
OBJECT_ID	ID	Die ID des zugeordneten Objekts, z. B. der zugeordnete Prozess oder die zugeordnete Task
ASSOC_OBJECT_TYPE	Ganzzahl	Der Typ des von dem Attribut ASSOC_OID referenzierten Objekts, z. B. Task, Prozess oder externes Objekt (verwenden Sie die Werte für das Attribut OBJECT_TYPE)
ASSOC_OID	ID	Die ID des dem Arbeitselement zugeordneten Objekts (z. B. die ID der Prozessinstanz (PIID), in der die Aktivitätsinstanz enthalten ist, für die dieses Arbeitselement erstellt wurde)
REASON	Ganzzahl	Der Grund für die Zuordnung des Arbeitselements; gültige Werte sind: 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)

Tabelle 23. Spalten in der Ansicht WORK_ITEM (Forts.)

Spaltenname	Typ	Kommentare
CREATION_TIME	Zeitmarke	Das Datum und die Uhrzeit der Erstellung des Arbeitselements

Daten mithilfe von Variablen in Abfragen filtern

Im Abfrageergebnis werden die Objekte zurückgegeben, die mit den Abfragekriterien übereinstimmen. Es kann hilfreich sein, die Ergebnismenge anhand von Variablenwerten zu filtern.

Sie können Variablen definieren, die zur Laufzeit im Prozessmodell eines Prozesses verwendet werden. Deklarieren Sie für diese Variablen, welche Teile abgefragt werden dürfen.

Angenommen, Karl Schmidt ruft die Servicenummer seiner Versicherungsgesellschaft an, um sich über den Stand des Versicherungsfalls für sein beschädigtes Auto zu erkundigen. Der Sachbearbeiter findet den Schadensfall anhand der Kunden-ID.

1. Optional: Listen Sie die Merkmale der Variablen in einem Prozess auf, die abgefragt werden können.

   Finden Sie den Prozess anhand der Prozessschablonen-ID. Überspringen Sie diesen Schritt, wenn Sie bereits wissen, welche Variablen abgefragt werden können.

   ```
   List variableProperties = process.getQueryProperties(ptid);
   for (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. Listen Sie die Prozessinstanzen mit Variablen auf, die mit den Filterkriterien übereinstimmen.

   Die Kunden-ID für diesen Prozess wird als Teil der Variablen customerClaim modelliert, die abgefragt werden kann. Sie können den Schadensfall anhand der Kunden-ID finden.

   ```
   QueryResultSet result = process.query
       ("PROCESS_INSTANCE.NAME, QUERY_PROPERTY.STRING_VALUE",
        "QUERY_PROPERTY.VARIABLE_NAME = 'Kundenschadensfall' AND " +
        "QUERY_PROPERTY.NAME = 'Kunden-ID' AND " +
        "QUERY_PROPERTY.STRING_VALUE like 'Schmidt%'",
         (String)null, (Integer)null,
         (Integer)null, (TimeZone)null );
   ```

   Diese Aktion gibt ein Abfrageergebnis zurück, das die Prozessinstanznamen und die Kunden-IDs der Kunden enthält, deren ID mit 'Schmidt' beginnt.

Gespeicherte Abfragen verwalten

Gespeicherte Abfragen bieten eine Möglichkeit zum Aufbewahren häufig verwendeter Abfragen. Die gespeicherte Abfrage kann eine für alle Benutzer verfügbare Abfrage sein (öffentliche Abfrage) oder eine Abfrage, die einem bestimmten Benutzer zugeordnet ist (private Abfrage).

Eine gespeicherte Abfrage ist eine Abfrage, die in der Datenbank gespeichert und mit einem Namen versehen ist. Eine private gespeicherte Abfrage kann denselben

Namen wie eine öffentliche gespeicherte Abfrage haben. Private gespeicherte Abfragen von verschiedenen Eignern können ebenfalls denselben Namen haben.

Sie können Abfragen für Business-Prozessobjekte, Taskobjekte oder für eine Kombination dieser beiden Objekttypen speichern.

Zugehörige Konzepte

„Parameter in gespeicherten Abfragen" auf Seite 33
Eine gespeicherte Abfrage ist eine Abfrage, die in der Datenbank gespeichert und mit einem Namen versehen ist. Die dazugehörigen Tupel werden beim Ausführen der Abfrage dynamisch zusammengesetzt. Um gespeicherte Abfragen wiederverwendbar zu machen, können Sie Parameter in der Abfragedefinition verwenden, die während der Laufzeit aufgelöst werden.

Öffentliche gespeicherte Abfragen verwalten:

Öffentliche gespeicherte Abfragen werden vom Systemadministrator erstellt. Diese Abfragen stehen für alle Benutzer zur Verfügung.

Als Systemadministrator können Sie öffentliche gespeicherte Abfragen erstellen, anzeigen und löschen. Wenn Sie im API-Aufruf keine Benutzer-ID angeben, wird angenommen, dass es sich bei der gespeicherten Abfrage um eine öffentliche gespeicherte Abfrage handelt.

1. Erstellen Sie eine öffentliche gespeicherte Abfrage.

   Beispielsweise erstellt das folgende Codefragment eine gespeicherte Abfrage für Prozessinstanzen und speichert sie mit dem Namen 'Kundenbestellungen mit A beginnend'.

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

   Das Ergebnis der gespeicherten Abfrage ist eine sortierte Liste aller Prozessinstanznamen, die mit dem Buchstaben A beginnen, mit den zugeordneten Prozessinstanz-IDs (PIID).

2. Führen Sie die in der gespeicherten Abfrage definierte Abfrage aus.

   ```
   QueryResultSet result = process.query("Kundenbestellungen mit A beginnend",
                new Integer(0));
   ```

   Diese Aktion gibt die Objekte zurück, die die Bedingungen erfüllen. In diesem Beispiel sind dies alle Kundenbestellungen, die mit A beginnen.

3. Listen Sie die Namen der verfügbaren, öffentlichen gespeicherten Abfragen auf.

   Das folgende Codefragment zeigt, wie die Liste der zurückgegebenen Abfragen auf die öffentlichen Abfragen begrenzt werden kann.

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

4. Optional: Überprüfen Sie die von einer bestimmten gespeicherten Abfrage definierte Abfrage.

   Eine private gespeicherte Abfrage kann denselben Namen haben wie eine öffentliche gespeicherte Abfrage. Wenn die Namen identisch sind, wird die private gespeicherte Abfrage zurückgegeben. Das folgende Codefragment zeigt, wie nur die öffentliche Abfrage mit dem angegebenen Namen zurückgegeben werden kann. Wenn Sie diese Abfrage für taskbezogene Objekte ausführen wollen, geben Sie als zurückzugebenden Objekttyp StoredQuery an und nicht StoredQueryData.

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

5. Löschen Sie eine öffentliche gespeicherte Abfrage.

   Das folgende Codefragment zeigt, wie die in Schritt 1 erstellte gespeicherte Abfrage gelöscht wird.

   ```
   process.deleteStoredQuery("Kundenbestellungen mit A beginnend");
   ```

Private gespeicherte Abfragen für andere Benutzer verwalten:

Jeder Benutzer kann private Abfragen erstellen. Diese Abfragen stehen nur für den Eigner der Abfrage und für den Systemadministrator zur Verfügung.

Als Systemadministrator können Sie private gespeicherte Abfragen verwalten, die einem bestimmten Benutzer gehören.

1. Erstellen Sie eine private gespeicherte Abfrage für die Benutzer-ID Schmidt.

   Beispielsweise erstellt das folgende Codefragment eine gespeicherte Abfrage für Prozessinstanzen und speichert sie mit dem Namen 'Kundenbestellungen mit A beginnend' für die Benutzer-ID Schmidt.

   ```
   process.createStoredQuery("Schmidt", "Kundenbestellungen mit A beginnend",
               "DISTINCT PROCESS_INSTANCE.PIID, PROCESS_INSTANCE.NAME",
               "PROCESS_INSTANCE.NAME LIKE 'A%'",
               "PROCESS_INSTANCE.NAME",
                (Integer)null, (TimeZone)null,
                (List)null, (String)null);
   ```

   Das Ergebnis der gespeicherten Abfrage ist eine sortierte Liste aller Prozessinstanznamen, die mit dem Buchstaben A beginnen, mit den zugeordneten Prozessinstanz-IDs (PIID).

2. Führen Sie die in der gespeicherten Abfrage definierte Abfrage aus.

   ```
   QueryResultSet result = process.query
                         ("Schmidt", "Kundenbestellungen beginnend mit A"),
                           (Integer)null, (Integer)null, (List)null);
   new Integer(0));
   ```

   Diese Aktion gibt die Objekte zurück, die die Bedingungen erfüllen. In diesem Beispiel sind dies alle Kundenbestellungen, die mit A beginnen.

3. Rufen Sie eine Namensliste der privaten Abfragen ab, die einem bestimmten Benutzer gehören.

   Beispielsweise zeigt das folgende Codefragment, wie eine Liste der privaten Abfragen abgerufen wird, die dem Benutzer Schmidt gehören.

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

4. Zeigen Sie die Details einer bestimmten Abfrage an.

   Das folgende Codefragment zeigt, wie die Details der Abfrage 'Kundenbestellungen mit A beginnend' abgerufen werden können, die dem Benutzer Schmidt gehören.

   ```
   StoredQuery storedQuery = process.getStoredQuery
      ("Schmidt", "Kundenbestellungen beginnend mit A");
   String selectClause = storedQuery.getSelectClause();
   String whereClause = storedQuery.getWhereClause();
   String orderByClause = storedQuery.getOrderByClause();
   Integer threshold = storedQuery.getThreshold();
   String owner = storedQuery.getOwner();
   ```

5. Löschen Sie eine private gespeicherte Abfrage.

   Das folgende Codefragment zeigt, wie eine private Abfrage gelöscht wird, die dem Benutzer Schmidt gehört.

   ```
   process.deleteStoredQuery("Schmidt", "Kundenbestellungen mit A beginnend");
   ```

Mit privaten gespeicherten Abfragen arbeiten:

Wenn Sie kein Systemadministrator sind, können Sie eigene private gespeicherte Abfragen erstellen, ausführen und löschen. Außerdem können Sie die vom Systemadministrator erstellten öffentlichen gespeicherten Abfragen verwenden.

1. Erstellen Sie eine private gespeicherte Abfrage.

   Beispielsweise erstellt das folgende Codefragment eine gespeicherte Abfrage für Prozessinstanzen und speichert sie mit einem bestimmten Namen. Wenn keine Benutzer-ID angegeben ist, wird angenommen, dass es sich bei der gespeicherten Abfrage um eine private gespeicherte Abfrage für den angemeldeten Benutzer handelt.

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

   Diese Abfrage gibt eine sortierte Liste aller Namen von Prozessinstanzen, die mit dem Buchstaben A beginnen, mit den zugeordneten Prozessinstanz-IDs (PIID) zurück.

2. Führen Sie die in der gespeicherten Abfrage definierte Abfrage aus.

   ```
   QueryResultSet result = process.query("Kundenbestellungen mit A beginnend",
                new Integer(0));
   ```

   Diese Aktion gibt die Objekte zurück, die die Bedingungen erfüllen. In diesem Beispiel sind dies alle Kundenbestellungen, die mit A beginnen.

3. Rufen Sie eine Namensliste der gespeicherten Abfragen ab, auf die der angemeldete Benutzer zugreifen kann.

   Das folgende Codefragment zeigt, wie die öffentlichen und die privaten gespeicherten Abfragen abgerufen werden können, auf die der Benutzer zugreifen kann.

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

4. Zeigen Sie die Details einer bestimmten Abfrage an.

   Das folgende Codefragment zeigt, wie die Details der Abfrage 'Kundenbestellungen mit A beginnend' abgerufen werden können, deren Eigner der Benutzer Schmidt ist.

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

5. Löschen Sie eine private gespeicherte Abfrage.

   Das folgende Codefragment zeigt, wie eine private gespeicherte Abfrage gelöscht wird.

   ```
   process.deleteStoredQuery("Kundenbestellungen mit A beginnend");
   ```

Anwendungen für Business-Prozesse entwickeln

Ein Business-Prozess ist eine Gruppe geschäftsbezogener Aktivitäten, die in einer bestimmten Reihenfolge aufgerufen werden, um ein Business-Ziel zu erreichen. Die bereitgestellten Beispiele zeigen, wie Sie Anwendungen für typische Aktionen für Prozesse entwickeln können.

Ein Business-Prozess kann ein Mikroprozess oder ein Dauerprozess sein:

- Mikroprozesse sind Business-Prozesse mit kurzer Laufzeit, die synchron ausgeführt werden. Nach kurzer Zeit wird das Ergebnis an den Aufrufenden zurückgegeben.
- Unterbrechbare Prozesse mit langer Laufzeit werden als verkettete Folge von Aktivitäten ausgeführt. Die Verwendung bestimmter Konstrukte in einem Prozess führt zu Unterbrechungen im Prozessablauf (z. B. Aufrufen einer Benutzertask, Aufrufen eines Service mit synchroner Bindung, oder Verwenden zeitgebergesteuerter Aktivitäten).

  In parallelen Prozessverzweigungen wird in der Regel asynchron navigiert, d. h. Aktivitäten in parallelen Verzweigungen werden gleichzeitig ausgeführt. Eine Aktivität kann je nach Typ und Transaktionseinstellung der Aktivität in einer eigenen Transaktion ausgeführt werden.

Erforderliche Aufgabenbereiche der Aktionen für Prozessinstanzen

Die Zugriffsmöglichkeit auf die Schnittstelle BusinessFlowManager bedeutet nicht unbedingt, dass der Aufrufende alle Aktionen an einem Prozess ausführen darf. Der Aufrufende muss außerdem mit einem Aufgabenbereich bei der Clientanwendung angemeldet sein, der zum Ausführen der Aktion berechtigt ist.

Die folgende Tabelle enthält die Aktionen, die von Benutzern mit einem bestimmten Aufgabenbereich für eine Prozessinstanz ausgeführt werden können.

Aktion	**Aufgabenbereich des Aufrufenden**		
	Leser	**Starter**	**Administrator**
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

Aktion	Aufgabenbereich des Aufrufenden		
	Leser	**Starter**	**Administrator**
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

Erforderliche Aufgabenbereiche der Aktionen für Business-Prozess-Aktivitäten

Die Zugriffsmöglichkeit auf die Schnittstelle BusinessFlowManager bedeutet nicht unbedingt, dass der Aufrufende alle Aktionen an einer Aktivität ausführen darf. Der Aufrufende muss außerdem mit einem Aufgabenbereich bei der Clientanwendung angemeldet sein, der zum Ausführen der Aktion berechtigt ist.

Die folgende Tabelle enthält die Aktionen, die von Benutzern mit einem bestimmten Aufgabenbereich für eine Aktivitäteninstanz ausgeführt werden können.

Aktion	Aufgabenbereich des Aufrufenden				
	Leser	**Editor**	**Potenzieller Eigner**	**Eigner**	**Administrator**
cancelClaim				x	x
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

Aktion	Aufgabenbereich des Aufrufenden				
	Leser	**Editor**	**Potenzieller Eigner**	**Eigner**	**Administrator**
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 Nur für potenzielle Eigner oder Administratoren	x

Lebenszyklus eines Business-Prozesses verwalten

Eine Prozessinstanz entsteht, wenn eine Business Process Choreographer-API-Methode aufgerufen wird, die einen Prozess starten kann. Die Verarbeitung der Prozessinstanz wird fortgesetzt bis alle dazugehörigen Aktivitäten einen Endstatus erreicht haben. An der Prozessinstanz können verschiedene Aktionen ausgeführt werden, um ihren Lebenszyklus zu verwalten.

Die bereitgestellten Beispiele zeigen, wie Sie Anwendungen für die folgenden typischen Lebenszyklusaktionen für Prozesse entwickeln können.

Business-Prozesse starten:

Wie ein Business-Prozess gestartet wird, hängt davon ab, ob der Prozess ein Mikroprozess oder ein Dauerprozess ist. Welcher Service den Prozess startet, ist für die Art, wie der Prozess gestartet wird, ebenfalls von Bedeutung. Der Prozess kann entweder über einen eindeutigen Startservice oder über mehrere Startservices verfügen.

Die bereitgestellten Beispiele zeigen, wie Sie Anwendungen für typische Startszenarios für Mikroprozesse und Dauerprozesse entwickeln können.

Mikroprozess mit eindeutigem Startservice ausführen:

Ein Mikroprozess kann durch eine Empfangs- oder Auswahlaktivität gestartet werden. Der Startservice ist eindeutig, wenn der Mikroprozess mit einer Empfangsaktivität beginnt, oder wenn die Auswahlaktivität nur über eine einzige onMessage-Definition verfügt.

Wenn der Mikroprozess eine Operation für Antwortanforderung implementiert, d. h., der Prozess enthält eine Antwort, können Sie den Prozess mit der Aufrufmethode ausführen, wobei der Name der Prozessschablone als Parameter im Aufruf übergeben wird.

Wenn es sich bei dem Mikroprozess um eine unidirektionale Operation handelt, verwenden Sie die Methode sendMessage zum Ausführen des Prozesses. Diese Methode wird in diesem Beispiel nicht dargelegt.

1. Optional: Listen Sie die Prozessschablonen auf, um den Namen des Prozesses zu finden, den Sie ausführen möchten.

   Dieser Schritt ist optional, wenn Sie den Namen des Prozesses bereits kennen.

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

   Die Ergebnisse sind nach dem Namen sortiert. Die Abfrage gibt eine Matrix mit den ersten 50 sortierten Schablonen zurück, die mit der Aufrufmethode gestartet werden können.

2. Starten Sie den Prozess mit einer Eingabenachricht des passenden Typs.

   Beim Erstellen der Nachricht müssen Sie den Nachrichtentypnamen so angeben, dass er die Nachrichtendefinition enthält.

   ```
   ProcessTemplateData template = processTemplates[0];
   // Erstellen einer Nachricht für die einzelne Empfangsstartaktivität
   ClientObjectWrapper input = process.createMessage
                              (template.getID(),
                               template.getInputMessageTypeName());
   DataObject myMessage = null;
   if ( input.getObject()!= null && input.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)input.getObject();
     // Festlegen der Zeichenfolgen in der Nachricht (z. B. ein Kundenname)
     myMessage.setString("Kundenname", "Schmidt");
   }

   // Ausführen des Prozesses
   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("Bestellnr");
   }
   ```

   Diese Aktion erstellt eine Instanz der Prozessschablone 'Kundenschablone' und übergibt die Kundendaten. Die Operation gibt das Ergebnis erst zurück, wenn der Prozess abgeschlossen ist. Das Ergebnis des Prozesses (Bestellnr) wird an den Aufrufenden zurückgegeben.

Mikroprozess mit nicht eindeutigem Startservice ausführen:

Ein Mikroprozess kann durch eine Empfangs- oder Auswahlaktivität gestartet werden. Der Startservice ist nicht eindeutig, wenn der Mikroprozess mit einer Auswahlaktivität beginnt, die über mehrere onMessage-Definitionen verfügt.

Wenn der Mikroprozess eine Operation für Antwortanforderung implementiert, d. h., der Prozess enthält eine Antwort, können Sie den Prozess mit der Aufrufmethode ausführen, wobei die ID des Startservice im Aufruf übergeben wird.

Wenn es sich bei dem Mikroprozess um eine unidirektionale Operation handelt, verwenden Sie die Methode sendMessage zum Ausführen des Prozesses. Diese Methode wird in diesem Beispiel nicht dargelegt.

1. Optional: Listen Sie die Prozessschablonen auf, um den Namen des Prozesses zu finden, den Sie ausführen möchten.

   Dieser Schritt ist optional, wenn Sie den Namen des Prozesses bereits kennen.

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

   Die Ergebnisse sind nach dem Namen sortiert. Die Abfrage gibt eine Matrix mit den ersten 50 sortierten Schablonen zurück, die als Mikroprozesse gestartet werden können.

2. Bestimmen Sie den aufzurufenden Startservice.

   In diesem Beispiel wird die erste gefundene Schablone verwendet.

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

3. Starten Sie den Prozess mit einer Eingabenachricht des passenden Typs.

   Beim Erstellen der Nachricht müssen Sie den Nachrichtentypnamen so angeben, dass er die Nachrichtendefinition enthält.

   ```
   ActivityServiceTemplateData activity = startActivities[0];
   // Erstellen einer Nachricht für den aufzurufenden Service
   ClientObjectWrapper input =
         process.createMessage(activity.getServiceTemplateID(),
                               activity.getActivityTemplateID(),
                               activity.getInputMessageTypeName());
   DataObject myMessage = null;
   if ( input.getObject()!= null && input.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)input.getObject();
     // Festlegen der Zeichenfolgen in der Nachricht (z. B. ein Kundenname)
     myMessage.setString("Kundenname", "Schmidt");
   }
   // Ausführen des Prozesses
   ClientObjectWrapper output = process.call(activity.getServiceTemplateID(),
                                             activity.getActivityTemplateID(),
                                             input);
   // Prüfen der Prozessausgabe (z. B. eine Bestellnummer)
   DataObject myOutput = null;
   if ( output.getObject() != null && output.getObject() instanceof DataObject )
   {
     myOutput  = (DataObject)output.getObject();
     int order = myOutput.getInt("Bestellnr");
   }
   ```

   Diese Aktion erstellt eine Instanz der Prozessschablone 'Kundenschablone' und übergibt die Kundendaten. Die Operation gibt das Ergebnis erst zurück, wenn der Prozess abgeschlossen ist. Das Ergebnis des Prozesses (Bestellnr) wird an den Aufrufenden zurückgegeben.

Starten eines unterbrechbaren Prozesses, der einen eindeutigen Startservice enthält:

Wenn der Startservice eindeutig ist, können Sie die Einleitungsmethode verwenden und den Namen der Prozessschablone als Parameter übergeben. Dies ist der Fall, wenn der Dauerprozess mit einer einzelnen Empfangs- oder Auswahlaktivität startet und die einzelne Auswahlaktivität nur über eine onMessage-Definition verfügt.

1. Optional: Listen Sie die Prozessschablonen auf, um den Namen des Prozesses zu finden, den Sie starten möchten.

   Dieser Schritt ist optional, wenn Sie den Namen des Prozesses bereits kennen.

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

   Die Ergebnisse sind nach dem Namen sortiert. Die Abfrage gibt eine Matrix mit den ersten 50 sortierten Schablonen zurück, die mit der Einleitungsmethode gestartet werden können.

2. Starten Sie den Prozess mit einer Eingabenachricht des passenden Typs.

   Beim Erstellen der Nachricht müssen Sie den Nachrichtentypnamen so angeben, dass er die Nachrichtendefinition enthält. Wenn Sie einen Prozessinstanznamen angeben, darf dieser nicht mit einem Unterstreichungszeichen beginnen. Wird kein Prozessinstanzname angegeben, wird die Prozessinstanz-ID (PIID) im Zeichenfolgeformat als Name verwendet.

   ```
   ProcessTemplateData template = processTemplates[0];
   // Erstellen einer Nachricht für die einzelne Empfangsstartaktivität
   ClientObjectWrapper input = process.createMessage
                               (template.getID(),
                               template.getInputMessageTypeName());
   DataObject myMessage = null;
   if ( input.getObject()!= null && input.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)input.getObject();
     // Festlegen der Zeichenfolgen in der Nachricht (z. B. ein Kundenname)
     myMessage.setString("Kundenname", "Schmidt");
   }
   // Starten des Prozesses
   PIID piid = process.initiate(template.getName(), "Kundenbestellung", input);
   ```

   Diese Aktion erstellt eine Instanz 'Kundenbestellung' und übergibt einige Kundendaten. Beim Starten des Prozesses gibt die Operation die Objekt-ID der neuen Prozessinstanz an den Aufrufenden zurück.

   Als Starter der Prozessinstanz wird der Aufrufende der Anforderung festgelegt. Diese Person erhält ein Arbeitselement für die Prozessinstanz. Die Prozessadministratoren, Leser und Editoren der Prozessinstanz werden ermittelt und erhalten Arbeitselemente für die Prozessinstanz. Die Folgeaktivitätsinstanzen werden ermittelt. Diese werden automatisch gestartet oder, wenn sie Benutzertask-, Empfangs- oder Auswahlaktivitäten sind, werden Arbeitselemente für die möglichen Eigner erstellt.

Starten eines Dauerprozesses, der einen nicht eindeutigen Startservice enthält:

Ein Dauerprozess kann durch mehrere einleitende Empfangs- oder Auswahlaktivitäten gestartet werden. Zum Starten des Prozesses können Sie die Einleitungsmethode verwenden. Wenn der Startservice nicht eindeutig ist, (z. B. wenn der Prozess mit mehreren Empfangs- oder Auswahlaktivitäten startet oder bei einer Auswahlaktivität mit mehreren onMessage-Definitionen) müssen Sie den aufzurufenden Service identifizieren.

1. Optional: Listen Sie die Prozessschablonen auf, um den Namen des Prozesses zu finden, den Sie starten möchten.

   Dieser Schritt ist optional, wenn Sie den Namen des Prozesses bereits kennen.

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

Die Ergebnisse sind nach dem Namen sortiert. Die Abfrage gibt eine Matrix mit den ersten 50 sortierten Schablonen zurück, die als Dauerprozesse gestartet werden können.

2. Bestimmen Sie den aufzurufenden Startservice.

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

3. Starten Sie den Prozess mit einer Eingabenachricht des passenden Typs.

   Beim Erstellen der Nachricht müssen Sie den Nachrichtentypnamen so angeben, dass er die Nachrichtendefinition enthält. Wenn Sie einen Prozessinstanznamen angeben, darf dieser nicht mit einem Unterstreichungszeichen beginnen. Wird kein Prozessinstanzname angegeben, wird die Prozessinstanz-ID (PIID) im Zeichenfolgeformat als Name verwendet.

```
ActivityServiceTemplateData activity = startActivities[0];
// Erstellen einer Nachricht für den aufzurufenden Service
ClientObjectWrapper input = process.createMessage
                           (activity.getServiceTemplateID(),
                            activity.getActivityTemplateID(),
                            activity.getInputMessageTypeName());
DataObject myMessage = null;
if ( input.getObject()!= null && input.getObject() instanceof DataObject )
{
  myMessage = (DataObject)input.getObject();
  // Festlegen der Zeichenfolgen in der Nachricht (z. B. ein Kundenname)
  myMessage.setString("Kundenname", "Schmidt");
}
// Starten des Prozesses
PIID piid = process.sendMessage(activity.getServiceTemplateID(),
                            activity.getActivityTemplateID(),
                            input);
```

   Diese Aktion erstellt eine Instanz und übergibt einige Kundendaten. Beim Starten des Prozesses gibt die Operation die Objekt-ID der neuen Prozessinstanz an den Aufrufenden zurück.

   Als Starter der Prozessinstanz wird der Aufrufende der Anforderung festgelegt. Er erhält ein Arbeitselement für die Prozessinstanz. Die Prozessadministratoren, Leser und Editoren der Prozessinstanz werden ermittelt und erhalten Arbeitselemente für die Prozessinstanz. Die Folgeaktivitätsinstanzen werden ermittelt. Diese werden automatisch gestartet oder, wenn sie Benutzertask-, Empfangs- oder Auswahlaktivitäten sind, werden Arbeitselemente für die möglichen Eigner erstellt.

Business-Prozess aussetzen und wieder aufnehmen:

Eine Dauerprozessinstanz der höchsten Ebene kann während der Ausführung ausgesetzt und später fortgesetzt werden, um sie zu beenden.

Der Aufrufende muss ein Administrator der Prozessinstanz oder ein Business-Prozessadministrator sein. Eine Prozessinstanz kann nur ausgesetzt werden, wenn sie sich in einem aktiven oder fehlgeschlagenen Status befindet.

Beispielsweise kann es erforderlich sein, eine Prozessinstanz auszusetzen, um den Zugriff auf ein Back-End-System zu konfigurieren, das später in dem Prozess verwendet wird. Wenn die Voraussetzungen für den Prozess erfüllt sind, können Sie die Prozessinstanz wieder aufnehmen. Sie können einen Prozess auch aussetzen, um ein Problem zu beheben, das zum Fehlschlagen der Prozessinstanz führt, und den Prozess nach dem Beheben des Problems fortsetzen.

1. Rufen Sie den aktiven Prozess Kundenbestellung ab, den Sie aussetzen möchten.

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

2. Setzen Sie die Prozessinstanz aus.

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

   Diese Aktion setzt die angegebene Prozessinstanz der höchsten Ebene aus. Die Prozessinstanz wird in den Status für ausgesetzt versetzt. Unterprozesse, bei denen das Attribut autonomy auf `child` gesetzt ist, werden ebenfalls ausgesetzt, wenn Sie sich im Status für aktiv, fehlgeschlagen, beendet oder kompensiert befinden. Integrierte Tasks, die dieser Prozessinstanz zugeordnet sind, werden ebenfalls ausgesetzt. Der Prozessinstanz zugeordnete eigenständige Tasks werden jedoch nicht ausgesetzt.

   In diesem Zustand können gestartete Aktivitäten weiterhin beendet werden, aber es werden keine neuen Aktivitäten aktiviert (z. B. kann eine beanspruchte Benutzertaskaktivität beendet werden).

3. Nehmen Sie die Prozessinstanz wieder auf.

   ```
   process.resume( piid );
   ```

   Diese Aktion versetzt die Prozessinstanz und die dazugehörigen Unterprozesse in den Status, den sie vor dem Aussetzen hatten.

Business-Prozess erneut starten:

Sie können eine Prozessinstanz neu starten, die sich im Status 'Fertig gestellt', 'Beendet', 'Fehlgeschlagen' oder 'Kompensiert' befindet.

Der Aufrufende muss ein Administrator der Prozessinstanz oder ein Business-Prozessadministrator sein.

Das erneute Starten einer Prozessinstanz ist vergleichbar mit dem ersten Starten einer Prozessinstanz. Beim erneuten Starten einer Prozessinstanz ist die Prozess-ID jedoch bekannt und die Eingabenachricht für die Instanz ist verfügbar.

Wenn der Prozess über mehr als eine Empfangsaktivität bzw. Auswahlaktivität (auch Empfangsaktivität für Auswahlmöglichkeiten genannt) verfügt, kann dadurch die Prozessinstanz erstellt werden; alle Nachrichten, die zu diesen Aktivitäten gehören, werden für den Neustart der Prozessinstanz verwendet. Wenn beliebig viele dieser Aktivitäten eine Request/Response-Operation implementieren, wird die Antwort noch einmal gesendet, wenn zur zugeordneten Antwortaktivität navigiert wurde.

1. Rufen Sie den Prozess ab, den Sie neu starten möchten.

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

2. Starten Sie den Prozess erneut.

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

   Diese Aktion startet die angegebene Prozessinstanz erneut.

Prozessinstanz beenden:

Manchmal muss eine Person mit Prozessadministratorberechtigung eine Prozessinstanz der höchsten Ebene beenden, die sich in einem nicht wiederherstellbaren Zustand befindet. Da die Beendigung einer Prozessinstanz sofort wirksam wird, ohne noch ausstehende Unterprozesse oder Aktivitäten abzuwarten, sollten Sie nur in Ausnahmefällen eine Prozessinstanz beenden.

1. Rufen Sie die Prozessinstanz ab, die beendet werden soll.

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

2. Beenden Sie die Prozessinstanz.

   Eine Prozessinstanz kann mit oder ohne Kompensation beendet werden.

   Gehen Sie wie folgt vor, um eine Prozessinstanz mit Kompensation zu beenden:

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

   Gehen Sie wie folgt vor, um eine Prozessinstanz ohne Kompensation zu beenden:

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

   Beim Beendigen der Prozessinstanz mit Kompensation wird die Kompensation des Prozesses so ausgeführt, als wenn im oberen Bereich ein Fehler aufgetreten wäre. Wenn Sie die Prozessinstanz ohne Kompensation beenden, erfolgt die Beendigung der Prozessinstanz sofort, ohne den ordnungsgemäßen Abschluss der Aktivitäten, unerledigten Tasks oder integrierten Aufruftasks abzuwarten.

   Auf Anwendungen, die durch den Prozess gestartet wurden, und auf eigenständige Tasks, die zu dem Prozess gehören, hat die Anforderung für eine erzwungene Beendigung keinen Einfluss. Falls diese Anwendungen beendet werden müssen, müssen Sie Anwendungen zur Prozessanwendung hinzufügen, mit denen die vom Prozess gestarteten Anwendungen explizit beendet werden.

Prozessinstanzen löschen:

Abgeschlossene Prozessinstanzen werden automatisch aus der Business Process Choreographer-Datenbank gelöscht, wenn das entsprechende Merkmal für die Prozessschablone in dem Prozessmodell gesetzt ist. Möglicherweise möchten Sie Prozessinstanzen in Ihrer Datenbank aufbewahren, um beispielsweise Daten aus Prozessinstanzen abzufragen, die nicht im Prüfprotokoll erfasst werden. Die gespeicherten Prozessinstanzdaten beeinflussen jedoch nicht nur Plattenspeicherplatz und Leistung. Sie verhindern auch die Erstellung von Prozessinstanzen, die dieselben Korrelationsgruppenwerte verwenden. Darum sollten Sie Prozessinstanzdaten regelmäßig aus der Datenbank löschen.

Wenn Sie eine Prozessinstanz löschen möchten, müssen Sie über Prozessadministratorrechte verfügen, und die Prozessinstanz muss eine Prozessinstanz der höchsten Ebene sein.

Das folgende Beispiel zeigt, wie alle beendeten Prozessinstanzen gelöscht werden können.

1. Listen Sie die beendeten Prozessinstanzen auf.

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

Diese Aktion gibt ein Abfrageergebnis zurück, in dem die beendeten Prozessinstanzen aufgelistet sind.

2. Löschen Sie die beendeten Prozessinstanzen.

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

Diese Aktion löscht die ausgewählte Prozessinstanz mit ihren integrierten Tasks aus der Datenbank.

Benutzertaskaktivitäten verarbeiten

Benutzertaskaktivitäten werden verschiedenen Personen in Ihrem Unternehmen durch Arbeitselemente zugewiesen. Sobald ein Prozess gestartet wird, werden für die potenziellen Eigner Arbeitselemente erstellt.

Bei der Aktivierung einer Benutzertaskaktivität werden sowohl eine Aktivitätsinstanz als auch eine zugeordnete unerledigte Task erstellt. Die Verarbeitung der Benutzertaskaktivität sowie das Management von Arbeitselementen wird an den Human Task Manager delegiert. Jede Änderung des Status der Aktivitätsinstanz wird in der Taskinstanz reflektiert und umgekehrt.

Ein potenzieller Eigner beansprucht die Aktivität. Diese Person ist dafür zuständig, die relevanten Informationen bereitzustellen und die Aktivität abzuschließen.

1. Listen Sie die einer angemeldeten Person zugeordneten Aktivitäten auf, die zum Bearbeiten bereit sind:

```
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);
```

Diese Aktion gibt ein Abfrageergebnis zurück, in dem die Aktivitäten enthalten sind, die von der angemeldeten Person bearbeitet werden können.

2. Beanspruchen Sie die zu bearbeitende Aktivität:

```
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 ClientObjectWrapper input = process.claim(aiid);
 DataObject activityInput = null ;
  if ( input.getObject()!= null && input.getObject() instanceof DataObject )
  {
    activityInput = (DataObject)input.getObject();
    // Lesen der Werte
    ...
  }
}
```

Wenn die Aktivität beansprucht ist, wird die Eingabenachricht der Aktivität zurückgegeben.

3. Beenden Sie die Aktivität, nachdem ihre Bearbeitung abgeschlossen ist. Die Aktivität kann erfolgreich oder mit einer Fehlernachricht beendet werden. Ist die Aktivität erfolgreich beendet, wird eine Ausgabenachricht übergeben. Falls die Aktivität nicht erfolgreich beendet wird, wird sie in den Fehlschlagstatus

oder den Stoppstatus versetzt, und es wird eine Fehlernachricht übergeben. Sie müssen die entsprechenden Nachrichten für diese Aktionen erstellen. Beim Erstellen der Nachricht müssen Sie den Nachrichtentypnamen so angeben, dass er die Nachrichtendefinition enthält.

a. Erstellen Sie eine Ausgabenachricht für die erfolgreiche Beendigung der Aktivität.

```
ActivityInstanceData activity = process.getActivityInstance(aiid);
ClientObjectWrapper output =
      process.createMessage(aiid, activity.getOutputMessageTypeName());
DataObject myMessage = null ;
if ( output.getObject()!= null && output.getObject() instanceof DataObject )
{
  myMessage = (DataObject)output.getObject();
  // Festlegen der Teile in der Nachricht (z. B. eine Bestellnummer)
  myMessage.setInt("Bestellnr", 4711);
}

// Beenden der Task
process.complete(aiid, output);
```

Diese Aktion legt eine Ausgabenachricht fest, die die Bestellnummer enthält.

b. Erstellen Sie eine Fehlernachricht für die Beendigung einer Aktivität nach Auftreten eines Fehlers.

```
// Abrufen der für die Benutzertaskaktivität modellierten Fehler
List faultNames = process.getFaultNames(aiid);

// Erstellen einer Nachricht des gewünschten Typs
ClientObjectWrapper myFault =
      process.createMessage(aiid, faultNames.get(0));

// Festlegen der Teile in der Fehlernachricht (z. B. eine Fehlernummer)
DataObject myMessage = null ;
if ( myFault.getObject()!= null && input.getObject() instanceof DataObject )
{
  myMessage = (DataObject)myFault.getObject();
  // Festlegen der Teile in der Nachricht (z. B. ein Kundenname)
  myMessage.setInt("Fehler",1304);
}

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

Diese Aktion versetzt die Aktivität entweder in den Fehlschlagstatus oder in den Stoppstatus. Falls der Parameter **continueOnError** für die Aktivität im Prozessmodell auf TRUE gesetzt ist, wird die Aktivität in den Fehlschlagstatus versetzt, und die Navigation wird fortgesetzt. Wenn der Parameter **continueOnError** auf 'false' gesetzt ist und der Fehler im umgebenden Bereich nicht abgefangen wird, wird die Aktivität in den Stoppstatus versetzt. In diesem Status kann die Aktivität durch eine erzwungene Beendigung oder eine erzwungene Wiederholung repariert werden.

Workflow für Einzelperson verarbeiten

Manche Workflows werden nur von Einzelpersonen ausgeführt (z. B. das Bestellen von Büchern bei einem Online-Buchversand). Bei diesem Workflowtyp gibt es keine parallelen Pfade. Die API completeAndClaimSuccessor unterstützt die Verarbeitung dieses Workflowtyps.

Auf der Website eines Online-Buchversands führt der Käufer nacheinander eine Reihe von Aktionen aus, um ein Buch zu bestellen. Die Abfolge dieser Aktionen kann als eine Serie von Benutzertaskaktivitäten (unerledigte Tasks) implementiert werden. Wenn der Käufer beschließt, mehrere Bücher zu bestellen, entspricht dies

dem Beanspruchen der nächsten Benutzertaskaktivität. Dieser Workflowtyp wird auch als *Seitenabfolge* bezeichnet, weil den Aktivitäten Benutzerschnittstellendefinitionen zugeordnet sind, um die Abfolge der Dialogfenster in der Benutzerschnittstelle zu steuern.

Die API completeAndClaimSuccessor beendet eine Benutzertaskaktivität und beansprucht die nächste innerhalb derselben Prozessinstanz für den angemeldeten Benutzer. Sie gibt Informationen zur nächsten beanspruchten Aktivität zurück, einschließlich der zu bearbeitenden Eingabenachricht. Da die nächste Aktivität innerhalb derselben Transaktion wie die beendete Aktivität zur Verfügung gestellt wird, muss das Transaktionsverhalten aller Benutzertaskaktivitäten im Prozessmodell auf `participates` (nimmt teil) gesetzt sein.

Vergleichen Sie dieses Beispiel mit dem Beispiel, bei dem sowohl die Business Flow Manager-API als auch die Human Task Manager-API verwendet wird.

1. Beanspruchen Sie die erste Aktivität in der Aktivitätsfolge.

```
//
//Liste der Aktivitäten abrufen, die der angemeldete Benutzer beanspruchen kann
//
QueryResultSet result =
     process.query("ACTIVITY.AIID",
                   "PROCESS_INSTANCE.NAME = 'Kundenbestellung' 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);
...
//
//Erste Aktivität beanspruchen
//
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 ClientObjectWrapper input = process.claim(aiid);
 DataObject activityInput = null ;
  if ( input.getObject()!= null && input.getObject() instanceof DataObject )
  {
    activityInput = (DataObject)input.getObject();
    // Lesen der Werte
    ...
  }
}
```

   Wenn die Aktivität beansprucht ist, wird die Eingabenachricht der Aktivität zurückgegeben.

2. Beenden Sie die Aktivität, nachdem ihre Bearbeitung abgeschlossen ist, und beanspruchen Sie die nächste Aktivität.

   Zum Beenden der Aktivität wird eine Ausgabenachricht übergeben. Beim Erstellen der Ausgabenachricht müssen Sie den Nachrichtentypnamen so angeben, dass er die Nachrichtendefinition enthält.

```
ActivityInstanceData activity = process.getActivityInstance(aiid);
ClientObjectWrapper output =
      process.createMessage(aiid, activity.getOutputMessageTypeName());
DataObject myMessage = null ;
if ( output.getObject()!= null && output.getObject() instanceof DataObject )
{
  myMessage = (DataObject)output.getObject();
  // Festlegen der Teile in der Nachricht (z. B. eine Bestellnummer)
  myMessage.setInt("Bestellnr", 4711);
```

```
}

//Aktivität beenden und nächste beanspruchen
CompleteAndClaimSuccessorResult successor =
   process.completeAndClaimSuccessor(aiid, output);
```

Diese Aktion legt eine Ausgabenachricht fest, die die Bestellnummer enthält, und beansprucht die nächste Aktivität in der Abfolge. Wenn `AutoClaim` für Folgeaktivitäten gesetzt ist, und mehrere mögliche Pfade vorhanden sind, werden alle Folgeaktivitäten beansprucht und eine beliebige Aktivität als nächste Aktivität übergeben. Wenn dem aktuellen Benutzer keine weiteren Folgeaktivitäten zugeordnet werden können, wird `Null` zurückgegeben.

Wenn der Prozess parallele Pfade enthält, die beschritten werden können, und diese Pfade Benutzertaskaktivitäten enthalten, für die der angemeldete Benutzer ein möglicher Eigner ist, wird automatisch eine beliebige Aktivität beansprucht und als nächste Aktivität übergeben.

3. Bearbeiten Sie die nächste Aktivität.

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

ClientObjectWrapper nextInput = successor.getInputMessage();
if ( nextInput.getObject()!=
               null && nextInput.getObject() instanceof DataObject )
{
  activityInput = (DataObject)input.getObject();
  // Lesen der Werte
  ...
}

aiid = successor.getAIID();
```

4. Fahren Sie mit Schritt 2 fort, um die Aktivität abzuschließen.

Zugehörige Tasks

„Workflow für Einzelperson mit Benutzertasks verarbeiten" auf Seite 104
Manche Workflows werden nur von Einzelpersonen ausgeführt (z. B. das Bestellen von Büchern bei einem Online-Buchversand). In diesem Beispiel wird gezeigt, wie die Abfolge von Aktionen für die Buchbestellung als eine Serie von Benutzertaskaktivitäten (unerledigte Tasks) implementiert werden kann. Sowohl die Business Flow Manager- als auch die Human Task Manager-APIs werden für die Verarbeitung des Workflows verwendet.

Nachricht an wartende Aktivität senden

Aktivitäten für ankommende Nachrichten (Empfangsaktivitäten, onMessage in Auswahlaktivitäten, onEvent in Ereignishandlern) können zum Synchronisieren eines aktiven Prozesses mit externen Ereignissen verwendet werden. Ein solches Ereignis kann der Empfang einer E-Mail-Nachricht von einem Kunden als Antwort auf eine Informationsanfrage sein.

Sie können die ursprünglichen Tasks verwenden, um die Nachricht an die Aktivität zu senden.

1. Listen Sie die Serviceschablonen für Aktivitäten, die auf eine Nachricht von dem angemeldeten Benutzer warten, in einer Prozessinstanz mit einer bestimmten Prozessinstanz-ID auf.

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

2. Senden Sie eine Nachricht an den ersten wartenden Service.

Es wird davon ausgegangen, dass der erste Service derjenige ist, den Sie bedienen wollen. Der Aufrufende muss ein potenzieller Starter der Aktivität sein, die die Nachricht empfängt, oder ein Administrator der Prozessinstanz.

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

// Erstellen einer Nachricht für den aufzurufenden Service
  ClientObjectWrapper message = process.createMessage(vtid,atid,inputMessageTypeName);
  DataObject myMessage = null;
  if ( message.getObject()!= null && message.getObject() instanceof DataObject )
  {
    myMessage = (DataObject)message.getObject();
  // Festlegen der Zeichenfolge in der Nachricht
     (es soll z. B. Schokolade bestellt werden)
     myMessage.setString("Schokolade", "bestellen");
  }

  // Senden der Nachricht an die wartende Aktivität
  process.sendMessage(vtid, atid, message);
}
```

Diese Aktion sendet die angegebene Nachricht an den wartenden Aktivitätsservice und übergibt Bestellungsdaten.

Sie können auch die ID der Prozessinstanz angeben, um sicherzustellen, dass die Nachricht an die angegebene Prozessinstanz gesendet wird. Wenn die ID der Prozessinstanz nicht angegeben ist, wird die Nachricht an den Aktivitätsservice gesendet und an die Prozessinstanz, die durch die Korrelationswerte in der Nachricht identifiziert wird. Ist die ID der Prozessinstanz angegeben, wird die anhand der Korrelationswerte gefundene Prozessinstanz überprüft, um sicherzustellen, dass sie die angegebene Prozessinstanz-ID hat.

Ereignishandhabung

Ein vollständiger Business-Prozess und jeder seiner Bereiche kann Ereignissteuerroutinen zugeordnet werden, die aufgerufen werden, wenn das zugeordnete Ereignis auftritt. Eine Gemeinsamkeit zwischen Ereignissteuerroutinen und Empfangs- oder Auswahlaktivitäten besteht darin, dass ein Prozess anhand von Ereignissteuerroutinen Webserviceoperationen bereitstellen kann.

Eine Ereignissteuerroutine kann beliebig oft aufgerufen werden, solange der entsprechende Bereich ausgeführt wird. Zusätzlich können mehrere Instanzen einer Ereignissteuerroutine gleichzeitig aktiviert werden.

Das folgende Codefragment zeigt, wie die aktiven Ereignissteuerroutinen für eine angegebene Prozessinstanz abgerufen werden, und wie eine Eingabenachricht gesendet wird.

1. Ermitteln Sie die Daten für die Prozessinstanz-ID, und listen Sie die aktiven Ereignissteuerroutinen für den Prozess auf.

   ```
   ProcessInstanceData processInstance =
          process.getProcessInstance( "Kundenbestellung2711");
   EventHandlerTemplateData[] events =
          process.getActiveEventHandlers(processInstance.getID() );
   ```

2. Senden Sie die Eingabenachricht.

   In diesem Beispiel wird der erste gefundene Ereignishandler verwendet.

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

       // Erstellen einer Nachricht für den aufzurufenden Service
       ClientObjectWrapper input = process.createMessage(
       event.getID(), event.getInputMessageTypeName());
   ```

```
        if (input.getObject() != null && input.getObject() instanceof DataObject )
        {
           DataObject inputMessage = (DataObject)input.getObject();
           // Festlegen des Nachrichteninhalts (z. B. Kundenname, Bestellnummer)
           inputMessage.setString("Kundenname", "Schmidt");
           inputMessage.setString("Bestellnr", "2711");

            // Senden der Nachricht
           process.sendMessage( event.getProcessTemplateName(),
                                event.getPortTypeNamespace(),
                                event.getPortTypeName(),
                                event.getOperationName(),
         input );
         }
     }
```

Diese Aktion sendet die angegebene Nachricht an die aktive Ereignissteuerroutine für den Prozess.

Prozessergebnisse analysieren

Ein Prozess kann Web-Services-Operationen zugänglich machen, die als unidirektionale WSDL-Operationen (WSDL = Web Services Description Language) oder als Anforderung/Antwort-Operationen modelliert sind. Die Ergebnisse von Dauerprozessen mit unidirektionalen Schnittstellen können nicht mit der Methode getOutputMessage abgerufen werden, da der Prozess keine Ausgabe aufweist. Sie können jedoch stattdessen den Inhalt von Variablen abfragen.

Die Prozessergebnisse werden nur in der Datenbank gespeichert, wenn die Prozessschablone, aus der die Prozessinstanz abgeleitet wurde, nicht das automatische Löschen der abgeleiteten Prozessinstanzen vorsieht.

Analysieren Sie die Prozessergebnisse (z. B. durch Prüfen der Bestellnummer).

```
QueryResultSet result = process.query
                       ("PROCESS_INSTANCE.PIID",
                        "PROCESS_INSTANCE.NAME = 'Kundenbestellung' 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("Bestellnr");
  }
}
```

Aktivitäten reparieren

Ein Dauerprozess kann Daueraktivitäten enthalten. Diese Aktivitäten können auch nicht abgefangene Fehler treffen und in den Stoppstatus gesetzt werden. Es ist auch möglich, dass eine Aktivität im aktiven Status nicht antwortet. In beiden Fällen kann ein Prozessadministrator auf verschiedene Arten mit der Aktivität verfahren, damit die Navigation des Prozesses fortgesetzt werden kann.

Die Business Process Choreographer-API stellt die Methoden forceRetry und forceComplete zum Reparieren von Aktivitäten bereit. Die bereitgestellten Beispiele zeigen, wie Sie Reparaturaktionen für Aktivitäten zu Ihren Anwendungen hinzufügen können.

Beendigung einer Aktivität erzwingen:

Aktivitäten in Dauerprozessen können gelegentlich Fehler aufweisen. Wenn diese Fehler vom Fehlerhandler im übergeordneten Bereich nicht gefunden werden und die zugeordnete Aktivitätsschablone angibt, dass die Aktivität beim Auftreten von Fehlern zu stoppen ist, wird die Aktivität in den Stoppstatus versetzt, damit sie repariert werden kann. In diesem Status können Sie die Beendigung der Aktivität erzwingen.

Sie können auch die Beendigung von Aktivitäten im aktiven Status im erzwingen, wenn eine Aktivität beispielsweise nicht antwortet.

Für bestimmte Aktivitätstypen gelten zusätzliche Voraussetzungen.

Benutzertaskaktivitäten
: In dem Aufruf für erzwungene Beendigung können Sie Parameter übergeben (z. B. eine Nachricht, die gesendet werden sollte, oder den Fehler der gemeldet werden sollte).

Scriptaktivitäten
: In dem Aufruf für erzwungene Beendigung können keine Parameter übergeben werden. Sie müssen jedoch die Variablen festlegen, die repariert werden müssen.

Aufrufaktivitäten
: Sie können auch die Beendigung von Aufrufaktivitäten erzwingen, die einen asynchronen Service aufrufen, der kein Unterprozess ist, wenn sich diese Aktivitäten um Laufstatus befinden. Dies kann beispielsweise hilfreich sein, wenn der aufgerufene asynchrone Service nicht reagiert.

1. Listen Sie die gestoppten Aktivitäten im Stoppstatus auf.

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

   Diese Aktion gibt die gestoppten Aktivitäten für die Prozessinstanz 'Kundenbestellung' zurück.

2. Führen Sie die Aktivität, z. B. eine gestoppte Benutzertaskaktivität, durch.

   In diesem Beispiel wird eine Ausgabenachricht übergeben.

```
if (result.size() > 0)
{
   result.first();
   AIID aiid = (AIID) result.getOID(1);
   ActivityInstanceData activity = process.getActivityInstance(aiid);
  ClientObjectWrapper output =
        process.createMessage(aiid, activity.getOutputMessageTypeName());
  DataObject myMessage = null;
  if ( output.getObject()!= null && output.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)output.getObject();
     // Festlegen der Teile in der Nachricht (z. B. eine Bestellnummer)
     myMessage.setInt("Bestellnr", 4711);
   }

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

Diese Aktion beendet die Aktivität. Bei Auftreten eines Fehlers legt der Parameter **continueOnError** fest, welche Aktion durchgeführt wird, wenn mit der Anforderung forceComplete ein Fehler übergeben wird.

In dem angegebenen Beispiel ist **continueOnError** auf 'true' gesetzt. Dieser Wert bedeutet, dass die Aktivität bei Übergabe eines Fehlers in den Fehlschlagstatus versetzt wird. Der Fehler wird an die übergeordneten Bereiche der Aktivität weitergegeben, bis er behoben oder der Prozessbereich erreicht ist. Anschließend wird der Prozess in den Status für anstehendes Fehlschlagen versetzt, bis er schließlich den Status für fehlgeschlagen erreicht.

Gestoppte Aktivität erneut ausführen:

Wenn in einer Aktivität eines Dauerprozesses ein nicht abgefangener Fehler im übergeordneten Bereich gefunden wird und die zugeordnete Aktivitätsschablone angibt, dass die Aktivität beim Auftreten von Fehlern zu stoppen ist, wird die Aktivität in den Stoppstatus versetzt, damit sie repariert werden kann. Sie können erneut versuchen, die Aktivität auszuführen.

Sie können die von der Aktivität verwendeten Variablen festlegen. Außerdem können Sie (außer bei Scriptaktivitäten) im Aufruf für erzwungene Wiederholung Parameter übergeben (z. B. die von der Aktivität erwartete Nachricht).

1. Listen Sie die gestoppten Aktivitäten auf.

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

   Diese Aktion gibt die gestoppten Aktivitäten für die Prozessinstanz 'Kundenbestellung' zurück.

2. Versuchen Sie erneut, die Aktivität auszuführen, z. B. eine gestoppte Benutzertaskaktivität.

```
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();
      // Legen Sie die Zeichenfolgen in der Nachricht fest
         (es soll z. B. Schokolade bestellt werden)
      myMessage.setString("Bestellnr", "Schokolade");
    }
   boolean continueOnError = true;
   process.forceRetry(aiid, input, continueOnError);
}
```

   Diese Aktion wiederholt die Aktivität. Wenn ein Fehler auftritt, bestimmt der Parameter **continueOnError** die durchzuführende Aktion bei einem Fehler während der Verarbeitung der Anforderung forceRetry.

   In dem angegebenen Beispiel ist **continueOnError** auf 'true' gesetzt. Dies bedeutet, dass die Aktivität beim Auftreten eines Fehlers während der Verarbeitung der forceRetry-Anforderung in den Fehlschlagstatus versetzt wird. Der Fehler wird an die übergeordneten Bereiche der Aktivität weitergegeben, bis er behoben oder der Prozessbereich erreicht ist. Anschließend wird der Prozess

in den Status 'Schlägt fehl' versetzt, und ein Fehlerbehandlungsprogramm auf der Prozessebene wird ausgeführt, bevor der Prozessstatus im Status 'Schlägt fehl' endet.

Schnittstelle BusinessFlowManagerService

Die Schnittstelle BusinessFlowManagerService stellt Funktionen für Business-Prozesse bereit, die von einer Clientanwendung aufgerufen werden können.

Welche Methoden über die Schnittstelle BusinessFlowManagerService aufgerufen werden können, hängt vom Status des Prozesses bzw. der Aktivität ab und von den Berechtigungen der Person, die die Anwendung verwendet, in der die Methoden enthalten sind. Die Hauptmethoden zum Bearbeiten von Business-Prozessobjekten werden nachfolgend aufgelistet. Weitere Informationen zu diesen Methoden und den übrigen Methoden, die in der Schnittstelle BusinessFlowManagerService zur Verfügung stehen, enthält das Javadoc im Paket com.ibm.bpe.api.

Prozessschablonen

Eine Prozessschablone ist ein versionsgesteuertes, implementiertes und installiertes Prozessmodell, das die Spezifikation für einen Business-Prozess enthält. Durch Absetzen entsprechender Anforderungen, z. B. senMessage(), kann eine Instanz der Prozessschablone erstellt und gestartet werden. Die Ausführung der Prozessinstanz wird vom Server automatisch gesteuert.

Tabelle 24. API-Methoden für Prozessschablonen

Methode	Beschreibung
getProcessTemplate	Ruft die angegebene Prozessschablone ab
queryProcessTemplates	Ruft in der Datenbank gespeicherte Prozessschablonen ab

Prozessinstanzen

Die folgenden API-Methoden beziehen sich auf den Start von Prozessinstanzen.

Tabelle 25. API-Methoden beziehen sich auf den Start von Prozessinstanzen

Methode	Beschreibung
call	Erstellt und startet einen Mikroprozess
callWithReplyContext	Erstellt und startet einen Mikroprozess mit einem eindeutigen Startservice oder einen Dauerprozess mit einem angegebenen Startservice aus der angegebenen Prozessschablone. Der Aufruf wartet im asynchronen Wartestatus auf das Ergebnis.
callWithUISettings	Erstellt und startet einen Mikroprozess und gibt die Ausgabenachricht und die Einstellungen der Benutzerschnittstelle (UI) zurück
initiate	Erstellt eine Prozessinstanz und leitet die Verarbeitung der Prozessinstanz ein. Verwenden Sie diese Methode für Dauerprozesse. Sie können diese Methode außerdem für Mikroprozesse einsetzen, die Sie starten und übergehen wollen.

Tabelle 25. API-Methoden beziehen sich auf den Start von Prozessinstanzen (Forts.)

Methode	Beschreibung
sendMessage	Sendet die angegebene Nachricht an den angegebenen Aktivitätsservice und die angegebene Prozessinstanz. Wenn keine Prozessinstanz mit denselben Korrelationsgruppenwerten vorhanden ist, wird sie erstellt. Der Prozess kann entweder eindeutige oder nicht eindeutige Startservices aufweisen.
getStartActivities	Gibt Informationen zu den Aktivitäten zurück, die eine Prozessinstanz aus der angegebenen Prozessschablone starten können
getActivityServiceTemplate	Ruft die angegebenen Schablone für Aktivitätsservice ab

Tabelle 26. API-Methoden zum Steuern des Lebenszyklus von Prozessinstanzen

Methode	Beschreibung
suspend	Stellt das Ausführen einer Prozessinstanz der höchsten Ebene mit langer Laufzeit zurück, die aktiv oder fehlgeschlagen ist
resume	Nimmt das Ausführen einer Prozessinstanz der höchsten Ebene mit langer Laufzeit wieder auf, die zurückgestellt ist
restart	Startet eine Prozessinstanz der höchsten Ebene mit langer Laufzeit, die fertig gestellt, fehlgeschlagen oder beendet ist, neu
forceTerminate	Beendet die angegebene Prozessinstanz der höchsten Ebene sowie die dazugehörigen Unterprozesse mit untergeordneter Autonomie und aktiven, beanspruchten oder anstehenden Aktivitäten
delete	Löscht die angegebene Prozessinstanz der höchsten Ebene und die dazugehörigen Unterprozesse mit untergeordneter Autonomie
query	Ruft die Merkmale aus der Datenbank ab, die mit den Suchbedingungen übereinstimmen

Aktivitäten

Für Aufrufaktivitäten können Sie im Prozessmodell angeben, dass diese Aktivitäten in Fehlersituationen fortgesetzt werden. Ist die Markierung continueOnError auf FALSE gesetzt, wird die Aktivität beim Auftreten eines Fehlers in den Stoppstatus versetzt. Die Aktivität kann anschließend von einem Administrator repariert werden. Die Markierung continueOnError und die dazugehörigen Reparaturfunktionen können beispielsweise in einem Dauerprozess verwendet werden, in dem von Zeit zu Zeit eine Aufruffunktion fehlschlägt, wenn der erforderliche Aufwand für das Einrichten der Modellkompensation und der Fehlerbehandlung zu hoch wäre.

Die folgenden Methoden zum Arbeiten mit und zum Reparieren von Aktivitäten stehen zur Verfügung.

Tabelle 27. API-Methoden zum Steuern des Lebenszyklus von Aktivitätsinstanzen

Methode	Beschreibung
claim	Beansprucht eine betriebsbereite Aktivitätsinstanz für einen Benutzer, der die Aktivität bearbeiten möchte
cancelClaim	Bricht die Beanspruchung einer Aktivitätsinstanz ab
complete	Beendet die Aktivitätsinstanz
completeAndClaimSuccessor	Beendet die Aktivitätsinstanz und beansprucht die nächste innerhalb derselben Prozessinstanz für den angemeldeten Benutzer.
forceComplete	Erzwingt die Beendigung einer Aktivitätsinstanz, die aktiv oder gestoppt ist
forceRetry	Erzwingt die Wiederholung einer Aktivitätsinstanz, die aktiv oder gestoppt ist
query	Ruft die Merkmale aus der Datenbank ab, die mit den Suchbedingungen übereinstimmen

Variablen und benutzerdefinierte Merkmale

Die Schnittstelle stellt eine Methode zum Abrufen und Festlegen sowie festgelegte Werte für Variablen bereit. Außerdem können Sie benannte Merkmale zu Prozess- und Aktivitätsinstanzen zuordnen und daraus abrufen. Namen und Werte benutzerdefinierter Merkmale müssen dem Typ java.lang.String angehören.

Tabelle 28. API-Methoden für Variablen und benutzerdefinierte Merkmale

Methode	Beschreibung
getVariable	Ruft die angegebene Variable ab
setVariable	Legt die angegebene Variable fest
getCustomProperty	Ruft das benannte benutzerdefinierte Merkmal der angegebenen Aktivitäts- oder Prozessinstanz ab
getCustomProperties	Ruft die benutzerdefinierten Merkmale der angegebenen Aktivitäts- oder Prozessinstanz ab.
getCustomPropertyNames	Ruft die Namen der benutzerdefinierten Merkmale für die angegebene Aktivitäts- oder Prozessinstanz ab
setCustomProperty	Speichert benutzerdefinierte Werte für die angegebene Aktivitäts- oder Prozessinstanz

Anwendungen für Benutzertasks entwickeln

Mit einer Task können Komponenten Benutzer als Services aufrufen, oder Benutzer können Services aufrufen. Beispiele für typische Anwendungen für Benutzertasks werden bereitgestellt.

Weitere Informationen zur Human Task Manager-API enthält das Javadoc im Paket com.ibm.task.api.

Aufruftask starten, die eine synchrone Schnittstelle aufruft

Eine Aufruftask wird einer SCA-Komponente (SCA = Service Component Architecture) zugeordnet. Wenn die Task gestartet ist, ruft sie die SCA-Komponente auf. Starten Sie eine Aufruftask nur dann synchron, wenn die zugeordnete SCA-Komponente synchron aufgerufen werden kann.

Eine solche SCA-Komponente kann beispielsweise als Mikroprozess oder als einfache Java-Klasse implementiert werden.

In diesem Szenario wird eine Instanz einer Taskschablone erstellt, und es werden einige Kundendaten übergeben. Die Task bleibt so lange im aktiven Status, bis die bidirektionale Operation zurückgegeben wird. Das Ergebnis der Task (Bestellnr) wird an den Aufrufenden zurückgegeben.

1. Optional: Listen Sie die Taskschablonen auf, um den Namen der Aufruftask zu finden, die Sie ausführen möchten.

   Dieser Schritt ist optional, wenn Sie den Namen der Task bereits kennen.

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

   Die Ergebnisse sind nach dem Namen sortiert. Die Abfrage gibt eine Matrix mit den ersten 50 sortierten ursprünglichen Schablonen zurück.

2. Erstellen Sie eine Eingabenachricht des passenden Typs.

   ```
   TaskTemplate template = taskTemplates[0];

   // Erstellen einer Nachricht für die ausgewählte Task
   ClientObjectWrapper input = task.createInputMessage( template.getID());
   DataObject myMessage = null ;
   if ( input.getObject()!= null && input.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)input.getObject();
     // Festlegen der Teile in der Nachricht (z. B. ein Kundenname)
     myMessage.setString("Kundenname", "Schmidt");
   }
   ```

3. Erstellen Sie die Task, und führen Sie sie synchron aus.

   Für eine Task, die synchron ausgeführt werden soll, ist eine bidirektionale Operation erforderlich. In dem Beispiel wird eine createAndCallTask-Methode zum Erstellen und Ausführen der Task verwendet.

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

4. Analysieren Sie das Ergebnis der Task.

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

Aufruftask starten, die eine asynchrone Schnittstelle aufruft

Eine Aufruftask wird einer SCA-Komponente (SCA = Service Component Architecture) zugeordnet. Wenn die Task gestartet ist, ruft sie die SCA-Komponente auf.

Starten Sie eine Aufruftask nur dann asynchron, wenn die zugeordnete SCA-Komponente asynchron aufgerufen werden kann.

Eine solche SCA-Komponente kann beispielsweise als Dauerprozess oder als unidirektionale Operation implementiert werden.

In diesem Szenario wird eine Instanz einer Taskschablone erstellt, und es werden einige Kundendaten übergeben.

1. Optional: Listen Sie die Taskschablonen auf, um den Namen der Aufruftask zu finden, die Sie ausführen möchten.

   Dieser Schritt ist optional, wenn Sie den Namen der Task bereits kennen.

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

   Die Ergebnisse sind nach dem Namen sortiert. Die Abfrage gibt eine Matrix mit den ersten 50 sortierten ursprünglichen Schablonen zurück.

2. Erstellen Sie eine Eingabenachricht des passenden Typs.

   ```
   TaskTemplate template = taskTemplates[0];

   // Erstellen einer Nachricht für die ausgewählte Task
   ClientObjectWrapper input = task.createInputMessage( template.getID());
   DataObject myMessage = null ;
   if ( input.getObject()!= null && input.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)input.getObject();
     // Festlegen der Teile in der Nachricht (z. B. ein Kundenname)
     myMessage.setString("Kundenname", "Schmidt");
   }
   ```

3. Erstellen Sie eine Task, und führen Sie sie asynchron aus.

   In dem Beispiel wird eine createAndStartTask-Methode zum Erstellen und Ausführen der Task verwendet.

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

Taskinstanz erstellen und starten

Dieses Szenario veranschaulicht, wie Sie eine Instanz einer Taskschablone erstellen und starten können, die eine Collaboration-Task (auch *Benutzertask* genannt) definiert.

1. Optional: Listen Sie die Taskschablonen auf, um den Namen der Collaboration-Task zu finden, die Sie ausführen möchten.

   Dieser Schritt ist optional, wenn Sie den Namen der Task bereits kennen.

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

   Die Ergebnisse sind nach dem Namen sortiert. Die Abfrage gibt eine Matrix mit den ersten 50 sortierten Taskschablonen zurück.

2. Erstellen Sie eine Eingabenachricht des passenden Typs.

   ```
   TaskTemplate template = taskTemplates[0];

   // Erstellen einer Nachricht für die ausgewählte Task
   ClientObjectWrapper input = task.createInputMessage( template.getID());
   ```

```
DataObject myMessage = null ;
if ( input.getObject()!= null && input.getObject() instanceof DataObject )
{
  myMessage = (DataObject)input.getObject();
  // Festlegen der Teile in der Nachricht (z. B. ein Kundenname)
  myMessage.setString("Kundenname", "Schmidt");
}
```

3. Erstellen und starten Sie die Collaboration-Task. Ein Antworthandler ist im vorliegenden Beispiel nicht angegeben.

   Das Beispiel verwendet die Methode createAndStartTask, um die Task zu erstellen und zu starten.

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

   Für die Mitarbeiter, die von der Taskinstanz betroffen sind, werden Arbeitselemente erstellt. Ein potenzieller Eigner kann beispielsweise die neue Taskinstanz beanspruchen.

4. Beanspruchen Sie die Taskinstanz.

   ```
   ClientObjectWrapper input2 = task.claim(tkiid);
   DataObject taskInput = null ;
   if ( input2.getObject()!= null && input2.getObject() instanceof DataObject )
   {
     taskInput = (DataObject)input2.getObject();
     // Lesen der Werte
     ...
   }
   ```

   Wenn die Taskinstanz beansprucht ist, wird die Eingabenachricht der Task zurückgegeben.

Unerledigte Tasks oder Collaboration-Tasks verarbeiten

Unerledigte Tasks (auch *Teilnehmende Tasks* in der API genannt) bzw. Collaboration-Tasks (auch *Benutzertasks* in der API genannt) werden durch Arbeitselemente verschiedenen Personen in Ihrem Unternehmen zugeordnet. Unerledigte Tasks und die zugeordneten Arbeitselemente werden beispielsweise erstellt, wenn ein Prozess eine bestimmte Benutzertaskaktivität aufruft.

Einer der potenziellen Eigner beansprucht die dem Arbeitselement zugeordnete Task. Diese Person ist dafür zuständig, die relevanten Informationen bereitzustellen und die Task abzuschließen.

1. Listen Sie die einer angemeldeten Person zugeordneten Tasks auf, die zum Bearbeiten bereit sind.

   ```
   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);
   ```

   Diese Aktion gibt ein Abfrageergebnis zurück, in dem die Tasks enthalten sind, die von der angemeldeten Person bearbeitet werden können.

2. Beanspruchen Sie die zu bearbeitende Task.

   ```
   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();
    // Lesen der Werte
    ...
  }
}
```

Wenn die Task beansprucht ist, wird die Eingabenachricht der Task zurückgegeben.

3. Beenden Sie die Task, nachdem ihre Bearbeitung abgeschlossen ist.

   Die Task kann erfolgreich oder mit einer Fehlernachricht beendet werden. Ist die Task erfolgreich beendet, wird eine Ausgabenachricht übergeben. Ist die Task nicht erfolgreich, wird eine Fehlernachricht übergeben. Sie müssen die entsprechenden Nachrichten für diese Aktionen erstellen.

   a. Erstellen Sie eine Ausgabenachricht für die erfolgreiche Beendigung der Task.

   ```
   ClientObjectWrapper output =
         task.createOutputMessage(tkiid);
   DataObject myMessage = null ;
   if ( output.getObject()!= null && output.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)output.getObject();
     // Festlegen der Teile in der Nachricht (z. B. eine Bestellnummer)
     myMessage.setInt("Bestellnr", 4711);
   }

   // Beenden der Task
   task.complete(tkiid, output);
   ```

   Diese Aktion legt eine Ausgabenachricht fest, die die Bestellnummer enthält. Die Task wird in den Status für Fertigstellung versetzt.

   b. Erstellen Sie eine Fehlernachricht für die Beendigung einer Task nach Auftreten eines Fehlers.

   ```
   // Abrufen der für die Task modellierten Fehler
   List faultNames = task.getFaultNames(tkiid);

   // Erstellen einer Nachricht des gewünschten Typs
   ClientObjectWrapper myFault =
         task.createFaultMessage(tkiid, (String)faultNames.get(0));

   // Festlegen der Teile in der Fehlernachricht (z. B. eine Fehlernummer)
   DataObject myMessage = null ;
   if ( myFault.getObject()!= null && input.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)myFault.getObject();
     // Festlegen der Teile in der Nachricht (z. B. ein Kundenname)
     myMessage.setInt("Fehler",1304);
   }

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

   Diese Aktion definiert eine Fehlernachricht, die den Fehlercode enthält. Die Task wird in den Fehlschlagstatus versetzt.

Taskinstanz aussetzen und wieder aufnehmen

Sie können Instanzen von Collaboration-Tasks (auch *Benutzertasks* in der API genannt) oder Instanzen von unerledigten Tasks (auch *Teilnehmende Tasks* in der API genannt) aussetzen.

Die Taskinstanz kann den Status ″Bereit″ oder ″Beansprucht″ aufweisen bzw. eskaliert sein. Der Aufrufende muss Eigner, Ersteller oder Administrator der Taskinstanz sein.

Sie können eine Taskinstanz aussetzen, während sie ausgeführt wird. Dies kann beispielsweise erforderlich sein, um Informationen zusammenzustellen, die Sie zum Abschluss der Task benötigen. Sobald die Informationen zur Verfügung stehen, können Sie die Task wieder aufnehmen.

1. Rufen Sie eine Liste der Tasks ab, die vom angemeldeten Benutzer beansprucht wurden.

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

   Diese Aktion gibt eine Abfrageergebnisliste zurück, in der eine Liste der Tasks enthalten ist, die vom angemeldeten Benutzer beansprucht wurden.

2. Setzen Sie die Taskinstanz aus.

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

   Diese Aktion setzt die angegebene Taskinstanz aus. Die Taskinstanz wird in den Status ″Ausgesetzt″ versetzt.

3. Nehmen Sie die Prozessinstanz wieder auf.

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

   Diese Aktion versetzt die Taskinstanz in den Status, den sie vor dem Aussetzen innehatte.

Taskergebnisse analysieren

Eine unerledigte Task (auch *Teilnehmende Task* in der API genannt) oder eine Collaboration-Task (auch *Benutzertask* in der API genannt) wird asynchron ausgeführt. Wenn ein Antworthandler beim Start der Task angegeben ist, wird die Ausgabenachricht automatisch zurückgegeben, sobald die Task durchgeführt wurde. Wenn der Antworthandler nicht angegeben ist, muss die Nachricht explizit abgerufen werden.

Die Taskergebnisse werden nur in der Datenbank gespeichert, wenn die Taskschablone, aus der die Taskinstanz abgeleitet wurde, nicht das automatische Löschen der abgeleiteten Taskinstanzen vorsieht.

Analysieren Sie die Ergebnisse der Task.

In dem Beispiel sehen Sie die Vorgehensweise zum Überprüfen der Bestellnummer einer erfolgreich durchgeführten Task.

```
QueryResultSet result = task.query("DISTINCT TASK.TKIID",
                                   "TASK.NAME = 'Kundenbestellung' 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("Bestellnr");
  }
}
```

Taskinstanz beenden

Manchmal muss eine Person mit Administratorberechtigungen eine Taskinstanz beenden, die sich in einem nicht wiederherstellbaren Zustand befindet. Da die Taskinstanz unverzüglich beendet wird, sollten Sie eine Taskinstanz nur in Ausnahmefällen beenden.

1. Rufen Sie die Taskinstanz ab, die beendet werden soll.

   ```
   Task taskInstance = task.getTask(tkiid);
   ```

2. Beenden Sie die Taskinstanz.

   ```
   TKIID tkiid = taskInstance.getID();
   task.terminate(tkiid);
   ```

   Die Taskinstanz wird sofort beendet, ohne auf anstehende Tasks zu warten.

Taskinstanzen löschen

Taskinstanzen werden beim Beenden nur automatisch gelöscht, wenn dies in der zugeordneten Taskschablone, aus der die Instanzen abgeleitet wurden, angegeben ist. Das folgende Beispiel zeigt, wie alle beendeten Taskinstanzen gelöscht werden können, die nicht automatisch gelöscht werden.

1. Listen Sie die fertig gestellten Taskinstanzen auf.

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

   Diese Aktion gibt ein Abfrageergebnis zurück, in dem die beendeten Taskinstanzen aufgelistet sind.

2. Löschen Sie die beendeten Taskinstanzen.

   ```
   while (result.next() )
   {
    TKIID tkiid = (TKIID) result.getOID(1);
    task.delete(tkiid);
   }
   ```

Beanspruchte Task freigeben

Wenn ein potenzieller Eigner eine Task beansprucht, ist diese Person dafür zuständig, die Task abzuschließen. Manchmal muss die beanspruchte Task jedoch freigegeben werden, damit sie von einem anderen potenziellen Eigner beansprucht werden kann.

Manchmal muss eine Person mit Administratorberechtigungen eine beanspruchte Task freigeben. Dies kann beispielsweise erforderlich sein, wenn eine Task abgeschlossen werden muss, deren Eigner abwesend ist. Der Eigner der Task kann auch eine beanspruchte Task freigeben.

1. Listen Sie die beanspruchten Tasks auf, deren Eigner eine bestimmte Person (z. B. Schmidt) ist.

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

Diese Aktion gibt ein Abfrageergebnis zurück, in dem die von der angegebenen Person (Schmidt) beanspruchten Tasks aufgelistet sind.

2. Geben Sie die beanspruchte Task frei.

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

Diese Aktion setzt die Task in den Bereitschaftsstatus zurück, d. h. die Task kann nun von einem anderen potenziellen Eigner beansprucht werden. Alle vom ursprünglichen Eigner festgelegten Ausgabe- oder Fehlerdaten werden beibehalten.

Arbeitselemente verwalten

Während der Laufzeit einer Aktivitätsinstanz oder Taskinstanz können sich die dem Objekt zugeordneten Personen ändern (z. B. weil eine Person in Urlaub geht, neue Mitarbeiter eingestellt werden oder die Arbeit anders verteilt werden muss). Um solche Änderungen zu berücksichtigen, können Sie Anwendungen zum Erstellen, Löschen oder Übertragen von Arbeitselementen entwickeln.

Ein Arbeitselement repräsentiert die Zuordnung eines Objekts zu einem Benutzer oder einer Benutzergruppe aus einem bestimmten Grund. Das Objekt ist in der Regel eine Benutzertaskaktivitätsinstanz, eine Prozessinstanz oder eine Taskinstanz. Die Gründe werden aus dem Aufgabenbereich abgeleitet, den der Benutzer für das Objekt hat. Ein Objekt kann mehrere Arbeitselemente enthalten, weil ein Benutzer verschiedene Aufgabenbereiche in Bezug auf das Objekt haben kann. Für jeden dieser Aufgabenbereiche wird ein eigenes Arbeitselement erstellt. So kann beispielsweise eine unerledigte Taskinstanz gleichzeitig über ein Arbeitselement für Administrator, Leser, Editor und Eigner verfügen.

Die Aktionen, die zur Verwaltung von Arbeitselementen durchgeführt werden können, sind vom Aufgabenbereich des Benutzers abhängig; beispielsweise kann ein Administrator Arbeitselemente erstellen, löschen und übertragen, aber der Taskeigner kann lediglich Arbeitselemente übertragen.

- Erstellen Sie ein Arbeitselement.

```
// Abfragen der Taskinstanz, für die ein weiterer
// Administrator angegeben werden soll
QueryResultSet result = task.query("TASK.TKIID",
                                   "TASK.NAME='Kundenbestellung'",
                                    (String)null, (Integer)null,
                                    (TimeZone)null);
if ( result.size() > 0 )
{
  result.first();
  // Erstellen des Arbeitselements
  task.createWorkItem((TKIID)(result.getOID(1)),
                          WorkItem.REASON_ADMINISTRATOR,"Schmidt");
}
```

Diese Aktion erstellt ein Arbeitselement für den Benutzer Schmidt, der den Administratoraufgabenbereich hat.

- Löschen Sie ein Arbeitselement.

```
// Abfragen der Taskinstanz, für die ein Arbeitselement gelöscht werden soll
QueryResultSet result = task.query("TASK.TKIID",
                                   "TASK.NAME='Kundenbestellung'",
                                    (String)null, (Integer)null,
                                    (TimeZone)null);
if ( result.size() > 0 )
```

```
{
  result.first();
  // Löschen des Arbeitselements
  task.deleteWorkItem((TKIID)(result.getOID(1)),
                          WorkItem.REASON_READER,"Schmidt");
}
```

Diese Aktion löscht das Arbeitselement für den Benutzer Schmidt, der den Aufgabenbereich 'Leser' hat.

- Übertragen Sie ein Arbeitselement.

```
// Abfragen der Task, die neu geplant werden soll
QueryResultSet result =
     task.query("DISTINCT TASK.TKIID",
                "TASK.NAME='Kundenbestellung' AND
                 TASK.STATE=TASK.STATE.STATE_READY AND
                 WORK_ITEM.REASON=WORK_ITEM.REASON.REASON_POTENTIAL_OWNER AND
                 WORK_ITEM.OWNER_ID='Müller'",
                 (String)null, (Integer)null, (TimeZone)null);
if ( result.size() > 0 )
{
  result.first();
  // Übertragen des Arbeitselements von Benutzer Müller auf Benutzer Schmidt,
  // damit Schmidt an der Task arbeiten kann
  task.transferWorkItem((TKIID)(result.getOID(1)),
                          WorkItem.REASON_POTENTIAL_OWNER,"Müller","Schmidt");
}
```

Diese Aktion überträgt das Arbeitselement auf den Benutzer Schmidt, damit er daran arbeiten kann.

Taskschablonen und Taskinstanzen zur Laufzeit erstellen

In der Regel verwenden Sie zum Erstellen von Taskschablonen ein Modellierungstool wie WebSphere Integration Developer. Sie installieren anschließend die Taskschablonen in WebSphere Process Server und erstellen Instanzen auf der Basis dieser Schablonen, z. B. mit Business Process Choreographer Explorer. Allerdings können Sie auch Instanzen oder Schablonen von Benutzertasks oder teilnehmenden Tasks zur Laufzeit erstellen.

Dies können Sie beispielsweise dann tun, wenn die Taskdefinition bei der Implementierung der Anwendung nicht verfügbar ist, die Tasks, die Teil eines Workflows sind, noch nicht bekannt sind, oder Sie eine Task benötigen, um Ad-hoc-Collaboration zwischen einer Gruppe von Personen abzudecken.

Sie können unerledigte Ad-hoc-Tasks oder Ad-hoc-Collaboration-Tasks durch die Erstellung von Instanzen der Klasse com.ibm.task.api.TaskModel und durch deren Verwendung zur Erstellung einer wiederverwendbaren Taskschablone oder zur direkten Erstellung einer einmal auszuführenden Taskinstanz modellieren. Zur Erstellung einer Instanz der Klasse TaskModel steht eine Reihe von Factorymethoden in der Factoryklasse com.ibm.task.api.ClientTaskFactory zur Verfügung. Das Modellieren von Benutzertasks zur Laufzeit basiert auf dem Eclipse Modeling Framework (EMF).

1. Erstellen Sie mithilfe der Factorymethode createResourceSet ein org.eclipse.emf.ecore.resource.ResourceSet.
2. Optional: Wenn Sie komplexe Nachrichtentypen verwenden möchten, können Sie sie entweder mit der org.eclipse.xsd.XSDFactory definieren, die Sie mit der Factorymethode getXSDFactory() abrufen können, oder Sie können sie unter Verwendung der Factorymethode loadXSDSchema direkt in ein vorhandenes XML-Schema importieren.

Um die komplexen Typen dem WebSphere Process Server verfügbar zu machen, müssen Sie sie im Rahmen einer Unternehmensanwendung implementieren.

3. Erstellen Sie oder importieren Sie eine WSDL-Definition vom Typ javax.wsdl.Definition.

   Mithilfe der Methode createWSDLDefinition können Sie eine neue WSDL-Definition erstellen. Anschließend können Sie ihr einen Porttyp und eine Operation hinzufügen. Sie können mithilfe der Factorymethode loadWSDLDefinition auch direkt eine vorhandene WSDL-Definition importieren.

4. Erstellen Sie die Taskdefinition mithilfe der Factorymethode createTTask.

   Wenn Sie mehr komplexe Taskelemente hinzufügen oder bearbeiten möchten, können Sie die Klasse com.ibm.wbit.tel.TaskFactory verwenden, die mit der Factorymethode getTaskFactory abgerufen werden kann.

5. Erstellen Sie das Taskmodell mithilfe der Factorymethode createTaskModel, und übergeben Sie ihm das Ressourcenpaket, das Sie in Schritt 1 erstellt haben und das alle weiteren Artefakte zusammenfasst, die Sie in der Zwischenzeit erstellt haben.

6. Optional: Prüfen Sie das Modell mithilfe der TaskModel-Methode validate.

Verwenden Sie eine der Human Task Manager-EJB-API-Erstellungsmethoden (create) mit einem Parameter **TaskModel**, um entweder eine wiederverwendbare Taskschablone oder eine einmal auszuführende Taskinstanz zu erstellen.

Laufzeittasks erstellen, die einfache Java-Typen verwenden:

In diesem Beispiel wird eine Laufzeittask erstellt, die in ihrer Schnittstelle nur einfache Java-Typen, z. B. ein Zeichenfolgeobjekt, verwendet.

Das Beispiel funktioniert nur im Kontext der aufrufenden Enterprise-Anwendung, für die die Ressourcen geladen werden.

1. Greifen Sie auf die ClientTaskFactory zu, und erstellen Sie eine Ressourcengruppe, die die Definitionen des neuen Taskmodells enthalten soll.

```
ClientTaskFactory factory = ClientTaskFactory.newInstance();
ResourceSet resourceSet = factory.createResourceSet();
```

2. Erstellen Sie die WSDL-Definition, und fügen Sie die Beschreibungen Ihrer Operationen hinzu.

```
// Erstellen der WSDL-Schnittstelle
Definition definition = factory.createWSDLDefinition
         ( resourceSet, new QName( "http://www.ibm.com/task/test/", "test" ) );

// Erstellen eines Porttyps
PortType portType = factory.createPortType( definition, "doItPT" );

// Erstellen einer Operation; die Eingabe- und Ausgabenachrichten sind vom Typ 'String':
// Eine Fehlernachricht wird nicht angegeben
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. Erstellen Sie das EMF-Modell Ihrer neuen Benutzertask.

   Wenn Sie eine Taskinstanz erstellen, ist kein Gültigkeitsstartdatum (UTCDate) erforderlich.

```
TTask humanTask = factory.createTTask( resourceSet,
                                       TTaskKinds.HTASK_LITERAL,
                                       "TestTask",
```

```
new UTCDate( "2005-01-01T00:00:00" ),
"http://www.ibm.com/task/test/",
portType,
operation );
```

Durch diesen Schritt werden die Merkmale des neuen Taskmodells mit den Standardwerten initialisiert.

4. Ändern Sie die Merkmale Ihres Benutzertaskmodells.

```
// Verwenden der Methoden aus dem Paket com.ibm.wbit.tel, z. B.:
humanTask.setBusinessRelevance( TBoolean, YES_LITERAL );

// Abrufen der Task-Factory zum Erstellen oder Ändern von zusammengesetzten Taskelementen
TaskFactory taskFactory = factory.getTaskFactory();

// Angeben der Eskalationseinstellungen
TVerb verb = taskFactory.createTVerb();
verb.setName("Karl");

// Erstellen des Eskalationsempfängers und Hinzufügen eines Verbs
TEscalationReceiver escalationReceiver =
                    taskFactory.createTEscalationReceiver();
escalationReceiver.setVerb(verb);

// Erstellen einer Eskalation und Hinzufügen eines Eskalationsempfängers
TEscalation escalation = taskFactory.createTEscalation();
escalation.setEscalationReceiver(escalationReceiver);
```

5. Erstellen Sie das Taskmodell, das alle Ressourcendefinitionen enthält.

```
TaskModel taskModel = ClientTaskFactory.createTaskModel( resourceSet );
```

6. Prüfen Sie das Taskmodell, und korrigieren Sie sämtliche gefundenen Prüffehler.

```
ValidationProblem[] validationProblems = taskModel.validate();
```

7. Erstellen Sie die Laufzeit-Taskinstanz bzw. die Laufzeitschablone.

   Verwenden Sie die Schnittstelle HumanTaskManagerService, um die Taskinstanz oder die Taskschablone zu erstellen. Da die Anwendung nur einfache Java-Typen verwendet, müssen Sie keinen Anwendungsnamen angeben.

   - Durch das folgende Snippet wird eine Taskinstanz erstellt:

     ```
     atask.createTask( taskModel, (String)null, "HTM" );
     ```

   - Durch das folgende Snippet wird eine Taskschablone erstellt:

     ```
     task.createTaskTemplate( taskModel, (String)null );
     ```

Wenn eine Laufzeit-Taskinstanz erstellt wurde, kann sie nun gestartet werden. Wenn eine Laufzeit-Taskschablone erstellt wurde, können Sie nun auf der Basis der Schablone Taskinstanzen erstellen.

Laufzeittasks erstellen, die komplexe Typen verwenden:

In diesem Beispiel wird eine Laufzeittask erstellt, die in ihrer Schnittstelle komplexe Typen verwendet. Die komplexen Typen sind bereits definiert, d. h., das lokale Dateisystem auf dem Client verfügt über XSD-Dateien, die die Beschreibung der komplexen Typen enthalten.

Das Beispiel funktioniert nur im Kontext der aufrufenden Enterprise-Anwendung, für die die Ressourcen geladen werden.

1. Greifen Sie auf die ClientTaskFactory zu, und erstellen Sie eine Ressourcengruppe, die die Definitionen des neuen Taskmodells enthalten soll.

```
ClientTaskFactory factory = ClientTaskFactory.newInstance();
ResourceSet resourceSet = factory.createResourceSet();
```

2. Fügen Sie die XSD-Definitionen Ihrer komplexen Typen zur Ressourcengruppe hinzu, sodass sie bei der Definition Ihrer Operationen verfügbar sind.

   Die Dateien befinden sich in relativer Position zu der Position, an der der Code ausgeführt wird.

   ```
   factory.loadXSDSchema( resourceSet, "InputBO.xsd" );
   factory.loadXSDSchema( resourceSet, "OutputBO.xsd" );
   ```

3. Erstellen Sie die WSDL-Definition, und fügen Sie die Beschreibungen Ihrer Operationen hinzu.

   ```
   // Erstellen der WSDL-Schnittstelle
   Definition definition = factory.createWSDLDefinition
            ( resourceSet, new QName( "http://www.ibm.com/task/test/", "test" ) );

   // Erstellen eines Porttyps
   PortType portType = factory.createPortType( definition, "doItPT" );

   // Erstellen einer Operation; die Eingabenachricht ist InputBO, und
   // die Ausgabenachricht ist OutputBO;
   // Eine Fehlernachricht wird nicht angegeben
   Operation operation = factory.createOperation
           ( definition, portType, "doIt",
             new QName( "http://Input", "InputBO" ),
             new QName( "http://Output", "OutputBO" ),
             (Map)null );
   ```

4. Erstellen Sie das EMF-Modell Ihrer neuen Benutzertask.

   Wenn Sie eine Taskinstanz erstellen, ist kein Gültigkeitsstartdatum (UTCDate) erforderlich.

   ```
   TTask humanTask = factory.createTTask( resourceSet,
                                          TTaskKinds.HTASK_LITERAL,
                                          "TestTask",
                                          new UTCDate( "2005-01-01T00:00:00" ),
                                          "http://www.ibm.com/task/test/",
                                          portType,
                                          operation );
   ```

   Durch diesen Schritt werden die Merkmale des neuen Taskmodells mit den Standardwerten initialisiert.

5. Ändern Sie die Merkmale Ihres Benutzertaskmodells.

   ```
   // Verwenden der Methoden aus dem Paket com.ibm.wbit.tel, z. B.:
   humanTask.setBusinessRelevance( TBoolean, YES_LITERAL );

   // Abrufen der Task-Factory zum Erstellen oder Ändern von zusammengesetzten Taskelementen
   TaskFactory taskFactory = factory.getTaskFactory();

   // Angeben der Eskalationseinstellungen
   TVerb verb = taskFactory.createTVerb();
   verb.setName("Karl");

   // Erstellen des Eskalationsempfängers und Hinzufügen eines Verbs
   TEscalationReceiver escalationReceiver =
                       taskFactory.createTEscalationReceiver();
   escalationReceiver.setVerb(verb);

   // Erstellen einer Eskalation und Hinzufügen eines Eskalationsempfängers
   TEscalation escalation = taskFactory.createTEscalation();
   escalation.setEscalationReceiver(escalationReceiver);
   ```

6. Erstellen Sie das Taskmodell, das alle Ressourcendefinitionen enthält.

   ```
   TaskModel taskModel = ClientTaskFactory.createTaskModel( resourceSet );
   ```

7. Prüfen Sie das Taskmodell, und korrigieren Sie sämtliche gefundenen Prüffehler.

   ```
   ValidationProblem[] validationProblems = taskModel.validate();
   ```

8. Erstellen Sie die Laufzeit-Taskinstanz bzw. die Laufzeitschablone.

   Verwenden Sie die Schnittstelle HumanTaskManagerService, um die Taskinstanz oder die Taskschablone zu erstellen. Sie müssen einen Anwendungsnamen angeben, der die Datentypdefinitionen enthält, damit ein Zugriff möglich ist. Die Anwendung muss auch eine Mustertask oder einen Musterprozess enthalten, sodass die Anwendung von Business Process Choreographer geladen wird.

   - Durch das folgende Snippet wird eine Taskinstanz erstellt:

     ```
     task.createTask( taskModel, "BOapplication", "HTM" );
     ```

   - Durch das folgende Snippet wird eine Taskschablone erstellt:

     ```
     task.createTaskTemplate( taskModel, "BOapplication" );
     ```

Wenn eine Laufzeit-Taskinstanz erstellt wurde, kann sie nun gestartet werden. Wenn eine Laufzeittaskschablone erstellt wurde, können Sie nun auf der Basis der Schablone Taskinstanzen erstellen.

Laufzeittasks erstellen, die eine vorhandene Schnittstelle verwenden:

Bei diesem Beispiel wird eine Laufzeittask erstellt, die eine Schnittstelle verwendet, die bereits definiert ist, d. h., das lokale Dateisystem auf dem Client verfügt über eine Datei, die die Beschreibung der Schnittstelle enthält.

Das Beispiel funktioniert nur im Kontext der aufrufenden Enterprise-Anwendung, für die die Ressourcen geladen werden.

1. Greifen Sie auf die ClientTaskFactory zu, und erstellen Sie eine Ressourcengruppe, die die Definitionen des neuen Taskmodells enthalten soll.

   ```
   ClientTaskFactory factory = ClientTaskFactory.newInstance();
   ResourceSet resourceSet = factory.createResourceSet();
   ```

2. Greifen Sie auf die WSDL-Definition und die Beschreibungen Ihrer Operationen zu.

   Die Schnittstellenbeschreibung befindet sich in relativer Position zu der Position, an der der Code ausgeführt wird.

   ```
   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. Erstellen Sie das EMF-Modell Ihrer neuen Benutzertask.

   Wenn Sie eine Taskinstanz erstellen, ist kein Gültigkeitsstartdatum (UTCDate) erforderlich.

   ```
   TTask humanTask = factory.createTTask( resourceSet,
                                          TTaskKinds.HTASK_LITERAL,
                                          "TestTask",
                                          new UTCDate( "2005-01-01T00:00:00" ),
                                          "http://www.ibm.com/task/test/",
                                          portType,
                                          operation );
   ```

   Durch diesen Schritt werden die Merkmale des neuen Taskmodells mit den Standardwerten initialisiert.

4. Ändern Sie die Merkmale Ihres Benutzertaskmodells.

   ```
   // Verwenden der Methoden aus dem Paket com.ibm.wbit.tel, z. B.:
   humanTask.setBusinessRelevance( TBoolean, YES_LITERAL );

   // Abrufen der Task-Factory zum Erstellen oder Ändern von zusammengesetzten Taskelementen
   TaskFactory taskFactory = factory.getTaskFactory();
   ```

```
// Angeben der Eskalationseinstellungen
TVerb verb = taskFactory.createTVerb();
verb.setName("Karl");

// Erstellen des Eskalationsempfängers und Hinzufügen eines Verbs
TEscalationReceiver escalationReceiver =
                    taskFactory.createTEscalationReceiver();
escalationReceiver.setVerb(verb);

// Erstellen einer Eskalation und Hinzufügen eines Eskalationsempfängers
TEscalation escalation = taskFactory.createTEscalation();
escalation.setEscalationReceiver(escalationReceiver);
```

5. Erstellen Sie das Taskmodell, das alle Ressourcendefinitionen enthält.

   ```
   TaskModel taskModel = ClientTaskFactory.createTaskModel( resourceSet );
   ```

6. Prüfen Sie das Taskmodell, und korrigieren Sie sämtliche gefundenen Prüffehler.

   ```
   ValidationProblem[] validationProblems = taskModel.validate();
   ```

7. Erstellen Sie die Laufzeit-Taskinstanz bzw. die Laufzeitschablone.

   Verwenden Sie die Schnittstelle HumanTaskManagerService, um die Taskinstanz oder die Taskschablone zu erstellen. Sie müssen einen Anwendungsnamen angeben, der die Datentypdefinitionen enthält, damit ein Zugriff möglich ist. Die Anwendung muss auch eine Mustertask oder einen Musterprozess enthalten, sodass die Anwendung von Business Process Choreographer geladen wird.

   - Durch das folgende Snippet wird eine Taskinstanz erstellt:

     ```
     task.createTask( taskModel, "BOapplication", "HTM" );
     ```

   - Durch das folgende Snippet wird eine Taskschablone erstellt:

     ```
     task.createTaskTemplate( taskModel, "BOapplication" );
     ```

Wenn eine Laufzeit-Taskinstanz erstellt wurde, kann sie nun gestartet werden. Wenn eine Laufzeittaskschablone erstellt wurde, können Sie nun auf der Basis der Schablone Taskinstanzen erstellen.

Laufzeittasks erstellen, die eine Schnittstelle der aufrufenden Anwendung verwenden:

Bei diesem Beispiel wird eine Laufzeittask erstellt, die eine Schnittstelle verwendet, die ein Teil der aufrufenden Anwendung ist. Die Laufzeittask wird in einem Java-Snippet eines Business-Prozesses erstellt und verwendet eine Schnittstelle der Prozessanwendung.

Das Beispiel funktioniert nur im Kontext der aufrufenden Enterprise-Anwendung, für die die Ressourcen geladen werden.

1. Greifen Sie auf die ClientTaskFactory zu, und erstellen Sie eine Ressourcengruppe, die die Definitionen des neuen Taskmodells enthalten soll.

   ```
   ClientTaskFactory factory = ClientTaskFactory.newInstance();

   // Angeben des Ladeprogramms für Kontextklassen, damit nach den folgenden
      Ressourcen gesucht wird
      ResourceSet resourceSet = factory.createResourceSet
                 ( Thread.currentThread().getContextClassLoader() );
   ```

2. Greifen Sie auf die WSDL-Definition und die Beschreibungen Ihrer Operationen zu.

   Geben Sie den Pfad in der übergeordneten JAR-Paketdatei an.

```
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. Erstellen Sie das EMF-Modell Ihrer neuen Benutzertask.

   Wenn Sie eine Taskinstanz erstellen, ist kein Gültigkeitsstartdatum (UTCDate) erforderlich.

   ```
   TTask humanTask = factory.createTTask( resourceSet,
                                          TTaskKinds.HTASK_LITERAL,
                                          "TestTask",
                                          new UTCDate( "2005-01-01T00:00:00" ),
                                          "http://www.ibm.com/task/test/",
                                          portType, operation );
   ```

   Durch diesen Schritt werden die Merkmale des neuen Taskmodells mit den Standardwerten initialisiert.

4. Ändern Sie die Merkmale Ihres Benutzertaskmodells.

   ```
   // Verwenden der Methoden aus dem Paket com.ibm.wbit.tel, z. B.:
   humanTask.setBusinessRelevance( TBoolean, YES_LITERAL );

   // Abrufen der Task-Factory zum Erstellen oder Ändern von zusammengesetzten Taskelementen
   TaskFactory taskFactory = factory.getTaskFactory();

   // Angeben der Eskalationseinstellungen
   TVerb verb = taskFactory.createTVerb();
   verb.setName("Karl");

   // Erstellen des Eskalationsempfängers und Hinzufügen eines Verbs
   TEscalationReceiver escalationReceiver =
                       taskFactory.createTEscalationReceiver();
   escalationReceiver.setVerb(verb);

   // Erstellen einer Eskalation und Hinzufügen eines Eskalationsempfängers
   TEscalation escalation = taskFactory.createTEscalation();
   escalation.setEscalationReceiver(escalationReceiver);
   ```

5. Erstellen Sie das Taskmodell, das alle Ressourcendefinitionen enthält.

   ```
   TaskModel taskModel = ClientTaskFactory.createTaskModel( resourceSet );
   ```

6. Prüfen Sie das Taskmodell, und korrigieren Sie sämtliche gefundenen Prüffehler.

   ```
   ValidationProblem[] validationProblems = taskModel.validate();
   ```

7. Erstellen Sie die Laufzeit-Taskinstanz bzw. die Laufzeitschablone.

   Verwenden Sie die Schnittstelle HumanTaskManagerService, um die Taskinstanz oder die Taskschablone zu erstellen. Sie müssen einen Anwendungsnamen angeben, der die Datentypdefinitionen enthält, damit ein Zugriff möglich ist.

   - Durch das folgende Snippet wird eine Taskinstanz erstellt:

     ```
     task.createTask( taskModel, "WorkflowApplication", "HTM" );
     ```

   - Durch das folgende Snippet wird eine Taskschablone erstellt:

     ```
     task.createTaskTemplate( taskModel, "WorkflowApplication" );
     ```

Wenn eine Laufzeit-Taskinstanz erstellt wurde, kann sie nun gestartet werden. Wenn eine Laufzeit-Taskschablone erstellt wurde, können Sie nun auf der Basis der Schablone Taskinstanzen erstellen.

Schnittstelle HumanTaskManagerService

Die Schnittstelle HumanTaskManagerService stellt taskbezogene Funktionen bereit, die von einem lokalen oder fernen Client aufgerufen werden können.

Welche Methoden aufgerufen werden können, hängt vom Status der Task ab und von den Berechtigungen der Person, die die Anwendung verwendet, in der die Methoden enthalten sind. Die Hauptmethoden zum Bearbeiten von Tasks werden nachfolgend aufgelistet. Weitere Informationen zu diesen Methoden und den übrigen Methoden, die in der Schnittstelle HumanTaskManagerService zur Verfügung stehen, enthält das Javadoc im Paket com.ibm.task.api.

Taskschablonen

Die folgenden Methoden zum Arbeiten mit Taskschablonen stehen zur Verfügung.

Tabelle 29. API-Methoden für Taskschablonen

Methode	Beschreibung
getTaskTemplate	Ruft die angegebene Taskschablone ab
createAndCallTask	Erstellt und startet eine Taskinstanz aus der angegebenen Taskschablone und wartet im synchronen Wartestatus auf das Ergebnis
createAndStartTask	Erstellt und startet eine Taskinstanz aus der angegebenen Taskschablone
createTask	Erstellt eine Taskinstanz aus der angegebenen Taskschablone
createInputMessage	Erstellt eine Eingabenachricht für die angegebene Taskschablone (erstellen Sie z. B. eine Nachricht, die zum Starten einer Task verwendet werden kann)
queryTaskTemplates	Ruft in der Datenbank gespeicherte Taskschablonen ab

Taskinstanzen

Die folgenden Methoden zum Arbeiten mit Taskinstanzen stehen zur Verfügung.

Tabelle 30. API-Methoden für Taskinstanzen

Methode	Beschreibung
getTask	Ruft eine Taskinstanz ab, die jeden beliebigen Status aufweisen kann
callTask	Startet eine Aufruftask im synchronen Modus
startTask	Startet eine Task, die bereits erstellt ist
suspend	Setzt die Collaboration-Task oder die unerledigte Task aus
resume	Nimmt die Collaboration-Task oder die unerledigte Task wieder auf
terminate	Beendet die angegebene Taskinstanz (wird eine Aufruftask beendet, hat diese Aktion keine Auswirkung auf den aufgerufenen Service)
delete	Löscht die angegebene Taskinstanz
claim	Beansprucht die Task für die Verarbeitung
update	Aktualisiert die Taskinstanz
complete	Beendet die Taskinstanz

Tabelle 30. API-Methoden für Taskinstanzen (Forts.)

Methode	Beschreibung
cancelClaim	Gibt eine beanspruchte Taskinstanz frei, damit sie durch einen anderen potenziellen Eigner bearbeitet werden kann
createWorkItem	Erstellt ein Arbeitselement für die Taskinstanz
transferWorkItem	Überträgt das Arbeitselement an einen angegebenen Eigner
deleteWorkItem	Löscht das Arbeitselement

Eskalationen

Die folgenden Methoden zum Arbeiten mit Eskalationen stehen zur Verfügung.

Tabelle 31. API-Methoden zum Arbeiten mit Eskalationen

Methode	Beschreibung
getEscalation	Ruft die angegebene Eskalationsinstanz ab

Benutzerdefinierte Merkmale

Tasks, Taskschablonen und Eskalation können über benutzerdefinierte Merkmale verfügen. Die Schnittstelle stellt eine Methode zum Abrufen und Festlegen sowie festgelegte Werte für Variablen bereit. Außerdem können Sie benannte Merkmale zu Taskinstanzen zuordnen und daraus abrufen. Namen und Werte benutzerdefinierter Merkmale müssen dem Typ java.lang.String angehören. Die folgenden Methoden sind für Tasks, Taskschablonen und Eskalationen zulässig.

Tabelle 32. API-Methoden für Variablen und benutzerdefinierte Merkmale

Methode	Beschreibung
getCustomProperty	Ruft das benannte benutzerdefinierte Merkmal der angegebenen Taskinstanz ab
getCustomProperties	Ruft die benutzerdefinierten Merkmale der angegebenen Taskinstanz ab
getCustomPropertyNames	Ruft die Namen der benutzerdefinierten Merkmale für die Taskinstanz ab
setCustomProperty	Speichert benutzerdefinierte Werte für die angegebene Taskinstanz

Zulässige Aktionen für Tasks:

Welche Aktionen an einer Task ausgeführt werden können, ist davon abhängig, ob es sich um eine unerledigte Task, eine Collaboration-Task, eine Aufruftask oder eine Verwaltungstask handelt.

Nicht alle von der Schnittstelle HumanTaskManager bereitgestellten Aktionen können auf alle Taskarten angewendet werden. Die folgende Tabelle zeigt, welche Aktionen auf die verschiedenen Taskarten angewendet werden können.

Aktion	Art der Task			
	Unerledigte Task	Collaboration-Task	Aufruftask	Verwaltungstask
callTask			X	
cancelClaim	X	X[1]		
claim	X	X[1]		
complete	X	X[1]		X
completeWithFollowOnTask[4]	X	X[1]		
completeWithFollowOnTask[5]		X[3]	X[3]	
createFaultMessage	X	X	X	X
createInputMessage	X	X	X	X
createOutputMessage	X	X	X	X
createWorkItem	X	X[1]	X	X
delete	X[1]	X[1]	X	X[1]
deleteWorkItem	X	X[1]	X	X
getCustomProperty	X	X[1]	X	X
getDocumentation	X	X[1]	X	X
getFaultNames	X	X[1]		
getFaultMessage	X	X[1]	X	
getInputMessage	X	X[1]	X	
getOutputMessage	X	X[1]	X	
getUsersInRole	X	X[1]	X	X
getTask	X	X[1]	X	X
getUISettings	X	X[1]	X	X
resume	X	X[1]		
setCustomProperty	X	X[1]	X	X
setFaultMessage	X	X[1]		
setOutputMessage	X	X[1]		
startTask	X[1]	X[1]	X	X
startTaskAsSubtask[6]	X	X[1]		
startTaskAsSubtask[7]		X[3]	X[3]	
suspend	X	X[1]		
suspendWithCancelClaim	X	X[1]		
terminate	X[1]	X[1]	X[1]	
transferWorkItem	X	X[1]	X	X
update	X	X[1]	X	X

Aktion	Art der Task			
	Unerledigte Task	**Collaboration-Task**	**Aufruftask**	**Verwaltungstask**
Anmerkungen: 1. Nur für eigenständige Tasks, Ad-hoc-Tasks und Taskschablonen 2. Nur für eigenständige Tasks, integrierte Tasks in Business-Prozessen und Ad-hoc-Tasks 3. Nur für eigenständige Tasks und Ad-hoc-Tasks 4. Tasktypen, die über Folgetasks verfügen können 5. Tasktypen, die als Folgetasks verwendet werden können 6. Tasktypen, die über Subtasks verfügen können 7. Tasktypen, die als Subtasks verwendet werden können				

Anwendungen für Business-Prozesse und Benutzertasks entwickeln

Personal ist an den meisten Business-Prozessszenarios beteiligt. So ist beispielsweise eine Personalinteraktion für einen Business-Prozess erforderlich, wenn der Prozess gestartet oder verwaltet wird oder wenn Benutzertaskaktivitäten durchgeführt werden. Zur Unterstützung dieser Szenarios müssen sowohl die Business Flow Manager-API als auch die Human Task Manager-API verwendet werden.

Um Personal an Business-Prozessszenarios zu beteiligen, können Sie die folgenden Tasktypen in den Business-Prozess integrieren:

- Eine integrierte Aufruftask (auch *Ursprüngliche Task* in der API genannt).

  Sie können für jede receive-Aktivität, für jedes onMessage-Element einer pick-Aktivität sowie für jedes onEvent-Element eines Ereignishandlers eine Aufruftask bereitstellen. Diese Task steuert dann, welche Personen für den Start eines Prozesses oder die Kommunikation mit einer aktiven Prozessinstanz berechtigt sind.
- Eine Verwaltungstask.

  Sie können eine Verwaltungstask bereitstellen, um anzugeben, welche Person für die Verwaltung des Prozesses oder die Durchführung von Verwaltungsoperationen der fehlgeschlagenen Aktivitäten des Prozesses berechtigt ist.
- Eine unerledigte Task (auch *Teilnehmende Task* in der API genannt).

  Eine unerledigte Task implementiert eine Benutzertaskaktivität. Dieser Aktivitätstyp ermöglicht Ihnen die Integration von Personal in den Prozess.

Benutzertaskaktivitäten im Business-Prozess stellen die unerledigten Tasks dar, die Personal im Business-Prozessszenario durchführt. Für die Realisierung dieser Szenarios können Sie sowohl die Business Flow Manager-API als auch die Human Task Manager-API verwenden:

- Der Business-Prozess ist der Container für alle Aktivitäten, die zu dem Prozess gehören, einschließlich der Benutzertaskaktivitäten, die durch unerledigte Tasks dargestellt werden. Bei der Erstellung einer Prozessinstanz wird eine eindeutige Objekt-ID (PIID) zugeordnet.
- Wenn während der Ausführung der Prozessinstanz eine Benutzertaskaktivität aktiviert wird, wird eine Aktivitätsinstanz erstellt, die durch ihre eindeutige Objekt-ID (AIID) angegeben wird. Gleichzeitig wird auch eine integrierte unerledigte Aufgabe erstellt, die durch ihre Objekt-ID TKIID angegeben wird.

Die Beziehung zwischen der Benutzertaskaktivität und der Taskinstanz wird durch die Verwendung der Objekt-IDs hergestellt:
  - Die ID der unerledigten Task der Aktivitätsinstanz erhält den Wert der TKIID der zugeordneten unerledigten Task.
  - Die Kontext-ID des Einschlusses der Taskinstanz erhält den Wert der PIID der Prozessinstanz mit der zugeordneten Aktivitätsinstanz.
  - Die übergeordnete Kontext-ID der Taskinstanz erhält den Wert der AIID der zugeordneten Aktivitätsinstanz.
- Die Lebenszyklen aller integrierten unerledigten Taskinstanzen werden von der Prozessinstanz verwaltet. Wenn die Prozessinstanz gelöscht wird, werden auch die Taskinstanzen gelöscht. d. h., alle Tasks, deren Kontext-ID des Einschlusses auf den Wert der PIID der Prozessinstanz gesetzt ist, werden automatisch gelöscht.

Prozessschablonen oder -aktivitäten ermitteln, die gestartet werden können

Ein Business-Prozess kann durch Aufrufen der Methoden call, initiate bzw. sendMessage der Business Flow Manager-API gestartet werden. Wenn der Prozess lediglich eine einzige Startaktivität aufweist, können Sie die Methodensignatur verwenden, für die ein Prozessschablonenname als Parameter erforderlich ist. Wenn der Prozess mehr als eine Startaktivität aufweist, müssen Sie diese explizit angeben.

Wenn ein Business-Prozess modelliert wird, kann die für die Modellierung zuständige Person festlegen, dass nur ein Teil der Benutzer eine Prozessinstanz auf der Basis der Prozessschablone erstellen darf. Dabei wird eine integrierte Aufruftask einer Startaktivität des Prozesses zugeordnet, und es werden Berechtigungseinschränkungen für diese Task angegeben. Es dürfen nur die Personen eine Instanz der Task und somit eine Instanz der Prozessschablone erstellen, bei denen es sich um potenzielle Starter bzw. Administratoren der Task handelt.

Wenn eine integrierte Aufruftask nicht der Startaktivität zugeordnet wird oder wenn für die Task keine Berechtigungseinschränkungen angegeben sind, kann jeder mit der Startaktivität eine Prozessinstanz erstellen.

Ein Prozess kann mehrere Startaktivitäten aufweisen, wobei jede andere Personalabfragen für potenzielle Starter oder Administratoren enthält. Dies bedeutet, dass ein Benutzer dazu berechtigt sein kann, einen Prozess mit der Aktivität A, aber nicht mit der Aktivität B zu starten.

1. Verwenden Sie die Business Flow Manager-API, um eine Liste der aktuellen Versionen der Prozessschablonen zu erstellen, die sich im Status 'Gestartet' befinden.

   Tipp: Mit der Methode queryProcessTemplates werden nur die Prozessschablonen ausgeschlossen, die Teil von Anwendungen sind, die noch nicht gestartet wurden. Wenn Sie diese Methode ohne Filterung der Ergebnisse verwenden, werden alle Versionen der Prozessschablonen zurückgegeben, unabhängig davon, welchen Status sie aufweisen.

   ```
   // Aktuelle Zeitmarke im UTC-Format, konvertiert in das Format 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);
```

Die Ergebnisse sind nach dem Namen der Prozessschablone sortiert.

2. Erstellen Sie die Liste der Prozessschablonen sowie die Liste der Startaktivitäten, für die der Benutzer berechtigt ist.

   Die Liste der Prozessschablonen enthält die Prozessschablonen, die über eine einzige Startaktivität verfügen. Diese Aktivitäten sind entweder nicht sicher, oder der angemeldete Benutzer darf sie nicht starten. Alternativ dazu können Sie die Prozessschablonen erfassen, die zumindest mit einer Startaktivität gestartet werden können.

   Tipp: Eine Prozessinstanz kann auch von einem Prozessadministrator gestartet werden. Zum Abrufen einer vollständigen Liste der Schablonen müssen Sie außerdem die Verwaltungstaskschablone lesen, die der Prozessschablone zugeordnet ist und prüfen, ob es sich bei dem angemeldeten Benutzer um einen Administrator handelt.

   ```
   List authorizedProcessTemplates = new ArrayList();
   List authorizedActivityServiceTemplates = new ArrayList();
   ```

3. Ermitteln Sie die Startaktivitäten für die einzelnen Prozessschablonen.

   ```
   for( int i=0; i<processTemplates.length; i++ )
   {
     ProcessTemplateData template = processTemplates[i];
     ActivityServiceTemplateData[] startActivities =
                   process.getStartActivities(template.getID());
   ```

4. Rufen Sie für jede Startaktivität die ID der zugeordneten integrierten Aufruftaskschablone ab.

   ```
   for( int j=0; j<startActivities.length; j++ )
   {
     ActivityServiceTemplateData activity = startActivities[j];
     TKTID tktid = activity.getTaskTemplateID();
   ```

   a. Falls keine Aufruftaskschablone vorhanden ist, wird die Prozessschablone nicht durch diese Startaktivität gesichert.

      In diesem Fall kann mithilfe dieser Startaktivität jedermann eine Prozessinstanz erstellen.

      ```
      boolean isAuthorized = false;
             if ( tktid == null )
          {
            isAuthorized = true;
            authorizedActivityServiceTemplates.add(activity);
          }
      ```

   b. Falls eine Aufruftaskschablone vorhanden ist, verwenden Sie die Human Task Manager-API, um die Berechtigung für den angemeldeten Benutzer zu prüfen.

      In dem angegebenen Beispiel heißt der angemeldete Benutzer 'Schmidt'. Bei dem angemeldeten Benutzer muss es sich um einen potenziellen Starter der Aufruftask oder um einen Administrator handeln.

      ```
      if ( tktid != null )
          {
            isAuthorized =
              task.isUserInRole
                  (tkid, "Schmidt", WorkItem.REASON_POTENTIAL_STARTER) ||
              task.isUserInRole(tktid, "Schmidt", WorkItem.REASON_ADMINISTRATOR);

            if ( isAuthorized )
      ```

```
        {
          authorizedActivityServiceTemplates.add(activity);
        }
      }
```

Wenn der Benutzer den angegebenen Aufgabenbereich aufweist oder wenn keine Personalzuordnungskriterien für den Aufgabenbereich angegeben sind, gibt die Methode isUserInRole den Wert `true` zurück.

5. Überprüfen Sie, ob der Prozess nur mithilfe des Namens der Prozessschablone gestartet werden kann.

```
if ( isAuthorized && startActivities.length == 1 )
    {
      authorizedProcessTemplates.add(template);
    }
```

6. Beenden Sie die Schleifen.

```
  } // Ende der Schleife für die einzelnen Aktivitätsserviceschablonen
}   // Ende der Schleife für die einzelnen Prozessschablonen
```

Workflow für Einzelperson mit Benutzertasks verarbeiten

Manche Workflows werden nur von Einzelpersonen ausgeführt (z. B. das Bestellen von Büchern bei einem Online-Buchversand). In diesem Beispiel wird gezeigt, wie die Abfolge von Aktionen für die Buchbestellung als eine Serie von Benutzertaskaktivitäten (unerledigte Tasks) implementiert werden kann. Sowohl die Business Flow Manager- als auch die Human Task Manager-APIs werden für die Verarbeitung des Workflows verwendet.

Auf der Website eines Online-Buchversands führt der Käufer nacheinander eine Reihe von Aktionen aus, um ein Buch zu bestellen. Die Abfolge dieser Aktionen kann als eine Serie von Benutzertaskaktivitäten (unerledigte Tasks) implementiert werden. Wenn der Käufer beschließt, mehrere Bücher zu bestellen, entspricht dies dem Beanspruchen der nächsten Benutzertaskaktivität. Informationen zur Abfolge von Tasks werden vom Business Flow Manager gepflegt; die Tasks selbst werden vom Human Task Manager gepflegt.

Vergleichen Sie dieses Beispiel mit dem Beispiel, bei dem lediglich die Business Flow Manager-API verwendet wird.

1. Verwenden Sie die Business Flow Manager-API, um die Prozessinstanz abzurufen, mit der Sie arbeiten möchten.

   In diesem Beispiel mit einer Instanz des Prozesses der Kundenbestellung.

```
ProcessInstanceData processInstance =
   process.getProcessInstance("Kundenbestellung");
String piid = processInstance.getID().toString();
```

2. Verwenden Sie die Human Task Manager-API, um die unerledigten Tasks im Status 'Bereit' (teilnehmende Tasks) abzufragen, die Teil der angegebenen Prozessinstanz sind.

   Geben Sie die übergeordnete Prozessinstanz mithilfe der Kontext-ID des Einschlusses der Task an. Für einen Workflow für Einzelpersonen gibt die Abfrage die unerledigte Task zurück, die der ersten Benutzertaskaktivität in der Abfolge der Benutzertaskaktivitäten zugeordnet ist.

```
//
 //Liste der unerledigten Tasks abrufen, die der angemeldete Benutzer
 // für die angegebene Prozessinstanz anfordern kann
//
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. Fordern Sie die unerledigte Task an, die zurückgegeben wird.

```
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();
    // Lesen der Werte
    ...
  }
}
```

   Wenn die Task beansprucht ist, wird die Eingabenachricht der Task zurückgegeben.

4. Ermitteln Sie die Benutzertaskaktivität, die der unerledigten Task zugeordnet ist.

   Sie können eine der folgenden Methoden für die Korrelation von Aktivitäten mit deren Tasks verwenden.

   - Methode task.getActivityID:

     ```
     AIID aiid = task.getActivityID(tkiid);
     ```

   - Übergeordnete Kontext-ID, die Teil des Taskobjekts ist:

     ```
     AIID aiid = null;
     Task taskInstance = task.getTask(tkiid);

     OID oid = taskInstance.getParentContextID();
     if ( oid != null and oid instanceof AIID )
     {
       aiid = (AIID)oid;
     }
     ```

5. Wenn die Arbeit für die Task fertig gestellt ist, verwenden Sie die Business Flow Manager-API, um die Task und ihre zugeordnete Benutzertaskaktivität abzuschließen und die nächste Benutzertaskaktivität in der Prozessinstanz zu beanspruchen.

   Zum Beenden der Benutzertaskaktivität wird eine Ausgabenachricht übergeben. Beim Erstellen der Ausgabenachricht müssen Sie den Nachrichtentypnamen so angeben, dass er die Nachrichtendefinition enthält.

```
ActivityInstanceData activity = process.getActivityInstance(aiid);
ClientObjectWrapper output =
      process.createMessage(aiid, activity.getOutputMessageTypeName());
DataObject myMessage = null ;
if ( output.getObject()!= null && output.getObject() instanceof DataObject )
{
  myMessage = (DataObject)output.getObject();
  // Festlegen der Teile in der Nachricht (z. B. eine Bestellnummer)
  myMessage.setInt("Bestellnr", 4711);
}

//Abschließen der Benutzertaskaktivität und der zugeordneten unerledigten Task
// sowie Beanspruchen der nächsten Benutzertaskaktivität
CompleteAndClaimSuccessorResult successor =
   process.completeAndClaimSuccessor(aiid, output);
```

Diese Aktion legt eine Ausgabenachricht fest, die die Bestellnummer enthält, und beansprucht die nächste Benuztertaskaktivität in der Abfolge. Wenn `AutoClaim` für Folgeaktivitäten gesetzt ist, und mehrere mögliche Pfade vorhanden sind, werden alle Folgeaktivitäten beansprucht und eine beliebige Aktivität als nächste Aktivität übergeben. Wenn dem aktuellen Benutzer keine weiteren Folgeaktivitäten zugeordnet werden können, wird `Null` zurückgegeben.

Wenn der Prozess parallele Pfade enthält, die beschritten werden können, und diese Pfade Benutzertaskaktivitäten enthalten, für die der angemeldete Benutzer ein möglicher Eigner ist, wird automatisch eine beliebige Aktivität beansprucht und als nächste Aktivität übergeben.

6. Bearbeiten Sie die nächste Benutzertaskaktivität.

```
ClientObjectWrapper nextInput = successor.getInputMessage();
if ( nextInput.getObject()!=
               null && nextInput.getObject() instanceof DataObject )
{
  activityInput = (DataObject)input.getObject();
  // Lesen der Werte
  ...
}

aiid = successor.getAIID();
```

7. Fahren Sie mit Schritt 5 fort, um die Benutzertaskaktivität abzuschließen und die nächste Benutzertaskaktivität anzurufen.

Zugehörige Tasks

„Workflow für Einzelperson verarbeiten" auf Seite 74
Manche Workflows werden nur von Einzelpersonen ausgeführt (z. B. das Bestellen von Büchern bei einem Online-Buchversand). Bei diesem Workflowtyp gibt es keine parallelen Pfade. Die API completeAndClaimSuccessor unterstützt die Verarbeitung dieses Workflowtyps.

Handhabung von Fehlern und Ausnahmebedingungen

In einem BPEL-Prozess können Fehler an verschiedenen Stellen des Prozesses auftreten.

BPEL-Fehler (BPEL = Business Process Execution Language) stammen aus folgenden Quellen:

- Web-Service-Aufruffehler (Web Services Description Language, WSDL)
- Auslöseaktivitäten
- Von Business Process Choreographer erkannte BPEL-Standardfehler

Mechanismen zur Handhabung dieser Fehler sind vorhanden. Verwenden Sie die folgenden Mechanismen zum Handhaben von Fehlern, die von einer Prozessinstanz generiert werden:

- Steuerung an die entsprechenden Fehlerhandler übergeben
- Vorherige Arbeit im Prozess kompensieren
- Prozess stoppen, damit die Fehlersituation von einem Benutzer repariert werden kann (erzwungene Wiederholung, erzwungene Beendigung)

Ein BPEL-Prozess kann Fehler auch an den Aufrufenden einer Operation zurückgeben, die von dem Prozess bereitgestellt wurde. Sie können den Fehler im Prozess als ein Antwortaktivität mit einem Fehlernamen und Fehlerdaten modellieren. Diese Fehler werden als geprüfte Ausnahmen an den Aufrufenden der API übergegeben.

Wenn ein BPEL-Prozess einen BPEL-Fehler nicht verarbeitet, oder wenn eine API-Ausnahmebedingung auftritt, wird eine Laufzeitausnahmebedingung an den Aufrufenden der API übergeben. Ein Beispiel für eine API-Ausnahmebedingung ist das Nichtvorhandensein des Prozessmodells, aus dem eine Instanz erstellt werden soll.

Die Handhabung von Fehlern und Ausnahmebedingungen wird in den folgenden Tasks beschrieben.

Handhabung von API-Ausnahmebedingungen

Wird eine Methode in der Schnittstelle BusinessFlowManagerService oder HumanTaskManagerService nicht erfolgreich ausgeführt, wird eine Ausnahmebedingung ausgelöst, die auf die Ursache des Fehlers hinweist. Sie können eine spezielle Handhabung für diese Ausnahmebedingung festlegen, um dem Aufrufenden eine Hilfestellung zu geben.

Üblicherweise wird jedoch nur für einen Teil der Ausnahmebedingungen eine spezielle Handhabung festgelegt. Für die übrigen potenziellen Ausnahmebedingungen wird eine allgemeine Anleitung zur Verfügung gestellt. Alle spezifischen Ausnahmebedingungen übernehmen Angaben von einer generischen ProcessException oder TaskException. Eine *empfohlene Methode* besteht darin, generische Ausnahmebedingungen mit einer abschließenden Anweisung `catch(ProcessException)` oder `catch(TaskException)` abzufangen. Diese Anweisung trägt dazu bei, die Aufwärtskompatibilität Ihres Anwendungsprogramms zu gewährleisten, da sie alle anderen Ausnahmebedingungen berücksichtigt, die auftreten können.

Prüfen, welcher Fehler für eine Aktivität festgelegt ist

1. Listen Sie die Taskaktivitäten auf, die sich in einem Fehlschlag- oder Stoppstatus befinden.

```
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);
```

   Diese Aktion gibt ein Abfrageergebnis zurück, das fehlgeschlagene bzw. gestoppte Aktivitäten enthält.

2. Lesen Sie den Namen des Fehlers.

```
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();
  }
}
```

   Diese Aktion gibt den Fehlernamen zurück. Sie können auch die nicht behobene Ausnahmebedingung für eine gestoppte Aktivität analysieren, anstatt den Fehlernamen abzurufen.

Prüfen, welcher Fehler für eine gestoppte Aufrufaktivität aufgetreten ist

Wenn eine Aktivität einen Fehler auslöst, bestimmt der Fehlertyp, welche Aktionen Sie ausführen können, um die Aktivität zu reparieren.

1. Listen Sie die Benutzertaskaktivitäten auf, die sich im Stoppstatus befinden.

```
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);
```

   Diese Aktion gibt ein Abfrageergebnis zurück, das gestoppte Aufrufaktivitäten enthält.

2. Lesen Sie den Namen des Fehlers.

```
if (result.size() > 0)
{
  result.first();
  AIID aiid = (AIID) result.getOID(1);
  ActivityInstanceData activity = process.getActivityInstance(aiid);

  ProcessException excp = activity.getUnhandledException();
  if ( excp instanceof ApplicationFaultException )
  {
   ApplicationFaultException fault = (ApplicationFaultException)excp;
   String faultName = fault.getFaultName();
  }
}
```

Web-Service-API-Clientanwendungen entwickeln

Sie können Clientanwendungen entwickeln, die über Web-Services-APIs auf Business-Prozessanwendungen und Benutzertaskanwendungen zugreifen.

Clientanwendungen können in jeder Web-Service-Clientumgebung entwickelt werden, einschließlich Java Web-Services und Microsoft .NET.

Einführung: Web-Services

Web-Services sind webbasierte Unternehmensanwendungen, die offene, XML-basierte Standards und Übertragungsprotokolle für den Datenaustausch mit Clientanwendungen verwenden. Web-Services ermöglichen die Verwendung eines sprach- und umgebungsneutralen Programmiermodells.

Web-Services verwenden die folgende Kerntechnologie:

- XML (Extensible Markup Language). XML löst das Problem der Datenunabhängigkeit. Mit XML können Sie Daten beschreiben und diese Daten innerhalb bzw. außerhalb jeder Anwendung oder Programmiersprache zuordnen.
- WSDL (Web Services Description Language). Mit dieser XML-basierten Sprache können Sie eine Beschreibung einer zu Grunde liegenden Anwendung erstellen. Diese Beschreibung macht aus einer Anwendung einen Web-Service. Sie fungiert als Schnittstelle zwischen der zu Grunde liegenden Anwendung und anderen webfähigen Anwendungen.
- SOAP (Simple Object Access Protocol). SOAP ist das zentrale Kommunikationsprotokoll für das Web. Die meisten Web-Services kommunizieren über dieses Protokoll miteinander.

Web-Service-Komponenten und Steuerungsabfolge

Mehrere clientseitige und serverseitige Komponenten sind an der Steuerungsabfolge beteiligt, aus der eine Web-Service-Anforderung mit dazugehöriger Antwort besteht.

Eine typische Steuerungsabfolge sieht wie folgt aus.

1. Auf der Clientseite:
   a. Eine Clientanwendung (vom Benutzer bereitgestellt) gibt eine Anforderung für einen Web-Service aus.
   b. Ein Proxy-Client (ebenfalls vom Benutzer bereitgestellt, kann aber auch von clientseitigen Dienstprogrammen automatisch generiert werden) hüllt die Serviceanforderung in eine SOAP-Anforderung ein.
   c. Die clientseitige Entwicklungsinfrastruktur leitet die Anforderung an eine URL-Adresse weiter, die als Endpunkt des Web-Service definiert ist.
2. Im Netz wird die Anforderung unter Verwendung von HTTP oder HTTPS zum Web-Service-Endpunkt übertragen.
3. Auf der Serverseite:
   a. Die generische Web-Services-API empfängt und dekodiert die Anforderung.
   b. Die Anforderung wird direkt von der generischen Business Flow Manager- oder Human Task Manager-Komponente verarbeitet oder zum angegebenen Business-Prozess bzw. zur angegebenen Benutzertask weitergeleitet.
   c. Die Rückgabedaten werden in einen SOAP-Antwortumschlag eingehüllt.
4. Im Netz wird die Antwort unter Verwendung von HTTP oder HTTPS zur clientseitigen Umgebung übertragen.
5. Zurück auf der Clientseite:
   a. Die clientseitige Entwicklungsinfrastruktur packt den SOAP-Antwortumschlag aus.
   b. Der Proxy-Client extrahiert die Daten aus der SOAP-Antwort und übergibt sie an die Clientanwendung.
   c. Die Clientanwendung verarbeitet die zurückgegebenen Daten entsprechend.

Übersicht über die Web-Services-APIs

Web-Services-APIs ermöglichen das Entwickeln von Clientanwendungen, die über Web-Services auf Business-Prozesse und Benutzertasks zugreifen, die in der Business Process Choreographer-Umgebung ausgeführt werden.

Die Web-Services-API von Business Process Choreographer stellt zwei separate Web-Service-Schnittstellen (WSDL-Porttypen) zur Verfügung:

- Die Business Flow Manager-API ermöglicht Clientanwendungen die Interaktion mit Mikroprozessen und mit Prozessen mit langer Laufzeit. Beispiele:
  – Prozessschablonen und Prozessinstanzen erstellen
  – Vorhandene Prozesse beanspruchen
  – Einen Prozess nach seiner ID abfragen

  Eine vollständige Liste der möglichen Aktionen finden Sie in „Anwendungen für Business-Prozesse entwickeln" auf Seite 64.
- Human Task Manager-API. Bietet folgende Möglichkeiten für Clientanwendungen:
  – Tasks erstellen und starten
  – Vorhandene Tasks beanspruchen

– Tasks abschließen
– Eine Task nach ihrer ID abfragen
– Eine Taskobjektgruppe abfragen

Eine vollständige Liste der möglichen Aktionen finden Sie in „Anwendungen für Benutzertasks entwickeln" auf Seite 83.

Clientanwendungen können eine oder beide Web-Service-Schnittstellen verwenden.

Beispiel:

Das folgende Beispiel enthält einen möglichen Entwurf für eine Clientanwendung, die auf die Web-Services-API von Human Task Manager zugreift, um eine bestimmte teilnehmende Benutzertask zu verarbeiten:

1. Die Clientanwendung gibt einen Web-Service-Aufruf query an WebSphere Process Server aus, in dem eine Liste der teilnehmenden Tasks angefordert wird, an denen ein Benutzer arbeiten soll.
2. Die Liste der teilnehmenden Tasks wird in einer SOAP/HTTP-Antwort zurückgegeben.
3. Die Clientanwendung gibt anschließend einen Web-Service-Aufruf claim aus, um eine der teilnehmenden Tasks zu beanspruchen.
4. WebSphere Process Server gibt die Eingabenachricht der Task zurück.
5. Die Clientanwendung gibt einen Web-Service-Aufruf complete aus, um die Task mit einer Ausgabe- oder Fehlernachricht abzuschließen.

Voraussetzungen für Business-Prozesse und Benutzertasks

Business-Prozesse und Benutzertasks, die mit WebSphere Integration Developer für die Ausführung unter Business Process Choreographer entwickelt wurden, müssen bestimmte Regeln erfüllen, damit die Web-Services-APIs auf sie zugreifen können.

Dabei gelten folgende Voraussetzungen:

1. Die Schnittstellen von Business-Prozessen und Benutzertasks müssen unter Verwendung des Stils "document/literal wrapped" definiert werden, der in der Spezifikation JAX-RPC 1.1 (Java API for XML-based RPC) definiert ist. Dies ist der Standardstil für alle mit WID entwickelten Business-Prozesse und Benutzertasks.
2. Fehlernachrichten, die von Business-Prozessen und Benutzertasks für Web-Service-Operationen ausgegeben werden, müssen aus einem einzigen WSDL-Nachrichtenteil bestehen, der mit einem XML-Schemaelement definiert ist. Beispiel:

   ```
   <wsdl:part name="meinFehler" element="meinNamensbereich:meinFehlerelement"/>
   ```

   Zugehörige Informationen

   Downloadseite der Java-API für XML-basierte RPC (JAX-RPC)

   Welcher WSDL-Stil sollte verwendet werden?

Clientanwendungen entwickeln

Der Entwicklungsprozess für Clientanwendungen besteht aus mehreren Schritten.

1. Legen Sie fest, welche Web-Services-API Ihre Clientanwendung verwenden sollte: Business Flow Manager-API, Human Task Manager-API oder beide.
2. Exportieren Sie die erforderlichen Dateien aus der WebSphere Process Server-Umgebung. Als Alternative können Sie die Dateien auch von der WebSphere Process Server-Client-CD kopieren.

3. Generieren Sie in der gewünschten Entwicklungsumgebung für Clientanwendungen einen *Proxy-Client* mithilfe der exportierten Artefakte.
4. Optional: Generieren Sie *Helper-Klassen*. Helper-Klassen sind erforderlich, wenn Ihre Clientanwendung direkt mit konkreten Prozessen oder Tasks auf dem WebSphere-Server interagiert. Sie sind jedoch nicht erforderlich, wenn Ihre Clientanwendung nur generische Tasks ausführen soll (z. B. Abfragen absetzen).
5. Entwickeln Sie den Code für Ihre Clientanwendung.
6. Fügen Sie alle erforderlichen Sicherheitsmechanismen für Ihre Clientanwendung hinzu.

Artefakte kopieren

Eine Anzahl Artefakte aus der WebSphere-Umgebung muss kopiert werden, um das Erstellen von Clientanwendungen zur unterstützen.

Diese Artefakte sind auf zwei Wegen erhältlich:

- Publizieren und Exportieren aus der WebSphere Process Server-Umgebung
- Kopieren der Dateien von der WebSphere Process Server-Client-CD

Artefakte aus der Serverumgebung veröffentlichen und exportieren

Bevor Sie mit der Entwicklung von Clientanwendungen für den Zugriff auf die Web-Services-APIs beginnen können, müssen Sie verschiedene Artefakte aus der WebSphere-Serverumgebung veröffentlichen und exportieren.

Folgende Artefakte müssen exportiert werden:

- WSDL-Dateien (WSDL = Web Service Definition Language) mit Beschreibungen der Porttypen und Operationen, aus denen die Web-Services-APIs bestehen.
- XML-Schemadefinitionsdateien (XSD-Dateien) mit Definitionen der Datentypen, auf die die Services und Methoden in den WSDL-Dateien verweisen.
- Zusätzliche WSDL- und XSD-Dateien mit Beschreibungen der Business-Objekte. Business-Objekte beschreiben konkrete Business-Prozesse oder Benutzertasks, die auf dem WebSphere-Server ausgeführt werden. Diese zusätzlichen Dateien sind nur erforderlich, wenn Ihre Clientanwendung über die Web-Services-APIs direkt mit den konkreten Business-Prozessen oder Benutzertasks interagieren muss. Sie sind nicht erforderlich, wenn Ihre Clientanwendung nur generische Tasks ausführen soll (z. B. Abfragen absetzen).

Nach dem Veröffentlichen müssen Sie diese Artefakte in Ihre Clientprogrammierumgebung kopieren, wo sie zum Generieren eines Proxy-Clients und von Helper-Klassen verwendet werden.

Web-Service-Endpunktadresse angeben:

Dies ist die URL-Adresse, die eine Clientanwendung angeben muss, um auf die Web-Services-APIs zuzugreifen. Die Endpunktadresse wird in die WSDL-Datei geschrieben, die Sie exportieren, um einen Proxy-Client für Ihre Clientanwendung zu generieren.

Welche Web-Service-Endpunktadresse verwendet werden sollte, hängt von Ihrer WebSphere-Serverkonfiguration ab:

- Szenario 1. Ein einzelner WebSphere-Server. Als WebSphere-Endpunktadresse sind Hostname und Portnummer des Servers anzugeben (z. B. **host1:9080**).

- Szenario 2. Ein aus mehreren Servern bestehender WebSphere-Cluster. Als WebSphere-Endpunktadresse sind Hostname und Port des Servers anzugeben, der als Host für die Web-Services-APIs fungiert (z. B. **host2:9081**).
- Szenario 3. Ein Web-Server wird als Front-End verwendet. Als WebSphere-Endpunktadresse sind Hostname und Port des Web-Servers anzugeben (z. B. **host:80**).

Das Standardformat der Web-Service-Endpunktadresse lautet *protokoll*`://`*host*`:`*port*`/`*stammkontext*`/`*fester_pfad*. Dabei gilt Folgendes:

- *protokoll*. Das Kommunikationsprotokoll, das zwischen der Clientanwendung und dem WebSphere-Server verwendet werden soll. Das Standardprotokoll ist HTTP. Sie können stattdessen auch das sicherere HTTPS-Protokoll (HTTP über SSL) verwenden. Es wird empfohlen, HTTPS zu verwenden.
- *host:port*. Hostname und Portnummer für den Zugriff auf die Maschine, die als Host für die Web-Services-APIs fungiert. Diese Werte variieren je nach WebSphere-Serverkonfiguration (z. B. abhängig davon, ob Ihre Clientanwendung direkt oder über ein Web-Server-Front-End auf die Anwendung zugreifen soll.
- *stammkontext*. Als Stammkontext können Sie einen beliebigen Wert angeben. Dieser Wert muss jedoch innerhalb jeder WebSphere-Zelle eindeutig sein. Der Standardwert verwendet ein Suffix "knotenserver/cluster", um das Risiko von Namensunverträglichkeiten zu vermeiden.
- *fester_pfad* ist entweder `/sca/com/ibm/bpe/api/BFMWS` (für die Business Flow Manager-API) oder `/sca/com/ibm/task/api/HTMWS` (für die Human Task Manager-API) und kann nicht geändert werden.

Die Web-Service-Endpunktadresse wird beim Konfigurieren des Business-Prozesscontainers oder Benutzertaskcontainers erstmals angegeben:

1. Melden Sie sich bei der Administrationskonsole mit einer Benutzer-ID mit Administratorberechtigung an.
2. Wählen Sie **Anwendungen → SCA-Module** aus.

   Anmerkung: Sie können auch **Anwendungen → Enterprise-Anwendungen** auswählen, um eine Liste aller verfügbaren Enterprise-Anwendungen aufzurufen.
3. Wählen Sie **BPEContainer** (für den Business-Prozesscontainer) oder **TaskContainer** (für den Benutzertaskcontainer) in der Liste der SCA-Module oder -Anwendungen aus.
4. Wählen Sie **Informationen zum HTTP-Endpunkt-URL angeben** in der Liste der **Weitere Merkmale** aus.
5. Wählen Sie eines der Standardpräfixe in der Liste aus, oder geben Sie ein eigenes Präfix ein. Verwenden Sie ein Präfix aus der Liste der Standardpräfixe, wenn Ihre Clientanwendungen eine direkte Verbindung zu dem Anwendungsserver herstellen sollen, der als Host für die Web-Services-API fungiert. Andernfalls geben Sie ein eigenes Präfix an.
6. Klicken Sie auf **Anwenden**, um das ausgewählte Präfix in das SCA-Modul zu kopieren.
7. Klicken Sie auf **OK**. Die URL-Informationen werden in Ihrem Arbeitsbereich gespeichert.

Den aktuellen Wert können Sie in der Administrationskonsole anzeigen (z. B. für den Business-Prozesscontainer über **Enterprise-Anwendungen → BPEContainer → Deployment-Deskriptor anzeigen**).

In der exportierten WSDL-Datei enthält das Attribut `location` des Elements `soap:address` die angegebene Web-Services-Endpunktadresse. Beispiel:

```
<wsdl:service name="BFMWSService">
  <wsdl:port name="BFMWSPort" binding="this:BFMWSBinding">
   <soap:address location=
         "https://meinserver:9080/WebServicesAPIs/sca/com/ibm/bpe/api/BFMWS"/>
```

Zugehörige Konzepte

„Sicherheit hinzufügen (Java-Web-Services)" auf Seite 124
Die Web-Service-Kommunikation muss durch Implementieren von Sicherheitsmechanismen in Ihrer Clientanwendung geschützt werden.

Zugehörige Tasks

„Sicherheit hinzufügen (.NET)" auf Seite 132
Die Web-Service-Kommunikation kann durch Implementieren von Sicherheitsmechanismen in Ihrer Clientanwendung geschützt werden.

WSDL-Dateien veröffentlichen:

Eine WSDL-Datei (WSDL = Web Service Definition Language) enthält eine detaillierte Beschreibung aller mit einer Web-Services-API verfügbaren Operationen. Für die Web-Services-APIs von Business Flow Manager und Human Task Manager stehen separate WSDL-Dateien zur Verfügung. Sie müssen diese WSDL-Dateien zunächst veröffentlichen und dann aus der WebSphere-Umgebung in Ihre Entwicklungsumgebung kopieren, wo sie zum Generieren eines Proxy-Clients verwendet werden.

Stellen Sie vor dem Veröffentlichen sicher, dass die richtige Web-Services-Endpunktadresse angegeben ist. Dies ist die URL-Adresse, unter der Ihre Clientanwendung auf die Web-Services-APIs zugreift.

Die WSDL-Dateien müssen nur ein Mal veröffentlicht werden.

Anmerkung: Wenn Sie über die WebSphere Process Server-Client-CD verfügen, können Sie die Dateien auch direkt von dieser CD in Ihre Client-Programmierungsumgebung kopieren.

WSDL-Datei des Business-Prozesses veröffentlichen:

Verwenden Sie die Administrationskonsole zum Veröffentlichen der WSDL-Datei.

1. Melden Sie sich bei der Administrationskonsole mit einer Benutzer-ID mit Administratorberechtigung an.
2. Wählen Sie **Anwendungen** → **SCA-Module** aus.

   Anmerkung: Sie können auch **Anwendungen** → **Enterprise-Anwendungen** auswählen, um eine Liste aller verfügbaren Enterprise-Anwendungen aufzurufen.
3. Wählen Sie die Anwendung **BPEContainer** aus der Liste der SCA-Module oder Anwendungen aus.
4. Wählen Sie **WSDL-Dateien veröffentlichen** in der Liste **Weitere Merkmale** aus.
5. Klicken Sie auf die ZIP-Datei in der Liste.
6. Klicken Sie in dem anschließend angezeigten Fenster Dateidownload auf **Speichern**.
7. Navigieren Sie zu einem lokalen Ordner, und klicken Sie auf **Speichern**.

Die exportierte komprimierte Datei hat den Namen BPEContainer_WSDLFiles.zip. Sie enthält eine WSDL-Datei mit der Beschreibung der Web-Services sowie alle XSD-Dateien, auf die in der WSDL-Datei verwiesen wird.

WSDL-Datei der Benutzertask veröffentlichen:

Verwenden Sie die Administrationskonsole zum Veröffentlichen der WSDL-Datei.

1. Melden Sie sich bei der Administrationskonsole mit einer Benutzer-ID mit Administratorberechtigung an.
2. Wählen Sie **Anwendungen → SCA-Module** aus.

   Anmerkung: Sie können auch **Anwendungen → Enterprise-Anwendungen** auswählen, um eine Liste aller verfügbaren Enterprise-Anwendungen aufzurufen.
3. Wählen Sie die Anwendung **TaskContainer** aus der Liste der SCA-Module oder Anwendungen aus.
4. Wählen Sie **WSDL-Dateien veröffentlichen** in der Liste **Weitere Merkmale** aus.
5. Klicken Sie auf die ZIP-Datei in der Liste.
6. Klicken Sie in dem anschließend angezeigten Fenster Dateidownload auf **Speichern**.
7. Navigieren Sie zu einem lokalen Ordner, und klicken Sie auf **Speichern**.

Die exportierte komprimierte Datei hat den Namen TaskContainer_WSDLFiles.zip. Sie enthält eine WSDL-Datei mit der Beschreibung der Web-Services sowie alle XSD-Dateien, auf die in der WSDL-Datei verwiesen wird.

Business-Objekte exportieren:

Business-Prozesse und Benutzertasks verfügen über klar strukturierte Schnittstellen, über die sie als externe Web-Services aufgerufen werden können. Wenn diese Schnittstellen auf Business-Objekte verweisen, müssen Sie die Schnittstellendefinitionen und Business-Objekte in Ihre Clientprogrammierumgebung exportieren.

Dieser Vorgang muss für jedes Business-Objekt ausgeführt werden, mit dem Ihre Clientanwendung interagieren soll.

In WebSphere Process Server definieren Business-Objekte das Format von Anforderungs-, Antwort- und Fehlernachrichten, die mit Business-Prozessen oder Benutzertasks interagieren. Diese Nachrichten können auch Definitionen komplexer Datentypen enthalten.

Beispielsweise müssen zum Erstellen und Starten einer Benutzertask folgende Informationselemente an die Taskschnittstelle übergeben werden:

- Name der Taskschablone
- Namespace der Taskschablone
- Eingabenachricht mit formatierten Business-Daten
- Antwort-Wrapper zum Übergeben der Antwortnachricht
- Fehlernachricht zum Übergeben von Fehlern und Ausnahmebedingungen

Diese Elemente sind in ein einziges Business-Objekt eingebunden. Alle Operationen der Web-Service-Schnittstelle werden als Operation vom Typ ″eingeschlossenes Dokument/Literal″ modelliert.

Eingabe- und Ausgabeparameter für diese Operationen sind in Wrapper-Dokumente eingebunden. Andere Business-Objekte definieren die entsprechenden Antwort- und Fehlernachrichtenformate.

Zum Erstellen und Starten des Business-Prozesses oder der Benutzertask über einen Web-Service müssen diese Wrapperobjekte auf der Clientseite für die Clientanwendung bereitgestellt werden.

Dies geschieht durch Exportieren der Business-Objekte aus der WebSphere-Umgebung als WSDL- und XSD-Dateien, Importieren der Datentypdefinitionen in Ihre Clientprogrammierumgebung und anschließendes Umwandeln in Helper-Klassen, die von der Clientanwendung verwendet werden können.

1. Starten Sie den WID-Arbeitsbereich (WID = WebSphere Integration Developer), sofern er noch nicht aktiviert ist.
2. Wählen Sie das Bibliotheksmodul aus, das die zu exportierenden Business-Objekte enthält. Ein Bibliotheksmodul ist eine komprimierte Datei, die die erforderlichen Business-Objekte enthält.
3. Exportieren Sie das Bibliotheksmodul.
4. Kopieren Sie die exportierten Dateien in Ihre Entwicklungsumgebung für Clientanwendungen.

Beispiel:

Angenommen, ein Business-Prozess macht die folgende Web-Serviceoperation zugänglich:

```
<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>
```

Dabei sind die WSDL-Nachrichten wie folgt definiert:

```
<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>
```

Die *konkreten* kundendefinierten Elemente `types:updateCustomer`, `types:updateCustomerResponse` und `types:updateCustomerFault` müssen von den Web-Services-APIs unter Verwendung von <xsd:any>-Parametern in allen *generischen* Operationen (call, sendMessage usw.) übergeben und empfangen werden, die von der Clientanwendung ausgeführt werden. Diese kundenbezogenen Elemente werden auf der Seite der Clientanwendung mithilfe von Helper-Klassen erstellt, serialisiert und entserialisiert, die unter Verwendung der exportierten XSD-Dateien generiert werden.

Zugehörige Tasks

„Helper-Klassen für BPEL-Prozesse erstellen (.NET)" auf Seite 129
Für bestimmte Operationen von Web-Services-APIs müssen Clientanwendungen eingeschlossene Elemente vom Typ "Dokument/Literal" verwenden. Clientanwendungen benötigen Helper-Klassen, die das Generieren der erforderlichen Wrapper-Elemente unterstützen.

Dateien von der Client-CD verwenden

Als Alternative zum Exportieren von Artefakten aus der WebSphere-Serverumgebung können Sie die erforderlichen Dateien zum Generieren einer Clientanwendung von der WebSphere Process Server-Client-CD kopieren.

In diesem Fall müssen Sie die Endpunktadresse für die Standard-Web-Services der Business Flow Manager-API oder der Human Task Manager-API manuell ändern.

Wenn die Clientanwendung auf beide APIs zugreifen soll, müssen Sie die Standardendpunktadressen für beide APIs bearbeiten.

Dateien von der Client-CD kopieren:

Die erforderlichen Dateien für den Zugriff auf die Web-Services-API sind auf der WebSphere Process Server-Client-CD verfügbar.

1. Legen Sie die Client-CD ein, und öffnen Sie das Verzeichnis ProcessChoreographer\client.
2. Kopieren Sie die erforderlichen Dateien in Ihre Entwicklungsumgebung für Clientanwendungen.

   Kopieren Sie für die Business Flow Manager-API folgende Dateien:

   BFMWS.wsdl
   Beschreibt die in der Web-Services-API von Business Flow Manager verfügbaren Web-Services. Diese Datei enthält die Endpunktadresse.

   BFMIF.wsdl
   Beschreibt die Parameter und Datentypen der einzelnen Web-Services in der Web-Services-API von Business Flow Manager.

   BFMIF.xsd
   Beschreibt die in der Web-Services-API von Business Flow Manager verwendeten Datentypen.

   BPCGEN.xsd
   Enthält die in den Web-Services-APIs von Business Flow Manager und Human Task Manager gemeinsam genutzten Datentypen.

   Kopieren Sie für die Human Task Manager-API folgende Dateien:

   HTMWS.wsdl
   Beschreibt die in der Web-Services-API von Human Task Manager verfügbaren Web-Services. Diese Datei enthält die Endpunktadresse.

   HTMIF.wsdl
   Beschreibt die Parameter und Datentypen der einzelnen Web-Services in der Web-Services-API von Human Task Manager.

   HTMIF.xsd
   Beschreibt die in der Web-Services-API von Human Task Manager verwendeten Datentypen.

BPCGEN.xsd
Enthält die in den Web-Services-APIs von Business Flow Manager und Human Task Manager gemeinsam genutzten Datentypen.

Anmerkung: Die Datei BPCGen.xsd wird von beiden APIs gemeinsam benutzt.

Nach dem Kopieren der Dateien müssen Sie die Endpunktadresse der Web-Services-API in den Dateien BFMWS.wsdl oder HTMWS.wsdl manuell in die Adresse des WebSphere-Anwendungsservers ändern, der als Host für die Web-Services-APIs fungiert.

Web-Service-Endpunktadresse manuell ändern:

Wenn Sie Dateien von der Client-CD kopieren, müssen Sie in den WSDL-Dateien als Web-Service-Standardendpunktadresse die Adresse des Servers angeben, der als Host für die Web-Services-APIs fungiert.

Mit der Administrationskonsole können Sie die Web-Service-Endpunktadresse festlegen, bevor Sie die WSDL-Dateien exportieren. Wenn Sie die WSDL-Dateien jedoch von der WebSphere Process Server-Client-CD kopieren, müssen Sie die Web-Service-Standardendpunktadresse manuell ändern.

Welche Web-Service-Endpunktadresse verwendet werden sollte, hängt von Ihrer WebSphere-Serverkonfiguration ab:

- Szenario 1. Ein einzelner WebSphere-Server. Als WebSphere-Endpunktadresse sind Hostname und Portnummer des Servers anzugeben (z. B. **host1:9080**).
- Szenario 2. Ein aus mehreren Servern bestehender WebSphere-Cluster. Als WebSphere-Endpunktadresse sind Hostname und Port des Servers anzugeben, der als Host für die Web-Services-APIs fungiert (z. B. **host2:9081**).
- Szenario 3. Ein Web-Server wird als Front-End verwendet. Als WebSphere-Endpunktadresse sind Hostname und Port des Web-Servers anzugeben (z. B. **host:80**).

Endpunkt der Business Flow Manager-API ändern:

Beim Kopieren der Business Flow Manager-API-Dateien von der WebSphere Process Server-Client-CD müssen Sie die Standardendpunktadresse manuell ändern.

1. Navigieren Sie zu dem Verzeichnis, das die von der Client-CD kopierten Dateien enthält.
2. Öffnen Sie die Datei BFMWS.wsdl in einem Texteditor oder XML-Editor.
3. Suchen Sie das Element `soap:address` (gegen Ende der Datei).
4. Geben Sie als Wert des Attributs `location` die HTTP-URL-Adresse des Servers an, auf dem die Web-Service-API ausgeführt wird. Gehen Sie dabei wie folgt vor:
   a. (Optional) Ersetzen Sie `http` durch `https`, um das sicherere HTTPS-Protokoll zu verwenden.
   b. Ersetzen Sie *localhost* durch den Hostnamen oder die IP-Adresse des Serverendpunkts der Web Services-API.
   c. Ersetzen Sie *9080* durch die Portnummer des Anwendungsservers.
   d. Ersetzen Sie *BPEContainer_N1_server1* durch den Stammkontext der Anwendung, in der die Web-Services-API ausgeführt wird. Der Standardwert für den Stammkontext besteht aus folgenden Elementen:
      - *BPEContainer*. Der Name der Anwendung.

- *N1*. Der Knotenname.
- *server1*. Der Servername.

e. Lassen Sie den festen Teil der URL-Adresse unverändert (`/sca/com/ibm/bpe/api/BFMWS`) .

Wenn die Anwendung beispielsweise auf dem Server **s1.n1.ibm.com** ausgeführt wird, und der Server SOAP/HTTP-Anforderungen am Port **9080** empfängt, ändern Sie das Element `soap:address` wie folgt:

```
<soap:address location="http://si.n1.ibm.com:9080/
                  BPEContainer_N1_server1/sca/com/ibm/bpe/api/BFMWS"/>
```

Zugehörige Konzepte

„Sicherheit hinzufügen (Java-Web-Services)" auf Seite 124
Die Web-Service-Kommunikation muss durch Implementieren von Sicherheitsmechanismen in Ihrer Clientanwendung geschützt werden.

Zugehörige Tasks

„Sicherheit hinzufügen (.NET)" auf Seite 132
Die Web-Service-Kommunikation kann durch Implementieren von Sicherheitsmechanismen in Ihrer Clientanwendung geschützt werden.

Endpunkt der Human Task Manager-API ändern:

Beim Kopieren der Human Task Manager-API-Dateien von der WebSphere Process Server-Client-CD müssen Sie die Standardendpunktadresse manuell bearbeiten.

1. Navigieren Sie zu dem Verzeichnis, das die von der Client-CD kopierten Dateien enthält.
2. Öffnen Sie die Datei HTMWS.wsdl in einem Texteditor oder XML-Editor.
3. Suchen Sie das Element `soap:address` (gegen Ende der Datei).
4. Geben Sie als Wert des Attributs `location` die korrekte Endpunktadresse ein. Gehen Sie dabei wie folgt vor:
   a. (Optional) Ersetzen Sie `http` durch `https`, um das sicherere HTTPS-Protokoll zu verwenden.
   b. Ersetzen Sie *localhost* durch den Hostnamen oder die IP-Adresse des Serverendpunkts der Web-Services-API.
   c. Ersetzen Sie *9080* durch die Portnummer des Anwendungsservers.
   d. Ersetzen Sie *HTMContainer_N1_server1* durch den Stammkontext der Anwendung, in der die Web-Services-API ausgeführt wird. Der Standardwert für den Stammkontext besteht aus folgenden Elementen:
      - *HTMContainer*. Der Name der Anwendung.
      - *N1*. Der Knotenname.
      - *server1*. Der Servername.
   e. Lassen Sie den festen Teil der URL-Adresse unverändert (`/sca/com/ibm/task/api/HTMWS`).

Wenn die Anwendung beispielsweise auf dem Server **s1.n1.ibm.com** ausgeführt wird, und der Server empfängt SOAP/HTTPS-Anforderungen am Port **9081**, ändern Sie das Element `soap:address` wie folgt:

```
<soap:address location="https://si.n1.ibm.com:9081/
                  HTMContainer_N1_server1/sca/com/ibm/task/api/HTMWS"/>
```

Zugehörige Konzepte

„Sicherheit hinzufügen (Java-Web-Services)" auf Seite 124
Die Web-Service-Kommunikation muss durch Implementieren von Sicherheitsmechanismen in Ihrer Clientanwendung geschützt werden.

Zugehörige Tasks

„Sicherheit hinzufügen (.NET)" auf Seite 132
Die Web-Service-Kommunikation kann durch Implementieren von Sicherheitsmechanismen in Ihrer Clientanwendung geschützt werden.

Clientanwendungen in der Java Web-Services-Umgebung entwickeln

Sie können jede Java-basierte Entwicklungsumgebung verwenden, die mit Java-Web-Services kompatibel ist, um Clientanwendungen für die Web-Services-APIs zu entwickeln.

Proxy-Client generieren (Java-Web-Services)

Java-Web-Service-Clientanwendungen verwenden einen *Proxy-Client* für die Interaktion mit den Web-Services-APIs.

Ein Proxy-Client für Java-Web-Services enthält eine Reihe von Java-Bean-Klassen, die von der Clientanwendung aufgerufen werden, um Web-Service-Anforderungen auszuführen. Der Proxy-Client stellt Serviceparameter zu SOAP-Nachrichten zusammen, sendet SOAP-Nachrichten über HTTP zum Web-Service, empfängt Antworten vom Web-Service und leitet die zurückgegebenen Daten an die Clientanwendung weiter.

Im Grunde ermöglicht ein Proxy-Client einer Clientanwendung, einen Web-Service wie eine lokale Funktion aufzurufen.

Anmerkung: Ein Proxy-Client muss nur einmal generiert werden. Danach können alle Clientanwendungen, die auf die Web-Services-API zugreifen, denselben Proxy-Client verwenden.

In der IBM Web-Services-Umgebung gibt es zwei Methoden zum Generieren eines Proxy-Clients:
- Mit der integrierte Entwicklungsumgebung von Rational Application Developer oder WebSphere Integration Developer
- Mit dem Befehlszeilentool WSDL2Java

Andere Entwicklungsumgebungen für Java-Web-Services beinhalten in der Regel entweder das Tool WSDL2Java oder eigene Generierungseinrichtungen für Clientanwendungen.

Proxy-Client unter Verwendung von Rational Application Developer generieren:

Die integrierte Entwicklungsumgebung von Rational Application Developer ermöglicht das Generieren eines Proxy-Clients für Ihre Clientanwendung.

Vor dem Generieren eines Proxy-Clients müssen Sie die WSDL-Dateien, die die Web-Services-Schnittstellen für Business-Prozesse oder Benutzertasks beschreiben, aus der WebSphere-Umgebung exportiert (oder von der WebSphere Process Server-Client-CD extrahiert) und in Ihre Clientprogrammierumgebung kopiert haben.

1. Fügen Sie die entsprechende WSDL-Datei Ihrem Projekt hinzu:
   - Für Business-Prozesse:
     a. Dekomprimieren Sie die exportierte Datei BPEContainer_*knotenname_servername*_WSDLFiles.zip in ein temporäres Verzeichnis.

b. Importieren Sie das Unterverzeichnis META-INF des dekomprimierten Verzeichnisses BPEContainer_*knotenname_servername*.ear/b.jar.

- Für Benutzertasks:
  a. Dekomprimieren Sie die exportierte Datei TaskContainer_*knotenname_servername*_WSDLFiles.zip in ein temporäres Verzeichnis.
  b. Importieren Sie das Unterverzeichnis META-INF des dekomprimierten Verzeichnisses TaskContainer_*knotenname_servername*.ear/h.jar.

In Ihrem Projekt werden ein neues WSDL-Verzeichnis (wsdl) sowie eine Unterverzeichnisstruktur erstellt.

2. Ändern Sie die Merkmale des Web-Service-Assistenten wie folgt:
   a. Wählen Sie in Rational Application Developer **Benutzervorgaben** → **Web-Services** → **Codegenerierung** → **IBM WebSphere-Laufzeit** aus.
   b. Wählen Sie die Option **Java aus WSDL unter Verwendung des Styles "no wrapped" generieren** aus.

   Anmerkung: Wenn Sie die Option **Web-Services** im Menü **Benutzervorgaben** nicht auswählen können, müssen Sie zuerst die erforderliche Funktionalität wie folgt aktivieren: **Fenster** → **Benutzervorgaben** → **Workbench** → **Leistungsmerkmale**. Klicken Sie auf **Web Service Developer** und auf **OK**. Öffnen Sie anschließend das Fenster Benutzervorgaben erneut, und ändern Sie die Option **Codegenerierung**.
3. Wählen Sie die Datei BFMWS.WSDL bzw. HTMWS.WSDL aus, die sich in dem neu erstellten Verzeichnis wsdl befindet.
4. Klicken Sie mit der rechten Maustaste, und wählen Sie **Web-Services** → **Client generieren** aus.

   Bevor Sie mit den verbleibenden Schritten fortfahren, müssen Sie sicherstellen, dass der Server gestartet wurde.
5. Klicken Sie im Fenster Web-Services auf **Weiter**, um alle Standardwerte zu übernehmen.
6. Klicken Sie im Fenster Web-Service-Auswahl auf **Weiter**, um alle Standardwerte zu übernehmen.
7. Führen Sie im Fenster Konfiguration der Clientumgebung die folgenden Schritte aus:
   a. Klicken Sie auf **Bearbeiten**, und ändern Sie die Option Web-Service-Laufzeit in IBM WebSphere.
   b. Ändern Sie die Option J2EE-Version in 1.4.
   c. Klicken Sie auf **OK**.
   d. Klicken Sie auf **Weiter**.
8. Dieser Schritt ist nur dann notwendig, wenn Sie einen Web-Services-Client generieren müssen, der sowohl Web-Services-APIs für Business-Prozesse als auch für Benutzertasks umfasst, da in beiden WSDL-Dateien doppelt vorhandene Methoden enthalten sind.
   a. Wählen Sie im Fenster für den Web-Service-Proxy die Option Benutzerdefinierte Zuordnung fnr Namensbereich-zu-Paket festlegen aus, und klicken Sie anschließend auf **OK**.
   b. Fügen Sie im Fenster "Namensbereich-zu-Paket-Zuordnung fnr Web-Service-Client" die folgenden Namespaces und das folgende Paket hinzu:

Für BFMWS.wsdl:

Namespace	Paket
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

Für HTMWS.wsdl:

Namespace	Paket
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

Klicken Sie auf **Ja, alle**, wenn Sie zur Bestätigung des Überschreibungsvorgangs aufgefordert werden.

9. Klicken Sie auf **Fertig stellen**.

Ein Proxy-Client, bestehend aus einer Reihe von Proxy-, Locator- und Helper-Java-Klassen, wird generiert und Ihrem Projekt hinzugefügt. Außerdem wird der Implementierungsdeskriptor aktualisiert.

Proxy-Client unter Verwendung von WSDL2Java generieren:

WSDL2Java ist ein Befehlszeilentool, das einen Proxy-Client generiert. Ein Proxy-Client erleichtert des Programmieren von Clientanwendungen.

Vor dem Generieren eines Proxy-Clients müssen Sie die WSDL-Dateien, die die Web-Services-APIs für Business-Prozesse oder Benutzertasks beschreiben, aus der WebSphere-Umgebung exportiert (oder von der WebSphere Process Server-Client-CD extrahiert) und in Ihre Clientprogrammierumgebung kopiert haben.

1. Generieren Sie mit dem Tool WSDL2Java einen Proxy-Client. Geben Sie Folgendes ein: **wsdl2java** *optionen WSDLdateipfad*

   Dabei gilt Folgendes:

   - Zu den *optionen* gehören:

     -noWrappedOperations (-w)
     Inaktiviert die Erkennung eingeschlossener Operationen. Java-Beans für Anforderungs- und Antwortnachrichten werden generiert. **Anmerkung:** Dies ist nicht die Standardeinstellung.

     -role (-r)
     Geben Sie den Wert **client** an, um Dateien und Bindungsdateien für clientseitige Entwicklung zu generieren.

-container (-c)
Der clientseitig zu verwendende Container. Zu den gültigen Argumenten gehören:
client Ein Client-Container
ejb Ein EJB-Container (EJB = Enterprise JavaBeans)
none Kein Container
web Ein Web-Container

-output (-o)
Der Ordner zum Speichern der generierten Dateien.

Eine vollständige Liste der Parameter für WSDL2Java erhalten Sie mit dem Befehlszeilenschalter **-help** oder in der Onlinehilfe für das Tool WSDL2Java in WID/RAD.

- *WSDLdateipfad* ist der Pfad mit Dateiname der WSDL-Datei, die Sie aus der WebSphere-Umgebung exportiert oder von der Client-CD kopiert haben.

Das folgende Beispiel generiert einen Proxy-Client für die Web-Services-API von Benutzertaskaktivitäten:

```
call wsdl2java.bat -r client -c client -noWrappedOperations
                   -output c:\ws\proxyClient  c:\ws\bin\HTMWS.wsdl
```

2. Nehmen Sie die generierten Klassendateien in Ihr Projekt auf.

Helper-Klassen für BPEL-Prozesse erstellen (Java-Web-Services)

Clientanwendungen benötigen für die Verarbeitung von Business-Objekten, auf die in konkreten API-Anforderungen verwiesen wird (z. B. sendMessage oder call) eingeschlossene Elemente vom Typ "Dokument/Literal". Die Clientanwendungen benötigen außerdem Helper-Klassen, die das Generieren der erforderlichen Wrapper-Elemente unterstützen.

Vor dem Erstellen von Helper-Klassen muss die WSDL-Datei der Web-Services-API aus der WebSphere Process Server-Umgebung exportiert werden.

Die Operationen call() und sendMessage() der Web-Services-APIs ermöglichen die Interaktion mit BPEL-Prozessen unter WebSphere Process Server. Die Eingabenachricht der Operation call() erwartet die Bereitstellung des Dokument/Literal-Wrappers der Prozesseingabenachricht.

Das Generieren der Helper-Klassen für einen BPEL-Prozess oder eine BPEL-Benutzertask kann mit verschiedenen Verfahren erfolgen, einschließlich der folgenden:

1. Verwenden Sie das Objekt SoapElement.

   In der Rational Application Developer-Umgebung, die in WebSphere Integration Developer zur Verfügung steht, unterstützt die Web-Service-Steuerkomponente JAX-RPC 1.1. In JAX-RPC 1.1 erweitert das Objekt SoapElement ein DOM-Element (DOM = Document Object Model), d. h. SOAP-Nachrichten können mit der DOM-API erstellt, gelesen, geladen und gespeichert werden.

   Angenommen, die WSDL-Datei enthält folgende Eingabenachricht für einen Workflowprozess oder eine Benutzertask:

```
<xsd:element name="operation1">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="input1" nillable="true" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
```

Die WSDL-Datei wird erstellt, wenn Sie einen Prozess oder ein Benutzertaskmodul erstellen.

Geben Sie Folgendes ein, um die entsprechende SOAP-Nachricht in Ihrer Clientanwendung mithilfe DOM-API zu erstellen:

```
SOAPFactory soapfactoryinstance = SOAPFactory.newInstance();
SOAPElement soapmessage = soapfactoryinstance.createElement
                          ("operation1", namespaceprefix, interfaceURI);
SOAPElement inputelement = soapfactoryinstance.createElement("input1");
inputelement.addTextNode( message value);
soapmessage.addChildElement(outputelement);
```

Das folgende Beispiel zeigt, wie Eingabeparameter für die Operation sendMessage in Ihrer Clientanwendung erstellt werden:

```
SendMessage inWsend = new SendMessage();
inWsend.setProcessTemplateName(processtemplatename);
inWsend.setPortType(porttype);
inWsend.setOperation(operationname);
inWsend.set_any(soapmessage);
```

2. Verwenden Sie die WebSphere-Funktion für benutzerdefinierte Datenbindungen.

   Dieses Verfahren wird in den folgenden developerWorks-Artikeln beschrieben:

   - How to choose a custom mapping technology for Web services
   - Developing Web Services with EMF SDOs for complex XML schema

   Interoperabilität mit Mustern und Strategien für dokumentbasierte Web-Services

   Web Services-Unterstützung für Schema/WSDL(s) mit optionalen JAX-RPC 1.0/1.1-XML-Schematypen

Clientanwendung erstellen (Java-Web-Services)

Eine Clientanwendung sendet Anforderungen an und empfängt Antworten von den Web-Services-APIs. Durch die Verwendung eines Proxy-Clients zum Verwalten der Kommunikation und von Helper-Klassen zum Formatieren komplexer Datentypen kann eine Clientanwendung Web-Service-Methoden wie lokale Funktionen aufrufen.

Bevor Sie mit dem Erstellen einer Clientanwendung beginnen, generieren Sie den Proxy-Client und alle erforderlichen Helper-Klassen.

Clientanwendungen können Sie mit jedem Web-Services-kompatiblen Entwicklungstool (z. B. IBM Rational Application Developer, RAD) entwickeln. Sie können einen beliebigen Web-Services-Anwendungstyp erstellen, der die Web-Services-APIs aufruft.

1. Erstellen Sie ein neues Clientanwendungsprojekt.
2. Generieren Sie den Proxy-Client, und fügen Sie die Java-Helper-Klassen zu Ihrem Projekt hinzu.
3. Codieren Sie Ihre Clientanwendung.
4. Erstellen Sie das Projekt.
5. Führen Sie die Clientanwendung aus.

Das folgende Beispiel zeigt die Verwendung der Web-Service-API von Business Flow Manager.

```
// Proxy erstellen
       BFMIFProxy proxy = new BFMIFProxy();
 // Eingabedaten für die Operation vorbereiten
```

```
    GetProcessTemplate iW = new GetProcessTemplate();
    iW.setIdentifier(name_ihrer_prozessschablone);

// Operation aufrufen
    GetProcessTemplateResponse oW = proxy.getProcessTemplate(iW);

// Prozessausgabe der Operation
    ProcessTemplateType ptd = oW.getProcessTemplate();
    System.out.println("getName= " + ptd.getName());
    System.out.println("getPtid= " + ptd.getPtid());
```

Zugehörige Tasks

„Proxy-Client generieren (Java-Web-Services)" auf Seite 119
Java-Web-Service-Clientanwendungen verwenden einen *Proxy-Client* für die Interaktion mit den Web-Services-APIs.

„Helper-Klassen für BPEL-Prozesse erstellen (Java-Web-Services)" auf Seite 122
Clientanwendungen benötigen für die Verarbeitung von Business-Objekten, auf die in konkreten API-Anforderungen verwiesen wird (z. B. sendMessage oder call) eingeschlossene Elemente vom Typ "Dokument/Literal". Die Clientanwendungen benötigen außerdem Helper-Klassen, die das Generieren der erforderlichen Wrapper-Elemente unterstützen.

Sicherheit hinzufügen (Java-Web-Services)

Die Web-Service-Kommunikation muss durch Implementieren von Sicherheitsmechanismen in Ihrer Clientanwendung geschützt werden.

WebSphere Application Server unterstützt derzeit folgende Sicherheitsmechanismen für die Web-Services-APIs:

- Benutzernamenstoken
- LTPA (Lightweight Third Party Authentication)

Zugehörige Konzepte

Berechtigungsaufgabenbereiche für Benutzertasks
Welche Aktionen Sie für Benutzertasks ausführen können, ist von Ihrem Berechtigungsaufgabenbereich abhängig. Bei diesem Aufgabenbereich kann es sich um einen J2EE-Aufgabenbereich auf Systemebene oder um einen instanzbasierten Aufgabenbereich handeln.

Berechtigungsaufgabenbereiche für Business-Prozesse
Ein Aufgabenbereich ist eine Gruppe von Personen, die eine gemeinsame Berechtigungsstufe haben. Welche Aktionen Sie für Business-Prozesse ausführen können, ist von Ihrem Berechtigungsaufgabenbereich abhängig. Bei diesem Aufgabenbereich kann es sich um einen J2EE-Aufgabenbereich oder einen instanzbasierten Aufgabenbereich handeln.

Benutzernamenstoken implementieren:

Der Sicherheitsmechanismus 'Benutzernamenstoken' stellt Benutzername und Kennwort als Berechtigungsnachweise bereit.

Mit dem Sicherheitsmechanismus 'Benutzernamenstoken' erhalten Sie die Möglichkeit, verschiedene *Callback-Handler* zu implementieren. Je nach Auswahl ist Folgendes möglich:

- Sie werden bei jedem Ausführen der Clientanwendung aufgefordert, Benutzername und Kennwort anzugeben
- Benutzername und Kennwort werden in den Implementierungsdeskriptor geschrieben

In beiden Fällen muss der angegebene Benutzername und das angegebene Kennwort mit dem Benutzernamen und dem Kennwort eines berechtigten Aufgabenbereichs im entsprechenden Business-Prozesscontainer oder Benutzertaskcontainer übereinstimmen.

Benutzername und Kennwort sind in den Umschlag der Antwortnachricht eingebunden, d. h. sie werden in der SOAP-Nachricht im Klartext angezeigt. Darum wird unbedingt empfohlen, die Clientanwendung so zu konfigurieren, dass sie das Kommunikationsprotokoll HTTPS (HTTP über SSL) verwendet. Wenn dies der Fall ist, werden alle Kommunikationsvorgänge verschlüsselt. Das Kommunikationsprotokoll HTTPS können Sie beim Angeben der Endpunkt-URL-Adresse für die Web-Service-API auswählen.

So definieren Sie ein Benutzernamenstoken:

1. Erstellen Sie ein Sicherheitstoken:
   a. Öffnen Sie den **Implementierungseditor** Ihres Moduls.
   b. Klicken Sie auf die Registerkarte **WS-Erweiterung**.
   c. Unter **Serviceverweise** werden möglicherweise die folgenden Web-Service-Referenzen aufgeführt:
      - service/BFMWSService für Business-Prozesse
      - service/HTMWSService für Benutzertasks

      Was aufgeführt wird, ist davon abhängig, ob BFMWS.wsdl (für Business-Prozesse) und/oder HTMWS.wsdl (für Benutzertasks) bei der Generierung des Proxy-Clients hinzugefügt wurde(n).
   d. Beide Servicereferenzen:
      1) Wählen Sie eine der **Servicereferenzen** aus.
      2) Erweitern Sie den Abschnitt **Anforderung Generator-Konfiguration**.
      3) Erweitern Sie den Unterabschnitt **Sicherheitstoken**.
      4) Klicken Sie auf **Hinzufügen**. Daraufhin wird das Fenster Sicherheitstoken geöffnet.
      5) Geben Sie im Feld **Name** einen Namen für das neue Sicherheitstoken ein: **UserNameTokenBFM** oder **UserNameTokenHTM**.
      6) Wählen Sie in der Dropdown-Liste **Tokentyp** die Option **Benutzername** aus. (Das Feld **Lokaler Name** wird automatisch mit einem Standardwert gefüllt.)
      7) Lassen Sie das Feld **URI** leer. Für ein Benutzernamenstoken ist kein URI-Wert erforderlich.
      8) Klicken Sie auf **OK**.
2. Erstellen Sie einen Tokengenerator:
   a. Öffnen Sie den **Implementierungseditor** Ihres Moduls.
   b. Klicken Sie auf die Registerkarte **WS-Binding**.
   c. Unter **Serviceverweise** werden dieselben Web-Service-Referenzen aufgeführt wie im vorherigen Schritt:
      - service/BFMWSService für Business-Prozesse
      - service/HTMWSService für Benutzertasks
   d. Beide Servicereferenzen:
      1) Wählen Sie eine der **Servicereferenzen** aus.
      2) Erweitern Sie den Abschnitt **Binding-Konfiguration für das Generieren von Anforderungen**.

3) Erweitern Sie den Unterabschnitt **Tokengenerator**.
4) Klicken Sie auf **Hinzufügen**. Daraufhin wird das Fenster 'Tokengenerator' geöffnet.
5) Geben Sie im Feld **Name** einen Namen für den neuen Tokengenerator ein, z. B. "UserNameTokenGeneratorBFM" oder "UserNameTokenGeneratorHTM".
6) Stellen Sie im Feld **Klasse für Tokengenerator** sicher, dass die folgende Tokengeneratorklasse ausgewählt ist: **com.ibm.wsspi.wssecurity-.token.UsernameTokenGenerator**.
7) Wählen Sie in der Dropdown-Liste **Sicherheitstoken** das entsprechende Sicherheitstoken aus, das Sie zuvor erstellt haben.
8) Aktivieren Sie das Markierungsfeld **Werttyp verwenden**.
9) Wählen Sie **UsernameToken** im Feld **Werttyp** aus. (Das Feld **Lokaler Name** wird automatisch gefüllt, um Ihre Auswahl des Benutzernamentokens widerzugeben.)
10) Geben Sie in das Feld **Rückrufsteuerroutine** den Wert "com.ibm.wsspi.wssecurity.auth.callback.GUIPromptCallbackHandler" (damit beim Ausführen der Clientanwendung Benutzername und Kennwort abgefragt werden) oder den Wert "com.ibm.wsspi.wssecurity.auth.callback.NonPromptCallbackHandler" ein.
11) Wenn Sie **NonPromptCallbackHandler** verwenden, müssen Sie im entsprechenden Feld des Implementierungsdeskriptors einen gültigen Benutzernamen mit Kennwort angeben.
12) Klicken Sie auf **OK**.

Zugehörige Tasks

„Web-Service-Endpunktadresse angeben" auf Seite 111
Dies ist die URL-Adresse, die eine Clientanwendung angeben muss, um auf die Web-Services-APIs zuzugreifen. Die Endpunktadresse wird in die WSDL-Datei geschrieben, die Sie exportieren, um einen Proxy-Client für Ihre Clientanwendung zu generieren.

Zugehörige Informationen

IBM WebSphere Developer Technical Journal: Web services security with WebSphere Application Server V6

LTPA-Sicherheitsmechanismus implementieren:

Der LTPA-Sicherheitsmechanismus (LTPA = Lightweight Third Party Authentication) kann verwendet werden, wenn die Clientanwendung in einem zuvor eingerichteten Sicherheitskontext ausgeführt wird.

Der LTPA-Sicherheitsmechanismus ist nur verfügbar, wenn Ihre Clientanwendung in einer geschützten Umgebung ausgeführt wird, in der bereits ein Sicherheitskontext eingerichtet ist. Wenn Ihre Clientanwendung beispielsweise in einem EJB-Container (EJB = Enterprise JavaBeans) ausgeführt wird, muss sich der EJB-Client anmelden, damit er die Clientanwendung aufrufen kann. Anschließend wird ein Sicherheitskontext eingerichtet. Wenn die EJB-Clientanwendung danach einen Web-Service aufruft, ruft der LTPA-Callback-Handler das LTPA-Token aus dem Sicherheitskontext ab und fügt es zur SOAP-Anforderungsnachricht hinzu. Auf der Serverseite wird das LTPA-Token vom LPTA-Mechanismus verarbeitet.

Gehen Sie wie folgt vor, um den LTPA-Sicherheitsmechanismus zu implementieren:

1. Wählen Sie in der Rational Application Developer-Umgebung, die in WebSphere Integration Developer zur Verfügung steht, **WS-Binding → Binding-Konfiguration für das Generieren von Anforderungen → Tokengenerator** aus.
2. Erstellen Sie ein Sicherheitstoken:
   a. Öffnen Sie den **Implementierungseditor** Ihres Moduls.
   b. Klicken Sie auf die Registerkarte **WS-Erweiterung**.
   c. Unter **Serviceverweise** werden möglicherweise die folgenden **Web-Service-Referenzen** aufgeführt:
      - service/BFMWSService für Business-Prozesse
      - service/HTMWSService für Benutzertasks

      Was aufgeführt wird, ist davon abhängig, ob BFMWS.wsdl (für Business-Prozesse) und/oder HTMWS.wsdl (für Benutzertasks) bei der Generierung des Proxy-Clients hinzugefügt wurden.
   d. Beide Serviceverweise:
      1) Wählen Sie eine der **Servicereferenzen** aus.
      2) Erweitern Sie den Abschnitt **Anforderung Generator-Konfiguration**.
      3) Erweitern Sie den Unterabschnitt **Sicherheitstoken**.
      4) Klicken Sie auf **Hinzufügen**. Daraufhin wird das Fenster Sicherheitstoken geöffnet.
      5) Geben Sie im Feld **Name** einen Namen für das neue Sicherheitstoken ein: **LTPATokenBFM** oder **LTPATokenHTM**.
      6) Wählen Sie in der Dropdown-Liste **Tokentyp** den Namen **LTPAToken** aus. (Die Felder **URI** und **Lokaler Name** werden automatisch mit Standardwerten gefüllt.)
      7) Klicken Sie auf **OK**.
3. Erstellen Sie einen Tokengenerator:
   a. Öffnen Sie den **Implementierungseditor** Ihres Moduls.
   b. Klicken Sie auf die Registerkarte **WS-Binding**.
   c. Unter **Serviceverweise** werden dieselben Web-Service-Referenzen aufgeführt wie im vorherigen Schritt:
      - service/BFMWSService für Business-Prozesse
      - service/HTMWSService für Benutzertasks
   d. Beide Serviceverweise:
      1) Wählen Sie eine der **Servicereferenzen** aus.
      2) Erweitern Sie den Abschnitt **Binding-Konfiguration für das Generieren von Anforderungen**.
      3) Erweitern Sie den Unterabschnitt **Tokengenerator**.
      4) Klicken Sie auf **Hinzufügen**. Daraufhin wird das Fenster 'Tokengenerator' geöffnet.
      5) Geben Sie im Feld **Name** einen Namen für den neuen Tokengenerator ein, z. B. "LTPATokenGeneratorBFM" oder "LTPATokenGeneratorHTM".
      6) Stellen Sie im Feld **Klasse für Tokengenerator** sicher, dass die folgende Tokengeneratorklasse ausgewählt ist: **com.ibm.wsspi.wssecurity-.token.LTPATokenGenerator**.
      7) Wählen Sie in der Dropdown-Liste **Sicherheitstoken** das entsprechende Sicherheitstoken aus, das Sie zuvor erstellt haben.
      8) Aktivieren Sie das Markierungsfeld **Werttyp verwenden**.

9) Wählen Sie **LTPAToken** im Feld **Werttyp** aus. (Die Felder **URI** und **Lokaler Name** werden automatisch gefüllt, um Ihre Auswahl des LTPA-Tokens widerzugeben.)
10) Geben Sie in das Feld **Rückrufsteuerroutine** den Wert "com.ibm.ws-spi.wssecurity.auth.callback.LTPATokenCallbackHandler" ein.
11) Klicken Sie auf **OK**.

Zur Laufzeit ruft **LTPATokenCallbackHandler** das LTPA-Token aus dem vorhandenen Sicherheitskontext ab, und fügt es zur SOAP-Anforderungsnachricht hinzu.

Transaktionsunterstützung hinzufügen (Java Web-Services)

Java Web-Service-Clientanwendungen können so konfiguriert werden, dass serverseitige Anforderungsverarbeitung an der Transaktion des Clients teilnehmen kann, indem ein Clientanwendungskontext als Bestandteil der Serviceanforderung übergeben wird. Diese atomare Transaktionsunterstützung ist in der WS-AT-Spezifikation (WS-AT = Web Services-Atomic Transaction) definiert.

WebSphere Application Server führt jede Web-Services-API-Anforderung als separate atomare Transaktion aus. Clientanwendungen können auf eine der folgenden Arten für die Verwendung von Transaktionsunterstützung konfiguriert werden:

- Teilnahme an der Transaktion. Die serverseitige Anforderungsverarbeitung erfolgt im Kontext der Clientanwendungstransaktion. Stellt der Server während der Ausführung der Web-Services-API-Anforderung einen Fehler fest und führt ein Rollback durch, wird auch für die Clientanwendungsanforderung ein Rollback durchgeführt.
- Ohne Transaktionsunterstützung. WebSphere Application Server erstellt trotzdem eine neue Transaktion, in der die Anforderung ausgeführt werden soll, aber die serverseitige Anforderungsverarbeitung erfolgt ohne den Transaktionskontext der Clientanwendung.

Clientanwendungen in der .NET-Umgebung entwickeln

Microsoft .NET stellt eine leistungsfähige Entwicklungsumgebung zur Verfügung, in der Anwendungen durch Web-Services verbunden werden können.

Proxy-Client generieren (.NET)

.NET-Clientanwendungen verwenden einen *Proxy-Client* für die Interaktion mit den Web-Service-APIs. Ein Proxy-Client schirmt Clientanwendungen von den komplexen Abläufen des Web-Service-Nachrichtenübertragungsprotokolls ab.

Zum Erstellen eines Proxy-Clients müssen Sie zunächst eine Reihe von WSDL-Dateien aus der WebSphere-Umgebung exportieren und in Ihre Clientprogrammierumgebung kopieren.

Anmerkung: Wenn Sie über die WebSphere Process Server-Client-CD verfügen, können Sie die Dateien auch von dieser CD kopieren.

Ein Proxy-Client umfasst eine Reihe von C#-Bean-Klassen. Jede Klasse enthält alle von einem einzigen Web-Service zugänglich gemachten Methoden und Objekte. Die Servicemethoden ermöglichen das Zusammenstellen von Parametern zu vollständigen SOAP-Nachrichten, das Senden von SOAP-Nachrichten zum Web-Service mit HTTP, das Empfangen der Antworten von dem Web-Service und das Verarbeiten der zurückgegebenen Daten.

Anmerkung: Ein Proxy-Client muss nur einmal generiert werden. Danach können alle Clientanwendungen, die auf die Web-Service-APIs zugreifen, denselben Proxy-Client verwenden.

1. Generieren Sie einen Proxy-Client mit dem Befehl WSDL. Geben Sie Folgendes ein:

   wsdl *optionen WSDLdateipfad*

   Dabei gilt Folgendes:

   - Zu den *optionen* gehören:
     /language
     Ermöglicht das Angeben der Sprache, die zum Erstellen der Proxy-Klasse verwendet wird. Der Standardwert ist C#. Sie können auch **VB** (Visual Basic), **JS** (JScript) oder **VJS** (Visual J#) als Sprache angeben.
     /output
     Der Name der Ausgabedatei mit dem entsprechenden Suffix. Beispiel: proxy.cs
     /protocol
     Das in der Proxy-Klasse implementierte Protokoll. Die Standardeinstellung ist **SOAP**.

     Eine vollständige Liste der Parameter für **WSDL.exe** erhalten Sie mit dem Befehlszeilenschalter **/?** oder in der Onlinehilfe für das WSDL-Tool in Visual Studio.
   - *WSDLdateipfad* ist der Pfad mit Dateiname der WSDL-Datei, die Sie aus der WebSphere-Umgebung exportiert oder von der Client-CD kopiert haben.

   Das folgende Beispiel generiert einen Proxy-Client für die Web-Services-API von Human Task Manager:

   ```
   wsdl /language:cs /output:proxyclient.cs c:\ws\bin\HTMWS.wsdl
   ```

2. Kompilieren Sie den Proxy-Client als DLL-Datei (DLL = Dynamic Link Library).

Helper-Klassen für BPEL-Prozesse erstellen (.NET)

Für bestimmte Operationen von Web-Services-APIs müssen Clientanwendungen eingeschlossene Elemente vom Typ "Dokument/Literal" verwenden. Clientanwendungen benötigen Helper-Klassen, die das Generieren der erforderlichen Wrapper-Elemente unterstützen.

Vor dem Erstellen von Helper-Klassen muss die WSDL-Datei der Web-Services-API aus der WebSphere Process Server-Umgebung exportiert werden.

Die Operationen call() und sendMessage() der Web-Services-APIs bewirken das Starten von BPEL-Prozessen in WebSphere Process Server. Die Eingabenachricht der Operation call() erwartet die Bereitstellung des Dokument/Literal-Wrappers der Eingabenachricht des BPEL-Prozesses. Um die erforderlichen Beans und Klassen für den BPEL-Prozess zu generieren, kopieren Sie das Element `<wsdl:types>` in eine neue XSD-Datei, und generieren Sie anschließend Helper-Klassen mit dem Tool xsd.exe.

1. Falls dies noch nicht geschehen ist, exportieren Sie die WSDL-Datei der BPEL-Prozessinstanz aus WebSphere Integration Developer.
2. Öffnen Sie die WSDL-Datei in einem Texteditor oder XML-Editor.
3. Kopieren Sie den Inhalt aller untergeordneten Elemente des Elements `<wsdl:types>`, und fügen Sie ihn in eine neue XSD-Gerüstdatei ein.
4. Wenden Sie das Tool `xsd.exe` wie folgt auf die XSD-Datei an:

   call xsd.exe file.xsd /classes /o

Dabei gilt Folgendes:

file.xsd
Die umzuwandelnde XML-Schemadefinitionsdatei.

/classes (/c)
Helper-Klassen generieren, die dem Inhalt des bzw. der XSD-Datei(en) entsprechen.

/output (/o)
Das Ausgabeverzeichnis für generierte Dateien angeben. Wenn dieses Verzeichnis nicht angegeben ist, wird standardmäßig das aktuelle Verzeichnis verwendet.

Beispiel:

call xsd.exe ProcessCustomer.xsd /classes /output:c:\temp

5. Fügen Sie die generierte Klassendatei zu Ihrer Clientanwendung hinzu. Wenn Sie beispielsweise mit Visual Studio arbeiten, können Sie dazu die Menüoption **Project → Add Existing Item** verwenden.

Angenommen, die Datei ProcessCustomer.wsdl enthält Folgendes:

```
<?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>
```

Die resultierende XSD-Datei hat den folgenden Inhalt:

```
<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>
```

Zugehörige Informationen

Microsoft-Dokumentation für das Tool für XML-Schemadefinition (XSD.EXE)

Clientanwendung erstellen (.NET)

Eine Clientanwendung sendet Anforderungen an und empfängt Antworten von den Web-Services-APIs. Durch die Verwendung eines Proxy-Clients zum Verwalten der Kommunikation und von Helper-Klassen zum Formatieren komplexer Datentypen kann eine Clientanwendung Web-Service-Methoden wie lokale Funktionen aufrufen.

Bevor Sie mit dem Erstellen einer Clientanwendung beginnen, generieren Sie den Proxy-Client und alle erforderlichen Helper-Klassen.

.NET-Clientanwendungen könnnen Sie mit jedem .NET-kompatiblen Entwicklungstool (z. B. Visual Studio .NET) entwickeln. Sie können einen beliebigen .NET-Anwendungstyp erstellen, der die generischen Web-Service-APIs aufruft.

1. Erstellen Sie ein neues Clientanwendungsprojekt. Erstellen Sie beispielsweise eine **WinFX-Windows-Anwendung** mit Visual Studio.
2. Fügen Sie in den Projektoptionen einen Verweis auf die DLL-Datei (DLL = Dynamic Link Library) des Proxy-Clients hinzu. Fügen Sie alle Helper-Klassen zu Ihrem Projekt hinzu, die Business-Objektdefinitionen enthalten. In Visual Studio können Sie dazu beispielsweise die Option **Project → Add existing item** verwenden.
3. Erstellen Sie ein Proxy-Client-Objekt. Beispiel:

```
HTMClient.HTMReference.HumanTaskManagerComponent1Export_HumanTaskManagerHttpService service =
    new HTMClient.HTMReference.HumanTaskManagerComponent1Export_HumanTaskManagerHttpService();
```

4. Deklarieren Sie alle Business-Objektdatentypen, die in den zum oder vom Web-Service gesendeten Nachrichten verwendet werden sollen. Beispiel:

```
HTMClient.HTMReference.TKIID id = new HTMClient.HTMReference.TKIID();

ClipBG bg = new ClipBG();
Clip clip = new Clip();
```

5. Rufen Sie bestimmte Web-Service-Funktionen auf, und geben Sie alle erforderlichen Parameter an. Beispiel zum Erstellen und Starten einer Benutzertask:

```
HTMClient.HTMReference.createAndStartTask task =
   new HTMClient.HTMReference.createAndStartTask();
HTMClient.HTMReference.StartTask sTask = new HTMClient.HTMReference.StartTask();

sTask.taskName = "EinfacheTask";
sTask.taskNamespace = "http://meinProzess/com/acme/task";
sTask.inputMessage = bg;
task.inputTask = sTask;

id = service.createAndStartTask(task).outputTask;
```

6. Ferne Prozesse und Tasks werden durch persistente IDs identifiziert (z. B. `id` im vorherigen Schritt). Beispiel zum Beanspruchen einer zuvor erstellten Benutzertask:

```
HTMClient.HTMReference.claimTask claim = new HTMClient.HTMReference.claimTask();
claim.inputTask = id;
```

Zugehörige Tasks

„Proxy-Client generieren (.NET)" auf Seite 128
.NET-Clientanwendungen verwenden einen *Proxy-Client* für die Interaktion mit den Web-Service-APIs. Ein Proxy-Client schirmt Clientanwendungen von den komplexen Abläufen des Web-Service-Nachrichtenübertragungsprotokolls ab.

„Helper-Klassen für BPEL-Prozesse erstellen (.NET)" auf Seite 129
Für bestimmte Operationen von Web-Services-APIs müssen Clientanwendungen eingeschlossene Elemente vom Typ "Dokument/Literal" verwenden. Clientanwendungen benötigen Helper-Klassen, die das Generieren der erforderlichen Wrapper-Elemente unterstützen.

Sicherheit hinzufügen (.NET)

Die Web-Service-Kommunikation kann durch Implementieren von Sicherheitsmechanismen in Ihrer Clientanwendung geschützt werden.

Zu diesen Sicherheitsmechanismen können Benutzernamenstoken (Benutzername und Kennwort) oder benutzerdefinierte binäre und XML-basierte Sicherheitstoken gehören.

1. Laden Sie Web Services Enhancements (WSE) 2.0 SP3 for Microsoft .NET herunter und installieren Sie es. Dieses Produkt ist unter folgender Webadresse verfügbar:

   http://www.microsoft.com/downloads/details.aspx?familyid=1ba1f631-c3e7-420a-bc1e-ef18bab66122&displaylang=en

2. Ändern Sie den generierten Proxy-Client-Code wie nachfolgend angegeben.

   Ändern Sie

```
public class Export1_MyMicroflowHttpService : System.Web.Services.Protocols.SoapHttpClientProtocol {
```

   in

```
public class Export1_MyMicroflowHttpService : Microsoft.Web.Services2.WebServicesClientProtocol {
```

   Anmerkung: Diese Änderungen gehen verloren, wenn Sie den Proxy-Client durch Ausführen des Tools WSDL.exe erneut generieren.

3. Ändern Sie den Client-Anwendungscode, indem Sie am Anfang der Datei die folgenden Zeilen hinzufügen:

```
using System.Web.Services.Protocols;
using Microsoft.Web.Services2;
using Microsoft.Web.Services2.Security.Tokens;
...
```

4. Fügen Sie Code hinzu, der den gewünschten Sicherheitsmechanismus implementiert. Der folgende Code fügt beispielsweise Schutz durch Benutzername und Kennwort hinzu:

```
string user = "U1";
string pwd = "kennwort";
UsernameToken token =
   new UsernameToken(user, pwd, PasswordOption.SendPlainText);

me._proxy.RequestSoapContext.Security.Tokens.Clear();
me._proxy.RequestSoapContext.Security.Tokens.Add(token);
```

Abfragen für Business-Prozessobjekte und taskbezogene Objekte

Mit den Web-Services-APIs können Sie Business-Prozessobjekte und taskbezogene Objekte in der Business Process Choreographer-Datenbank abfragen, um bestimmte Merkmale dieser Objekte abzurufen.

Die Business Process Choreographer-Datenbank speichert Schablonendaten (Modelldaten) und Instanzdaten (Laufzeitdaten) zum Verwalten von Business-Prozessen und Tasks.

Über die Web-Services-APIs können Clientanwendungen Abfragen absetzen, um Informationen zu Business-Prozessen und Tasks aus der Datenbank abzurufen.

Durch eine Einzelabfrage können Clientanwendungen ein bestimmtes Merkmal eines Objekts abrufen. Häufig verwendete Abfragen können gespeichert werden. Diese gespeicherten Abfragen können von Ihrer Clientanwendung abgerufen und verwendet werden.

Abfragen für Business-Prozessobjekte und taskbezogene Objekte

Verwenden Sie die Abfrageschnittstelle der Service-API zum Abrufen von Informationen zu Business-Prozessen und Tasks.

Client-Anwendungen verwenden eine SQL-ähnliche Syntax für Abfragen in der Datenbank.

Beispiel für Java Web-Services

```
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);
```

Die aus der Datenbank abgerufenen Informationen werden über die Web-Services-APIs als *Abfrageergebnis* zurückgegeben.

Beispiel:

```
QueryResultSetType queryResultSet = queryResponse1.queryResultSet;
if (queryResultSet != null) {
   Console.WriteLine("--> QueryResultSetType");
   Console.WriteLine(" . size= " + queryResultSet.size);
   Console.WriteLine(" . numberColumns= " + queryResultSet.numberColumns);
   string indent = " . ";

   // -- Informationen zu Abfragespalten
   QueryColumnInfoType[] queryColumnInfo = queryResultSet.QueryColumnInfo;
   if (queryColumnInfo.Length > 0) {
```

```
      Console.WriteLine();
      Console.WriteLine("= . QueryColumnInfoType size= " + queryColumnInfo.Length);
      Console.Write( " | tableName ");
      for (int i = 0; i < queryColumnInfo.Length ; i++) {
         Console.Write( " | " + queryColumnInfo[i].tableName.PadLeft(20) );
      }
      Console.WriteLine();
      Console.Write( " | columnName ");
      for (int i = 0; i < queryColumnInfo.Length ; i++) {
         Console.Write( " | " + queryColumnInfo[i].columnName.PadLeft(20) );
      }
      Console.WriteLine();
      Console.Write( " | data type ");
      for (int i = 0; i < queryColumnInfo.Length ; i++) {
         QueryColumnInfoTypeType tt = queryColumnInfo[i].type;
         Console.WriteLine( " | " + tt.ToString());
      }
         Console.WriteLine();
   }
   else {
      Console.WriteLine("--> queryColumnInfo= <null>");
   }

   // - Die Abfrageergebniswerte
   string[][] result = queryResultSet.result;
   if (result !=null) {
      Console.WriteLine();
      Console.WriteLine("= . result size= " + result.Length);
      for (int i = 0; i &lt; result.Length; i++) {
        Console.Write(indent +i );
        string[] row = result[i];
        for (int j = 0; j &lt; row.Length; j++ ) {
           Console.Write(" | " + row[j]);
        }
        Console.WriteLine();
      }
   }
   else {
      Console.WriteLine("--> result= <null>");
   }
}
else {
   Console.WriteLine("--> QueryResultSetType= <null>");
}
```

Die Abfragefunktion gibt entsprechend der Berechtigung des Aufrufenden Objekte zurück. Das Abfrageergebnis enthält nur die Merkmale derjenigen Objekte, für die der Aufrufende Lesezugriff hat.

Zum Abfragen der Objektmerkmale werden vordefinierte Datenbankansichten bereitgestellt. Für Prozessschablonen gilt in der Abfragefunktion die folgende Syntax:

```
ProcessTemplateData[] queryProcessTemplates
                      (java.lang.String whereClause,
                       java.lang.String orderByClause,
                       java.lang.Integer threshold,
                       java.util.TimeZone timezone);
```

Für Taskschablonen gilt in der Abfragefunktion die folgende Syntax:

```
TaskTemplate[] queryTaskTemplates
               (java.lang.String whereClause,
                java.lang.String orderByClause,
                java.lang.Integer threshold,
                java.util.TimeZone timezone);
```

Für die übrigen Business-Prozessobjekte und taskbezogenen Objekte gilt in der Abfragefunktion die folgende Syntax:

```
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);
```

Die Abfrageschnittstelle enthält außerdem eine Methode queryAll. Mit dieser Methoden können Sie alle relevanten Daten zu einem Objekt abrufen (z. B. für Überwachungszwecke). Der Aufrufende der Methode queryAll muss einem der folgenden J2EE-Aufgabenbereiche (J2EE = Java 2 Platform, Enterprise Edition) angehören: BPESystemAdministrator, BPESystemMonitor, TaskSystemAdministrator oder TaskSystemMonitor. Die Berechtigungsprüfung unter Verwendung des zugehörigen Arbeitselements des Objekts wird nicht angewendet.

Beispiel für .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("--> Keine Schablonen gefunden :-(");
        }
        else {
          for (int i = 0; i < templates.Length ; i++) {
            Console.Write("--> 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);
      }
```

Abfrageparameter:

Jede Abfrage muss eine Reihe SQL-ähnlicher Klauseln und Parameter angeben.

Eine Abfrage besteht aus folgenden Elementen:
- SELECT-Klausel
- WHERE-Klausel
- Sortierklausel
- Parameter 'skip-tuples'
- Grenzwertparameter
- Zeitzonenparameter

Vordefinierte Ansichten zum Abfragen von Business-Prozess- und Benutzertaskobjekten

Für Business-Prozess- und Benutzertaskobjekte werden vordefinierte Datenbankansichten bereitgestellt.

Verwenden Sie diese Ansichten zum Abfragen von Referenzdaten für diese Objekte. Bei Verwendung dieser Abfragen müssen Sie keine Verknüpfungsprädikate für Ansichtsspalten explizit hinzufügen. Diese Konstrukte werden automatisch hinzugefügt. Zum Abfragen dieser Daten können Sie die Abfragefunktion der Web-Services-APIs verwenden.

Gespeicherte Abfragen verwalten

Gespeicherte Abfragen bieten eine Möglichkeit zum Aufbewahren häufig verwendeter Abfragen. Die gespeicherte Abfrage kann eine für alle Benutzer verfügbare Abfrage sein (öffentliche Abfrage) oder eine Abfrage, die einem bestimmten Benutzer zugeordnet ist (private Abfrage).

Eine gespeicherte Abfrage ist eine Abfrage, die in der Datenbank gespeichert und mit einem Namen versehen ist. Eine private gespeicherte Abfrage kann denselben Namen wie eine öffentliche gespeicherte Abfrage haben. Private gespeicherte Abfragen von verschiedenen Eignern können ebenfalls denselben Namen haben.

Sie können Abfragen für Business-Prozessobjekte, Taskobjekte oder für eine Kombination dieser beiden Objekttypen speichern.

Öffentliche gespeicherte Abfragen verwalten
Öffentliche gespeicherte Abfragen werden vom Systemadministrator erstellt. Diese Abfragen stehen für alle Benutzer zur Verfügung.

Private gespeicherte Abfragen für andere Benutzer verwalten
Jeder Benutzer kann private Abfragen erstellen. Diese Abfragen stehen nur für den Eigner der Abfrage und für den Systemadministrator zur Verfügung.

Mit privaten gespeicherten Abfragen arbeiten
Wenn Sie kein Systemadministrator sind, können Sie eigene private gespeicherte Abfragen erstellen, ausführen und löschen. Außerdem können Sie die vom Systemadministrator erstellten öffentlichen gespeicherten Abfragen verwenden.

JMS-Clientanwendungen entwickeln

Sie können Clientanwendungen entwickeln, die über die JMS-APIs (JMS = Java Messaging Service) auf Business-Prozessanwendungen zugreifen.

JMS - Einführung

WebSphere Process Server Version 6.1 unterstützt die asynchrone Nachrichtenübermittlung auf der Basis der JMS-Programmierschnittstelle (JMS = Java Messaging Service) als Übertragungsmethode.

JMS stellt Java-Clients (Clientanwendungen oder J2EE-Anwendungen) eine einheitliche Methode zum Erstellen, Senden, Empfangen und Lesen von Anforderungen als JMS-Nachrichten bereit.

JMS ist eine asynchrone nachrichtenbasierte Schnittstelle, mit der Folgendes möglich ist:

- Sie verwendet das **Punkt-zu-Punkt- und Pub/Sub-Messaging**. Nachrichtenbasierte Frameworks können Informationen an andere Anwendungen

weitergeben, ohne dass diese die Informationen explizit anfordern. Dieselben Informationen können viele Subskribenten gleichzeitig bereitgestellt werden. Die JMS-Schnittstelle von Business Process Choreographer unterstützt nur das Punkt-zu-Punkt-Messaging.

- Sie ermöglicht eine **rhythmische Unabhängigkeit**. JMS-Frameworks arbeiten asynchron, können jedoch den synchronen Anforderungs-/Antwortmodus simulieren. Dadurch ist es möglich, dass Quellen- und Zielsysteme gleichzeitig arbeiten, ohne dass sie gegenseitig aufeinander warten müssen. Diese Funktionalität ist für Business Process Choreographer äußerst nützlich, da sie die Möglichkeit zur asynchronen Interaktion mit Business-Dauerprozessen bereitstellt.
- Sie unterstützt **Transaktionen**. Mithilfe von Transaktionen können Clientanwendungen Nachrichtengruppen verarbeiten, die als einzelne unteilbare Einheit gesendet oder empfangen wurden. JMS-Transaktionen werden innerhalb der Servertransaktion ausgeführt. Was die JMS-Schnittstelle von Business Process Choreographer betrifft, so senden und empfangen Sie in der Regel eine einzige Nachricht pro Transaktion.
- **Sie garantiert die Bereitstellung von Informationen**. Mit JMS-Frameworks können Nachrichten im Transaktionsmodus verwaltet werden; außerdem kann die Bereitstellung von Nachrichten sichergestellt werden (allerdings ohne Garantie auf eine rechtzeitige Bereitstellung). Für Business Process Choreographer ist diese zuverlässige Nachrichtenbereitstellung besonders wichtig, da mit Business-Prozessen gearbeitet wird.
- Sie stellt **Interoperabilität zwischen heterogenen Frameworks** sicher. Die Quellen- und Zielanwendungen können ohne Kommunikations- und Ausführungsprobleme mit den jeweiligen Frameworks in heterogenen Umgebungen arbeiten.
- **Sie sorgt für einen reibungsloseren Ablauf von Austauschvorgängen**. Das Wechseln in den Nachrichtenmodus ermöglicht den Austausch differenzierterer Informationen.

Anforderungen für Business-Prozesse

Business-Prozesse, die mit WebSphere Integration Developer für die Ausführung unter Business Process Choreographer entwickelt wurden, müssen bestimmte Regeln erfüllen, damit die JMS-APIs auf sie zugreifen können.

Dabei gelten folgende Voraussetzungen:

1. Die Schnittstellen von Business-Prozessen müssen unter Verwendung des Stils "document/literal wrapped" definiert werden, der in der Spezifikation JAX-RPC 1.1 (Java API for XML-based RPC) definiert ist. Dies ist der Standardstil für alle mit WebSphere Integration Developer entwickelten Business-Prozesse und Benutzertasks.
2. Fehlernachrichten, die von Business-Prozessen und Benutzertasks für Web-Service-Operationen ausgegeben werden, müssen aus einem einzigen WSDL-Nachrichtenteil bestehen, der mit einem XML-Schemaelement definiert ist. Beispiel:

   ```
   <wsdl:part name="meinFehler" element="meinNamensbereich:meinFehlerelement"/>
   ```

   Zugehörige Informationen

   Downloadseite der Java-API für XML-basierte RPC (JAX-RPC)

   Welcher WSDL-Stil sollte verwendet werden?

Auf die JMS-Schnittstelle zugreifen

Wenn Sie Nachrichten über die JMS-Schnittstelle senden und empfangen möchten, muss eine Anwendung zuerst eine Verbindung zum Bus BPC.zellenname.Bus herstellen, eine Sitzung erstellen und anschließend Nachrichtenproduzenten und -konsumenten generieren.

Der Process Server akzeptiert JMS-Nachrichten (JMS = Java Message Service), die auf dem Punkt-zu-Punkt-Konzept basieren. Eine Anwendung, die JMS-Nachrichten sendet oder empfängt muss die folgenden Aktionen durchführen.

Im folgenden Beispiel wird angenommen, dass der JMS-Client in einer verwalteten Umgebung (EJB, Anwendungsclient oder Web-Client-Container) ausgeführt wird. Wenn Sie den JMS-Client in einer J2SE-Umgebung ausführen möchten, ziehen Sie die Informationen unter ″IBM Client for JMS on J2SE with IBM WebSphere Application Server″ zu Rate, die Sie unter der folgenden Adresse finden: http://www-1.ibm.com/support/docview.wss?uid=swg24012804.

1. Erstellen Sie eine Verbindung zum Bus BPC.*zellenname*.Bus. Für Anforderungen einer Clientanwendung ist keine vorkonfigurierte Verbindungsfactory vorhanden: Eine Clientanwendung kann entweder die ReplyConnectionFactory der JMS-API verwenden oder ihre eigene Verbindungsfactory erstellen; in diesem Fall kann die JNDI-Suchfunktion zum Abrufen der Verbindungsfactory verwendet werden. Der Name der JNDI-Suchfunktion muss mit dem bei der Konfiguration der externen Anforderungswarteschlange des Business Process Choreographer angegebenen Namen identisch sein. Im folgenden Beispiel wird angenommen, dass die Clientanwendung ihre eigene Verbindungsfactory mit dem Namen ″jms/clientCF″ erstellt.

   ```
   //Fragen Sie den ursprünglichen JNDI-Standardkontext ab.
   Context initialContext = new InitialContext();

   // Suchen Sie die Verbindungsfactory.
   // Erstellen Sie eine Verbindungsfactory, die eine Verbindung zum BPC-Bus herstellt.
   // Nennen Sie sie z. B. "jms/clientCF".
   // Konfigurieren Sie außerdem einen entsprechenden Authentifizierungsaliasnamen.
   ConnectionFactory connectionFactory =
               (ConnectionFactory)initialcontext.lookup("jms/clientCF");

   // Erstellen Sie die Verbindung.
   Connection connection = connectionFactory.createConnection();
   ```

2. Erstellen Sie eine Sitzung, damit Nachrichtenproduzenten und -konsumenten erstellt werden können.

   ```
   // Erstellen Sie mithilfe der Funktion für eine automatische Bestätigung eine
      Transaktionssitzung.
   Session session = connection.createSession(true, Session.AUTO_ACKNOWLEDGE);
   ```

3. Erstellen Sie zum Senden von Nachrichten einen Nachrichtenproduzenten. Der Name der JNDI-Suchfunktion muss mit dem bei der Konfiguration der externen Anforderungswarteschlange des Business Process Choreographer angegebenen Namen identisch sein.

   ```
   // Suchen Sie nach dem Ziel der Business Process Choreographer-Eingabewarteschlange,
   // um Nachrichten dorthin zu senden.
   Queue sendQueue = (Queue) initialcontext.lookup("jms/BFMJMSAPIQueue");

   // Erstellen Sie einen Nachrichtenproduzenten.
   MessageProducer producer = session.createProducer(sendQueue);
   ```

4. Erstellen Sie für den Empfang von Antworten einen Nachrichtenkonsumenten. Der Name der JNDI-Suchfunktion des Antwortziels kann ein benutzerdefiniertes Ziel angeben, aber er kann auch das Standardantwortziel (vom Business

Process Choreographer definiert) angeben (jms/BFMJMSReplyQueue). In beiden Fällen muss sich das Antwortziel im Bus BPC.<zellenname>.Bus befinden.

```
// Suchen Sie nach dem Ziel der Antwortwarteschlange.
Queue  replyQueue = (Queue) initialcontext.lookup("jms/BFMJMSReplyQueue");

// Erstellen Sie einen Nachrichtenkonsumenten.
MessageConsumer consumer = session.createConsumer(replyQueue);
```

5. Senden Sie eine Nachricht.

```
// Starten Sie die Verbindung.
connection.start();

// Erstellen Sie eine Nachricht, und senden Sie sie; Beispiele finden Sie in
   den Taskbeschreibungen.
// Diese Methode wird anderweitig definiert ...
String payload = createXMLDocumentForRequest();
TextMessage requestMessage = session.createTextMessage(payload);

// Definieren Sie einen obligatorischen JMS-Header.
// targetFunctionName ist der Operationsname der JMS-API
// (z. B. getProcessTemplate, sendMessage)
requestMessage.setStringProperty("TargetFunctionName", targetFunctionName);

// Definieren Sie die Antwortwarteschlange; diese ist obligatorisch, wenn replyQueue
// nicht die Standardwarteschlange ist (wie in diesem Beispiel).
requestMessage.setJMSReplyTo(replyQueue);

// Senden Sie die Nachricht.
producer.send(requestMessage);

// Rufen Sie die Nachrichten-ID ab.
String jmsMessageID = requestMessage.getJMSMessageID();

session.commit();
```

6. Empfangen Sie die Antwort.

```
// Empfangen Sie die Antwortnachricht, und analysieren Sie die Antwort.
TextMessage  replyMessage = (TextMessage) consumer.receive();

// Rufen Sie die Nutzdaten ab.
String payload = replyMessage.getText();

session.commit();
```

7. Schließen Sie die Verbindung, und geben Sie die Ressourcen frei.

```
 // Finale Arbeiten; geben Sie die Ressourcen frei.
session.close();
connection.close();
```

Anmerkung: Es ist nicht notwendig, die Verbindung nach jeder Transaktion zu schließen. Wenn eine Verbindung erst einmal gestartet wurde, kann eine beliebige Anzahl von Anforderungs- und Antwortnachrichten ausgetauscht werden, bevor die Verbindung geschlossen wird. In dem Beispiel wird ein einfacher Fall mit einem einzigen Aufruf in einer einzigen Business-Methode dargestellt.

Struktur einer Business Process Choreographer-JMS-Nachricht

Der Header und der Hauptteil jeder JMS-Nachricht (JMS = Java Message Service) muss eine vordefinierte Struktur aufweisen.

Eine JMS-Nachricht besteht aus folgenden Teilen:

- Einem Nachrichtenheader zur Nachrichtenidentifikation und für Route-Informationen.
- Dem Hauptteil (Nutzdaten) der Nachricht mit dem Inhalt.

Business Process Choreographer unterstützt nur Textnachrichtenformate.

Nachrichtenheader

Mit JMS können Clients auf eine Reihe von Nachrichtenheaderfeldern zugreifen.

Die folgenden Headerfelder können von einem Business Process Choreographer-JMS-Client definiert werden:

- **JMSReplyTo**

  Das Ziel, an das eine Antwort auf die Anforderung gesendet werden soll. Wenn dieses Feld nicht in der Anforderungsnachricht angegeben ist, wird die Antwort an das Standardziel für Antworten der Exportschnittstelle gesendet (ein Export ist die Wiedergabe einer Clientschnittstelle einer Business-Prozesskomponente). Das Ziel kann mithilfe von `initialContext.lookup("jms/BFMJMSReplyQueue");` abgerufen werden.
- **TargetFunctionName**

  Der Name der WSDL-Operation, z. B. "queryProcessTemplates". Dieses Feld muss immer definiert sein. Beachten Sie, dass mit TargetFunctionName die Operation der hier beschriebenen generischen JMS-Nachrichtenschnittstelle angegeben wird. Diese sollte nicht mit Operationen verwechselt werden, die von konkreten Prozessen oder Tasks bereitgestellt werden, die indirekt mit beispielsweise **call**- oder **sendMessage**-Operationen aufgerufen werden können.

Ein Business Process Choreographer-Client hat außerdem Zugriff auf die folgenden Headerfelder:

- **JMSMessageID**

  Dadurch wird eine Nachricht eindeutig identifiziert. Sie wird beim Senden der Nachricht vom JMS-Provider gesetzt. Wenn der Client 'JMSMessageID' vor dem Senden der Nachricht setzt, wird sie vom JMS-Provider überschrieben. Wenn die ID der Nachricht für die Authentifizierung erforderlich ist, kann der Client 'JMSMessageID' nach dem Senden der Nachricht abrufen.
- **JMSCorrelationID**

  Dadurch werden Nachrichten verknüpft. Definieren Sie dieses Feld nicht. Eine Business Process Choreographer-Antwortnachricht enthält die JMSMessageID der Anforderungsnachricht.

In jeder Antwortnachricht sind die folgenden JMS-Headerfelder enthalten:

- **IsBusinessException**

  "False" für WSDL-Ausgabenachrichten bzw. "true" für WSDL-Fehlernachrichten.

Laufzeitausnahmebedingungen (ServiceRuntimeExceptions) werden nicht an asynchrone Clientanwendungen zurückgegeben. Wenn bei der Verarbeitung einer JMS-Anforderungsnachricht eine schwer wiegenden Ausnahmebedingung auftritt, kommt es zu einem Laufzeitfehler, wodurch die Transaktion, die die Anforderungsnachricht verarbeitet, rückgängig gemacht wird. Die JMS-Anforderungsnachricht wird anschließend erneut zugestellt. Wenn der Fehler bei der Verarbeitung der Nachricht im Rahmen des SCA-Exports frühzeitig auftritt (z. B. bei der Entserialisierung der Nachricht), werden bis zu maximal der Anzahl der fehlgeschlagenen Zustellungen Wiederholungen durchgeführt, die das Empfangsziel des SCA-Exports angegeben werden. Wenn die maximale Anzahl der fehlgeschlagenen

Zustellungsversuche erreicht ist, wird die Anforderungsnachricht zum Ziel der Systemausnahme des Business Process Choreographer-Busses hinzugefügt. Wenn der Fehler jedoch während der aktuellen Verarbeitung der Anforderung durch die SCA-Komponente des Business Flow Managers auftritt, wird die fehlgeschlagene Anforderungsnachricht von der WebSphere Process Server-Managementinfrastruktur für fehlgeschlagene Ereignisse verarbeitet, d. h., möglicherweise kommt er in die Datenbank des Managements für fehlgeschlagene Ereignisse, wenn durch wiederholte Durchführung des Vorgangs die Ausnahmesituation nicht behoben werden kann.

Nachrichtenhauptteil

Der JMS-Nachrichtenhauptteil ist eine Zeichenfolge mit einem XML-Dokument, wodurch das Dokument/Literal-Wrapperelement der Operation darstellt.

Im Folgenden sehen Sie ein einfaches Beispiel für einen gültigem Anforderungsnachrichtenhauptteil:

```
<?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>
```

Berechtigung für JMS-Wiedergaben

Für die Berechtigung zur Verwendung der JMS-Schnittstelle müssen Sicherheitseinstellungen in WebSphere Application Server aktiviert sein.

Nach der Installation des Business Process Container muss der Aufgabenbereich **JMSAPIUser** einer Benutzer-ID zugeordnet werden. Diese Benutzer-ID wird für alle JMS-API-Anforderungen abgesetzt. Beispiel: Wenn **JMSAPIUser** dem Benutzer A zugeordnet wird, werden alle JMS-API-Anforderungen in der Prozessengine als vom Benutzer A stammend angezeigt.

Dem Aufgabenbereich **JMSAPIUser** müssen die folgenden Berechtigungen zugeordnet werden:

Anforderung	Erforderliche Berechtigung
forceTerminate	Prozessadministrator
sendEvent	Potenzieller Aktivitätseigner oder Prozessadministrator

Anmerkung: Für alle anderen Anforderungen sind keine Sonderberechtigungen erforderlich.

Sonderberechtigungen werden einer Person mit dem Aufgabenbereich eines Business-Prozessadministrators erteilt. Ein Business-Prozessadministrator ist ein besonderer Aufgabenbereich; er unterscheidet sich vom Prozessadministrator einer Prozessinstanz. Ein Business-Prozessadministrator verfügt über alle Berechtigungen.

Es ist nicht möglich, die Benutzer-ID des Prozess-Starters aus Ihrer Benutzerregistry zu löschen, solange die Prozessinstanz vorhanden ist. Wenn Sie diese Benutzer-ID löschen, ist kein weiteres Navigieren in diesem Prozess möglich. In der Systemprotokolldatei wird die folgende Ausnahmebedingung gemeldet:

```
Keine eindeutige ID für: <Benutzer-ID>
```

Übersicht über die JMS-API

Mit der JMS-Nachrichtenschnittstelle (im Folgenden ″JMS-API″ genannt) können Sie Clientanwendungen entwickeln, die auf asynchrone Weise auf Business-Prozesse zugreifen, die in der Business Process Choreographer-Umgebung ausgeführt werden.

Die JMS-API ermöglicht Clientanwendungen die asynchrone Interaktion mit Mikroprozessen und Dauerprozessen.

Die JMS-API stellt dieselbe Schnittstelle wie die Web-Services-API bereit; es gelten jedoch die folgenden Ausnahmebedingungen:

- Was die Web-Services-API betrifft, so kann die Operation `call` nur zum Aufrufen von Mikroprozessen verwendet werden. Mit der JMS-API jedoch kann die Operation `call` zum Aufrufen sowohl von Mikroprozessen als auch von Dauerprozessen verwendet werden.
- Die folgenden Operationen werden von der JMS-API nicht bereitgestellt:
  - Die Operation `callAsync` (zusammen mit den zugeordneten Callback-Operationen).
  - Die Operationen `completeAndClaimSuccessor` und `getParticipatingTask`.

Beispiel - Ausführung eines Dauerprozesses

Damit eine generische Clientanwendung mit Dauerprozessen arbeiten kann, werden die Schritte in der folgenden Reihenfolge ausgeführt:

1. Richten Sie die JMS-Umgebung wie unter „Auf die JMS-Schnittstelle zugreifen” auf Seite 138 beschrieben ein.
2. Rufen Sie eine Liste der vorinstallierten Prozessdefinitionen ab:
   - Senden Sie `queryProcessTemplates`.
   - Daraufhin wird eine Liste der Prozessschablonenobjekte (**ProcessTemplate**) zurückgegeben.
3. Rufen Sie eine Liste der Startaktivitäten ab (’receive’ oder ’pick’ mit `createInstance="yes"`):
   - Senden Sie `getStartActivities`.
   - Daraufhin wird eine Liste der eingebundenen Operationsschablonenobjekte (**InboundOperationTemplate**) zurückgegeben.
4. Erstellen Sie eine Eingabenachricht. Diese ist umgebungsspezifisch; möglicherweise ist auch die Verwendung von vorimplementierten, prozessspezifischen Artefakten erforderlich.
5. Erstellen Sie eine Prozessinstanz:
   - Setzen Sie `sendMessage` ab.

   Bei der JMS-API können Sie auch die Operation `call` für die Interaktion mit Daueroperationen für Anforderungen und Antworten verwenden, die ein Business-Prozess bereitstellt. Diese Operation gibt ein Operationsergebnis bzw. einen Fehler für das angegebene Antwortziel zurück, auch nach einem langen Zeitraum. Aus diesem Grund ist bei der Verwendung der Operation `call` die Verwendung der Operationen `query` und `getOutputMessage` zum Abrufen der Prozessausgabe bzw. der Fehlernachricht nicht erforderlich.
6. Optional können Sie die Ausgabenachrichten aus den Prozessinstanzen abrufen; wiederholen Sie hierfür die folgenden Schritte:
   - Setzen Sie `query` ab, um den Zustand ’finished’ der Prozessinstanz abzurufen.

- Setzen Sie `getOutputMessage` ab.

7. Arbeiten Sie optional mit zusätzlichen Operationen, die der Prozess breitstellt:
   - Mit der Operation `getWaitingActivities` bzw. `getActiveEventHandlers` können Sie eine Liste der eingebundenen Operationsschablonenobjekte (**InboundOperationTemplate**) abrufen.
   - Erstellen Sie Eingabenachrichten.
   - Senden Sie Nachrichten mit `sendMessage`.
8. Optional können Sie angepasste Merkmale, die im Prozess definiert wurden oder Aktivitäten enthielten, mit `getCustomProperties` und `setCustomProperties` abrufen und setzen.
9. Beenden Sie optional die Arbeit mit einer Prozessinstanz:
   - Senden Sie `delete` und `terminate`, um die Arbeit mit Dauerprozessen zu beenden.

JMS-Anwendungen entwickeln

JMS-Clientanwendungen müssen in Java mithilfe der J2EE-Umgebung (J2EE = Java 2 Enterprise Edition) entwickelt werden.

JMS-Clientanwendungen tauschen Anforderungs- und Antwortnachrichten mit der JMS-API aus. Zur Erstellung einer Anforderungsnachricht füllt die Clientanwendung einen JMS-Nachrichtenhauptteil vom Typ TextMessage mit einem XML-Element, das den Wrapper vom Typ 'Dokument/Literal der entsprechenden Operation darstellt.

Artefakte kopieren

Eine Anzahl Artefakte kann aus der WebSphere-Umgebung kopiert werden, um das Erstellen von JMS-Clientanwendungen zur unterstützen.

Die Verwendung dieser Artefakte ist nur obligatorisch, wenn BOXMLSerializer für die Erstellung des JMS-Nachrichtenhauptteils verwendet wird.

Diese Artefakte sind auf zwei Wegen erhältlich:

- Publizieren und Exportieren aus der WebSphere Process Server-Umgebung

  Bei WebSphere Process Server 6.1 befinden sich alle Clientartefakte im Verzeichnis *installationsstammverzeichnis*\ProcessChoreographer\client. Die folgenden Artefakte gelten für die JMS-API:

  BFMIF.wsdl

  BFMIF.xsd

  BPCGen.xsd

- Kopieren der Dateien von der WebSphere Process Server-Client-CD

Artefakte aus der Serverumgebung veröffentlichen:

Für die Entwicklung von Clientanwendungen, die auf die JMS-API zugreifen können Sie eine Reihe von Artefakten aus der WebSphere-Serverumgebung veröffentlichen.

Bei WebSphere Process Server 6.1 befinden sich alle Clientartefakte im Verzeichnis *was-ausgangsverzeichnis*\ProcessChoreographer\client. Die folgenden Artefakte gelten für die JMS-API:

BFMIF.wsdl

BFMIF.xsd

BPCGen.xsd

Nach dem Veröffentlichen müssen Sie diese Artefakte in Ihre Clientprogrammierumgebung kopieren.

Dateien von der Client-CD kopieren:

Die erforderlichen Dateien für den Zugriff auf die JMS-API sind auf der WebSphere Process Server-Client-CD verfügbar.

1. Legen Sie die Client-CD ein, und öffnen Sie das Verzeichnis `ProcessChoreographer\client`.
2. Kopieren Sie die erforderlichen Dateien in Ihre Entwicklungsumgebung für Clientanwendungen.

   Bei WebSphere Process Server 6.1 befinden sich alle Clientartefakte im Verzeichnis \ProcessChoreographer\client. Die folgenden Artefakte gelten für die JMS-API:

   BFMIF.wsdl

   BFMIF.xsd

   BPCGen.xsd

Antwortnachricht für Business-Ausnahmebedingungen überprüfen

JMS-Clientanwendungen müssen die Nachrichtenheader aller Antwortnachrichten auf Business-Ausnahmebedingungen überprüfen.

Eine JMS-Clientanwendung muss zuerst das Merkmal **IsBusinessException** im Header der Antwortnachricht prüfen.

Beispiel:

```
// Antwortnachricht empfangen
Message receivedMessage = ((JmsProxy) getToBeInvokedUponObject().receiveMessage();
String strResponse = ((TextMessage) receivedMessage).getText();

if (receivedMessage.getStringProperty("IsBusinessException") {
  // strResponse ist ein Business-Fehler
  // Jede API kann ohne processFaultMsg beendet werden
  // Die Aufruf-API kann auch ohne businessFaultMsg beendet werden
 }
else {
  // strResponse ist die Ausgabenachricht
}
```

Webanwendungen für Business-Prozesse und Benutzertasks mit JSF-Komponenten entwickeln

Business Process Choreographer stellt verschiedene JSF-Komponenten (JSF = JavaServer Faces) bereit. Sie können diese Komponenten erweitern und integrieren, um Business-Prozess- und Benutzertaskfunktionen zu Webanwendungen hinzuzufügen.

Sie können WebSphere Integration Developer zur Erstellung Ihrer Webanwendung verwenden.

1. Erstellen Sie ein dynamisches Projekt, und ändern Sie die Merkmale für die Webprojektfunktionen, um die JSF-Basiskomponenten zu integrieren.

Weitere Informationen zur Erstellung eines Webprojekts finden Sie im Information Center für WebSphere Integration Developer.

2. Fügen Sie die erforderlichen Business Process Choreographer Explorer-JAR-Dateien (Java-Archiv) hinzu.

   Fügen Sie die folgenden Dateien zum Verzeichnis WEB-INF/lib Ihres Projekts hinzu:

   - bpcclientcore.jar
   - bfmclientmodel.jar
   - htmclientmodel.jar
   - bpcjsfcomponents.jar

   Wenn Sie Ihre Webanwendung auf einem fernen Server implementieren, müssen Sie außerdem die folgenden Dateien hinzufügen. Diese Dateien sind für den fernen Zugriff auf die Business Process Choreographer-APIs erforderlich.

   - bpe137650.jar
   - task137650.jar

   In WebSphere Process Server befinden sich alle diese Dateien im folgenden Verzeichnis:

   - Auf Windows-Systemen: *installationsstammverzeichnis*\ProcessChoreographer\client
   - Auf UNIX-, Linux- und i5/OS-Systemen: *installationsstammverzeichnis*/ProcessChoreographer/client

3. Fügen Sie die erforderlichen EJB-Referenzen zum Implementierungsdeskriptor für Webanwendungen hinzu (Datei 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. Fügen Sie die JSF-Komponenten des Business Process Choreographer Explorer zur JSF-Anwendung hinzu.

   a. Fügen Sie die für Ihre Anwendungen erforderlichen Tagbibliothekreferenzen zu den JSP-Dateien hinzu (JSP = JavaServer Pages). In der Regel benötigen Sie die JSF- und HTML-Tagbibliotheken sowie die für die JSF-Komponenten erforderliche Tagbibliothek.

      - `<%@ 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. Fügen Sie einen Tag `<f:view>` zum Hauptteil der JSP-Seite sowie einen Tag `<h:form>` zum Tag `<f:view>` hinzu.

c. Fügen Sie die JSF-Komponenten zu den JSP-Dateien hinzu.

   Fügen Sie in Abhängigkeit von Ihrer Anwendung die Komponente List, Details, CommandBar oder Message zu den JSP-Dateien hinzu. Sie können mehrere Instanzen der einzelnen Komponenten hinzufügen.

d. Konfigurieren Sie die Managed Beans in der JSF-Konfigurationsdatei.

   Standardmäßig befindet sich die Konfigurationsdatei in der Datei faces-config.xml. Diese Datei befindet sich im Verzeichnis WEB-INF der Webanwendung.

   In Abhängigkeit von der Komponente, die Sie zu Ihrer JSP-Datei hinzufügen, müssen Sie auch die Referenzen zu der Abfrage und andere Wrapperobjekte zur JSF-Konfigurationsdatei hinzufügen. Zur Sicherstellung der richtigen Fehlerbehandlung müssen Sie außerdem sowohl eine Fehlerbean als auch ein Navigationsziel für die fehlerhafte Seite in der JSF-Konfigurationsdatei definieren.

```
<faces-config>
...
<managed-bean>
  <managed-bean-name>BPCError</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>
<description>
Die allgemeine fehlerhafte Seite.
</description>
<from-outcome>error</from-outcome>
<to-view-id>/Error.jsp</to-view-id>
</navigation-case>
...
</navigation-rule>
</faces-config>
```

   In Fehlersituationen, die die Fehlerseite auslösen, ist die Ausnahmebedingung für die Fehlerbean festgelegt.

e. Implementieren Sie den angepassten Code, den Sie für die Unterstützung der JSF-Komponenten benötigen.

5. Implementieren Sie die Anwendung.

   Wenn Sie die Anwendung in einer Network Deployment-Umgebung implementieren, ändern Sie die JNDI-Namen (JNDI = Java Naming and Directory Interface) der Zielressource in Werte, bei denen die Business Flow Manager- und Human Task Manager-APIs in Ihrer Zelle zu finden sind.

   - Wenn Ihre Business Process Container auf einem anderen Server in derselben verwalteten Zelle konfiguriert sind, haben die Namen die folgende Struktur:

     zelle/*knoten*/*knotenname*/*server*/*servername*/com/ibm/bpe/api/BusinessManagerHome
     zelle/*knoten*/*knotenname*/*server*/*servername*/com/ibm/task/api/HumanTaskManagerHome

- Wenn Ihre Business Process Container in einem Cluster in derselben Zelle konfiguriert sind, haben die Namen die folgende Struktur:

  zelle/*cluster*/*clustername*/com/ibm/bpe/api/BusinessFlowManagerHome
  zelle/*cluster*/*clustername*/com/ibm/task/api/HumanTaskManagerHome

Ordnen Sie die EJB-Referenzen den JNDI-Namen zu, oder fügen Sie die Referenzen manuell der Datei ibm-web-bnd.xmi hinzu.

In der folgenden Tabelle werden die Referenzbindungen und ihre Standardzuordnungen aufgeführt.

Tabelle 33. Referenzbindungen zu JNDI-Namen zuordnen

Referenzbindung	JNDI-Name	Kommentare
ejb/BusinessProcessHome	com/ibm/bpe/api/BusinessFlowManagerHome	Ferne Session-Bean
ejb/LocalBusinessProcessHome	com/ibm/bpe/api/BusinessFlowManagerHome	Lokale Session-Bean
ejb/HumanTaskManagerEJB	com/ibm/task/api/HumanTaskManagerHome	Ferne Session-Bean
ejb/LocalHumanTaskManagerEJB	com/ibm/task/api/HumanTaskManagerHome	Lokale Session-Bean

Ihre implementierte Webanwendung enthält die durch die Business Process Choreographer Explorer-Komponenten bereitgestellte Funktion.

Wenn Sie angepasste JSPs für die Prozess- und Tasknachrichten verwenden, müssen Sie die Webmodule, die zur Implementierung der JSPs verwendet werden, denselben Servern zuordnen, denen der angepasste JSF-Client zugeordnet ist.

Zugehörige Konzepte

„Fehlerbehandlung in JSF-Komponenten (JSF = JavaServer Faces)" auf Seite 149
Die JSF-Komponenten nutzen eine vordefinierte Managed-Bean (BPCError) für die Fehlerbehandlung. In Fehlersituationen, die die Fehlerseite auslösen, ist für die Error-Bean eine Ausnahmebedingung festgelegt.

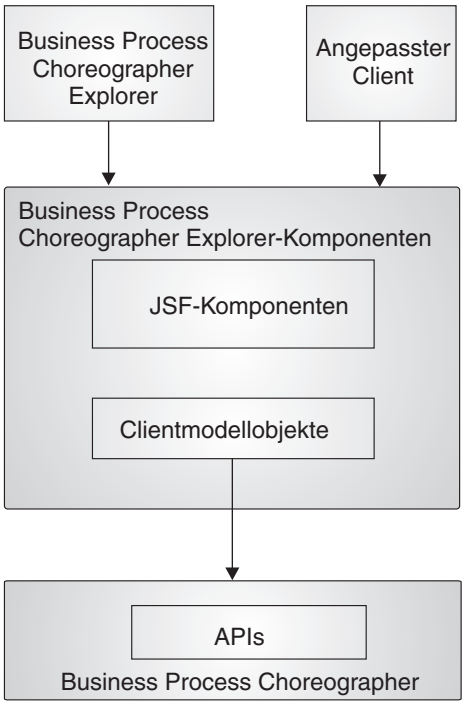
Zugehörige Tasks

„Auf die ferne Schnittstelle der Session-Bean zugreifen" auf Seite 22
Eine EJB-Clientanwendung greift auf die ferne Schnittstelle der Session-Bean über die ferne Home-Schnittstelle der Bean zu.

Business Process Choreographer Explorer-Komponenten

Die Business Process Choreographer Explorer-Komponenten sind eine Reihe von konfigurierbaren, wiederverwendbaren Elementen, die auf der JSF-Technologie (JSF = JavaServer Faces) basieren. Sie können diese Elemente in Webanwendungen einbetten. Die Webanwendungen können anschließend auf installierte Business-Prozess- und Benutzertaskanwendungen zugreifen.

Die Komponenten bestehen aus einer Reihe von JSF-Komponenten und einer Reihe von Clientmodellobjekten. Die Beziehung der Komponenten zu Business Process Choreographer, Business Process Choreographer Explorer und anderen angepassten Clients ist in der folgenden Abbildung zu sehen.



JSF-Komponenten

Die Business Process Choreographer Explorer-Komponenten umfassen die folgenden JSF-Komponenten. Sie betten diese JSF-Komponenten in Ihre JSP-Dateien (JSP = JavaServer Pages) ein, wenn Sie Ihre Webanwendungen für die Arbeit mit Business-Prozessen und Benutzertasks erstellen.

- Komponente List

  In der Komponente List wird eine Liste mit Anwendungsobjekten in einer Tabelle angezeigt (z. B. Tasks, Aktivitäten, Prozessinstanzen, Prozessschablonen, Arbeitselemente oder Eskalationen). Diese Komponente verfügt über einen zugeordneten Listenhandler.
- Komponente Details

  Die Komponente Details zeigt die Merkmale von Tasks, Arbeitselementen, Aktivitäten, Prozessinstanzen und Prozessschablonen an. Diese Komponente verfügt über einen zugeordneten Detailhandler.
- Komponente CommandBar

  Die Komponente CommandBar zeigt eine Leiste mit Schaltflächen an. Diese Schaltflächen stellen Befehle dar, die für ein Objekt in einer Detailsicht oder die ausgewählten Objekte in einer Liste zuständig sind. Diese Objekte werden durch einen Listenhandler oder einen Detailhandler bereitgestellt.
- Komponente Message

  Die Komponente Message zeigt eine Nachricht an, die entweder ein SDO (Service Data Object) oder einen einfachen Typ enthält.

Clientmodellobjekte

Die Clientmodellobjekte werden mit den JSF-Komponenten verwendet. Die Objekte implementieren einige Schnittstellen der zu Grunde liegenden Business Process Choreographer-API und umhüllen das ursprüngliche Objekt.

Die Clientmodellobjekte stellen die Unterstützung in der Landessprache für Kennungen und Konvertierungstools für einige Merkmale bereit.

Fehlerbehandlung in JSF-Komponenten (JSF = JavaServer Faces)

Die JSF-Komponenten nutzen eine vordefinierte Managed-Bean (BPCError) für die Fehlerbehandlung. In Fehlersituationen, die die Fehlerseite auslösen, ist für die Error-Bean eine Ausnahmebedingung festgelegt.

Diese Bean implementiert die Schnittstelle com.ibm.bpc.clientcore.util.ErrorBean. Die Fehlerseite wird in den folgenden Situationen angezeigt:

- Während der Ausführung einer Abfrage, die für einen Listenhandler definiert ist, tritt ein Fehler auf, und der Fehler wird als Fehler ClientException durch die Methode execute eines Befehls generiert.
- Ein Fehler ClientException wird durch die Methode execute generiert, und dieser Fehler ist weder ein Fehler ErrorsInCommandException noch implementiert er die Schnittstelle CommandBarMessage.
- In der Komponente wird eine Fehlernachricht angezeigt, und Sie wählen den Hyperlink für die Nachricht aus.

Für die Schnittstelle com.ibm.bpc.clientcore.util.ErrorBeanImpl gibt es eine Standardimplementierung.

Die Schnittstelle ist folgendermaßen definiert:

```
public interface ErrorBean {

    public void setException(Exception ex);

    /*
     * Dieser Aufruf der Setter-Methode ermöglicht die Übergabe
     * einer Ländereinstellung und der Ausnahmebedingung. Dadurch kann
     * die Methode getExceptionMessage übersetzte Zeichenfolgen zurückgeben.
     *
     */
    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();

    /*
     * Diese Methode gibt die Ausnahmebedingungsnachricht zurück,
     * die rekursiv mit den Nachrichten aller verschachtelten
     * Ausnahmebedingungen verknüpft ist.
     */
    public String getAllExceptionMessages();

    /*
     * Diese Methode gibt den Ausnahmebedingungsstack zurück,
     * der rekursiv mit den Stacks aller verschachtelten
     * Ausnahmebedingungen verknüpft ist.
     */
    public String getAllExceptionStacks();
}
```

Zugehörige Konzepte

„Fehlerbehandlung in der Komponente List" auf Seite 155
Bei der Verwendung der Komponente List zum Anzeigen von Linsten in Ihrer JSF-Anwendung können Sie die Fehlerbehandlungsfunktion nutzen, die von der Klasse com.ibm.bpe.jsf.handler.BPCListHandler bereitgestellt wird.

Standardkonvertierungstools und Bezeichnungen für Clientmodellobjekte

Die Clientmodellobjekte implementieren die entsprechenden Schnittstellen der Business Process Choreographer-API.

Die Komponenten List und Details sind für alle Beans zuständig. Sie können alle Merkmale einer Bean anzeigen. Wenn Sie jedoch die Konvertierungstools und Bezeichnungen definieren möchten, die für die Merkmale einer Bean verwendet werden, müssen Sie entweder den Tag `column` für die Komponente List bzw. den Tag `property` für die Komponente Details verwenden. Anstatt die Konvertierungstools und Bezeichnungen zu definieren, können Sie Standardkonvertierungstools und -bezeichnungen für die Merkmale definieren; hierfür müssen Sie die folgenden statischen Methoden definieren. Sie können die folgenden statischen Methoden definieren:

```
static public String getLabel(String property,Locale locale);
static public com.ibm.bpc.clientcore.converter.SimpleConverter
       getConverter(String property);
```

In der folgenden Tabelle sehen Sie die Clientmodellobjekte, die die entsprechenden Business Flow Manager- und Human Task Manager-API-Klassen implementieren und Standardbezeichnungen und -konvertierungstools für ihre Merkmale bereitstellen. Bei dieser Schnittstellenumhüllung werden für eine Reihe von Merkmalen lokalbezogene Kennungen und Konvertierungstools bereitgestellt. In der folgenden Tabelle ist die Zuordnung der Business Process Choreographer-Schnittstellen zu den entsprechenden Clientmodellobjekten zu sehen.

Tabelle 34. Vorgehensweise beim Zuordnen von Business Process Choreographer-Schnittstellen zu Clientmodellobjekten

Business Process Choreographer-Schnittstelle	Clientmodellobjektklasse
com.ibm.bpe.api.ActivityInstanceData	com.ibm.bpe.clientmodel.bean.ActivityInstanceBean
com.ibm.bpe.api.ActivityServiceTemplateData	com.ibm.bpe.clientmodel.bean.ActivityServiceTemplateBean
com.ibm.bpe.api.ProcessInstanceData	com.ibm.bpe.clientmodel.bean.ProcessInstanceBean
com.ibm.bpe.api.ProcessTemplateData	com.ibm.bpe.clientmodel.bean.ProcessTemplateBean
com.ibm.task.api.Escalation	com.ibm.task.clientmodel.bean.EscalationBean
com.ibm.task.api.Task	com.ibm.task.clientmodel.bean.TaskInstanceBean
com.ibm.task.api.TaskTemplate	com.ibm.task.clientmodel.bean.TaskTemplateBean

Komponente List zu einer JSF-Anwendung hinzufügen

Mit der Komponente List von Business Process Choreographer Explorer können Sie eine Liste mit Clientmodellobjekten (z. B. Business-Prozessinstanzen oder Taskinstanzen) anzeigen.

1. Fügen Sie die Komponente List zur JSP-Datei (JSP = JavaServer Pages) hinzu.

Fügen Sie den Tag bpe:list zum Tag h:form hinzu. Der Tag bpe:list muss ein Modellattribut umfassen. Fügen Sie die bpe:column-Tags zum Tag bpe:list hinzu, um die Merkmale des Objekts hinzuzufügen, die in den einzelnen Zeilen in der Liste aufgeführt werden sollen.

Das folgende Beispiel zeigt, wie eine Komponente List zum Anzeigen von Taskinstanzen hinzugefügt wird.

```
<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>
```

Das Modellattribut bezieht sich auf eine Managed Bean TaskPool. Die Managed Bean stellt die Liste mit Java-Objekten bereit, über die die Liste iteriert; anschließend werden einzelne Zeilen angezeigt.

2. Konfigurieren Sie die Managed Bean, auf die im Tag bpe:list verwiesen wird.

   Für die Komponente List muss diese Managed Bean eine Instanz der Klasse com.ibm.bpe.jsf.handler.BPCListHandler sein.

   Im folgenden Beispiel sehen Sie, wie die Managed Bean TaskPool zur Konfigurationsdatei hinzugefügt wird.

```
<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>
```

In dem Beispiel sehen Sie, dass TaskPool über zwei konfigurierbare Merkmale verfügt: query und type. Der Wert des Merkmals query weist auf eine andere Managed Bean (TaskPoolQuery). Der Wert dieses Merkmals gibt die Bean-

Klasse an, deren Merkmale in den Spalten der angezeigten Liste angezeigt werden. Die zugeordnete Abfrageinstanz kann auch über einen Merkmaltyp verfügen. Wenn ein Merkmaltyp angegeben ist, muss es sich es sich um denselben Typ handeln, der für den Listenhandler angegeben wurde.

Sie können der JSF-Anwendung jeden beliebigen Abfragelogiktyp hinzufügen, solange das Ergebnis der Abfrage im Rahmen einer Liste von stark typisierten Beans dargestellt werden kann. Beispiel: Die Abfrage TaskPoolQuery wird mit einer Liste von com.ibm.task.clientmodel.bean.TaskInstanceBean-Objekten implementiert.

3. Fügen Sie den angepassten Code für die Managed Bean hinzu, auf die der Listenhandler verweist.

   Im folgenden Beispiel sehen Sie, wie angepasster Code zur Managed Bean TaskPool hinzugefügt wird.

```
public class TaskPoolQuery implements Query {

  public List execute throws ClientException {

    // Datei faces-config nach einer Managed Bean "htmConnection" durchsuchen.
    //
    FacesContext ctx = FacesContext.getCurrentInstance();
    Application  app = ctx.getApplication();
    ValueBinding htmVb = app.createValueBinding("#{htmConnection}");
    htmConnection = (HTMConnection) htmVb.getValue(ctx);
    HumanTaskManagerService taskService =
         htmConnection.getHumanTaskManagerService();

    // Anschließendes Abrufen der tatsächlichen Abfragemethode für den Service 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 );

// Setzt jede Zeile im QueryResultSet in eine Taskinstanzbean um.
 for (int i = 0; i < entries; i++) {
   result.next();
   array.add( new TaskInstanceBean( result, connection ));
 }
 return array ;
  }
}
```

   Die Bean TaskPoolQuery fragt die Merkmale der Java-Objekte ab. Diese Bean muss die Schnittstelle com.ibm.bpc.clientcore.Query implementieren. Wenn der Listenhandler seinen Inhalt aktualisiert, ruft er die Methode execute der Abfrage auf. Der Aufruf gibt eine Liste der Java-Objekte zurück. Die Methode getType muss den Klassennamen der zurückgegebenen Java-Objekte zurückgeben.

Ihre JSF-Anwendung enthält nun eine JSP (JavaServer Pages), die die Merkmale der angeforderten Liste der Objekte, z. B. den Status, die Art, den Eigner und den Ersteller der Ihnen zur Verfügung stehenden Taskinstanzen, anzeigt.

Zugehörige Konzepte

„Benutzerspezifische Zeitzoneninformationen" auf Seite 154
Die JSF-Komponenten stellen ein Dienstprogramm für die Verarbeitung benutzerspezifischer Zeitzoneninformationen in der Komponente List bereit.

Zugehörige Verweise

„Komponente List: Tagdefinitionen" auf Seite 156
In der Komponente List von Business Process Choreographer Explorer wird eine Liste mit Objekten in einer Tabelle angezeigt (z. B. Tasks, Aktivitäten, Prozessinstanzen, Prozessschablonen, Arbeitselemente und Eskalationen).

Verarbeitung von Listen

Jeder Instanz der Komponente List ist eine Instanz der Klasse com.ibm.bpe.jsf.handler.BPCListHandler zugeordnet.

Dieser Listenhandler protokolliert, welche Elemente in der zugeordneten Liste ausgewählt werden, und stellt einen Benachrichtigungsmechanismus bereit, um den Listeneinträgen die Detailseiten für die unterschiedlichen Elementtypen zuzuordnen. Der Listenhandler ist über das Attribut **model** des Tags `bpe:list` an die Komponente List gebunden.

Der Benachrichtigungsmechanismus des Listenhandlers wird unter Verwendung der Schnittstelle com.ibm.bpe.jsf.handler.ItemListener implementiert. Sie können Implementierungen dieser Schnittstelle in der Konfigurationsdatei der JSF-Anwendung (JSF = JavaServer Faces) registrieren.

Die Benachrichtigung wird ausgelöst, wenn in der Liste auf einen Link geklickt wird. Links werden für alle Spalten wiedergegeben, für die das Attribut **action** gesetzt ist. Bei dem Wert des Attributs **action** handelt es sich entweder um ein JSF-Navigationsziel oder um eine JSF-Aktionsmethode, die ein JSF-Navigationsziel zurückgibt.

Die Klasse BPCListHandler stellt außerdem eine Methode refreshList bereit. Mit dieser Methode können Sie in JSF-Methodenbindungen ein Steuerelement für die Benutzerschnittstelle implementieren, mit dem die Abfrage erneut ausgeführt werden kann.

Abfrageimplementierungen

Mit dem Listenhandler können Sie alle Arten von Objekten und deren Merkmale anzeigen. Der Inhalt der angezeigten Liste ist von der Liste der Objekte abhängig, die durch die Implementierung der Schnittstelle com.ibm.bpc.clientcore.Query zurückgegeben wurde, die für den Listenhandler konfiguriert ist. Sie können die Abfrage entweder über das Programm mit der Methode setQuery der Klasse BPCListHandler festlegen oder in den JSF-Konfigurationsdateien der Anwendung konfigurieren.

Abfragen können nicht nur für die APIs von Business Process Choreographer ausgeführt werden, sondern auch für alle anderen Informationsquellen, auf die Ihre Anwendung zugreifen kann (z. B. ein Content-Management-System oder eine Datenbank). Die einzige Voraussetzung besteht darin, dass das Ergebnis der Abfrage durch die Methode execute in Form von Objekten in einer Liste des Typs `java.util.List` zurückgegeben wird.

Der Typ der zurückgegebenen Objekte muss garantieren, dass die entsprechenden Getter-Methoden für alle Merkmale verfügbar sind, die in den Spalten der Liste angezeigt werden, für die die Abfrage definiert wird. Um sicherzustellen, dass der Typ des zurückgegebenen Objekts zu den Listendefinitionen passt, können Sie als Wert des Merkmals 'type' für die Instanz von BPCListHandler, die in der Faces-Konfigurationsdatei definiert ist, den vollständig qualifizierten Klassennamen der zurückgegebenen Objekte festlegen. Diesen Namen können Sie im Aufruf getType der Abfrageimplementierung zurückgeben. Zur Laufzeit überprüft der Listenhandler, ob die Objekttypen mit den Definitionen konform sind.

Um Fehlernachrichten zu bestimmten Einträgen in einer Liste zuzuordnen, müssen die von der Abfrage zurückgegebenen Objekte eine Methode mit der Signatur `public Object getID()` implementieren.

Standardkonvertierungstools und Bezeichnungen

Bei den durch eine Abfrage zurückgegebenen Elementen muss es sich um Beans handeln, und ihre Klasse muss der Klasse entsprechen, die als Typ in der Definition der Klasse BPCListHandler oder der Schnittstelle com.ibm.bpc.clientcore-.Query angegeben wurde. Außerdem wird durch die Komponente List geprüft, ob die Elementklasse bzw. eine Superklasse die folgenden Methoden implementiert:

```
static public String getLabel(String property,Locale locale);
static public com.ibm.bpc.clientcore.converter.SimpleConverter
       getConverter(String property);
```

Wenn diese Methoden für die Beans definiert sind, verwendet die Komponente List die Bezeichnung als Standardbezeichnung für die Liste sowie SimpleConverter als Standardkonvertierungstool für das Merkmal. Sie können diese Einstellungen mit den Attributen **label** und **converterID** des Tags `bpe:list` überschreiben. Weitere Informationen finden Sie in der Javadoc, wenn Sie nach der Schnittstelle SimpleConverter und der Klasse ColumnTag suchen.

Benutzerspezifische Zeitzoneninformationen

Die JSF-Komponenten stellen ein Dienstprogramm für die Verarbeitung benutzerspezifischer Zeitzoneninformationen in der Komponente List bereit.

Die Klasse BPCListHandler verwendet die Schnittstelle com.ibm.bpc.clientcore.util.User, um Informationen zur Zeitzone und zur Ländereinstellung eines Benutzers abzurufen. Die Komponente List geht davon aus, dass die Implementierung der Schnittstelle mit dem Wert **user** als Name der Managed-Bean in der JSF-Konfigurationsdatei konfiguriert ist. Falls dieser Eintrag in der Konfigurationsdatei fehlt, wird die Zeitzone zurückgegeben, in der WebSphere Process Server ausgeführt wird.

Die Schnittstelle com.ibm.bpc.clientcore.util.User ist folgendermaßen definiert:

```
public interface User {

    /**
     * Vom Client des Benutzers verwendete Ländereinstellung.
     * @return Locale.
     */
    public Locale getLocale();
   /**
    * Vom Client des Benutzers verwendete Zeitzone.
    * @return TimeZone.
    */
    public TimeZone getTimeZone();
```

```
    /**
     * Name des Benutzers.
     * @return name of the user.
     */
     public String getName();
}
```

Fehlerbehandlung in der Komponente List

Bei der Verwendung der Komponente List zum Anzeigen von Linsten in Ihrer JSF-Anwendung können Sie die Fehlerbehandlungsfunktion nutzen, die von der Klasse com.ibm.bpe.jsf.handler.BPCListHandler bereitgestellt wird.

Fehler bei der Ausführung von Abfragen oder Befehlen

Falls während der Ausführung einer Abfrage ein Fehler auftritt, unterscheidet die Klasse BPCListHandler zwischen Fehlern, die durch unzureichende Zugriffsberechtigungen verursacht wurden, und anderen Ausnahmebedingungen. Um Fehler aufgrund von unzureichenden Zugriffsberechtigungen abzufangen, muss der Parameter **rootCause** der Ausnahmebedingung ClientException, die durch die Methode execute der Abfrage ausgelöst wird, eine Ausnahmebedingung com.ibm.bpe.api.EngineNotAuthorizedException oder com.ibm.task.api.NotAuthorizedException sein. Die Komponente List zeigt anstelle des Abfrageergebnisses die Fehlernachricht an.

Falls der Fehler nicht durch unzureichende Zugriffsberechtigungen verursacht wurde, übergibt die Klasse BPCListHandler das Ausnahmebedingungsobjekt an die Implementierung der Schnittstelle com.ibm.bpc.clientcore.util.ErrorBean, die durch den Schlüssel BPCError in der JSF-Anwendungskonfigurationsdatei definiert ist. Wenn die Ausnahmebedingung festgelegt ist, wird das Fehlernavigationsziel aufgerufen.

Fehler beim Arbeiten mit Elementen, die in einer Liste angezeigt werden

Die Klasse BPCListHandler implementiert die Schnittstelle com.ibm.bpe.jsf.handler.ErrorHandler. Informationen zu diesen Fehlern können Sie mit dem Zuordnungsparameter des Typs java.util.Map in der Methode setErrors bereitstellen. Diese Zuordnung enthält Kennungen als Schlüssel und Ausnahmebedingungen als Werte. Die Kennungen müssen Werte sein, die von der Methode getID des Objekts zurückgegeben werden, das den Fehler verursacht hat. Falls die Zuordnung festgelegt ist, und eine der IDs mit einem der in der Liste angezeigten Elemente übereinstimmt, fügt der Listenhandler automatisch eine Spalte zur Liste hinzu, in der die Fehlernachricht enthalten ist.

Damit die Liste keine veralteten Fehlernachrichten enthält, setzen Sie die Fehlerzuordnung zurück. In den folgenden Situationen wird die Zuordnung automatisch zurückgesetzt:

- Die Methode refreshList der Klasse BPCListHandler wird aufgerufen.
- Für die Klasse BPCListHandler wird eine neue Abfrage festgelegt.
- Mit der Komponente CommandBar werden Aktionen für Elemente der Liste ausgelöst. Die Komponente CommandBar verwendet unter anderem diesen Mechanismus als Methode zur Fehlerbehandlung.

Zugehörige Konzepte

„Fehlerbehandlung in JSF-Komponenten (JSF = JavaServer Faces)" auf Seite 149
Die JSF-Komponenten nutzen eine vordefinierte Managed-Bean (BPCError) für

die Fehlerbehandlung. In Fehlersituationen, die die Fehlerseite auslösen, ist für die Error-Bean eine Ausnahmebedingung festgelegt.

Komponente List: Tagdefinitionen

In der Komponente List von Business Process Choreographer Explorer wird eine Liste mit Objekten in einer Tabelle angezeigt (z. B. Tasks, Aktivitäten, Prozessinstanzen, Prozessschablonen, Arbeitselemente und Eskalationen).

Die Komponente List besteht aus den JSF-Komponententags `bpe:list` und `bpe:column`. Der Tag `bpe:column` ist ein Unterelement des Tags `bpe:list`.

Komponentenklasse

com.ibm.bpe.jsf.component.ListComponent

Beispielsyntax

```
<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>
```

Tagattribute

Der Hauptteil des Tags `bpe:list` darf nur `bpe:column`-Tags enthalten. Wenn die Tabelle wiedergegeben wird, iteriert die Komponente List über die Liste der Anwendungsobjekte und gibt alle Spalten für die einzelnen Objekte wieder.

Tabelle 35. bpe:list - Attribute

Attribut	Erforderlich	Beschreibung
buttonStyleClass	nein	Die CSS-Darstellungsklasse für die Wiedergabe der Schaltflächen im Fußzeilenbereich.
cellStyleClass	nein	Die CSS-Darstellungsklasse für die Wiedergabe einzelner Tabellenzellen.
checkbox	nein	Stellt fest, ob das Markierungsfeld zum Auswählen mehrerer Elemente wiedergegeben wurde. Das Attribut hat den Wert `true` oder `false`. Wenn der Wert auf `true` gesetzt ist, wird die Markierungsfeldspalte wiedergegeben.
headerStyleClass	nein	Die CSS-Darstellungsklasse für die Wiedergabe der Tabellenüberschrift.
model	ja	Eine Wertebindung für eine Managed Bean der Klasse com.ibm.bpe.jsf.handler.BPCListHandler.

Tabelle 35. bpe:list - Attribute (Forts.)

Attribut	Erforderlich	Beschreibung
rows	nein	Die Anzahl der Zeilen, die auf einer Seite angezeigt werden. Wenn die Anzahl an Elementen die Anzahl an Zeilen überschreitet, werden am Ende der Tabelle Schaltflächen zum Blättern angezeigt. Werteausdrücke werden für dieses Attribut nicht unterstützt.
rowClasses	nein	Die CSS-Darstellungsklasse für die Wiedergabe der Zeilen in der Tabelle.
selectAll	nein	Wenn dieses Attribut auf `true` gesetzt ist, werden standardmäßig alle Elemente in der Liste ausgewählt.
styleClass	nein	Die CSS-Darstellungsklasse für die Wiedergabe der gesamten Tabelle mit den Schaltflächen für die Titel, die Zeilen und zum Blättern.

Tabelle 36. bpe:column - Attribute

Attribut	Erforderlich	Beschreibung
action	nein	Wenn dieses Attribut angegeben ist, wird in der Spalte ein Link wiedergegeben. Wenn Sie auf diesen Link klicken, wird entweder eine JSF-Aktionsmethode oder das Faces-Navigationsziel ausgelöst. Eine JSF-Aktionsmethode hat die folgende Signatur: String method().
converterID	nein	Die ID für das Faces-Konvertierungstool, die für die Konvertierung des Merkmalwerts verwendet wird. Wenn dieses Attribut nicht gesetzt ist, wird eine beliebige ID für das Faces-Konvertierungstool verwendet, die das Modell für dieses Merkmal bereitstellt.
label	nein	Ein Literal oder Wertebindungsausdruck, das bzw. der als Bezeichnung für die Überschrift der Spalte oder der Zelle der Tabellenüberschriftszeile. Wenn dieses Attribut nicht gesetzt ist, wird eine beliebige Bezeichnung verwendet, die das Modell für dieses Merkmal bereitstellt.
name	ja	Der Name des Merkmals, das in dieser Spalte angezeigt wird.

Komponente Details zu einer JSF-Anwendung hinzufügen

Die Komponente Details von Business Process Choreographer Explorer zeigt die Merkmale von Tasks, Arbeitselementen, Aktivitäten, Prozessinstanzen und Prozessschablonen an.

1. Fügen Sie die Komponente Details zur JSP-Datei (JSP = JavaServer Pages) hinzu.

   Fügen Sie den Tag `bpe:details` zum Tag `<h:form>` hinzu. Der Tag `bpe:details` muss ein Modellattribut (**model**) enthalten. Sie können Merkmale zur Komponente Details mit dem Tag `bpe:property` hinzufügen.

Das folgende Beispiel zeigt, wie eine Komponente Details hinzugefügt wird, die einige der Merkmale einer Taskinstanz anzeigt.

```
<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>
```

Das Attribut **model** bezieht sich auf eine Managed Bean TaskInstanceDetails. Die Bean stellt die Merkmale des Java-Objekts bereit.

2. Konfigurieren Sie die Managed Bean, auf die im Tag `bpe:details` verwiesen wird.

   Für die Komponente Details muss diese Managed Bean eine Instanz der Klasse com.ibm.bpe.jsf.handler.BPCDetailsHandler sein. Diese Handlerklasse umhüllt ein Java-Objekt und legt die allgemein zugänglichen Merkmale der Komponente Details fest.

   Das folgende Beispiel zeigt, wie die Managed Bean TaskInstanceDetails zur Konfigurationsdatei hinzugefügt wird.

```
<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>
```

   In diesem Beispiel sehen Sie, dass die Bean TaskInstanceDetails über ein konfigurierbares Merkmal `type` verfügt. Der Wert dieses Merkmals gibt die Bean-Klasse (com.ibm.task.clientmodel.bean.TaskInstanceBean) an, deren Merkmale in den Zeilen der angezeigten Details angezeigt werden. Bei der Bean-Klasse kann es sich um eine beliebige JavaBeans-Klasse handeln. Wenn die Bean ein Standardkonvertierungstool und Merkmalbezeichnungen angibt, werden das Konvertierungstool und die Bezeichnung für die Wiedergabe ebenso wie für die Komponente List verwendet.

Ihre JSF-Anwendung enthält nun eine JSP (JavaServer Pages), die die Details des angegebenen Objekts anzeigt, z. B. die Details einer Taskinstanz.

Zugehörige Verweise

„Komponente Details: Tagdefinitionen"
Die Komponente Details von Business Process Choreographer Explorer zeigt die Merkmale von Tasks, Arbeitselementen, Aktivitäten, Prozessinstanzen und Prozessschablonen an.

Komponente Details: Tagdefinitionen

Die Komponente Details von Business Process Choreographer Explorer zeigt die Merkmale von Tasks, Arbeitselementen, Aktivitäten, Prozessinstanzen und Prozessschablonen an.

Die Komponente Details besteht aus den JSF-Komponententags `bpe:details` und `bpe:property`. Der Tag `bpe:property` ist ein Unterelement des Tags `bpe:details`.

Komponentenklasse

com.ibm.bpe.jsf.component.DetailsComponent

Beispielsyntax

```
<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>
```

Tagattribute

Mit den `bpe:property`-Tags können Sie sowohl die Untergruppe von Attributen, die angezeigt werden, als auch die Reihenfolge angeben, in der diese Attribute angezeigt werden. Wenn der Detailtag keine Attributtags enthält, gibt er alle verfügbaren Attribute des Modellobjekts wieder.

Tabelle 37. bpe:details - Attribute

Attribut	Erforderlich	Beschreibung
columnClasses	nein	Eine Liste mit CSS-Darstellungsklassen, die durch Kommata getrennt sind und der Wiedergabe von Spalten dienen.
id	nein	Die JSF-ID der Komponente.
model	ja	Eine Wertebindung für eine Managed Bean der Klasse com.ibm.bpe.jsf.handler.BPCDetailsHandler.
rowClasses	nein	Eine Liste mit CSS-Darstellungsklassen, die durch Kommata getrennt sind und der Wiedergabe von Zeilen dienen.
styleClass	nein	Die CSS-Klasse, die für die Wiedergabe des HTML-Elements verwendet wird.

Tabelle 38. bpe:property - Attribute

Attribut	Erforderlich	Beschreibung
converterID	nein	Die ID, die zum Registrieren des Konvertierungstools in der JSF-Konfigurationsdatei (JSF = JavaServer Faces) verwendet wird.
label	nein	Die Kennung des Merkmals. Wenn dieses Attribut nicht gesetzt ist, stellt die Clientmodellklasse eine Standardkennung bereit.
name	ja	Der Name des anzuzeigenden Merkmals. Dieser Name muss einem angegebenen Merkmal entsprechen, das in der entsprechenden Clientmodellklasse definiert wurde.

Komponente CommandBar zu einer JSF-Anwendung hinzufügen

Mit der Komponente CommandBar von Business Process Choreographer Explorer können Sie eine Leiste mit Schaltflächen anzeigen. Diese Schaltflächen stellen Befehle dar, die für die Detailsicht eines Objekts oder der ausgewählten Objekten in einer Liste zuständig sind.

Wenn der Benutzer auf eine Schaltfläche in der Benutzerschnittstelle klickt, wird der entsprechende Befehl für die ausgewählten Objekte ausgeführt. Sie können die Komponente CommandBar in Ihrer JSF-Anwendung hinzufügen und erweitern.

1. Fügen Sie die Komponente CommandBar zur JSP-Datei (JSP = JavaServer Pages) hinzu.

   Fügen Sie den Tag `bpe:commandbar` zum Tag `<h:form>` hinzu. Der Tag `bpe:commandbar` muss ein Modellattribut enthalten.

   Das folgende Beispiel zeigt, wie eine Komponente CommandBar hinzugefügt wird, die Befehle zum Aktualisieren und Beanspruchen für eine Taskinstanzliste bereitstellt.

   ```
   <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>
   ```

   Das Attribut **model** bezieht sich auf eine Managed Bean. Diese Bean muss die Schnittstelle ItemProvider implementieren und die ausgewählten Java-Objekte bereitstellen. Die Komponente CommandBar wird in der Regel entweder mit der Komponente List oder der Komponente Details in derselben JSP-Datei verwendet. Im Allgemeinen handelt es sich bei dem im Tag angegebenen Modell um dasselbe Modell, das in der Komponente List oder in der Komponente Details auf derselben Seite angegeben wird. Beispielsweise wirkt sich der Befehl also bei der Komponente List auf die ausgewählten Elemente in der Liste aus.

   In diesem Beispiel bezieht sich das Attribut **model** auf die Managed Bean TaskInstanceList. Diese Bean stellt die ausgewählten Objekte in der Taskinstanzliste bereit. Die Bean muss die Schnittstelle ItemProvider implementieren. Diese Schnittstelle wird durch die Klasse BPCListHandler und die Klasse BPCDetailsHandler implementiert.

2. Optional: Konfigurieren Sie die Managed Bean, auf die im Tag `bpe:commandbar` verwiesen wird.

   Wenn das Attribut **model** der Befehlsleiste (CommandBar) auf eine bereits konfigurierte Managed Bean verweist (z. B. für einen Listen- oder Detailshandler), ist keine weitere Konfiguration erforderlich. Wenn Sie die Konfiguration eines dieser beiden Handler geändert oder eine andere Managed Bean verwendet haben, fügen Sie eine Managed Bean, die die Schnittstelle ItemProvider implementiert, zur JSF-Konfigurationsdatei hinzu.

3. Fügen Sie den Code hinzu, mit dem die angepassten Befehle in die JSF-Anwendung implementiert werden.

   Das folgende Code-Snippet veranschaulicht, wie Sie eine Befehlsklasse schreiben können, durch die die Schnittstelle Command implementiert wird. Auf diese Befehlsklasse (MyClaimCommand) bezieht sich der Tag `bpe:command` in der JSP-Datei.

```
public class MyClaimCommand implements Command {

  public String execute(List selectedObjects) throws ClientException {
     if( selectedObjects != null && selectedObjects.size() > 0 ) {
        try {
         // Festlegen der HumanTaskManagerService-API aus einer HTMConnection-Bean.
         // Konfigurieren der Bean in der Datei faces-config.xml für einen einfachen
         // Zugriff auf die JSF-Anwendung.
         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){
             ;   // Fehler beim Iterieren oder Beanspruchen der Taskinstanz.
                  // Bitte zum besseren Verständnis des Beispiels ignorieren.
           }
         }
       }
       catch(Exception e){
         ;  // Konfigurations- oder Kommunikationsfehler.
           // Bitte zum besseren Verständnis des Beispiels ignorieren.
       }
     }
     return null;
  }

 // Standardimplementierungen
 public boolean isMultiSelectEnabled() { return false; }
 public boolean[] isApplicable(List itemsOnList) {return null; }
 public void setContext(Object targetModel) {; // Wird hier nicht verwendet }
}
```

   Der Befehl wird wie folgt verarbeitet:

   a. Ein Befehl wird aufgerufen, wenn ein Benutzer auf die entsprechende Schaltfläche in der Befehlsleiste klickt. Mit der Komponente CommandBar werden die ausgewählten Elemente aus dem im Attribut **model** angegebenen Elementprovider abgerufen, und es wird die Liste der ausgewählten Objekte an die Methode execute der Instanz commandClass übergeben.

b. Das Attribut **commandClass** bezieht sich auf die Implementierung eines angepassten Befehls, der die Schnittstelle Command implementiert. Der Befehl muss die Methode public String execute(List selectedObjects) throws ClientException implementieren. Der Befehl gibt ein Ereignis zurück, das zur Festlegung der nächsten Navigationsregel für die JSF-Anwendung verwendet wird.

c. Nachdem der Befehl ausgeführt wurde, wertet die Komponente CommandBar das Attribut **action** aus. Bei dem Attribut **action** kann es sich um eine statische Zeichenfolge oder eine Methodenbindung an eine JSF-Aktionsmethode mit der Signatur `public String Method()` handeln. Verwenden Sie das Attribut **action**, um das Ergebnis einer Befehlsklasse zu überschreiben oder um ein Ergebnis für die Navigationsregeln explizit anzugeben. Das Attribut **action** wird nicht verarbeitet, wenn der Befehl eine andere Ausnahmebedingung als ErrorsInCommandException auslöst.

d. Wenn für das Attribut **commandClass** keine Befehlsklasse angegeben ist, wird die Aktion unmittelbar aufgerufen. Was z. B. den Aktualisierungsbefehl in dem Beispiel betrifft, so wird der JSF-Werteausdruck `#{TaskInstanceList.refreshList}` anstelle eines Befehls aufgerufen.

Ihre JSF-Anwendung enthält nun eine JSP (JavaServer Pages), die eine angepasste Befehlsleiste implementiert.

Zugehörige Verweise

„Komponente CommandBar: Tagdefinitionen" auf Seite 163
Die Befehlszeilenkomponente (CommandBar) von Business Process Choreographer Explorer zeigt eine Leiste mit Schaltflächen an. Diese Schaltflächen sind für ein Objekt in einer Detailsicht oder die ausgewählten Objekte in einer Liste zuständig.

Verarbeitung von Befehlen

Mit der Komponente CommandBar können Sie Aktionsschaltflächen zu Ihrer Anwendung hinzufügen. Die Komponente erstellt die Schaltflächen für die Aktionen in der Benutzerschnittstelle und verarbeitet die Ereignisse, die beim Klicken auf eine Schaltfläche erstellt werden.

Diese Schaltflächen lösen Funktionen für die Objekte aus, die von einer Schnittstelle com.ibm.bpe.jsf.handler.ItemProvider zurückgegeben werden, beispielsweise die Klasse BPCListHandler oder die Klasse BPCDetailsHandler. Die Komponente CommandBar verwendet den Elementprovider, der durch den Wert des Attributs **model** im Tag `bpe:commandbar` definiert ist.

Sobald ein Benutzer auf eine Schaltfläche im Befehlsleistenabschnitt der Benutzerschnittstelle für die Anwendung klickt, wird das zugeordnete Ereignis folgendermaßen von der Komponente CommandBar verarbeitet:

1. Die Komponente CommandBar ermittelt die Implementierung der Schnittstelle com.ibm.bpc.clientcore.Command, die für die Schaltfläche angegeben ist, von der das Ereignis generiert wurde.
2. Falls das Modell, das der Komponente CommandBar zugeordnet ist, die Schnittstelle com.ibm.bpe.jsf.handler.ErrorHandler implementiert, wird die Methode clearErrorMap aufgerufen, um die Fehlernachrichten der vorherigen Ereignisse zu entfernen.

3. Die Methode getSelectedItems der Schnittstelle ItemProvider wird aufgerufen. Die zurückgegebene Liste der Elemente wird an die Methode execute des Befehls übergeben, und der Befehl wird aufgerufen.
4. Die Komponente CommandBar ermittelt das JSF-Navigationsziel (JSF = JavaServer Faces). Falls kein Attribut **action** im Tag `bpe:commandbar` angegeben ist, gibt der Rückgabewert der Methode execute das Navigationsziel an. Ist das Attribut **action** auf eine JSF-Methodenbindung gesetzt, wird die von der Methode zurückgegebene Zeichenfolge als Navigationsziel interpretiert. Da Attribut **action** kann auch ein explizites Navigationsziel angeben.

Komponente CommandBar: Tagdefinitionen

Die Befehlszeilenkomponente (CommandBar) von Business Process Choreographer Explorer zeigt eine Leiste mit Schaltflächen an. Diese Schaltflächen sind für ein Objekt in einer Detailsicht oder die ausgewählten Objekte in einer Liste zuständig.

Die Komponente CommandBar besteht aus den JSF-Komponententags `bpe:commandbar` und `bpe:command`. Der Tag `bpe:command` ist ein Unterelement des Tags `bpe:commandbar`.

Komponentenklasse

com.ibm.bpe.jsf.component.CommandBarComponent

Beispielsyntax

```
<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>
```

Tagattribute

Tabelle 39. bpe:commandbar - Attribute

Attribut	Erforderlich	Beschreibung
buttonStyleClass	nein	Die CSS-Darstellungsklasse für die Wiedergabe der Schaltflächen in der Befehlsleiste.
id	nein	Die JSF-ID der Komponente.
model	ja	Ein Wertebindungsausdruck für eine Managed Bean, die die Schnittstelle ItemProvider implementiert. Diese Managed Bean ist in der Regel die Klasse com.ibm.bpe.jsf.handler.BPCListHandler oder die Klasse com.ibm.bpe.jsf.handler.BPCDetailsHandler, die von der Komponente List oder Details in derselben JSP-Datei (JSP = JavaServer Pages) wie die Komponente CommandBar verwendet wird.

Tabelle 39. bpe:commandbar - Attribute (Forts.)

Attribut	Erforderlich	Beschreibung
styleClass	nein	Die CSS-Darstellungsklasse, die für die Wiedergabe der Befehlsleiste verwendet wird.

Tabelle 40. bpe:command - Attribute

Attribut	Erforderlich	Beschreibung
action	nein	Eine JSF-Aktionsmethode oder das Faces-Navigationsziel, das durch die Befehlsschaltfläche ausgelöst wird. Das Navigationsziel, das durch die Aktion zurückgegeben wird, überschreibt alle anderen Navigationsregeln. Die Aktion wird aufgerufen, wenn entweder eine Ausnahmebedingung nicht ausgegeben wird oder wenn die Ausnahmebedingung ErrorsInCommandException von dem Befehl ausgegeben wird.
commandClass	nein	Der Name der Befehlsklasse. Eine Instanz der Klasse wird durch die Komponente CommandBar erstellt und ausgeführt, wenn die Befehlsschaltfläche ausgewählt wird.
commandID	ja	Die ID des Befehls.
context	nein	Ein Objekt, das Kontext für Befehle bereitstellt, die mit dem Attribut **commandClass** angegeben werden. Das Kontextobjekt wird abgerufen, wenn zuerst auf die Befehlsleiste zugegriffen wird.
immediate	nein	Gibt an, wann der Befehl ausgelöst wird. Wenn der Wert für dieses Attribut `true` lautet, wird der Befehl vor der Verarbeitung der Eingabe der Seite ausgelöst. Der Standardwert lautet `false`.
label	ja	Die Bezeichnung der wiedergegebenen Schaltfläche in der Befehlsleiste.
rendered	nein	Legt fest, ob eine Schaltfläche wiedergegeben wird. Der Wert des Attributs kann entweder ein boolescher Wert oder ein Werteausdruck sein.
styleClass	nein	Die CSS-Darstellungsklasse, die für die Wiedergabe der Schaltfläche verwendet wird. Diese Darstellung überschreibt die für die Befehlsleiste definierte Schaltflächendarstellung.

Komponente Message zu einer JSF-Anwendung hinzufügen

Mit der Komponente Message von Business Process Choreographer Explorer können Sie Datenobjekte und primitive Typen in einer JSF-Anwendung (JSF = JavaServer Faces) wiedergeben.

Wenn es sich bei dem Nachrichtentyp um einen primitiven Typ handelt, wird ein Feld für die Bezeichnung und die Eingabe wiedergegeben. Wenn es sich bei dem Nachrichtentyp um ein Datenobjekt handelt, durchquert die Komponente das Objekt und gibt die Elemente in dem Objekt wieder.

1. Fügen Sie die Komponente Message zur JSP-Datei (JSP = JavaServer Pages) hinzu.

   Fügen Sie den Tag `bpe:form` zum Tag `<h:form>` hinzu. Der Tag `bpe:form` muss ein Attribut model umfassen.

   Das folgende Beispiel zeigt, wie eine Komponente Message hinzugefügt wird.

   ```
   <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>
   ```

   Das Attribut **model** der Komponente Message bezieht sich auf ein Objekt com.ibm.bpc.clientcore.MessageWrapper. Dieses Wrapperobjekt umhüllt entweder ein SDO-Objekt (SDO = Service Data Object) oder einen primitiven Java-Typ, z. B. `int` oder `boolean`. In diesem Beispiel wird die Nachricht durch ein Merkmal der Managed Bean MyHandler bereitgestellt.

2. Konfigurieren Sie die Managed Bean, auf die im Tag `bpe:form` verwiesen wird.

   Im folgenden Beispiel sehen Sie, wie die Managed Bean MyHandler zur Konfigurationsdatei hinzugefügt wird.

   ```
   <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. Fügen Sie den angepassten Code zur JSF-Anwendung hinzu.

   Das folgende Beispiel veranschaulicht die Implementierung von Eingabe- und Ausgabenachrichten:

   ```
   public class MyHandler implements ItemListener {
     private TaskInstanceBean taskBean;
     private MessageWrapper inputMessage, outputMessage

     /* Listenermethode z. B., wenn eine Taskinstanz in einem Listenhandler ausgewählt
      * wurde. Sicherstellen, dass der Handler in der Datei faces-config.xml
      * registriert ist; oder manuell registrieren.
      */
     public void itemChanged(Object item) {
       if( item instanceof TaskInstanceBean ) {
           taskBean = (TaskInstanceBean) item ;
       }
     }
   ```

```
  /* Eingabenachrichtenwrapper abrufen
   */
  public MessageWrapper getInputMessage() {
    try{
        inputMessage = taskBean.getInputMessageWrapper() ;
    }
    catch(Exception e){
        ;       //...Fehler der Einfachheit halber ignorieren
    }
    return inputMessage;
  }

  /* Ausgabenachrichtenwrapper abrufen
   */
  public MessageWrapper getOutputMessage() {
    // Abrufen der Nachricht aus der Bean. Wenn keine Nachricht vorhanden ist, erstellen
    // sie eine, wenn die Task vom Benutzer beansprucht wurde. Stellen Sie sicher, dass
    // nur potenzielle Eigner oder Eigner die Ausgabenachricht bearbeiten können.
    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){
        ;       //...Fehler der Einfachheit halber ignorieren
    }
    return outputMessage
  }
}
```

Die Managed Bean MyHandler implementiert die Schnittstelle com.ibm.jsf.handler.ItemListener, sodass sie sich selbst als Elementlistener bei Listenhandlern registrieren kann. Wenn der Benutzer auf ein Element in der Liste klickt, wird die Bean MyHandler in der zugehörigen Methode itemChanged(Object item) über das ausgewählte Element benachrichtigt. Der Handler prüft den Elementtyp und speichert anschließend einen Verweis auf das zugeordnete Objekt TaskInstanceBean. Wenn Sie diese Schnittstelle verwenden möchten, fügen Sie der Liste itemListener im entsprechenden Listenhandler in der Datei faces-config.xml einen Eintrag hinzu.

Die Bean MyHandler stellt die Methoden getInputMessage und getOutputMessage bereit. Beide Methoden geben ein MessageWrapper-Objekt zurück. Die Methoden delegieren die Aufrufe an die referenzierte Taskinstanzbean. Wenn die Taskinstanzbean null zurückgibt, weil beispielsweise eine Nachricht nicht gesetzt wurde, erstellt der Handler eine neue, leere Nachricht und speichert sie. Die Komponente Message zeigt die von der Bean MyHandler bereitgestellten Nachrichten an.

Ihre JSF-Anwendung enthält nun eine JSP (JavaServer Pages), die Datenobjekte und primitive Typen wiedergeben kann.

Zugehörige Verweise

„Komponente Message: Tagdefinitionen” auf Seite 167
Die Komponente Message von Business Process Choreographer Explorer übergibt `commonj.sdo.DataObject`-Objekte und primitive Typen (z. B. ganze Zahlen oder Zeichenfolgen) in einer JSF-Anwendung (JSF = JavaServer Faces).

Komponente Message: Tagdefinitionen

Die Komponente Message von Business Process Choreographer Explorer übergibt `commonj.sdo.DataObject`-Objekte und primitive Typen (z. B. ganze Zahlen oder Zeichenfolgen) in einer JSF-Anwendung (JSF = JavaServer Faces).

Die Komponente Message besteht aus dem JSF-Komponententag `bpe:form`.

Komponentenklasse

com.ibm.bpe.jsf.component.MessageComponent

Beispielsyntax

```
<bpe:form model="#{TaskInstanceDetailsBean.inputMessageWrapper}"
          simplification="true" readOnly="true"
          styleClass4table="messageData"
          styleClass4output="messageDataOutput">
</bpe:form>
```

Tagattribute

Tabelle 41. bpe:form - Attribute

Attribut	Erforderlich	Beschreibung
id	nein	Die JSF-ID der Komponente
model	ja	Ein Wertebindungsausdruck, der sich entweder auf ein `commonj.sdo.DataObject`-Objekt oder auf ein `com.ibm.bpc.clientcore.MessageWrapper`-Objekt bezieht.
readOnly	nein	Wenn dieses Attribut auf `true` gesetzt ist, wird ein schreibgeschütztes Format wiedergegeben. Standardmäßig ist dieses Attribut auf `false` gesetzt.
simplification	nein	Wenn dieses Attribut auf `true` gesetzt ist, werden Merkmale mit einfachen Typen und einer Kardinalität von null oder eins angezeigt. Standardmäßig ist dieses Attribut auf `true` gesetzt.
style4validinput	nein	Die CSS-Darstellung für die Wiedergabe einer gültigen Eingabe.
style4invalidinput	nein	Die CSS-Darstellung für die Wiedergabe einer nicht gültigen Eingabe.
styleClass4invalidInput	nein	Der CSS-Darstellungsklassenname für die Wiedergabe einer nicht gültigen Eingabe.
styleClass4output	nein	Der Name der CSS-Darstellungsklasse für die Wiedergabe der Ausgabeelemente.
styleClass4table	nein	Der Klassenname der CSS-Tabellenvorlage für die Wiedergabe der Tabellen, die über die Nachrichtenkomponente wiedergegeben wurden.
styleClass4validInput	nein	Der CSS-Darstellungsklassenname für die Wiedergabe einer gültigen Eingabe.

JSP-Seiten für Task- und Prozessnachrichten entwickeln

Die Business Process Choreographer Explorer-Schnittstelle stellt Standardein- und Standardausgabeformate für die Anzeige und Eingabe von Business-Daten bereit. Mit JSP-Seiten (JSP = JavaServer Pages) können Sie angepasste Ein- und Ausgabeformate bereitstellen.

Wenn Sie benutzerdefinierte JSP-Seiten in den Web-Client aufnehmen möchten, müssen Sie sie bei der Modellierung einer Benutzertask in WebSphere Integration Developer angeben. Sie können beispielsweise JSP-Seiten für eine bestimmte Task und die entsprechenden Ein- und Ausgabenachrichten sowie für einen bestimmten Benutzeraufgabenbereich bzw. alle Benutzeraufgabenbereiche bereitstellen. Zur Laufzeit werden die benutzerdefinierten JSP-Seiten in der Benutzerschnittstelle integriert, damit Ausgabedaten angezeigt und Eingabedaten gesammelt werden.

Die angepassten Formate sind keine eigenständigen Webseiten; es handelt sich um HTML-Fragmente, die Business Process Choreographer Explorer in ein HTML-Format einbettet wie z. B. Fragmente für alle Bezeichnungen und Eingabefelder einer Nachricht.

Wird auf eine Schaltfläche auf der Seite mit den angepassten Formaten geklickt, wird die Eingabe übergeben und in Business Process Choreographer Explorer überprüft. Die Überprüfung basiert auf dem Typ der bereitgestellten Merkmale und der im Browser verwendeten Ländereinstellung. Wenn die Eingabe nicht geprüft werden kann, wird dieselbe Seite erneut angezeigt, und es werden Informationen zu den Gültigkeitsfehlern im Anforderungsattribut messageValidationErrors bereitgestellt. Die Informationen werden als Zuordnung bereitgestellt, wobei der XML-Pfadausdruck (XPath) der ungültigen Merkmale den aufgetretenen Ausnahmebedingungen bei der Überprüfung zugeordnet wird.

Um Business Process Choreographer Explorer angepasste Formate hinzuzufügen, müssen Sie die folgenden Schritte mithilfe von WebSphere Integration Developer durchführen.

1. Erstellen Sie die angepassten Formate.

   Für die benutzerdefinierten JSP-Seiten für die Ein- und Ausgabeformate, die in der Webschnittstelle verwendet werden, ist Zugriff auf die Nachrichtendaten erforderlich. Verwenden Sie für den Zugriff auf die Nachrichtendaten Java-Snippets in einer JSP bzw. die JSP-Ausführungssprache. Auf Daten in den Formaten kann über den Anforderungskontext zugegriffen werden.
2. Ordnen Sie die JSP-Seiten einer Task zu.

   Öffnen Sie die Benutzertask im Editor für Benutzertasks. Geben Sie in den Clienteinstellungen die Position der benutzerdefinierten JSP-Seiten sowie den Aufgabenbereich an, für den das angepasste Format gilt, z. B. Administrator. Die Clienteinstellungen für Business Process Choreographer Explorer werden in der Taskschablone gespeichert. Zur Laufzeit werden diese Einstellungen mit der Taskschablone abgerufen.
3. Packen Sie die benutzerdefinierten JSP-Seiten in ein Webarchiv (WAR-Datei).

   Sie können entweder die WAR-Datei in die EAR-Datei mit dem Modul integrieren, das die Tasks enthält, oder Sie können die WAR-Datei separat implementieren. Wenn die JSPs separat implementiert werden, müssen sie auf dem Server bereitgestellt werden, auf dem der Business Process Choreographer Explorer bzw. der angepasste Client implementiert ist.

Wenn Sie angepasste JSPs für die Prozess- und Tasknachrichten verwenden, müssen Sie die Webmodule, die zur Implementierung der JSPs verwendet werden, denselben Servern zuordnen, denen der angepasste JSF-Client zugeordnet ist.

Die angepassten Formate werden zur Laufzeit im Business Process Choreographer Explorer bereitgestellt.

Benutzerdefinierte JSP-Fragmente

Die benutzerdefinierten JSP-Fragmente (JSP = JavaServer Pages) werden in einen HTML-Tag 'Form' integriert. Während der Laufzeit gliedert Business Process Choreographer Explorer diese Fragmente in die wiedergegebene Seite ein.

Das benutzerdefinierte JSP-Fragment für die Eingabenachricht wird vor dem JSP-Fragment für die Ausgabenachricht integriert.

```
<html....>
   ...
   <form...>
     Eingabe-JSP (Taskeingabenachricht anzeigen)

     Ausgabe-JSP (Taskausgabenachricht anzeigen)

   </form>
   ...
</html>
```

Da die benutzerdefinierten JSP-Fragmente in einen HTML-Tag 'Form' integriert sind, können Sie Eingabeelemente hinzufügen. Der Name des Eingabeelements muss mit dem XPath-Ausdruck des Datenelements übereinstimmen. Es ist wichtig, dass der angegebene Präfixwert als Präfix vor den Namen gesetzt wird:

```
<input id="address"
       type="text"
       name="${prefix}/selectPromotionalGiftResponse/address"
       value="${messageMap['/selectPromotionalGiftResponse/address"]}
       size="60"
       align="left" />
```

Der Präfixwert wird als Anforderungsattribut bereitgestellt. Das Attribut stellt sicher, dass der eingegebene Name in dem einschließenden Format eindeutig ist. Das Präfix wird vom Business Process Choreographer Explorer generiert und sollte nicht geändert werden:

```
String prefix = (String)request.getAttribute("prefix");
```

Das Präfixelement wird nur definiert, wenn die Nachricht in dem angegebenen Kontext bearbeitet werden kann. Ausgabedaten können in Abhängigkeit von dem Status der Benutzertask unterschiedlich angezeigt werden. Beispiel: Wenn sich die Task im Status 'Angefordert' befindet, können die Ausgabedaten geändert werden. Wenn sich die Task jedoch im Status 'Fertig gestellt' befindet, können die Daten lediglich angezeigt werden. In Ihrem JSP-Fragment können Sie testen, ob das Präfixelement vorhanden ist und die Nachricht entsprechend zurückgeben. In der folgenden JSTL-Anweisung sehen Sie, wie Sie testen könne, ob das Präfixelement definiert ist.

```
...
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
...
<c:choose>
  <c:when test="${not empty prefix}">
    <!--Schreib-/Lesemodus-->
```

```
  </c:when>
  <c:otherwise>
    <!--Nur-Lese-Modus-->
  </c:otherwise>
</c:choose>
```

Plug-ins erstellen, um Benutzertaskfunktionen anzupassen

Business Process Choreographer stellt eine Verarbeitungsinfrastruktur für Ereignisse bereit, die während der Verarbeitung von Benutzertasks auftreten. Außerdem werden Plug-in-Punkte bereitgestellt, damit Sie die Funktionen an Ihre Anforderungen anpassen können. Mit den SPIs (SPI = Service Provider Interface) können Sie benutzerdefinierte Plug-ins für das Ereignishandling und für die Verarbeitung von Mitarbeiterabfragen erstellen.

Sie können Plug-ins für API-Benutzertaskereignisse und für Eskalationsbenachrichtigungsereignisse erstellen. Außerdem können Sie ein Plug-in erstellen, das die von der Personalauflösung zurückgegebenen Ergebnisse verarbeitet. Beispielsweise kann es in Zeiten hoher Auslastung hilfreich sein, Benutzer zur Ergebnisliste hinzuzufügen, um die Auslastung zu verteilen.

Sie können Ihre Plug-ins auf verschiedenen Ebenen registrieren, z. B. für alle Tasks auf einer globalen Ebene, für die Tasks in einer Anwendungskomponente, für alle einer Taskschablone zugeordneten Tasks oder für eine einzelne Taskinstanz.

API-Ereignishandler erstellen

Ein API-Ereignis tritt auf, wenn eine Benutzertask von einer API-Methode bearbeitet wird. Verwenden Sie das Plug-in des API-Ereignishandlers der Serviceanbieterschnittstelle (Service Provider Interface, SPI), um Plug-ins zum Verarbeiten der von der API übermittelten Taskereignisse zu erstellen, oder zum Verarbeiten der internen Ereignisse, die über entsprechende API-Ereignisse verfügen.

Führen Sie die folgenden Schritte aus, um einen API-Ereignishandler zu erstellen.

1. Erstellen Sie eine Klasse, die die Schnittstelle APIEventHandlerPlugin2 implementiert oder die Implementierungsklasse APIEventHandler erweitert. Diese Klasse kann die Methoden anderer Klassen aufrufen.
   - Wenn Sie die Schnittstelle APIEventHandlerPlugin2 verwenden, müssen Sie alle Methoden der Schnittstelle APIEventHandlerPlugin2 und der Schnittstelle APIEventHandlerPlugin implementieren.
   - Wenn Sie die SPI-Implementierungsklasse erweitern, überschreiben Sie die Methoden, die Sie benötigen.

   Diese Klasse wird im Kontext einer EJB-Anwendung (EJB = Enterprise JavaBeans) von J2EE (J2EE = Java 2 Enterprise Edition) ausgeführt. Stellen Sie sicher, dass diese Klasse und ihre Helper-Klassen die EJB-Spezifikation einhalten.

   Tipp: Wenn Sie die Schnittstelle HumanTaskManagerService über diese Klasse aufrufen wollen, rufen Sie keine Methode auf, die die Task aktualisiert, von der das Ereignis erzeugt wurde. Diese Aktion würde zu einem Datenbank-Deadlock führen.
2. Assemblieren Sie die Plug-in-Klasse und ihre Helper-Klassen in eine JAR-Datei.

   Wenn die Helper-Klassen von mehreren J2EE-Anwendungen verwendet werden, können Sie diese Klassen in eine separate JAR-Datei einbinden, die Sie als gemeinsam genutzte Bibliothek registrieren.

3. Erstellen Sie eine Serviceanbieter-Konfigurationsdatei für das Plug-in im Verzeichnis META-INF/services/ Ihrer JAR-Datei.

   Die Konfigurationsdatei stellt den Mechanismus zum Identifizieren und Laden des Plug-ins zur Verfügung. Diese Datei entspricht der SPI-Spezifikation von Java 2 (SPI = Service Provider Interface).

   a. Erstellen Sie eine Datei mit dem Namen `com.ibm.task.spi.`*`plug-in_name`*`APIEventHandlerPlugin`, wobei *plug-in_name* der Name des Plug-ins ist.

      Beispiel: Wenn Ihr Plug-in `Kunde` heißt und die Schnittstelle com.ibm-.task.spi.APIEventHandlerPlugin implementiert, lautet der Name der Konfigurationsdatei `com.ibm.task.spi.`**`Kunde`**`APIEventHandlerPlugin`.

   b. Geben Sie in der ersten Zeile dieser Datei, die weder eine Kommentarzeile noch eine Leerzeile ist, den vollständig qualifizierten Namen der Plug-in-Klasse an, die Sie in Schritt 1 erstellt haben.

      Wenn Ihre Plug-in-Klasse beispielsweise `MyAPIEventHandler` heißt und sich im Paket `com.customer.plugins` befindet, muss die erste Zeile der Konfigurationsdatei den folgenden Eintrag enthalten: `com.customer.plugins.MyAPIEventHandler`.

Sie verfügen nun über eine installierbare JAR-Datei, die ein Plug-in zum Verarbeiten von API-Ereignissen enthält, und über eine Serviceanbieter-Konfigurationsdatei, die zum Laden des Plug-ins verwendet werden kann.

Tipp: Es steht nur ein Merkmal eventHandlerName zur Verfügung, mit dem sowohl API-Ereignishandler als auch Benachrichtigungsereignishandler registriert werden können. Wenn Sie sowohl einen API-Ereignishandler als auch einen Benachrichtigungsereignishandler verwenden möchten, müssen die Plug-in-Implementierungen denselben Namen haben wie der Ereignishandler für die SPI-Implementierungen (z. B. `Kunde`).

Sie können beide Plug-ins mithilfe einer einzigen Klasse oder mit zwei separaten Klassen implementieren. In beiden Fällen müssen im Verzeichnis META-INF/services/ Ihrer JAR-Datei zwei Dateien erstellt werden, z. B. com.ibm.task.spi.CustomerNotificationEventHandlerPlugin und com.ibm.task.spi.CustomerAPIEventHandlerPlugin.

Packen Sie die Plug-in-Implementierung und die Helper-Klassen in eine einzige JAR-Datei.

Sie müssen das Plug-in nun installieren und registrieren, damit es zur Laufzeit für den Benutzertaskcontainer verfügbar ist. API-Ereignishandler können Sie mit einer Taskinstanz, einer Taskschablone oder einer Anwendungskomponente registrieren.

API-Ereignishandler

API-Ereignisse treten auf, wenn eine Benutzertask modifiziert wird, oder wenn sich ihr Status ändert. Zum Verarbeiten dieser API-Ereignisse wird der Ereignishandler unmittelbar vor dem Ändern der Task (vorgeschaltete Methode) und unmittelbar vor der Rückgabe des API-Aufrufs (nachgeschaltete Methode) aufgerufen.

Wenn die vorgeschaltete Methode eine ApplicationVetoException-Ausnahmebedingung auslöst, wird die API-Aktion nicht ausgeführt. Die Ausnahmebedingung wird an den Aufrufenden der API zurückgegeben, und die dem Ereignis zugeordnete Transaktion wird rückgängig gemacht. Wenn die vorgeschaltete Methode durch ein internes Ereignis ausgelöst wurde und eine ApplicationVetoException-Ausnahmebedingung verursacht hat, wird das interne Ereignis (z. B. die

automatische Beanspruchung) nicht ausgeführt, aber es wird keine Ausnahmebedingung an die Clientanwendung zurückgegeben. In diesem Fall wird eine Informationsnachricht in die Protokolldatei SystemOut.log geschrieben. Löst die API-Methode während der Verarbeitung eine Ausnahmebedingung aus, wird die Ausnahmebedingung abgefangen und an die nachgeschaltete Methode übergeben. Die Ausnahmebedingung wird nach Rückgabe der nachgeschalteten Methode erneut an den Aufrufenden übergeben.

Folgende Regeln gelten für vorgeschaltete Methoden:

- Vorgeschaltete Methoden empfangen die Parameter der zugeordneten API-Methode oder des internen Ereignisses.
- Vorgeschaltete Methoden können eine ApplicationVetoException-Ausnahmebedingung auslösen, damit die Verarbeitung nicht fortgesetzt wird.

Folgende Regeln gelten für nachgeschaltete Methoden:

- Nachgeschaltete Methoden empfangen die an den API-Aufruf übergebenen Parameter und den Rückgabewert. Wenn die Implementierung der API-Methode eine Ausnahmebedingung auslöst, empfängt die nachgeschaltete Methode diese Ausnahmebedingung ebenfalls.
- Nachgeschaltete Methoden können keine Rückgabewerte ändern.
- Nachgeschaltete Methoden können keine Ausnahmebedingungen auslösen (Laufzeitausnahmebedingungen werden zwar protokolliert, aber sie werden ignoriert).

API-Ereignishandler können Sie mit der Schnittstelle APIEventHandlerPlugin2 (einer Erweiterung der Schnittstelle APIEventHandlerPlugin) implementieren oder durch Erweitern der SPI-Standardimplementierungsklasse com.ibm.task.spi.APIEventHandler. Wenn Ihr Ereignishandler von der Standardimplementierungsklasse abgeleitet wird, implementiert er stets die aktuelle SPI-Version. Wenn Sie ein Upgrade auf eine neuere Version von Business Process Choreographer durchführen, sind weniger Änderungen erforderlich, um die neuen SPI-Methoden zu nutzen.

Wenn Sie über einen Ereignishandler für Benachrichtigungen und über einen API-Ereignishandler verfügen, müssen beide über denselben Namen verfügen, weil nur ein einziger Ereignishandlername registriert werden kann.

Benachrichtigungsereignishandler erstellen

Benachrichtigungsereignisse werden beim Eskalieren von Benutzertasks erzeugt. Business Process Choreographer stellt Funktionen für die Eskalationsverarbeitung zur Verfügung (z. B. Eskalationsarbeitselemente erstellen oder E-Mails senden). Sie können Benachrichtigungsereignishandler erstellen, um die Verarbeitung von Eskalationen anzupassen.

Zum Implementieren von Benachrichtigungsereignishandlern können Sie die Schnittstelle NotificationEventHandlerPlugin verwenden, oder Sie können die Standardklasse für SPI-Implementierung (com.ibm.task.spi.NotificationEventHandler) erweitern.

Führen Sie die folgenden Schritte aus, um einen Benachrichtigungsereignishandler zu erstellen.

1. Erstellen Sie eine Klasse, die die Schnittstelle NotificationEventHandlerPlugin implementiert oder die Implementierungsklasse NotificationEventHandler erweitert. Diese Klasse kann die Methoden anderer Klassen aufrufen.

Wenn Sie die Schnittstelle NotificationEventHandlerPlugin verwenden, müssen Sie alle Methoden der Schnittstelle implementieren. Wenn Sie die SPI-Implementierungsklasse erweitern, überschreiben Sie die Methoden, die Sie benötigen.

Diese Klasse wird im Kontext einer EJB-Anwendung (EJB = Enterprise JavaBeans) von J2EE (J2EE = Java 2 Enterprise Edition) ausgeführt. Stellen Sie sicher, dass diese Klasse und ihre Helper-Klassen die EJB-Spezifikation einhalten.

Das Plug-in wird mit der Berechtigung des Aufgabenbereichs EscalationUser aufgerufen. Dieser Aufgabenbereich wird beim Konfigurieren des Benutzertaskcontainers definiert.

Tipp: Wenn Sie die Schnittstelle HumanTaskManagerService über diese Klasse aufrufen wollen, rufen Sie keine Methode auf, die die Task oder Eskalation aktualisiert, von der das Ereignis erzeugt wurde. Diese Aktion würde zu einem Datenbank-Deadlock führen.

2. Assemblieren Sie die Plug-in-Klasse und ihre Helper-Klassen in eine JAR-Datei.

   Wenn die Helper-Klassen von mehreren J2EE-Anwendungen verwendet werden, können Sie diese Klassen in eine separate JAR-Datei einbinden, die Sie als gemeinsam genutzte Bibliothek registrieren.

3. Erstellen Sie eine Serviceanbieter-Konfigurationsdatei für das Plug-in im Verzeichnis META-INF/services/ Ihrer JAR-Datei.

   Die Konfigurationsdatei stellt den Mechanismus zum Identifizieren und Laden des Plug-ins zur Verfügung. Diese Datei entspricht der SPI-Spezifikation von Java 2 (SPI = Service Provider Interface).

   a. Erstellen Sie eine Datei mit dem Namen `com.ibm.task.spi.`*`plug-in_name`*`NotificationEventHandlerPlugin`, wobei *plug-in_name* der Name des Plug-ins ist.

      Beispiel: Wenn Ihr Plug-in `HelpDeskRequest` heißt (Name des Ereignishandlers) und die Schnittstelle com.ibm.task.spi.NotificationEventHandlerPlugin implementiert, lautet der Name der Konfigurationsdatei `com.ibm.task.spi.`**`HelpDeskRequest`**`NotificationEventHandlerPlugin`.

   b. Geben Sie in der ersten Zeile dieser Datei, die weder eine Kommentarzeile noch eine Leerzeile ist, den vollständig qualifizierten Namen der Plug-in-Klasse an, die Sie in Schritt 1 erstellt haben.

      Wenn Ihre Plug-in-Klasse beispielsweise `MyEventHandler` heißt und sich im Paket `com.customer.plugins` befindet, muss die erste Zeile der Konfigurationsdatei den folgenden Eintrag enthalten: `com.customer.plugins.MyEventHandler`.

Sie verfügen nun über eine installierbare JAR-Datei, die ein Plug-in zum Verarbeiten von Benachrichtigungsereignissen enthält, und über eine Serviceanbieter-Konfigurationsdatei, die zum Laden des Plug-ins verwendet werden kann. API-Ereignishandler können Sie mit einer Taskinstanz, einer Taskschablone oder einer Anwendungskomponente registrieren.

Tipp: Es steht nur ein Merkmal eventHandlerName zur Verfügung, mit dem sowohl API-Ereignishandler als auch Benachrichtigungsereignishandler registriert werden können. Wenn Sie sowohl einen API-Ereignishandler als auch einen Benachrichtigungsereignishandler verwenden möchten, müssen die Plug-in-Implementierungen denselben Namen haben wie der Ereignishandler für die SPI-Implementierungen (z. B. `Kunde`).

Sie können beide Plug-ins mithilfe einer einzigen Klasse oder mit zwei separaten Klassen implementieren. In beiden Fällen müssen im Verzeichnis META-INF/services/ Ihrer JAR-Datei zwei Dateien erstellt werden, z. B. com.ibm.task.spi.CustomerNotificationEventHandlerPlugin und com.ibm.task.spi.CustomerAPIEventHandlerPlugin.

Packen Sie die Plug-in-Implementierung und die Helper-Klassen in eine einzige JAR-Datei.

Sie müssen das Plug-in nun installieren und registrieren, damit es zur Laufzeit für den Benutzertaskcontainer verfügbar ist. Benachrichtigungsereignishandler können Sie mit einer Taskinstanz, einer Taskschablone oder einer Anwendungskomponente registrieren.

Plug-ins für die Nachbearbeitung von Personalabfrageergebnissen erstellen

Die Staff-Auflösung gibt eine Liste der Benutzer zurück, die einem bestimmten Aufgabenbereich zugeordnet sind (z. B. potenzieller Eigner einer Task). Sie können ein Plug-in erstellen, um die Ergebnisse von Personalabfragen zu ändern, die von der Personalauflösung zurückgegeben wurden. Um den Lastausgleich zu verbessern, können Sie ein Plug-in erstellen, das Benutzer, die bereits über eine hohe Auslastung verfügen, aus dem Abfrageergebnis entfernt.

Es darf nur ein Plug-in für Nachverarbeitung vorhanden sein, d. h. dieses Plug-in muss die Personalergebnisse von allen Tasks verarbeiten. Ihr Plug-in kann Benutzer hinzufügen bzw. entfernen sowie Benutzer- oder Gruppeninformationen ändern. Es kann außerdem den Ergebnistyp ändern, z. B. von einer Benutzerliste in eine Gruppe oder in alle Benutzer.

Da das Plug-in erst ausgeführt wird, nachdem die Personalauflösung abgeschlossen ist, wurden Ihre Regeln zur Wahrung der Vertraulichkeit oder Sicherheit bereits angewendet. Das Plug-in empfängt Informationen zu Benutzern, die während der Personalauflösung entfernt wurden (im Zuordnungsschlüssel HTM_REMOVED_USERS). Sie müssen sicherstellen, dass Ihr Plug-in diese Kontextinformationen verwendet, um alle von Ihnen definierten Regeln zur Wahrung von Vertraulichkeit und Sicherheit zu erfüllen.

Verwenden Sie zum Implementieren der Nachbearbeitung von Personalabfrageergebnissen die Schnittstelle StaffQueryResultPostProcessorPlugin. Diese Schnittstelle enthält Methoden zum Ändern der Abfrageergebnisse für Tasks, Eskalationen, Taskschablonen und Anwendungskomponenten.

Führen Sie die folgenden Schritte aus, um ein Plug-in für die Nachbearbeitung von Personalabfrageergebnissen zu erstellen.

1. Erstellen Sie eine Klasse, die die Schnittstelle 'StaffQueryResultPostProcessorPlugin' implementiert.

   Sie müssen alle Methoden der Schnittstelle implementieren. Diese Klasse kann Methoden anderer Klassen aufrufen.

   Diese Klasse wird im Kontext einer EJB-Anwendung (EJB = Enterprise JavaBeans) von J2EE (J2EE = Java 2 Enterprise Edition) ausgeführt. Stellen Sie sicher, dass diese Klasse und ihre Helper-Klassen die EJB-Spezifikation einhalten.

Tipp: Wenn Sie die Schnittstelle HumanTaskManagerService über diese Klasse aufrufen wollen, rufen Sie keine Methode auf, die die Task aktualisiert, von der das Ereignis erzeugt wurde. Diese Aktion würde zu einem Datenbank-Deadlock führen.

Das folgende Beispiel zeigt, wie Sie den Editoraufgabenbereich einer Task mit dem Namen SpecialTask ändern können.

```
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. Assemblieren Sie die Plug-in-Klasse und ihre Helper-Klassen in eine JAR-Datei.

   Wenn die Helper-Klassen von mehreren J2EE-Anwendungen verwendet werden, können Sie diese Klassen in eine separate JAR-Datei einbinden, die Sie als gemeinsam genutzte Bibliothek registrieren.

3. Erstellen Sie eine Serviceanbieter-Konfigurationsdatei für das Plug-in im Verzeichnis META-INF/services/ Ihrer JAR-Datei.

   Die Konfigurationsdatei stellt den Mechanismus zum Identifizieren und Laden des Plug-ins zur Verfügung. Diese Datei entspricht der SPI-Spezifikation von Java 2 (SPI = Service Provider Interface).

   a. Erstellen Sie eine Datei mit dem Namen `com.ibm.task.spi.`*`plug-in_name`*`StaffQueryResultPostProcessorPlugin`, wobei *plug-in_name* der Name des Plug-ins ist.

      Beispiel: Wenn Ihr Plug-in `MeinHandler` heißt und die Schnittstelle com.ibm-.task.spi.StaffQueryResultPostProcessorPlugin implementiert, lautet der Name der Konfigurationsdatei `com.ibm.task.spi.`**`MeinHandler`**`StaffQueryResultPostProcessorPlugin`.

   b. Geben Sie in der ersten Zeile dieser Datei, die weder eine Kommentarzeile noch eine Leerzeile ist, den vollständig qualifizierten Namen der Plug-in-Klasse an, die Sie in Schritt 1 erstellt haben.

      Wenn Ihre Plug-in-Klasse beispielsweise `StaffPostProcessor` heißt und sich im Paket `com.customer.plugins` befindet, muss die erste Zeile der Konfigurationsdatei den folgenden Eintrag enthalten: `com.customer.plugins.StaffPostProcessor`. Sie verfügen nun über eine installierbare JAR-Datei, die ein Plug-in enthält, das Personalabfrageergebnisse nachträglich verarbeitet, und über eine Serviceanbieter-Konfigurationsdatei, die zum Laden des Plug-ins verwendet werden kann.

4. Installieren Sie das Plug-in.

   Es darf nur ein Plug-in für die Nachbearbeitung von Personalabfrageergebnissen vorhanden sein. Sie müssen das Plug-in als gemeinsam genutzte Bibliothek installieren.
5. Registrieren Sie das Plug-in.
   a. Rufen Sie in der Administrationskonsole die Seite für benutzerdefinierte Merkmale des Human Task Managers auf (über **Anwendungsserver** → *servername* → **Human Task Container** → **Benutzerdefinierte Merkmale**).
   b. Fügen Sie ein benutzerdefiniertes Merkmal mit dem Namen **Staff.PostProcessorPlugin** hinzu und einen Wert mit dem Namen, den Sie Ihrem Plug-in zugeordnet haben (in diesem Beispiel `MeinHandler`).

Plug-ins installieren

Um ein Plug-in zu verwenden, müssen Sie es so installieren, dass der Taskcontainer darauf zugreifen kann.

Wie das Plug-in installiert wird, ist davon abhängig, ob es nur von einer einzigen J2EE-Anwendung (J2EE = Java 2 Enterprise Edition) oder von mehreren Anwendungen verwendet werden soll.

Führen Sie einen der folgenden Schritte aus, um ein Plug-in zu installieren.

- Installieren Sie ein Plug-in für die Verwendung durch eine einzige J2EE-Anwendung.

  Fügen Sie die JAR-Datei Ihres Plug-ins zur EAR-Datei der Anwendung hinzu. Installieren Sie die JAR-Datei für Ihr Plug-in im Implementierungsdeskriptoreditor von WebSphere Integration Developer als JAR-Datei des Projektdienstprogramms für die J2EE-Anwendung des JavaBeans-Hauptmoduls (EJB).
- Installieren Sie ein Plug-in für die Verwendung durch mehrere J2EE-Anwendungen.

  Fügen Sie die JAR-Datei in eine gemeinsam genutzte WebSphere Application Server-Bibliothek ein, und ordnen Sie der Bibliothek die Anwendungen zu, die auf das Plug-in zugreifen sollen. Um die JAR-Datei in einer Netzimplementierungsumgebung bereitzustellen, verteilen Sie die JAR-Datei manuell auf jeden Server, und installieren Sie die gemeinsam genutzte Bibliothek anschließend für jede Zelle einmal.

Sie können nun das Plug-in registrieren.

Plug-ins registrieren

Sie können Ihre Plug-ins auf verschiedenen Ebenen in der Hierarchie der Taskcontainer-Artefakte registrieren. Beispielsweise für alle Tasks auf einer globalen Ebene, für die Tasks einer Anwendungskomponente, für alle einer Taskschablone zugeordneten Tasks oder für eine einzelne Taskinstanz.

Beim Registrieren mehrerer Plug-ins wird Scoping unterstützt. Dies bedeutet, dass ein auf einer unteren Ebene der Hierarchie für Taskcontainer-Artefakte registriertes Plug-in (z. B. eine Taskinstanz) anstelle des Plug-ins verwendet wird, das auf einer höheren Ebene registriert ist (z. B. eine Taskschablone oder Anwendungskomponente). Scoping wird für alle Hierarchieebenen unterstützt. Der Taskcontainer verwendet das auf der untersten Ebene der Hierarchie registrierte Plug-in.

Ein Plug-in kann auf einer der folgenden Arten registriert werden.

- Registrieren Sie das Plug-in im Taskmodell.

  Geben Sie im Taskeditor von WebSphere Integration Developer auf der Seite 'Details' des Merkmalsbereichs für die Task den Namen des Ereignishandlers im Feld **Name des Ereignishandlers** an.
- Registrieren Sie das Plug-in für Ad-hoc-Tasks oder Taskschablonen, die Sie zur Laufzeit erstellen.

  Verwenden Sie die Methode setEventHandlerName der Klasse TTask, um den Namen des Ereignishandlers zu registrieren.
- Ändern Sie den registrierten Ereignishandler für eine Taskinstanz während der Laufzeit.

  Verwenden Sie die Methode update(Task task), um zur Laufzeit einen anderen Ereignishandler für eine Taskinstanz zu verwenden. Der Aufrufende muss über die Taskadministratorberechtigung verfügen, um dieses Merkmal zu aktualisieren.
- Registrieren Sie das Plug-in auf einer globalen Ebene.

  Definieren Sie in der Administrationskonsole auf der Seite Benutzerdefinierte Merkmale für den Benutzertaskcontainer ein benutzerdefiniertes Merkmal für das Plug-in. Der Wert des benutzerdefinierten Merkmals ist der Plug-in-Name.

Teil 2. Anwendungen implementieren

Kapitel 3. Übersicht über das Vorbereiten und Installieren von Modulen

Beim Installieren (Implementieren) von Modulen werden die Module in einer Testumgebung oder einer Produktionsumgebung aktiviert. Diese Übersicht enthält eine Kurzbeschreibung der Test- und Produktionsumgebungen sowie einige der erforderlichen Schritte beim Installieren von Modulen.

Anmerkung: Der Prozess zum Installieren von Anwendungen in einer Produktionsumgebung entspricht ungefähr dem im Abschnitt „Anwendungen entwickeln und implementieren" im Information Center für WebSphere Application Server Network Deployment, Version 6 beschriebenen Prozess. Wenn Sie diese Abschnitte noch nicht kennen, lesen Sie sie zuerst.

Verifizieren Sie Änderungen zunächst in einer Testumgebung, bevor Sie ein Modul in einer Produktionsumgebung installieren. Verwenden Sie WebSphere Integration Developer, um Module in einer Testumgebung zu installieren (weitere Informationen enthält das Information Center von WebSphere Integration Developer). Verwenden Sie WebSphere Process Server, um Module in einer Produktionsumgebung zu installieren.

In diesem Abschnitt werden die erforderlichen Konzepte und Tasks zum Vorbereiten und Installieren von Modulen in einer Produktionsumgebung beschrieben. Andere Abschnitte beschreiben die Dateien, in denen die von Ihren Modulen verwendeten Objekte enthalten sind, und unterstützen Sie beim Versetzen Ihres Moduls aus der Testumgebung in die Produktionsumgebung. Machen Sie sich mit diesen Dateien und ihrem Inhalt unbedingt gut vertraut, um sicherzugehen, dass Sie Ihre Module korrekt installiert haben.

Übersicht über Bibliotheken und JAR-Dateien

Module verwenden häufig Artefakte, die sich in Bibliotheken befinden. Artefakte und Bibliotheken sind in Java-Archivdateien (JAR-Dateien) enthalten, die Sie beim Implementieren eines Moduls angeben.

Beim Entwickeln eines Moduls geben Sie möglicherweise bestimmte Ressourcen oder Komponenten an, die von verschiedenen Teilen des Moduls verwendet werden können. Diese Ressourcen oder Komponenten können Objekte sein, die Sie beim Entwickeln des Moduls erstellt haben, oder bereits bestehende Objekte, die sich in einer bereits auf dem Server implementierten Bibliothek befinden. In diesem Abschnitt werden die Bibliotheken und Dateien beschrieben, die Sie zum Installieren einer Anwendung benötigen.

Was ist eine Bibliothek?

Eine Bibliothek enthält Objekte oder Ressourcen, die von mehreren Modulen in WebSphere Integration Developer verwendet werden. Die Artefakte können in JAR-, RAR- (Resource Archive) oder WAR-Dateien (Web Service Archive) enthalten sein. Zu diesen Artefakten gehören:

- Schnittstellen oder Web-Services-Deskriptoren (Dateien mit der Erweiterung .wsdl)

- XML-Schemadefinitionen für Business-Objekte (Dateien mit der Erweiterung .xsd)
- Business-Objektzuordnungen (Dateien mit der Erweiterung .map)
- Beziehungs- und Aufgabenbereichsdefinitionen (Dateien mit den Erweiterungen .rel und .rol)

Wenn ein Modul ein Artefakt benötigt, lokalisiert und lädt der Server das Artefakt aus dem EAR-Klassenpfad, wenn es nicht bereits in den Speicher geladen ist. Ab diesem Zeitpunkt verwendet jede Anforderung für das Artefakt diese geladene Kopie, bis sie ersetzt wird. Abb. 5 zeigt, dass eine Anwendung Komponenten und dazugehörige Bibliotheken enthält.

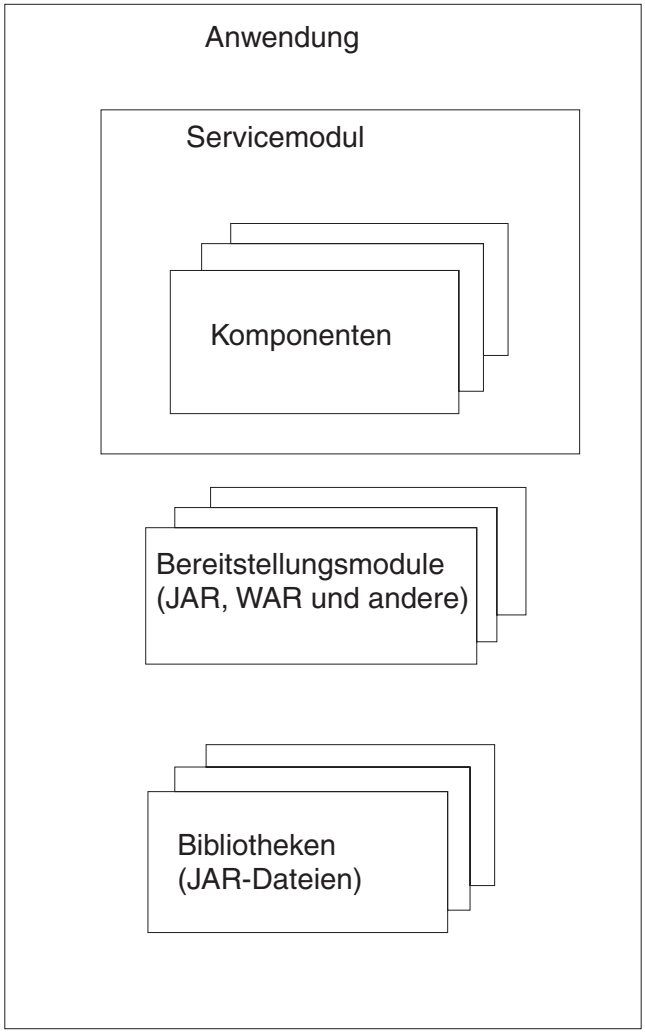


Abbildung 5. Beziehung zwischen Modul, Komponente und Bibliothek

Was sind JAR-, RAR- und WAR-Dateien?

Verschiedene Dateien können Komponenten eines Moduls enthalten. Diese Dateien werden in der Spezifikation Java Platform, Enterprise Edition ausführlich beschrieben. Details zu JAR-Dateien sind in der JAR-Spezifikation zu finden.

WebSphere Process Server enthält eine JAR-Datei außerdem eine Anwendung, d. h. die assemblierte Version des Moduls mit allen unterstützenden Referenzen und Schnittstellen zu allen anderen, von dem Modul verwendeten Servicekomponenten.

Zum vollständigen Installieren der Anwendung benötigen Sie diese JAR-Datei, alle weiteren Bibliotheken, z. B. JAR-Dateien, WAR-Dateien (Web Services Archives), RAR-Dateien (Resources Archives), JAR-Dateien für Bereitstellungsbibliotheken (Enterprise Java Beans - EJB) bzw. alle anderen Archive. Daraus erstellen Sie mit dem Befehl serviceDeploy eine installierbare EAR-Datei.

Namenskonventionen für Bereitstellungsmodule

Innerhalb der Bibliothek gelten bestimmte Voraussetzungen für die Namen der Bereitstellungsmodule. Diese Namen sind für ein bestimmtes Modul eindeutig. Benennen Sie alle anderen zum Implementieren der Anwendung erforderlichen Module so, dass keine Konflikte mit den Namen der Bereitstellungsmodule entstehen. Für ein Modul namens *myService* lauten die Namen der Bereitstellungsmodule wie folgt:

- *myService*App
- *myService*EJB
- *myService*EJBClient
- *myService*Web

Anmerkung: Der Befehl serviceDeploy erstellt das Webbereitstellungsmodul *myService* nur, wenn der Service einen Service mit dem Porttyp WSDL enthält.

Überlegungen zur Verwendung von Bibliotheken

Durch die Verwendung von Bibliotheken wird die Konsistenz von Business-Objekten und die Konsistenz der Verarbeitung zwischen Modulen gewährleistet, da jedes aufrufende Modul über eine eigene Kopie einer spezifischen Komponente verfügt. Zur Vermeidung von Inkonsistenzen und Fehlern sollte unbedingt sichergestellt werden, dass Änderungen an den von aufrufenden Modulen verwendeten Komponenten und Business-Objekten zwischen allen aufrufenden Modulen koordiniert werden. Aktualisieren Sie die aufrufenden Module wie folgt:

1. Kopieren Sie das Modul und die aktuelle Kopie der Bibliotheken auf den Produktionsserver
2. Erstellen Sie die installierbare EAR-Datei erneut mit dem Befehl serviceDeploy
3. Stoppen Sie die aktive Anwendung, die das aufrufende Modul enthält, und installieren Sie sie erneut
4. Starten Sie die Anwendung erneut, die das aufrufende Modul enthält

Zugehörige Verweise

Befehl serviceDeploy
Mit dem Befehl serviceDeploy können Sie SCA-kompatible Module als Java-Anwendungen einbinden, die auf einem Server installiert werden können. Dieser Befehl ist bei der Durchführung von Stapelinstallationen über wsadmin nützlich.

Übersicht über die EAR-Datei

Eine EAR-Datei ist von zentraler Bedeutung beim Implementieren einer Serviceanwendung auf einem Produktionsserver.

Eine EAR-Datei (Enterprise Archive-Datei) ist eine komprimierte Datei, die Bibliotheken, Enterprise-Beans und JAR-Dateien enthält, die zum Implementieren einer Anwendung erforderlich sind.

Eine JAR-Datei wird erstellt, wenn Sie Ihre Anwendungsmodule aus WebSphere Integration Developer exportieren. Verwenden Sie diese JAR-Datei und alle anderen Artefaktbibliotheken oder -objekte als Eingabedaten für den Installationsprozess. Der Befehl serviceDeploy erstellt aus den Eingabedateien mit Komponentenbeschreibungen und Java-Code, aus denen die Anwendung besteht, eine EAR-Datei.

Zugehörige Verweise

Befehl serviceDeploy
Mit dem Befehl serviceDeploy können Sie SCA-kompatible Module als Java-Anwendungen einbinden, die auf einem Server installiert werden können. Dieser Befehl ist bei der Durchführung von Stapelinstallationen über wsadmin nützlich.

Implementierung auf einem Server vorbereiten

Nach dem Entwickeln und Testen eines Moduls müssen Sie das Modul von dem Testsystem exportieren und zum Implementieren in eine Produktionsumgebung übertragen. Zum Installieren einer Anwendung müssen Sie außerdem die erforderlichen Pfade zum Exportieren des Moduls und alle für das Modul erforderlichen Bibliotheken kennen.

Vor dem Ausführen dieser Task müssen die Module auf einem Testserver entwickelt und getestet sein. Außerdem müssen Fehler und Leistungsprobleme behoben sein.

Durch diese Task wird sichergestellt, dass alle erforderlichen Bestandteile einer Anwendung verfügbar sind und in die richtigen Dateien zur Übertragung auf den Produktionsserver gepackt sind.

Anmerkung: Sie können auch eine EAR-Datei (Enterprise Archive-Datei) aus WebSphere Integration Developer exportieren und direkt in WebSphere Process Server installieren.

Wichtig: Wenn die Services in einer Komponente auf eine Datenbank zurückgreifen, installieren Sie die Anwendung auf einem Server mit direkter Verbindung zu der Datenbank.

1. Lokalisieren Sie den Ordner, der die Komponenten für das Modul enthält, das Sie implementieren möchten.

   Der Komponentenordner muss *modulname* heißen und eine Datei mit dem Namen *modul*.module (der Name des Basismoduls) enthalten.
2. Stellen Sie sicher, dass sich alle in dem Modul enthaltenen Komponenten in untergeordneten Komponentenordnern des Modulordners befinden.

   Ordnen Sie dem untergeordneten Ordner zur einfacheren Erkennung den Namen *modul/komponente* oder einen ähnlichen Namen zu.
3. Stellen Sie sicher, dass alle Dateien, die zu den verschiedenen Komponenten gehören, in dem entsprechenden Komponentenordner enthalten sind und einen ähnlichen Namen wie *komponentendateiname*.component haben.

   Die Komponentendateien enthalten die Definitionen für jede einzelne Komponente in dem Modul.
4. Stellen Sie sicher, dass sich alle anderen Komponenten und Artefakte in den entsprechenden Unterverzeichnissen der Komponente befinden, für die sie erforderlich sind.

Mit diesem Schritt wird sichergestellt, dass alle Verweise auf Artefakte, die für eine Komponente erforderlich sind, zur Verfügung stehen. Die Namen von Komponenten dürfen nicht mit den Namen in Konflikt stehen, die der Befehl serviceDeploy für Bereitstellungsmodule verwendet. Siehe Namenskonventionen für Bereitstellungsmodule.

5. Stellen Sie sicher, dass eine Verweisdatei *modul*.references in dem Modulordner aus dem Schritt 1 auf Seite 184 vorhanden ist.

   Die Verweisdatei definiert die Verweise und die Schnittstellen in dem Modul.

6. Vergewissern Sie sich, dass eine Verbindungsdatei *modul*.wires, in dem Komponentenordner vorhanden ist.

   Die Verbindungsdatei vervollständigt die Verbindungen zwischen den Verweisen und den Schnittstellen in dem Modul.

7. Vergewissern Sie sich, dass eine Manifestdatei *modul*.manifest in dem Komponentenordner vorhanden ist.

   In der Manifestdatei ist das Modul mit allen dazugehörigen Komponenten aufgelistet. Die Datei enthält außerdem eine Klassenpfadanweisung, damit der Befehl serviceDeploy alle weiteren für das Modul erforderlichen Module lokalisieren kann.

8. Erstellen Sie eine komprimierte Datei oder JAR-Datei des Moduls als Eingabe für den Befehl serviceDeploy, der das Modul für die Installation auf dem Produktionsserver vorbereitet.

Beispiel der Ordnerstruktur für das Modul MyValue vor der Implementierung

Das folgende Beispiel veranschaulicht die Verzeichnisstruktur für das Modul MyValueModule, das aus den Komponenten MyValue, CustomerInfo und StockQuote besteht.

```
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
```

Installieren Sie das Modul auf den Produktionssystemen wie unter Modul auf einem Produktionsserver installierenModul auf einem Produktionsserver installieren beschrieben.

Zugehörige Verweise

Befehl serviceDeploy
Mit dem Befehl serviceDeploy können Sie SCA-kompatible Module als Java-Anwendungen einbinden, die auf einem Server installiert werden können. Dieser Befehl ist bei der Durchführung von Stapelinstallationen über wsadmin nützlich.

Überlegungen zur Installation von Serviceanwendungen auf Clustern

Die Installation einer Serviceanwendung auf einem Cluster stellt zusätzliche Anforderungen an Sie. Es ist wichtig, dass Sie diese Überlegungen bei der Installation von Serviceanwendungen auf einem Cluster beachten.

Cluster können für Ihre Verarbeitungsumgebung von Vorteil sein, indem sie Größenvorteile bereitstellen, die Ihnen beim Ausgleich Ihres Anforderungsworkloads serverübergreifend behilflich sind; darüber hinaus geben sie Auskunft über die Verfügbarkeit von Clients der Anwendungen. Beachten Sie Folgendes, bevor Sie eine Anwendung installieren, die Services auf einem Cluster enthält:

- Ist für Benutzer der Anwendung die durch das Clustering bereitgestellte Verarbeitungsleistung und Verfügbarkeit erforderlich?

  Ist dies der Fall, so ist Clustering die richtige Lösung. Durch Clustering erhöht sich die Verfügbarkeit und die Kapazität Ihrer Anwendungen.
- Wurde der Cluster für Serviceanwendungen richtig vorbereitet?

  Vor der Installation und dem Start der ersten Anwendung, die einen Service enthält, müssen Sie den Cluster richtig konfigurieren. Wenn der Cluster nicht richtig konfiguriert wird, können die Anwendungen die Anforderungen nicht ordnungsgemäß verarbeiten.
- Gibt es einen Ausweichcluster?

  Sie müssen die Anwendung auch auf dem Ausweichcluster installieren.

Kapitel 4. Modul auf einem Produktionsserver installieren

In diesem Artikel werden die Schritte zum Übertragen einer Anwendung von einem Testserver auf einen Produktionsserver beschrieben.

Vor dem Implementieren auf einem Produktionsserver sollte die betreffende Serviceanwendung zunächst auf einem Testserver installiert, und getestet werden. Exportieren Sie die relevanten Dateien nach dem Testen, wie unter *Implementierung auf einem Server vorbereiten* in der vorliegenden PDF 'Module entwickeln und implementieren' beschrieben, und übertragen Sie die Dateien zum Implementieren auf das Produktionssystem. Weitere Informationen finden Sie in den Information Centers zu WebSphere Integration Developer und WebSphere Application Server Network Deployment.

1. Kopieren Sie das Modul und die übrigen Dateien auf den Produktionsserver.

   Die für die Anwendung erforderlichen Module und Ressourcen (EAR-, JAR-, RAR- und WAR-Dateien) werden in Ihre Produktionsumgebung versetzt.
2. Führen Sie den Befehl serviceDeploy aus, um eine installierbare EAR-Datei zu erstellen.

   In diesem Schritt wird das Modul für den Server definiert, als Vorbereitung zum Installieren der Anwendung in der Produktionsumgebung.

   a. Lokalisieren Sie die JAR-Datei, in der das zu implementierende Modul enthalten ist.
   b. Rufen Sie den Befehl auf, und verwenden Sie dabei die JAR-Datei aus dem vorherigen Schritt als Eingabedatei.
3. Installieren Sie die EAR-Datei wie in Schritt 2 beschrieben. Die Vorgehensweise bei der Installation der Anwendungen ist davon abhängig, ob Sie die Anwendung auf einem eigenständigen Server oder auf einem Server in einer Zelle installieren.

   Anmerkung: Für die Installation der Anwendung können Sie entweder die Administrationskonsole oder ein Script verwenden. Weitere Informationen finden Sie im Information Center von WebSphere Application Server.
4. Speichern Sie die Konfiguration. Das Modul ist jetzt als Anwendung installiert.
5. Starten Sie die Anwendung.

Die Anwendung ist jetzt aktiv und das Modul müsste Datenfluss abwickeln.

Überwachen Sie die Anwendung, um sicherzustellen, dass der Server Anforderungen korrekt verarbeitet.

Zugehörige Verweise

Befehl serviceDeploy
Mit dem Befehl serviceDeploy können Sie SCA-kompatible Module als Java-Anwendungen einbinden, die auf einem Server installiert werden können. Dieser Befehl ist bei der Durchführung von Stapelinstallationen über wsadmin nützlich.

Installierbare EAR-Datei mit serviceDeploy erstellen

Erstellen Sie zum Installieren einer Anwendung in der Produktionsumgebung eine installierbare EAR-Datei aus den auf den Produktionsserver kopierten Dateien.

Damit diese Task gestartet werden kann, muss eine JAR-Datei vorhanden sein, in der die Module und Services enthalten sind, die Sie auf dem Server implementieren möchten. Weitere Informationen finden Sie unter Implementierung auf einem Server vorbereiten.

Der Befehl serviceDeploy erstellt aus einer JAR-Datei und allen abhängigen EAR-, JAR-, RAR-, WAR- und ZIP-Dateien eine EAR-Datei, die Sie auf einem Server installieren können.

1. Lokalisieren Sie die JAR-Datei, in der das zu implementierende Modul enthalten ist.
2. Rufen Sie den Befehl auf, und verwenden Sie dabei die JAR-Datei aus dem vorherigen Schritt als Eingabedatei.

   In diesem Schritt wird eine EAR-Datei erstellt.

   Anmerkung: Führen Sie die folgenden Schritte in der Administrationskonsole aus.
3. Wählen Sie die EAR-Datei aus, um die Administrationskonsole des Servers zu installieren.
4. Klicken Sie auf **Speichern**, um die EAR-Datei zu installieren.

Zugehörige Verweise

Befehl serviceDeploy
Mit dem Befehl serviceDeploy können Sie SCA-kompatible Module als Java-Anwendungen einbinden, die auf einem Server installiert werden können. Dieser Befehl ist bei der Durchführung von Stapelinstallationen über wsadmin nützlich.

Anwendungen über Apache Ant-Tasks implementieren

In diesem Artikel wird beschrieben, wie die Implementierung von Anwendungen in WebSphere Process Server mithilfe von Apache Ant-Tasks automatisiert werden kann. Bei Verwendung von Apache Ant-Tasks können Sie die Implementierung mehrerer Anwendungen definieren und unbeaufsichtigt auf einem Server ausführen lassen.

In dieser Task wird Folgendes vorausgesetzt:
- Die zu implementierenden Anwendungen sind bereits entwickelt und getestet
- Die Anwendungen sollen auf demselben Server oder denselben Servern installiert werden
- Sie verfügen über Grundkenntnisse zu Apache Ant-Tasks
- Sie kennen den Implementierungsprozess

Informationen zum Entwickeln und Testen von Anwendungen sind im Information Center von WebSphere Integration Developer enthalten.

Der Referenzteil des Information Center für WebSphere Application Server Network Deployment enthält einen Abschnitt über Anwendungsprogrammierschnittstellen. Apache Ant-Tasks werden im Paket com.ibm.websphere.ant.tasks beschrieben. Für diesen Artikel sind die Tasks ServiceDeploy und InstallApplication von Bedeutung.

Wenn Sie mehrere Anwendungen gleichzeitig installieren müssen, erstellen Sie vor dem Implementieren eine Apache Ant-Task. Diese Apache Ant-Task kann anschließend das Implementieren und Installieren der Anwendungen auf den Servern ohne Ihr Eingreifen erledigen.

1. Geben Sie die zu implementierenden Anwendungen an.
2. Erstellen Sie für jede Anwendung eine JAR-Datei.
3. Kopieren Sie die JAR-Dateien auf die Zielserver.
4. Erstellen Sie eine Apache Ant-Task, die den Befehl ServiceDeploy aufruft, um die EAR-Datei für jeden Server zu erstellen.
5. Erstellen Sie eine Apache Ant-Task, die den Befehl InstallApplication für jede EAR-Datei aus dem Schritt 4 auf den betreffenden Servern ausführt.
6. Führen Sie die Apache Ant-Task für ServiceDeploy aus, um die EAR-Datei für die Anwendungen zu erstellen.
7. Führen Sie die Apache Ant-Task für InstallApplication aus, um die EAR-Dateien aus dem Schritt 6 zu installieren.

Die Anwendungen sind auf den Zielservern korrekt implementiert.

Beispiel für unbeaufsichtigtes Implementieren einer Anwendung

Dieses Beispiel zeigt eine Apache Ant-Task, die in der Datei myBuildScript.xml enthalten ist.

```
<?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>
```

Die folgende Anweisung veranschaulicht, wie die Apache Ant-Task aufgerufen wird.

```
${WAS}/bin/ws_ant -f myBuildScript.xml
```

Tipp: Es können mehrere Anwendungen unbeaufsichtigt implementiert werden, indem weitere Projektanweisungen in die Datei eingefügt werden.

Vergewissern Sie sich mithilfe der Administrationskonsole, dass die neu installierten Anwendungen gestartet sind und den Datenfluss korrekt verarbeiten.

Zugehörige Verweise

Befehl serviceDeploy
Mit dem Befehl serviceDeploy können Sie SCA-kompatible Module als Java-Anwendungen einbinden, die auf einem Server installiert werden können. Dieser Befehl ist bei der Durchführung von Stapelinstallationen über wsadmin nützlich.

Kapitel 5. Business-Prozess- und Benutzertaskanwendungen installieren

SCA-Module (SCA = Service Component Architecture), die Business-Prozesse und/oder Benutzertasks enthalten, können auf Implementierungsziele verteilt werden. Implementierungsziel kann ein Server oder ein Cluster sein.

Stellen Sie sicher, dass der Business Flow Manager und/oder der Human Task Manager für jeden Anwendungsserver bzw. Cluster installiert und konfiguriert ist/sind, auf dem Sie Ihre Anwendung installieren möchten.

Sie können Business-Prozess- und Taskanwendungen beispielsweise über die Administrationskonsole installieren, über die Befehlszeile oder durch Ausführen eines Verwaltungsscripts.

Nach der Installation einer Business-Prozess- oder Benutzertaskanwendung werden alle Business-Prozess- und Benutzertaskschablonen in den Startstatus versetzt. Aus diesen Schablonen können Sie Prozessinstanzen und Taskinstanzen erstellen.

Bevor Prozessinstanzen oder Taskinstanzen erstellt werden können, müssen Sie die Anwendung starten.

Zugehörige Konzepte

„Implementierung von Business-Prozessen und Benutzertasks" auf Seite 192
Wenn WebSphere Integration Developer oder die Serviceimplementierung den Implementierungscodes für Ihren Prozess oder Ihre Task generiert, werden die einzelnen Prozesskomponenten oder Taskkomponenten einer Session-Enterprise-Bean zugeordnet. Der gesamte Implementierungscode wird in der Datei der Unternehmensanwendung (EAR-Datei) zusammengefasst. Außerdem wird bei der Installation der Unternehmensanwendung für jeden Prozess eine Java-Klasse, die Java-Code in diesem Prozess darstellt, generiert und in die EAR-Datei integriert. Jede neue Version eines Modells, das implementiert werden soll, muss in eine neue Unternehmensanwendung geschrieben werden.

„Installation von Business-Prozessanwendungen und Benutzertaskanwendungen in einer Network Deployment-Umgebung"
Bei der Installation von Prozessschablonen oder Benutzertaskschablonen in einer Network Deployment-Umgebung werden die folgenden Aktionen von der Anwendungsinstallation automatisch durchgeführt.

Installation von Business-Prozessanwendungen und Benutzertaskanwendungen in einer Network Deployment-Umgebung

Bei der Installation von Prozessschablonen oder Benutzertaskschablonen in einer Network Deployment-Umgebung werden die folgenden Aktionen von der Anwendungsinstallation automatisch durchgeführt.

Die Anwendung wird schrittweise asynchron installiert. Jede Stufe muss vollständig abgeschlossen sein, bevor mit der nächsten Stufe begonnen werden kann.

1. Die Anwendungsinstallation wird mit dem Deployment Manager gestartet.

   Während dieser Stufe werden die Business-Prozessschablonen und die Benutzertaskschablonen im WebSphere-Konfigurationsrepository konfiguriert. Die Anwendung wird auch geprüft. Falls Fehler auftreten, werden diese in der

© IBM Corporation 2005, 2007

Datei 'System.out' oder in der Datei 'System.err' aufgezeichnet oder als FFDC-Einträge (FFDC = First-Failure Data Capture) im Deployment Manager registriert.

2. Die Anwendungsinstallation wird auf dem Knotenagenten fortgesetzt.

   Bei dieser Stufe wird die Installation der Anwendung auf einer Anwendungsserverinstanz ausgelöst. Diese Anwendungsserverinstanz ist entweder Teil des Implementierungsziels, oder sie ist das Implementierungsziel. Wenn das Implementierungsziel ein Cluster mit mehreren Cluster-Membern ist, wird die Serverinstanz aus den Cluster-Membern dieses Clusters beliebig ausgewählt. Wenn bei dieser Stufe Fehler auftreten, werden diese in der Datei 'SystemOut-.log' oder in der Datei 'SystemErr.log' aufgezeichnet oder als FFDC-Einträge (FFDC = First-Failure Data Capture) im Knotenagenten registriert.

3. Die Anwendung wird auf der Serverinstanz ausgeführt.

   Während dieser Stufe werden die Business-Prozessschablonen und die Benutzertaskschablonen in die Business Process Choreographer-Datenbank auf dem Implementierungsziel implementiert. Falls Fehler auftreten, werden diese in der Datei 'System.out' oder in der Datei 'SystemErr.log' aufgezeichnet oder als FFDC-Einträge (FFDC = First-Failure Data Capture) auf dieser Serverinstanz registriert.

   Zugehörige Tasks

   Kapitel 5, „Business-Prozess- und Benutzertaskanwendungen installieren", auf Seite 191
   SCA-Module (SCA = Service Component Architecture), die Business-Prozesse und/oder Benutzertasks enthalten, können auf Implementierungsziele verteilt werden. Implementierungsziel kann ein Server oder ein Cluster sein.

Implementierung von Business-Prozessen und Benutzertasks

Wenn WebSphere Integration Developer oder die Serviceimplementierung den Implementierungscodes für Ihren Prozess oder Ihre Task generiert, werden die einzelnen Prozesskomponenten oder Taskkomponenten einer Session-Enterprise-Bean zugeordnet. Der gesamte Implementierungscode wird in der Datei der Unternehmensanwendung (EAR-Datei) zusammengefasst. Außerdem wird bei der Installation der Unternehmensanwendung für jeden Prozess eine Java-Klasse, die Java-Code in diesem Prozess darstellt, generiert und in die EAR-Datei integriert. Jede neue Version eines Modells, das implementiert werden soll, muss in eine neue Unternehmensanwendung geschrieben werden.

Beim Installieren einer Unternehmensanwendung mit Business-Prozessen oder Benutzertasks werden diese als Business-Prozessschablonen bzw. Benutzertaskschablonen in der Business Process Choreographer-Datenbank gespeichert. Neu installierte Schablonen werden standardmäßig gestartet. Die neu installierte Unternehmensanwendung ist jedoch standardmäßig gestoppt. Jede installierte Unternehmensanwendung kann einzeln gestartet und gestoppt werden.

Sie können viele verschiedene Versionen einer Prozess- oder Taskschablone implementieren, jede Version in einer anderen Anwendung. Wenn Sie eine neue Unternehmensanwendung installieren, wird die Version der Schablone, die installiert ist, wie folgt ermittelt:

- Wenn der Name der Schablone und der Zielnamespace noch nicht vorhanden sind, wird eine neue Schablone installiert.

- Wenn der Schablonenname und der Zielnamespace mit dem Namen und dem Zielnamespace einer vorhandenen Schablone übereinstimmten, das Gültigkeitsstartdatum aber ein anderes ist, wird eine neue Version einer vorhandenen Schablone installiert.

Anmerkung: Der Schablonenname wird aus dem Namen der Komponente und nicht aus dem Business-Prozess oder der Benutzertask abgeleitet.

Wenn Sie kein zulässiges Gültigkeitsstartdatum angeben, wird das Datum wie folgt festgelegt:

- Bei der Verwendung von WebSphere Integration Developer ist das Gültigkeitsstartdatum das Datum, an dem die Benutzertask oder der Business-Prozess modelliert wurde.
- Wenn Sie die Serviceimplementierung verwenden, ist das Gültigkeitsstartdatum das Datum, an dem der Befehl serviceDeploy ausgeführt wurde. Nur Collaboration-Tasks erhalten das Datum, an dem die Anwendung mit dem Gültigkeitsstartdatum installiert wurde.

Zugehörige Tasks

Kapitel 5, „Business-Prozess- und Benutzertaskanwendungen installieren", auf Seite 191
SCA-Module (SCA = Service Component Architecture), die Business-Prozesse und/oder Benutzertasks enthalten, können auf Implementierungsziele verteilt werden. Implementierungsziel kann ein Server oder ein Cluster sein.

Business-Prozess- und Benutzertaskanwendungen im Dialogbetrieb installieren

Mit dem Tool wsadmin und dem Script installInteractive können Sie eine Anwendung während der Laufzeit im Dialogbetrieb installieren. Mit diesem Script können Sie Einstellungen ändern, die beim Installieren mit der Administrationskonsole nicht geändert werden können.

Führen Sie die folgenden Schritte aus, um Business-Prozessanwendungen im Dialogbetrieb zu installieren.

1. Starten Sie das Tool wsadmin.

   Geben Sie im Verzeichnis *profile_root*/bin die Zeichenfolge `wsadmin` ein.
2. Installieren Sie die Anwendung.

   Geben Sie an der Eingabeaufforderung von wsadmin den folgenden Befehl ein:

   ```
   $AdminApp installInteractive application.ear
   ```

   Dabei ist *application.ear* der qualifizierte Name der Unternehmensarchivdatei, die Ihre Prozessanwendung enthält. Sie werden im Dialogbetrieb durch eine Reihe von Tasks geführt, mit denen Sie Werte für die Anwendung ändern können.
3. Speichern Sie die Konfigurationsänderungen.

   Geben Sie an der Eingabeaufforderung von wsadmin den folgenden Befehl ein:

   ```
   $AdminConfig save
   ```

   Sie müssen Ihre Änderungen speichern, damit sie in die Hauptkonfigurationsdatei übertragen werden. Wenn die Änderungen bei Beendigung eines Scripting-Prozesses noch nicht gespeichert sind, gehen sie verloren.

Datenquelle für Prozessanwendungen und Satzreferenzeinstellungen konfigurieren

Möglicherweise müssen Sie Prozessanwendungen konfigurieren, die SQL-Anweisungen für die spezielle Datenbankinfrastruktur ausführen. Diese SQL-Anweisungen können von Informationsserviceaktivitäten stammen, oder sie können Anweisungen sein, die Sie bei der Prozessinstallation oder beim Starten von Instanzen ausführen.

Beim Installieren der Anwendung können Sie die folgenden Datenquellentypen angeben:

- Datenquellen zum Ausführen von SQL-Anweisungen während der Prozessinstallation
- Datenquellen zum Ausführen von SQL-Anweisungen beim Starten einer Prozessinstanz
- Datenquellen zum Ausführen von SQL-Snippet-Aktivitäten

Die zum Ausführen einer SQL-Snippet-Aktivität erforderliche Datenquelle wird in einer BPEL-Variablen des Typs tDataSource definiert. Das für eine SQL-Snippet-Aktivität erforderliche Datenbankschema und die erforderlichen Tabellennamen werden in BPEL-Variablen des Typs tSetReference definiert. Sie können die Anfangswerte für diese beiden Variablen konfigurieren.

Mit dem Tool wsadmin können Sie die Datenquellen angeben.

1. Installieren Sie die Prozessanwendung im Dialogbetrieb mit dem Tool wsadmin.
2. Gehen Sie die Tasks schrittweise durch, bis Sie zu den Tasks zum Aktualisieren von Datenquellen und Satzreferenzen kommen.

   Konfigurieren Sie diese Einstellungen für Ihre Umgebung. Das folgende Beispiel zeigt die Einstellungen, die Sie für jede dieser Tasks ändern können.
3. Speichern Sie Ihre Änderungen.

Beispiel: Datenquellen und Satzreferenzen mit dem Tool wsadmin aktualisieren

In der Task **Datenquellen aktualisieren** können Sie Datenquellenwerte für Variablenanfangswerte und Anweisungen ändern, die beim Installieren oder Starten des Prozesses verwendet werden. In der Task **Satzreferenzen aktualisieren** können Sie die Einstellungen für das Datenbankschema und die Tabellennamen konfigurieren.

```
Task [24]: Datenquellen aktualisieren

//Datenquellenwerte für Variablenanfangswerte beim Prozessstart ändern

Process name:  Test
// Name der Prozessschablone
Process start or installation time:  Prozessstart
// Gibt an, ob der angegebene Wert beim Prozessstart oder
//bei der Prozessinstallation ausgewertet wird
Statement or variable:  Variable
// Gibt an, dass eine Datenquellenvariable geändert werden soll
Data source name:  MeineDatenquelle
// Name der Variablen
JNDI-Name:[jdbc/sample]:jdbc/neuerName
// Legt jdbc/neuerName als JNDI-Name fest
```

```
Task [25]: Satzreferenzen aktualisieren

// Satzreferenzwerte ändern, die als Anfangswerte für BPEL-Variablen verwendet werden

Process name:  Test
// Name der Prozessschablone
Variable:  SatzRef
// Der Name der BPEL-Variablen
JNDI-Name:[jdbc/sample]:jdbc/neuerName
// Legt jdbc/neuerName als JNDI-Name für die Datenquelle der Satzreferenz fest
Schema name: [IISAMPLE]
// Der Name des Datenbankschemas
Schema prefix: []:
// Das Präfix des Schemanamens
// Diese Einstellung wird nur angewendet, wenn der Schemaname generiert wird
Table name: [SETREFTAB]: NEUETABELLE
// Legt NEUETABELLE als Name für die Datenbanktabelle fest
Table prefix: []:
// Das Präfix des Tabellennamens
// Diese Einstellung wird nur angewendet, wenn der Präfixname generiert wird
```

Business-Prozess- und Benutzertaskanwendungen mit der Administrationskonsole deinstallieren

Mithilfe der Administrationskonsole können Sie Anwendungen deinstallieren, die Business-Prozesse oder Benutzertasks enthalten.

Für die Deinstallation einer Unternehmensanwendung, die Business-Prozesse oder Benutzertasks enthält, müssen die folgenden Voraussetzungen gegeben sein:

- Wenn die Anwendung auf einem eigenständigen Server installiert wird, muss der Server aktiv sein, und es muss Zugriff auf die Business Process Choreographer-Datenbank möglich sein.
- Wenn die Anwendung in einem Cluster installiert wird, müssen der Deployment Manager und mindestens ein Cluster-Member aktiv sein. Cluster-Member haben Zugriff auf die Business Process Choreographer-Datenbank.
- Wenn die Anwendung auf einem verwalteten Server installiert wird, müssen der Deployment Manager und dieser Server aktiv sein. Der Server hat Zugriff auf die Business Process Choreographer-Datenbank.
- Alle Business-Prozess- und Benutzertaskschablonen, die zu der Anwendung gehören, müssen den Stoppstatus aufweisen.
- Es sind keine Instanzen von Business-Prozess- oder Benutzertaskschablonen in irgendeinem Status vorhanden.

Was eigenständige Serverumgebungen betrifft, die als Entwicklungs- und Einheitentestumgebungen verwendet werden, kann der Server so konfiguriert werden, dass er im Entwicklungsmodus ausgeführt wird. Für diese Konfiguration ist es nicht erforderlich, dass die Schablonen gestoppt werden und keine Instanzen vorhanden sind. Diese Konfiguration ist jedoch für Produktionsumgebungen nicht gültig.

Führen Sie die folgenden Aktionen aus, um eine Unternehmensanwendung zu deinstallieren, die Business-Prozesse oder Benutzertasks enthält:

1. Stoppen Sie alle Prozess- und Taskschablonen in der Anwendung.

   Dadurch wird das Erstellen von Prozess- und Taskinstanzen verhindert.

   a. Klicken Sie im Navigationsfenster der Administrationskonsole auf **Anwendungen → SCA-Module**.

b. Wählen Sie das Modul aus, das die Schablonen enthält, die Sie stoppen möchten.
c. Klicken Sie unter "Erweiterte Merkmale" nach Bedarf auf **Business-Prozesse** und/oder auf **Benutzertasks**.
d. Wählen Sie alle Prozess- und Taskschablonen aus, indem Sie die dazugehörigen Markierungsfelder aktivieren.
e. Klicken Sie auf **Stoppen**.

Wiederholen Sie diesen Schritt für alle EJB-Module, die Business-Prozess- oder Benutzertaskschablonen enthalten.

2. Vergewissern Sie sich, dass die Datenbank, für jeden Cluster mindestens ein Anwendungsserver sowie der eigenständige Server aktiv ist, auf dem die Anwendung implementiert ist.

   In einer Network Deployment-Umgebung müssen der Deployment Manager, alle verwalteten eigenständigen Anwendungsserver und mindestens ein Anwendungsserver für jeden Cluster, in dem die Anwendung installiert ist, aktiv sein.
3. Stellen Sie sicher, dass die Anwendung keine Business-Prozess- oder Benutzertaskinstanzen aufweist.

   Falls erforderlich, kann ein Administrator mit Business Process Choreographer Explorer alle vorhandenen Prozess- oder Taskinstanzen löschen.
4. Stoppen und deinstallieren Sie die Anwendung:
   a. Klicken Sie im Navigationsfenster der Administrationskonsole auf **Anwendungen → Enterprise-Anwendungen**.
   b. Wählen Sie die Anwendung aus, die Sie deinstallieren möchten, und klicken Sie auf **Stoppen**.

      Dieser Schritt schlägt fehl, wenn in der Anwendung noch Prozessinstanzen oder Taskinstanzen vorhanden sind.
   c. Wählen Sie erneut die Anwendung aus, die Sie deinstallieren möchten, und klicken Sie auf **Deinstallieren**.
   d. Klicken Sie auf **Speichern**, um die Änderungen zu speichern.

Die Anwendung wird nun deinstalliert.

Anwendungen für Business-Prozesse und Benutzertask mit Verwaltungsbefehlen deinstallieren

Bei der Deinstallation von Anwendungen, die Business-Prozesse oder Benutzertasks enthalten, stellen Verwaltungsbefehle eine Alternative zur Verwendung der Administrationskonsole dar.

Für die Deinstallation einer Unternehmensanwendung, die Business-Prozesse oder Benutzertasks enthält, müssen die folgenden Voraussetzungen gegeben sein:

- Wenn die Anwendung auf einem eigenständigen Server installiert wird, muss der Server aktiv sein, und es muss Zugriff auf die Business Process Choreographer-Datenbank möglich sein.
- Wenn die Anwendung in einem Cluster installiert wird, müssen der Deployment Manager und mindestens ein Cluster-Member aktiv sein. Cluster-Member haben Zugriff auf die Business Process Choreographer-Datenbank.
- Wenn die Anwendung auf einem verwalteten Server installiert wird, müssen der Deployment Manager und dieser Server aktiv sein. Der Server hat Zugriff auf die Business Process Choreographer-Datenbank.

- Alle Business-Prozess- und Benutzertaskschablonen, die zu der Anwendung gehören, müssen den Stoppstatus aufweisen.
- Es sind keine Instanzen von Business-Prozess- oder Benutzertaskschablonen in irgendeinem Status vorhanden.

Was eigenständige Serverumgebungen betrifft, die als Entwicklungs- und Einheitentestumgebungen verwendet werden, kann der Server so konfiguriert werden, dass er im Entwicklungsmodus ausgeführt wird. Für diese Konfiguration ist es nicht erforderlich, dass die Schablonen gestoppt werden und keine Instanzen vorhanden sind. Diese Konfiguration ist jedoch für Produktionsumgebungen nicht gültig.

Falls die globale Sicherheit aktiviert ist, müssen Sie außerdem sicherstellen, dass Ihre Benutzer-ID eine Bedienerberechtigung besitzt.

Vergewissern Sie sich, dass der Serverprozess aktiv ist, zu dem der Verwaltungsclient eine Verbindung herstellt. Um sicherzustellen, dass der Verwaltungsclient automatisch eine Verbindung zum Serverprozess herstellt, dürfen Sie die Option `-conntype NONE` nicht als Befehlsoption verwenden.

Die folgenden Schritte beschreiben, wie Sie mit dem Script bpcTemplates.jacl Anwendungen deinstallieren, die Schablonen für Business-Prozesse oder Benutzertasks enthalten. Sie müssen eine Schablone stoppen, bevor Sie die Anwendung deinstallieren können, zu der die Schablone gehört. Mit dem Script bpcTemplates.jacl können Sie Schablonen in einem einzigen Schritt stoppen und deinstallieren.

Vor dem Deinstallieren von Anwendungen können Sie Prozessinstanzen oder Taskinstanzen löschen, die den Schablonen in den Anwendungen zugeordnet sind (z. B. mit Business Process Choreographer Explorer). Sie können auch die Option **-force** mit dem Script bpcTemplates.jacl verwenden, um in einem Schritt die den Schablonen zugeordneten Instanzen zu löschen, die Schablonen zu stoppen und zu deinstallieren.

Achtung:
Da die Option -force alle Prozessinstanz- und Taskinstanzdaten löscht, sollten Sie sie mit besonderer Sorgfalt verwenden.

1. Wechseln Sie in das Beispielverzeichnis von Business Process Choreographer.

   Geben Sie auf Windows-Plattformen Folgendes ein:

   ```
   cd installationsstammverzeichnis\ProcessChoreographer\admin
   ```

   Geben Sie auf Linux-, UNIX- und i5/OS-Plattformen Folgendes ein:

   ```
   cd installationsstammverzeichnis/ProcessChoreographer/admin
   ```

2. Stoppen Sie die Schablonen, und deinstallieren Sie die entsprechende Anwendung.

   Geben Sie auf Windows-Plattformen Folgendes ein:

   ```
   installationsstammverzeichnis\bin\wsadmin -f bpcTemplates.jacl
                                [-user benutzername]
                                [-password benutzerkennwort]
                                -uninstall anwendungsname
                                [-force]
   ```

Geben Sie auf Linux-, UNIX- und i5/OS-Plattformen Folgendes ein:

```
installationsstammverzeichnis/bin/wsadmin -f bpcTemplates.jacl
                              [-user benutzername]
                              [-password benutzerkennwort]
                              -uninstall anwendungsname
                              [-force]
```

Für diese Syntax gilt Folgendes:

benutzername
Falls die globale Sicherheit aktiviert ist, geben Sie die Benutzer-ID für die Authentifizierung an.

benutzerkennwort
Falls die globale Sicherheit aktiviert ist, geben Sie das Benutzerkennwort für die Authentifizierung an.

anwendungsname
Falls die globale Sicherheit aktiviert ist, geben Sie das Benutzerkennwort für die Authentifizierung an.

Die Anwendung wird nun deinstalliert.

Kapitel 6. Adapter installieren

Adapter ermöglichen Ihrer Anwendung die Kommunikation mit anderen Komponenten in einem unternehmensweiten Informationssystem (EIS, Enterprise Information System).

Die Vorgehensweise zur Installation von Adaptern wird im Artikel Configuring and using adapters im WebSphere Integration Developer Information Center erläutert.

Kapitel 7. EIS-Anwendungen installieren

Ein EIS-Anwendungsmodul kann auf einer J2EE-Plattform implementiert werden. Dabei entsteht eine Anwendung, die als auf dem Server implementierte EAR-Datei ausgeprägt ist. Alle J2EE-Artefakte und -Ressourcen werden erstellt, und die Anwendung wird konfiguriert und kann sofort ausgeführt werden.

Beim Implementieren auf der J2EE-Plattform werden die folgenden J2EE-Artefakte und -Ressourcen erstellt:

Tabelle 42. Zuordnung zwischen Bindungen und J2EE-Artefakten

Bindung im SCA-Modul	Generierte J2EE-Artefakte	Erstellte J2EE-Ressourcen
EIS-Import	In der Session-EJB des Moduls generierte Ressourcenreferenzen	ConnectionFactory
EIS-Export	Nachrichtengesteuerte Bean (generiert oder implementiert), abhängig von der durch den Ressourcenadapter unterstützten Listenerschnittstelle	ActivationSpec
JMS-Import	Zur Laufzeit bereitgestellte MDB (Message Driven Bean, nachrichtengesteuerte Bean) wird implementiert; Ressourcenreferenzen werden für die Session-EJB des Moduls generiert. Die MDB wird nur generiert, wenn der Importvorgang über ein Empfangsziel verfügt.	• ConnectionFactory • ActivationSpec • Ziele
JMS-Export	Zur Laufzeit bereitgestellte MDB (Message Driven Bean, nachrichtengesteuerte Bean) wird implementiert; Ressourcenreferenzen werden für die Session-EJB des Moduls generiert	• ActivationSpec • ConnectionFactory • Ziele

Wenn der Import- oder Exportvorgang eine Ressource wie z. B. ConnectionFactory definiert, wird der Ressourcenverweis im Implementierungsdeskriptor des Stateless Session-EJB des Moduls generiert. Außerdem wird die entsprechende Bindung in der EJB-Bindungsdatei generiert. Die Ressourcenreferenz wird an einen Namen gebunden, der entweder der Wert des Zielattributs ist (falls vorhanden) oder der Standard-JNDI-Suchname der Ressource, basierend auf dem Modulnamen und dem Importnamen.

Beim Implementieren wird die Session-Bean des Moduls lokalisiert und zum Suchen der Ressourcen verwendet.

Beim Implementieren der Anwendung auf dem Server prüft die EIS-Installationstask, ob die Elementressource vorhanden ist, an die sie gebunden ist. Ist sie nicht vorhanden und die SCDL-Datei gibt mindestens ein Merkmal an, wird die Ressource von der EIS-Installationstask erstellt und konfiguriert. Ist die Ressource

nicht vorhanden, erfolgt keine Aktion, weil davon ausgegangen wird, dass vor dem Ausführen der Anwendung Ressourcen erstellt werden.

Beim Implementieren des JMS-Importvorgangs mit einem Empfangsziel wird eine (Message Driven Bean, nachrichtengesteuerte Bean) implementiert. Es wird nach Antworten auf abgesetzte Anforderungen gesucht. Die MDB ist dem Ziel zugeordnet, das mit der Anforderung im Headerfeld JMSreplyTo der JMS-Nachricht gesendet wird. Beim Eintreffen der Antwortnachricht ruft die MDB mithilfe ihrer Korrelations-ID die Rückrufinformationen ab, die im Rückrufziel gespeichert sind, und ruft anschließend das Rückrufobjekt auf.

Die Installationstask erstellt ConnectionFactory und drei Ziele aus den Informationen in der Importdatei. Außerdem erstellt sie ActivationSpec, um der Laufzeit-MDB das Empfangen von Antworten im Empfangsziel zu ermöglichen. Die Merkmale für ActivationSpec werden aus den Merkmalen von Destination/ ConnectionFactory abgeleitet. Wenn der JMS-Provider ein SIBus-Ressourcenadapter ist, werden die entsprechenden SIBus-Ziele für das JMS-Ziel erstellt.

Beim Implementieren des JMS-Exportvorgangs wird eine (Message Driven Bean, nachrichtengesteuerte Bean) implementiert (nicht die gleiche MDB, die beim JMS-Import implementiert wurde). Die eingehenden Anforderungen am Empfangsziel werden überwacht und die Anforderungen zur Verarbeitung an den SCA gesendet. Die Installationstask erstellt ein ähnliches Ressourcenset wie für den JMS-Import, eine ActivationSpec, eine ConnectionFactory zum Senden einer Antwort und zwei Ziele. Alle Merkmale für diese Ressourcen werden in der Exportdatei angegeben. Ist der JMS-Provider ein SIBus-Ressourcenadapter, werden die entsprechenden SIBus-Ziele für das JMS-Ziel erstellt.

EIS-Anwendungsmodul auf der J2SE-Plattform implementieren

Das EIS-Modul kann auf der J2SE-Plattform implementiert werden, es wird jedoch nur der EIS-Import unterstützt.

Vor dem Starten dieser Task müssen Sie ein EIS-Anwendungsmodul mit einer JMS-Importbindung in der WebSphere Integration Development-Umgebung erstellen.

Ein EIS-Anwendungsmodul wird mit einer JMS-Importbindung bereitgestellt, wenn Sie über Nachrichtenwarteschlangen asynchron auf EIS-Systeme zugreifen möchten.

Das Implementieren auf der J2SE-Plattform ist der einzige Anwendungsfall, in dem die Bindungsimplementierung im nicht verwalteten Modus ausgeführt werden kann. Für die JMS-Bindung ist asynchrone Unterstützung und JNDI-Unterstützung erforderlich, die weder von der Basisarchitektur für Servicekomponenten noch von J2SE zur Verfügung gestellt werden. Die J2EE Connector-Architektur unterstützt keine nicht verwaltete, eingehende Kommunikation, d. h. es ist kein EIS-Export möglich.

Beim Implementieren des EIS-Anwendungsmoduls mit dem EIS-Import unter J2SE muss zusätzlich zu den Modulabhängigkeiten der beim Import verwendete WebSphere Adapter als Abhängigkeit angegeben werden (im Manifest oder in jeder anderen von SCA unterstützten Form).

EIS-Anwendungsmodul auf der J2EE-Plattform implementieren

Beim Implementieren eines EIS-Moduls auf der J2EE-Plattform entsteht eine Anwendung, die als auf dem Server implementierte EAR-Datei ausgeprägt ist. Alle J2EE-Artefakte und -Ressourcen werden erstellt, und die Anwendung wird konfiguriert und kann sofort ausgeführt werden.

Vor dem Starten dieser Task müssen Sie ein EIS-Modul mit einer JMS-Importbindung in der WebSphere Integration Development-Umgebung erstellen.

Beim Implementieren auf der J2EE-Plattform werden die folgenden J2EE-Artefakte und -Ressourcen erstellt:

Tabelle 43. Zuordnung zwischen Bindungen und J2EE-Artefakten

Bindung im SCA-Modul	**Generierte J2EE-Artefakte**	**Erstellte J2EE-Ressourcen**
EIS-Import	In der Session-EJB des Moduls generierte Ressourcenreferenzen	ConnectionFactory
EIS-Export	Nachrichtengesteuerte Bean (generiert oder implementiert), abhängig von der durch den Ressourcenadapter unterstützten Listenerschnittstelle	ActivationSpec
JMS-Import	Zur Laufzeit bereitgestellte MDB (Message Driven Bean, nachrichtengesteuerte Bean) wird implementiert; Ressourcenreferenzen werden für die Session-EJB des Moduls generiert. Die MDB wird nur generiert, wenn der Importvorgang über ein Empfangsziel verfügt.	• ConnectionFactory • ActivationSpec • Ziele
JMS-Export	Zur Laufzeit bereitgestellte MDB (Message Driven Bean, nachrichtengesteuerte Bean) wird implementiert; Ressourcenreferenzen werden für die Session-EJB des Moduls generiert	• ActivationSpec • ConnectionFactory • Ziele

Wenn der Import- oder Exportvorgang eine Ressource wie z. B. ConnectionFactory definiert, wird der Ressourcenverweis im Implementierungsdeskriptor des Stateless Session-EJB des Moduls generiert. Außerdem wird die entsprechende Bindung in der EJB-Bindungsdatei generiert. Die Ressourcenreferenz wird an einen Namen gebunden, der entweder der Wert des Zielattributs ist (falls vorhanden) oder der Standard-JNDI-Suchname der Ressource, basierend auf dem Modulnamen und dem Importnamen.

Beim Implementieren wird die Session-Bean des Moduls lokalisiert und zum Suchen der Ressourcen verwendet.

Beim Implementieren der Anwendung auf dem Server prüft die EIS-Installationstask, ob die Elementressource vorhanden ist, an die sie gebunden ist. Ist sie nicht vorhanden und die SCDL-Datei gibt mindestens ein Merkmal an, wird die Ressource von der EIS-Installationstask erstellt und konfiguriert. Ist die Ressource nicht vorhanden, erfolgt keine Aktion, weil davon ausgegangen wird, dass vor dem Ausführen der Anwendung Ressourcen erstellt werden.

Beim Implementieren des JMS-Importvorgangs mit einem Empfangsziel wird eine MDB (Message Driven Bean, nachrichtengesteuerte Bean) implementiert. Es wird nach Antworten auf abgesetzte Anforderungen gesucht. Die MDB ist dem Ziel zugeordnet, das mit der Anforderung im Headerfeld JMSreplyTo der JMS-Nachricht gesendet wird. Beim Eintreffen der Antwortnachricht ruft die MDB mithilfe ihrer Korrelations-ID die Rückrufinformationen ab, die im Rückrufziel gespeichert sind, und ruft anschließend das Rückrufobjekt auf.

Die Installationstask erstellt ConnectionFactory und drei Ziele aus den Informationen in der Importdatei. Außerdem erstellt sie ActivationSpec, um der Laufzeit-MDB das Empfangen von Antworten im Empfangsziel zu ermöglichen. Die Merkmale für ActivationSpec werden aus den Merkmalen von Destination/ ConnectionFactory abgeleitet. Wenn der JMS-Provider ein SIBus-Ressourcenadapter ist, werden die entsprechenden SIBus-Ziele für das JMS-Ziel erstellt.

Beim Implementieren des JMS-Exportvorgangs wird eine MDB (Message Driven Bean, nachrichtengesteuerte Bean) implementiert (nicht die gleiche MDB, die beim JMS-Import implementiert wurde). Die eingehenden Anforderungen am Empfangsziel werden überwacht und die Anforderungen zur Verarbeitung an den SCA gesendet. Die Installationstask erstellt ein ähnliches Ressourcenset wie für den JMS-Import, eine ActivationSpec, eine ConnectionFactory zum Senden einer Antwort und zwei Ziele. Alle Merkmale für diese Ressourcen werden in der Exportdatei angegeben. Ist der JMS-Provider ein SIBus-Ressourcenadapter, werden die entsprechenden SIBus-Ziele für das JMS-Ziel erstellt.

Kapitel 8. Fehlerbehebung bei fehlgeschlagener Implementierung

In diesem Artikel werden die Schritte zum Ermitteln der Fehlerursache beim Implementieren einer Anwendung beschrieben. Außerdem werden einige mögliche Lösungen vorgestellt.

Dabei wird Folgendes vorausgesetzt:
- Sie verfügen über Grundkenntnisse im Debugging eines Moduls
- Protokollierung und Traceverarbeitung sind beim Implementieren des Moduls aktiv

Die Fehlerbehebung für eine Implementierung beginnt, sobald Sie eine Fehlernachricht erhalten. Bei einer fehlgeschlagenen Implementierung sollten zunächst verschiedene Symptome überprüft werden, bevor Sie Maßnahmen ergreifen.

1. Stellen Sie fest, ob das Installieren der Anwendung fehlgeschlagen ist.

   Überprüfen Sie die Datei SystemOut.log auf Nachrichten, die auf die Fehlerursache hinweisen. Zu den möglichen Ursachen für das Fehlschlagen einer Anwendungsinstallation gehören die folgenden:
   - Sie versuchen, eine Anwendung auf mehreren Servern in derselben Netzimplementierungszelle zu installieren
   - Eine Anwendung trägt den gleichen Namen wie ein vorhandenes Modul in der Netzimplementierungszelle, in der Sie die Anwendung installieren möchten
   - Sie versuchen, J3EE-Module in einer EAR-Datei auf verschiedenen Zielservern zu implementieren

   Wichtig: Wenn die Installation fehlgeschlagen ist und die Anwendung Services enthält, müssen Sie sämtliche SIBus-Ziele oder J2C-Aktivierungsspezifikationen entfernen, die vor dem Fehlschlagen erstellt wurden, bevor Sie versuchen, die Anwendung erneut zu installieren. Am einfachsten können Sie diese Artefakte entfernen, wenn Sie nach dem Fehler auf die Optionen zum Speichern und Löschen aller Fehler klicken. Wenn Sie die Änderungen versehentlich speichern, müssen Sie die SIBus-Ziele und J2C-Aktivierungsspezifikationen manuell entfernen (siehe SIBus-Ziele löschen und J2C-Aktivierungsspezifikationen löschen).
2. Wenn die Anwendung korrekt installiert ist, überprüfen Sie, ob die Anwendung erfolgreich gestartet wurde.

   Wenn die Anwendung nicht erfolgreich gestartet wurde, trat der Fehler bei dem Versuch des Servers auf, die Ressourcen für die Anwendung zu initialisieren.

   a. Überprüfen Sie die Datei SystemOut.log auf Nachrichten, die die weitere Vorgehensweise angeben.

   b. Stellen Sie fest, ob für die Anwendung erforderliche Ressourcen verfügbar sind und/oder erfolgreich gestartet wurden.

      Nicht gestartete Ressourcen verhindern das Ausführen der Anwendung. Dadurch werden Datenverluste vermieden. Zu den möglichen Gründen für das Nichtstarten einer Ressource gehören die folgenden:
      - Bindungen wurden falsch angegeben
      - Ressourcen sind nicht korrekt konfiguriert
      - Ressourcen sind nicht im Ressourcenarchiv (RAR-Datei) enthalten

- Webressourcen sind nicht im Web-Service-Archive (WAR-Datei) enthalten

c. Stellen Sie fest, ob Komponenten fehlen.

Die Ursache für das Fehlen einer Komponente ist ein fehlerhaftes Unternehmensarchiv (EAR-Datei). Stellen Sie sicher, dass sich alle für das Modul erforderlichen Komponenten in den richtigen Ordnern auf dem Testsystem befinden, auf dem Sie das Java-Archiv (JAR-Datei) erstellt haben. Weitere Informationen finden Sie unter „Implementierung auf einem Server vorbereiten".

3. Überprüfen Sie die Anwendung, um festzustellen, ob darin Datenfluss stattfindet.

Selbst in einer aktiven Anwendung kann es vorkommen, dass keine Daten verarbeitet werden. Die Gründe dafür sind ähnlich wie im Schritt 2b auf Seite 205 angegeben.

a. Stellen Sie fest, ob die Anwendung Services verwendet, die in einer anderen Anwendung enthalten sind. Stellen Sie sicher, dass die andere Anwendung installiert ist und erfolgreich gestartet wurde.

b. Stellen Sie fest, ob die Import- und Exportbindungen für Einheiten, die in anderen Anwendungen enthalten sind, die die fehlgeschlagene Anwendung verwendet, ordnungsgemäß konfiguriert sind. Verwenden Sie die Administrationskonsole, um die Bindungen zu überprüfen und zu korrigieren.

4. Beheben Sie den Fehler, und starten Sie die Anwendung erneut.

J2C-Aktivierungsspezifikationen löschen

Bei der Installation einer Anwendung, die Services enthält, erstellt das System J2C-Anwendungsspezifikationen. Es gibt Fälle, in denen Sie diese Spezifikationen vor der Neuinstallation der Anwendung löschen müssen.

Wenn Sie die Spezifikation aufgrund einer fehlgeschlagenen Anwendungsinstallation löschen, stellen Sie sicher, dass das Modul im JNDI-Namen (JNDI = Java Naming and Directory Interface) mit dem Namen des Moduls übereinstimmt, dessen Installation fehlgeschlagen ist. Der zweite Teil des JNDI-Namens ist der Name des Moduls, das das Ziel implementiert hat. Beispiel: In sca/SimpleBOCrsmA/ActivationSpec ist **SimpleBOCrsmA** der Modulname.

Erforderlicher Sicherheitsaufgabenbereich für diese Task: Wenn die Sicherheit und die aufgabenbereichsbasierte Berechtigung aktiviert sind, müssen Sie als Administrator oder Konfigurator angemeldet sein, um diese Task durchzuführen.

Löschen Sie die J2C-Aktivierungsspezifikationen, wenn Sie nach der Installation einer Anwendung, die Services enthält, eine Konfiguration versehentlich gespeichert haben und Sie die Spezifikationen nicht mehr benötigen.

1. Suchen Sie nach der zu löschenden Aktivierungsspezifikation.

Die Spezifikationen befinden sich in dem Ressourcenadapterfenster. Navigieren Sie zu diesem Fenster, indem Sie auf die Optionen **Ressourcen > Ressourcenadapter** klicken.

a. Suchen Sie den **Platform Messaging Component SPI Resource Adapter.**

Um nach diesem Adapter zu suchen, müssen Sie sich im Knotengeltungsbereich für einen eigenständigen Server oder im Servergeltungsbereich einer Implementierungsumgebung befinden.

2. Zeigen Sie die dem Platform Messaging Component SPI Resource Adapter zugeordneten J2C-Aktivierungsspezifikationen an.

Klicken Sie auf den Ressourcenadapternamen; daraufhin werden im nächsten Fenster die zugeordneten Spezifikationen angezeigt.

3. Löschen Sie alle Spezifikationen mit einem **JNDI-Namen**, der mit dem Modulnamen übereinstimmt, den Sie löschen.
   a. Klicken Sie auf das Markierungsfeld neben den entsprechenden Spezifikationen.
   b. Klicken Sie auf die Option zum Löschen.

Das System entfernt ausgewählte Spezifikationen aus der Anzeige.

Speichern Sie die Änderungen.

Ziele des SIBus löschen

Ziele des SIBus sind Verbindungen, mit deren Hilfe Services für Anwendungen zur Verfügung gestellt werden. Es gibt Situationen, in denen Sie Ziele entfernen müssen.

Wenn Sie das Ziel aufgrund einer fehlgeschlagenen Anwendungsinstallation löschen, stellen Sie sicher, dass das Modul im Zielnamen mit dem Namen des Moduls übereinstimmt, dessen Installation fehlgeschlagen ist. Der zweite Teil des Namens ist der Name des Moduls, das das Ziel implementiert hat . Beispiel: In sca/SimpleBOCrsmA/component/test/sca/cros/simple/cust/Customer ist **SimpleBOCrsmA** der Modulname.

Erforderlicher Sicherheitsaufgabenbereich für diese Task: Wenn die Sicherheit und die aufgabenbereichsbasierte Berechtigung aktiviert sind, müssen Sie als Administrator oder Konfigurator angemeldet sein, um diese Task durchzuführen.

Löschen Sie Ziele des SIBus, wenn Sie nach der Installation einer Anwendung, die Services enthält, eine Konfiguration versehentlich gespeichert haben oder Sie keine Ziele mehr benötigen.

Anmerkung: Bei dieser Task wird das Ziel nur aus dem SCA-Systembus gelöscht. Sie müssen auch die Einträge aus dem Anwendungsbus entfernen, bevor Sie eine Anwendung neu installieren, die Services enthält (siehe 'J2C-Aktivierungsspezifikationen löschen' im Abschnitt zur Verwaltung in diesem Information Center).

1. Melden Sie sich bei der Administrationskonsole an.
2. Rufen Sie die Ziele des SCA-Systembusses auf.

   Navigieren Sie zu dieser Anzeige, indem Sie auf die Option **Serviceintegration > Busse** klicken.
3. Wählen Sie die Ziele des SCA-Systembusses aus.

   Klicken Sie in der Anzeige auf **SCA.SYSTEM.*zellenname*.Bus**, wobei *zellenname* der Name der Zelle ist, die das Modul mit den Zielen enthält, die Sie löschen.
4. Löschen Sie die Ziele, die einen Modulnamen enthalten, der mit dem Modul übereinstimmt, das Sie entfernen.
   a. Klicken Sie auf das Markierungsfeld neben den relevanten Zielen.
   b. Klicken Sie auf die Option zum Löschen.

In der Anzeige werden nur die verbliebenen Ziele angezeigt.

Löschen Sie die J2C-Aktivierungsspezifikationen, die sich auf das Modul beziehen, das diese Ziele erstellt hat.

Teil 3. Schlussteil

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden.

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb von Produkten, Programmen und Services anderer Anbieter liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanfragen können Sie schriftlich an die folgende Adresse richten (Anfragen an die unten stehende Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Defense
France

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die Angaben in diesem Handbuch werden in regelmäßigen Zeitabständen aktualisiert. Die Änderungen werden in Überarbeitungen oder in Technical News Letters (TNLs) bekannt gegeben. IBM kann ohne weitere Mitteilung jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter dienen lediglich als Benutzerinformationen und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängig erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Corporation
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A.

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des im Handbuch aufgeführten Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM-Rahmenvereinbarung sowie der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer gesteuerten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Garantie, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichten Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Die oben genannten Erklärungen bezüglich der Produktstrategien und Absichtserklärungen von IBM stellen die gegenwärtige Absicht von IBM dar, unterliegen Änderungen oder können zurückgenommen werden, und repräsentieren nur die Ziele von IBM.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes. Sie sollen nur die Funktionen des Lizenzprogrammes illustrieren; sie können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden; Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Musteranwendungsprogramme, die in Quellensprache geschrieben sind und die Programmiertechnik für verschiedene Betriebsumgebungen demonstrieren. Sie dürfen diese Musterprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, verwenden, vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle konform sind, für die diese Musterprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten.

Kopien oder Teile der Musterprogramme bzw. daraus abgeleiteter Code müssen folgenden Copyrightvermerk beinhalten: (c) (Ihr Firmenname) (Jahr). Teile des vorliegenden Codes wurden aus Musterprogrammen der IBM Corp. abgeleitet. (c) Copyright IBM Corp. _Jahr/Jahre angeben_. Alle Rechte vorbehalten.

Informationen zur Programmierschnittstelle

Bereitgestellte Informationen zur Programmierschnittstelle sind als Unterstützung für die Erstellung von Anwendungssoftware mit diesem Programm gedacht.

Mit allgemeinen Programmierschnittstellen können Sie Anwendungssoftware schreiben, mit der die Services dieser Programmtools abgerufen werden können.

Diese Informationen können aber auch Informationen zu Diagnose, Änderung und Optimierung enthalten. Die Informationen zu Diagnose, Änderung und Optimierung sollen Sie bei der Behebung von Fehlern in Ihrer Anwendungssoftware unterstützen.

Warnung: Verwenden Sie diese Informationen zu Diagnose, Änderung und Optimierung nicht als Programmierschnittstelle, da sie jederzeit geändert werden können.

Marken und Servicemarken

IBM, das IBM Logo, developerWorks, WebSphere und z/OS sind eingetragene Marken der International Business Machines Corporation in den USA und/oder anderen Ländern.

Adobe ist eine eingetragene Marke von Adobe Systems Incorporated in den USA und/oder anderen Ländern.

Java und alle Java-basierten Marken sind Marken von Sun Microsystems, Inc. in den USA und/oder anderen Ländern.

Weitere Unternehmens-, Produkt- oder Servicenamen können Marken anderer Hersteller sein.

Dieses Produkt enthält Software, die vom Eclipse-Projekt entwickelt wurde (http://www.eclipse.org).

IBM WebSphere Process Server for Multiplatforms, Version 6.1.0