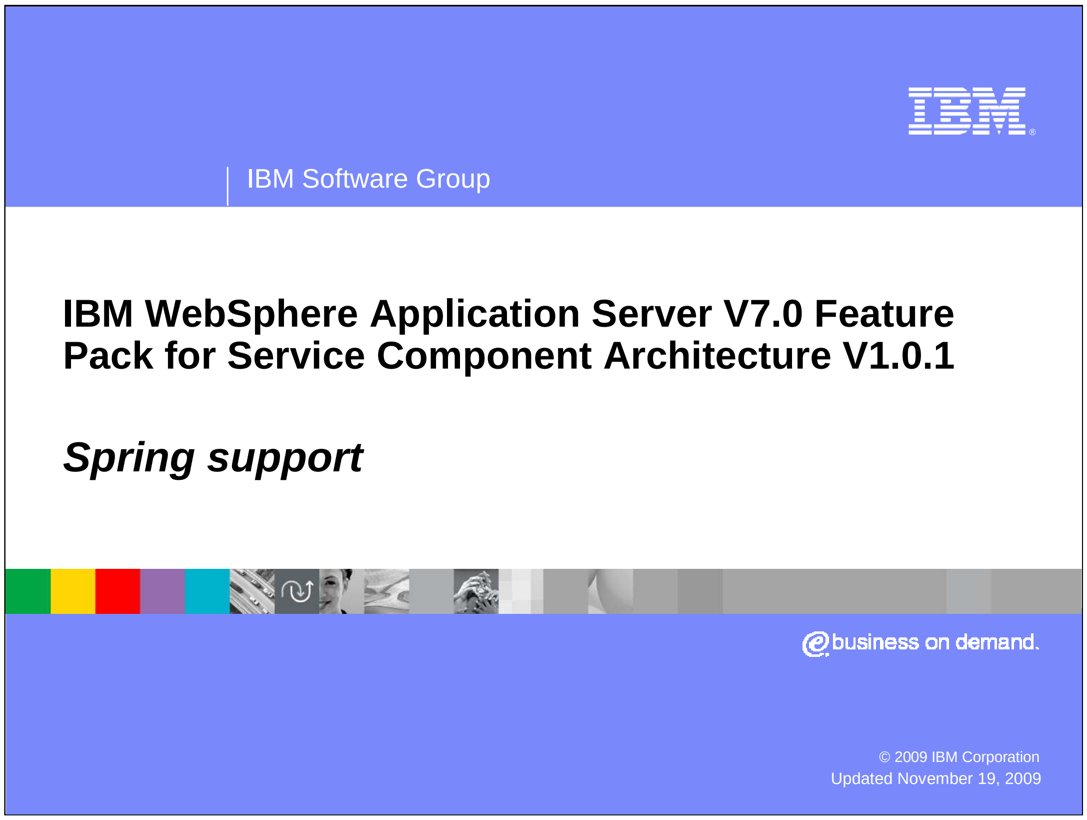


IBM Software Group

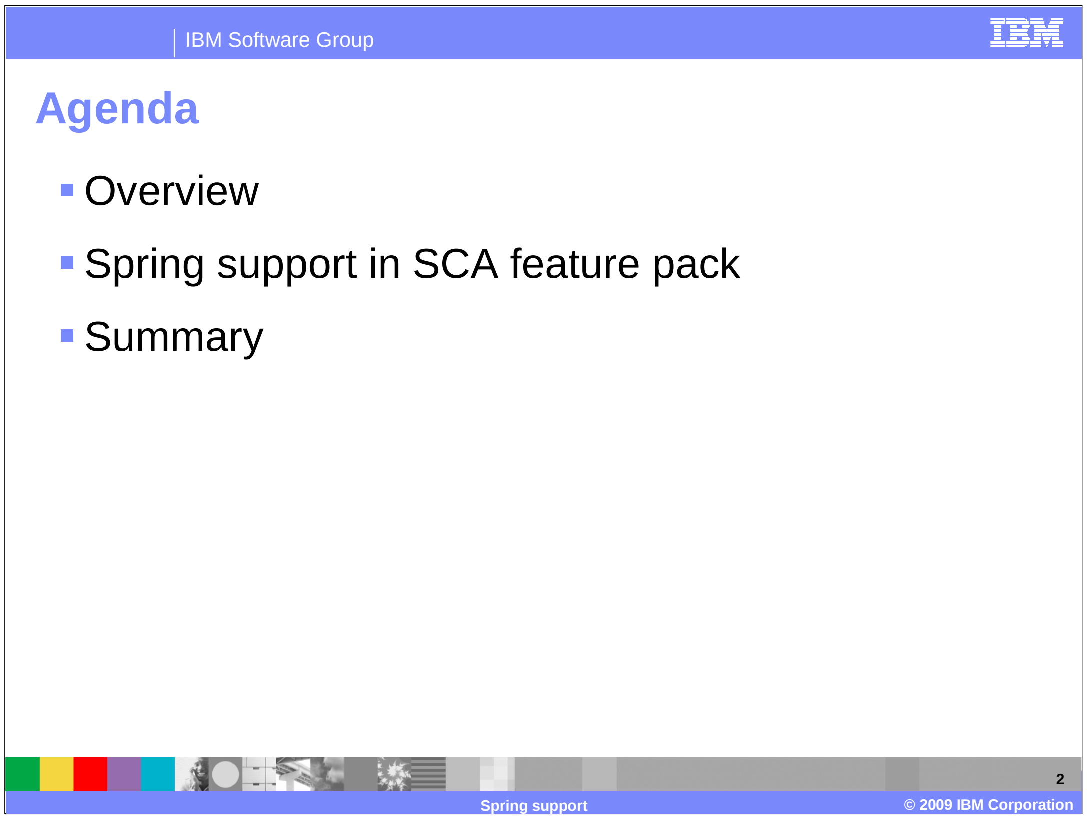
IBM WebSphere Application Server V7.0 Feature Pack for Service Component Architecture V1.0.1

Spring support

business on demand.

Updated November 19, 2009

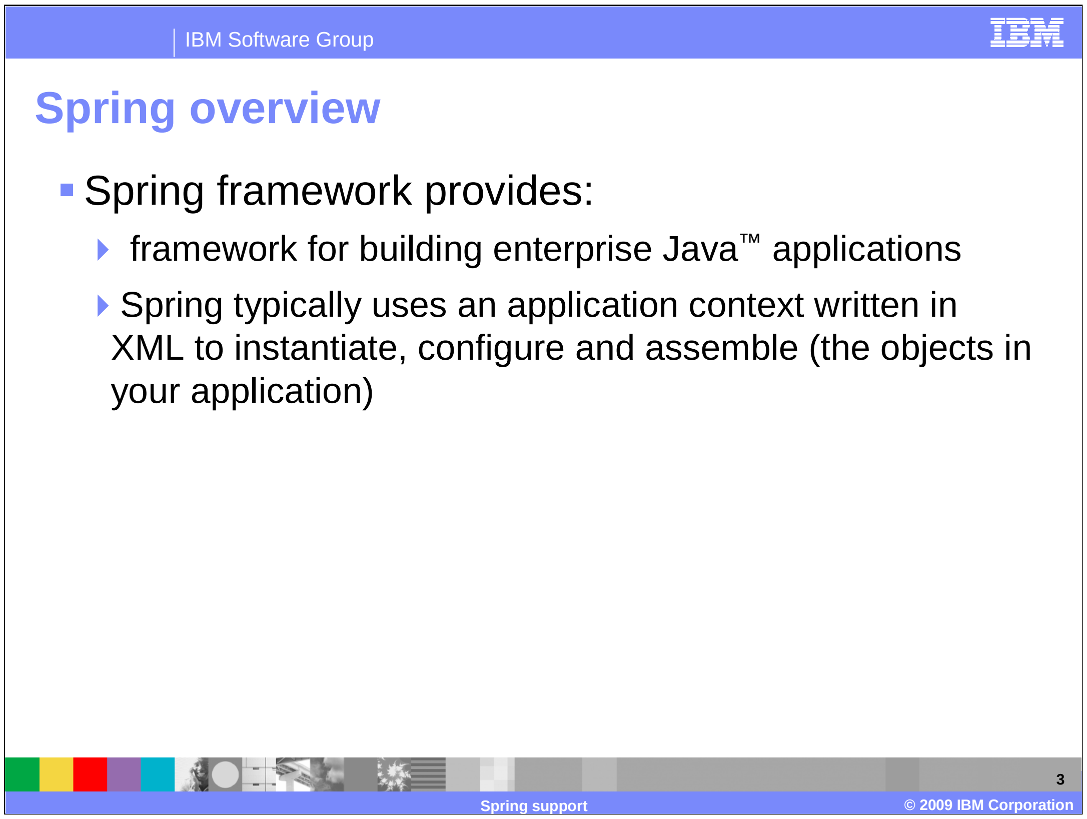
This presentation covers spring support in the SCA feature pack.



Agenda

- Overview
- Spring support in SCA feature pack
- Summary

You will start with a short overview of spring framework followed by its support in SCA feature pack.



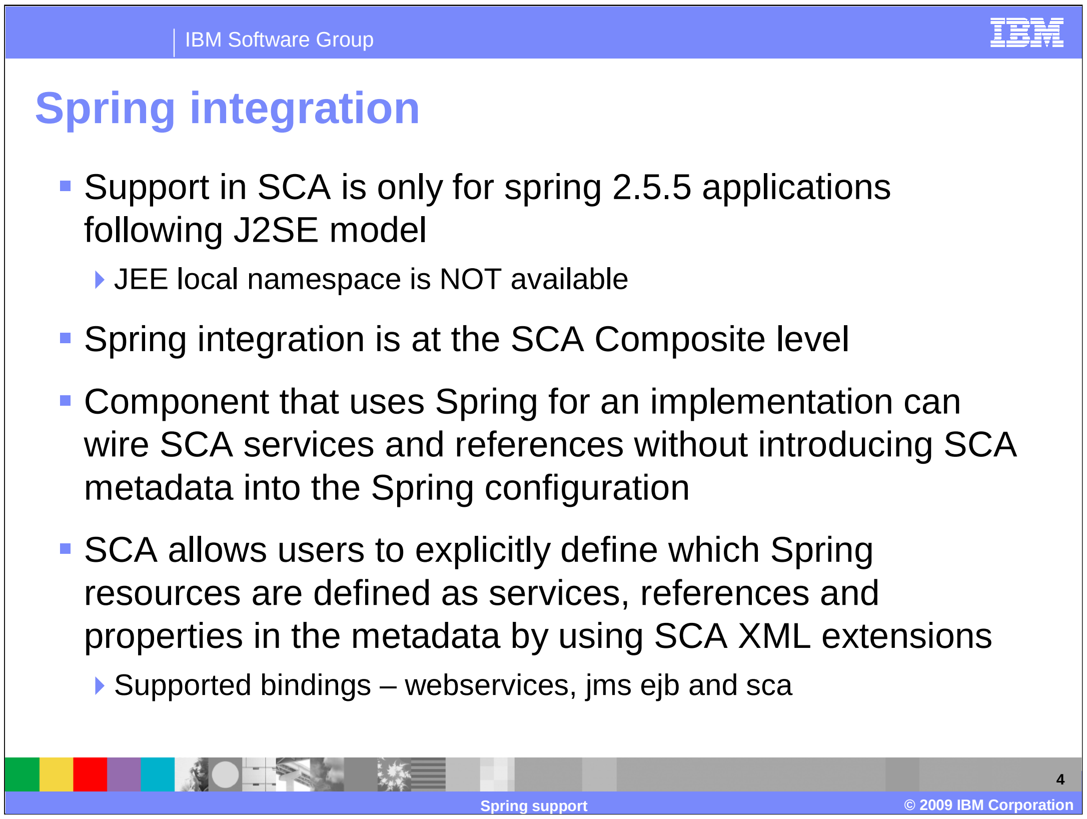
Spring overview

- Spring framework provides:
  - framework for building enterprise Java™ applications
  - Spring typically uses an application context written in XML to instantiate, configure and assemble (the objects in your application)

The Spring framework is a popular platform used to construct Java applications. It aims to reduce the complexity of the programming environment and it shares many of the same design principles as SCA. In particular, Spring provides a runtime container that provides ***dependency injection*** so that application components can avoid the need to program directly to middleware APIs. This is also one of the key principles of SCA.

The Spring framework also uses an application context written in XML to instantiate, configure and assemble the objects in your application.

The Spring framework is an open source project. It provides a framework for simple Java objects that enables them to use the Java EE container through wrapper classes and XML configuration.


IBM Software Group

Spring integration

- Support in SCA is only for spring 2.5.5 applications following J2SE model
  - JEE local namespace is NOT available
- Spring integration is at the SCA Composite level
- Component that uses Spring for an implementation can wire SCA services and references without introducing SCA metadata into the Spring configuration
- SCA allows users to explicitly define which Spring resources are defined as services, references and properties in the metadata by using SCA XML extensions
  - Supported bindings – webservices, jms ejb and sca

4

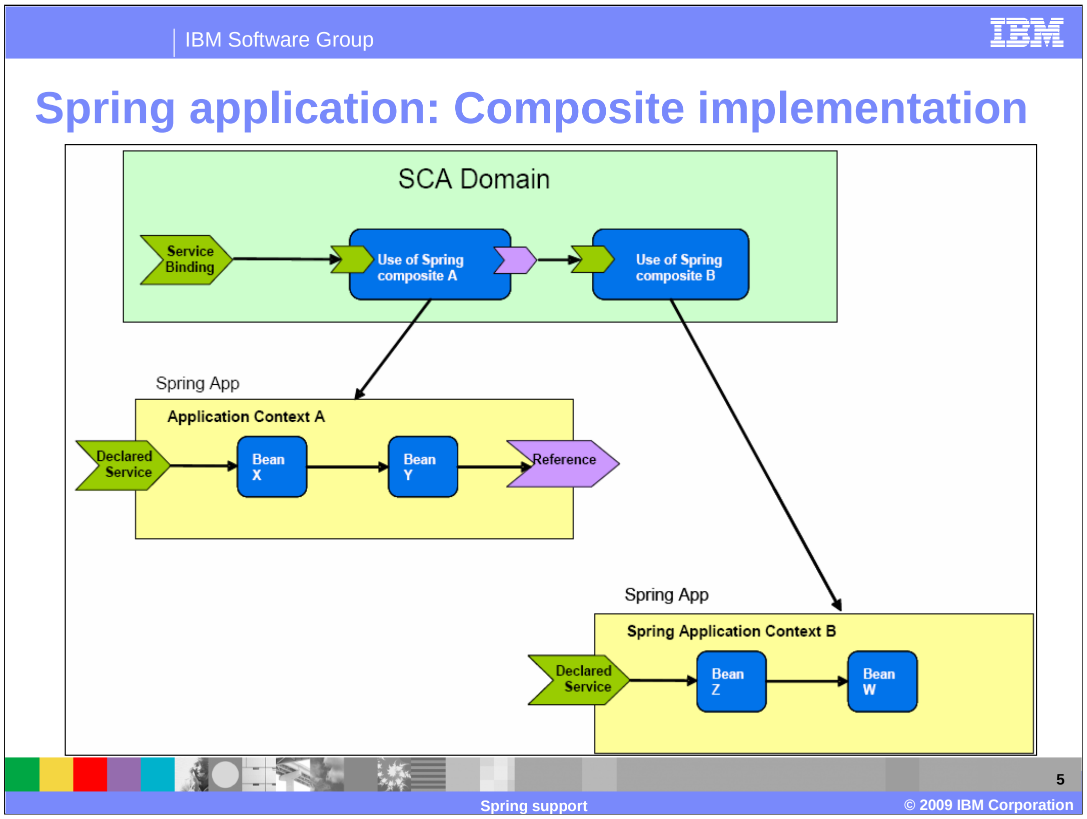
Spring support © 2009 IBM Corporation

Support for the Spring framework in SCA is provided at a **coarse-grained level**. It is possible to use an existing Spring application context as a component implementation in SCA. An SCA runtime that supports Spring integration can use an application context as-is in an SCA assembly. For such a component it is possible to wire Spring services and references without the need to introduce SCA metadata into the Spring configuration. The integration with Spring is at the SCA Composite level, where a Spring application context provides a complete composite, exposing services and using references through SCA. This means that a Spring application context defines the internal structure of a composite implementation.

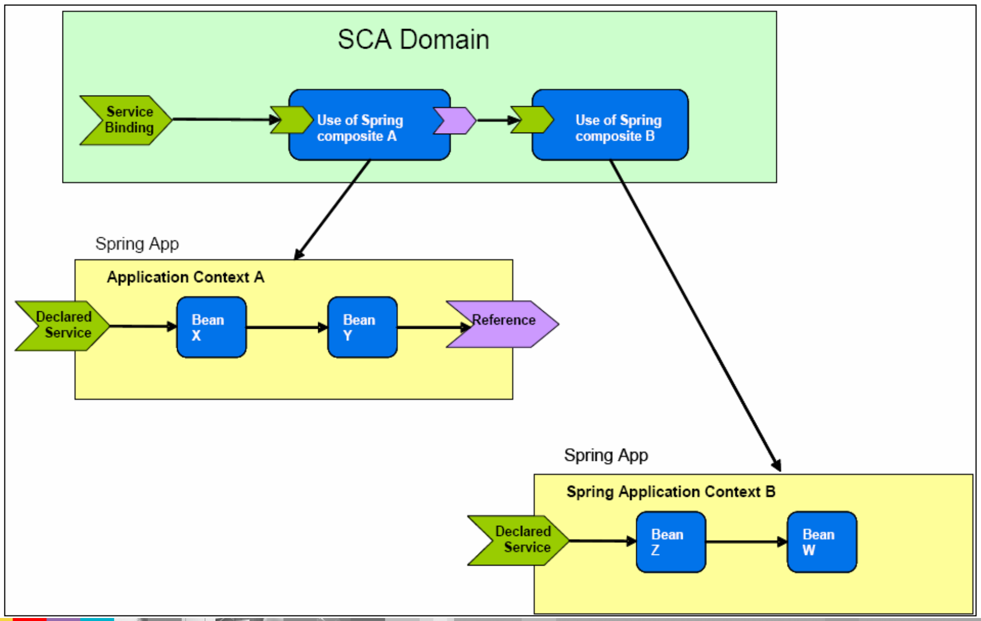
SCA allows you to explicitly define which Spring resources are defined as services, references and properties. It does so by using SCA XML extensions. It also provides default mapping.

Note that you must use Spring 2.5.5 for the SCA composite component implementation. The product does not support other levels of Spring.

Also note that the support is for Spring applications following J2SE model. JEE local namespace is NOT available.


IBM Software Group

Spring application: Composite implementation



5

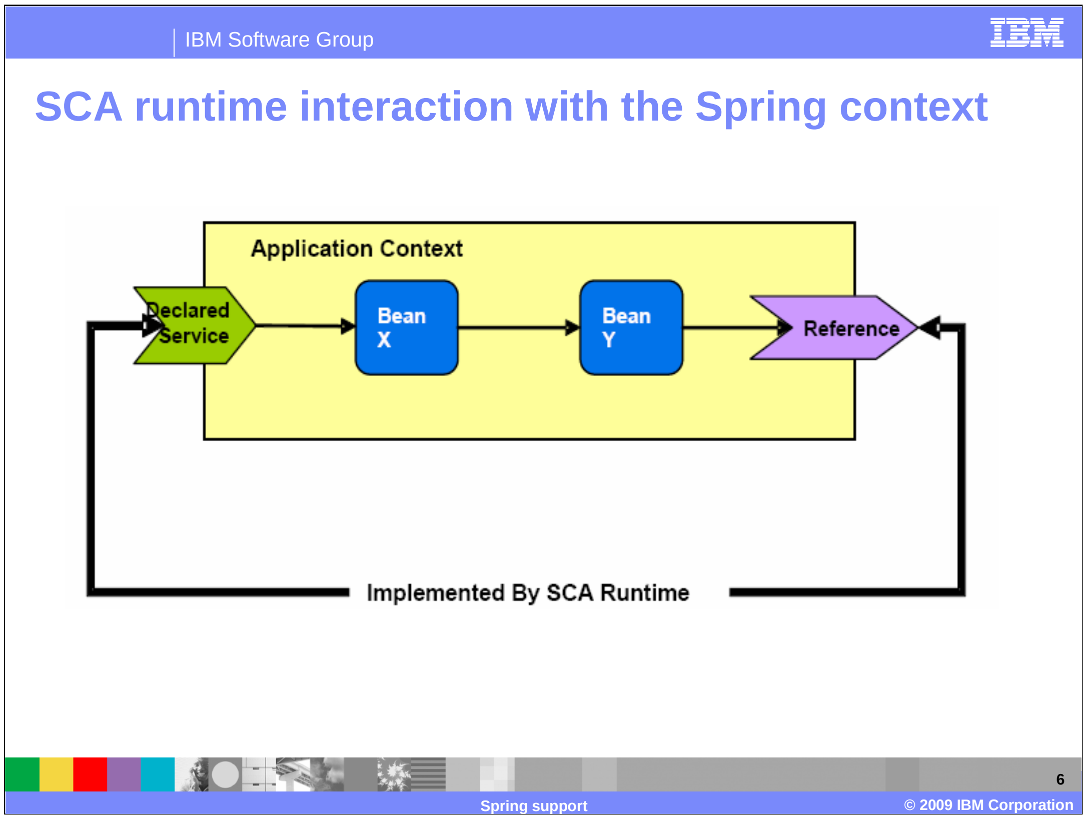
Spring support

© 2009 IBM Corporation

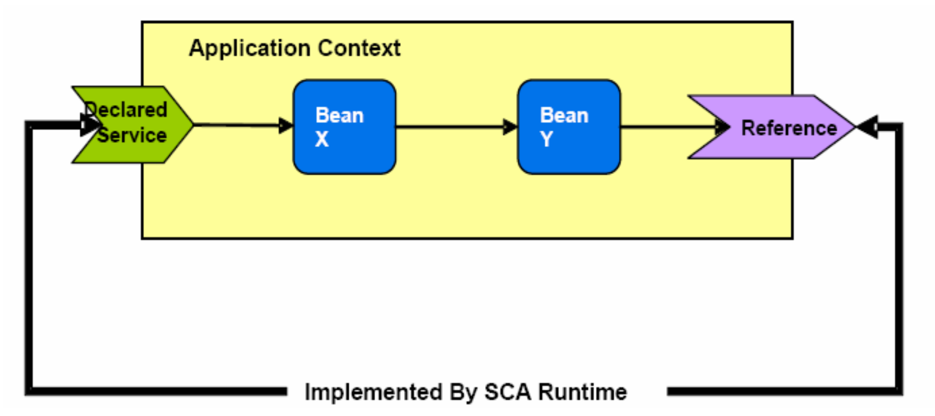
A Spring application context is used as an implementation within an SCA composite component.

Conceptually, this can be represented as shown on the diagram. The diagram shows a simple SCA domain composed of two composites, both of which are implemented by Spring application contexts. In this diagram, there are two composites defined by separate Spring application contexts, each with one declared service. Composite A is composed of two Spring beans, and bean X is exposed to SCA through an SCA service. Bean Y has a reference to an external SCA service. This service reference is wired to another Spring context, Composite B, which has a single declared service entry point, which is wired to Bean Z.

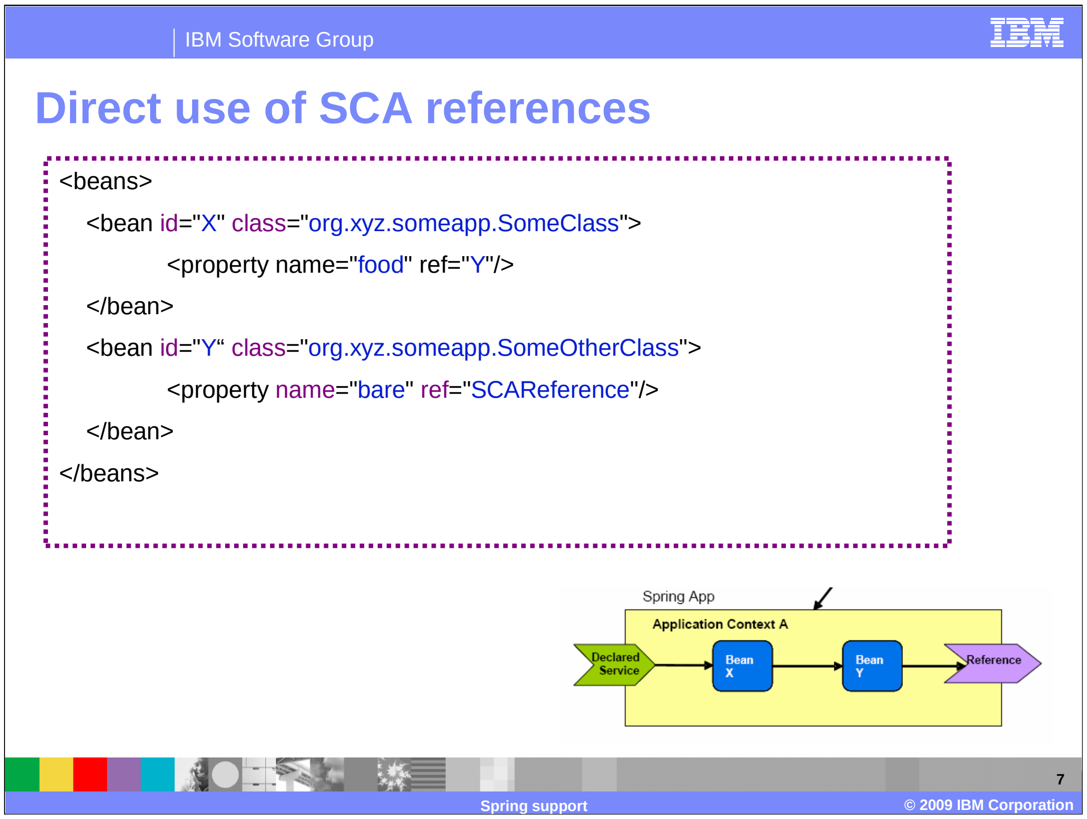
As mentioned in the previous slide, a component that uses Spring for an implementation can wire SCA services and references without introducing SCA metadata into the Spring configuration. The Spring context knows very little about the SCA environment. All policy enforcement occurs in the SCA runtime implementation and does not enter into the Spring space.



SCA runtime interaction with the Spring context

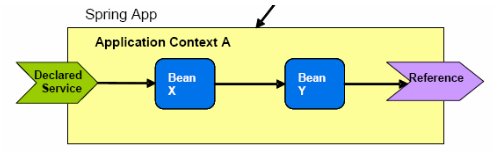


This diagram shows two of the points where the SCA runtime interacts with the Spring context: services and references. Any policy enforcement is done by the SCA runtime on calls into the Spring application context before the final message is delivered to the target Spring bean. On outbound calls from the application context, references supplied by the SCA can provide policy enforcement.



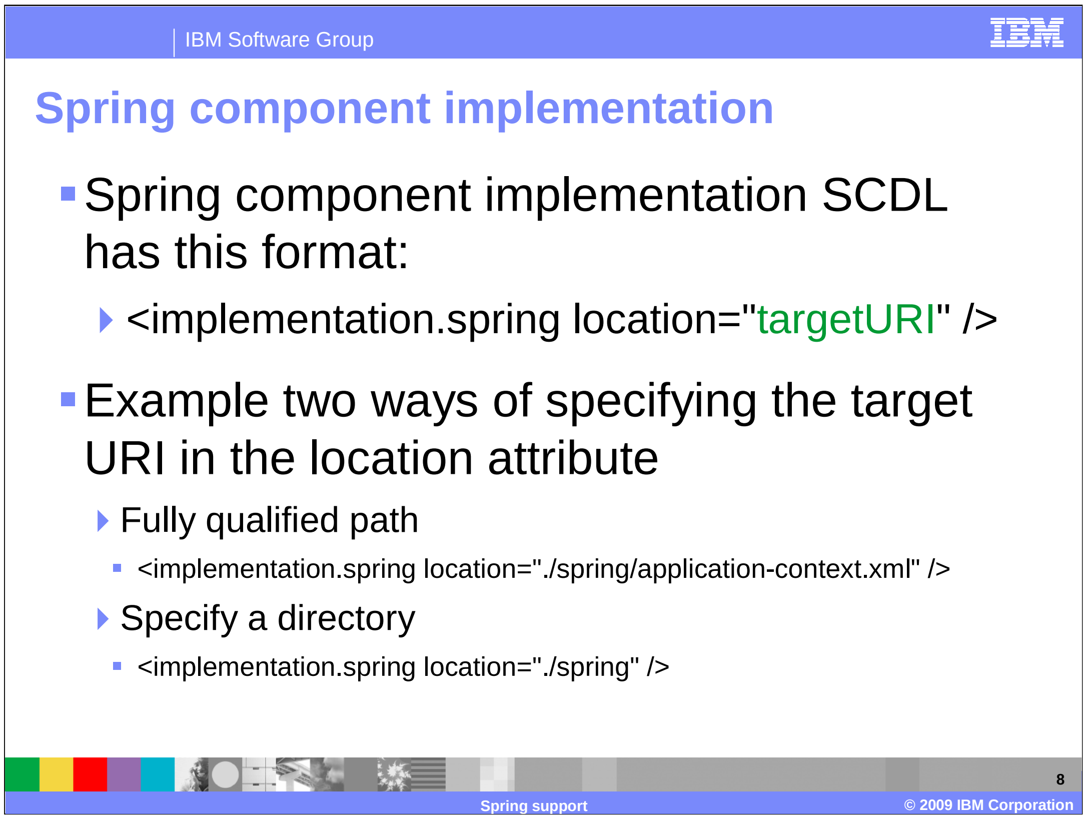
Direct use of SCA references

```
<beans>
   <bean id="X" class="org.xyz.someapp.SomeClass">
            <property name="food" ref="Y"/>
   </bean>
   <bean id="Y“ class="org.xyz.someapp.SomeOtherClass">
            <property name="bare" ref="SCAReference"/>
   </bean>
</beans>
```



Here is a look at a direct use of SCA references.

The SCA runtime hosting the Spring application context implementing a composite creates a parent application context in which all SCA references are defined as beans using the SCA reference name as the bean name. These beans are automatically visible in the child (or user application) context. The shown Spring configuration provides a model for Spring application context A shown in a previous slide. In that example, there are two Spring beans, X and Y. The bean named “X” is the entry point from SCA into the Spring context and Spring bean Y contains a reference to a service supplied by SCA. Two beans are defined. Taking a closer look at the code, the bean named “X” contains one property named “food” which refers to the second bean in the context, named “Y”. The bean “Y” also has a single property named “bare” which refers to the SCA service reference, given the name “SCAReference.”

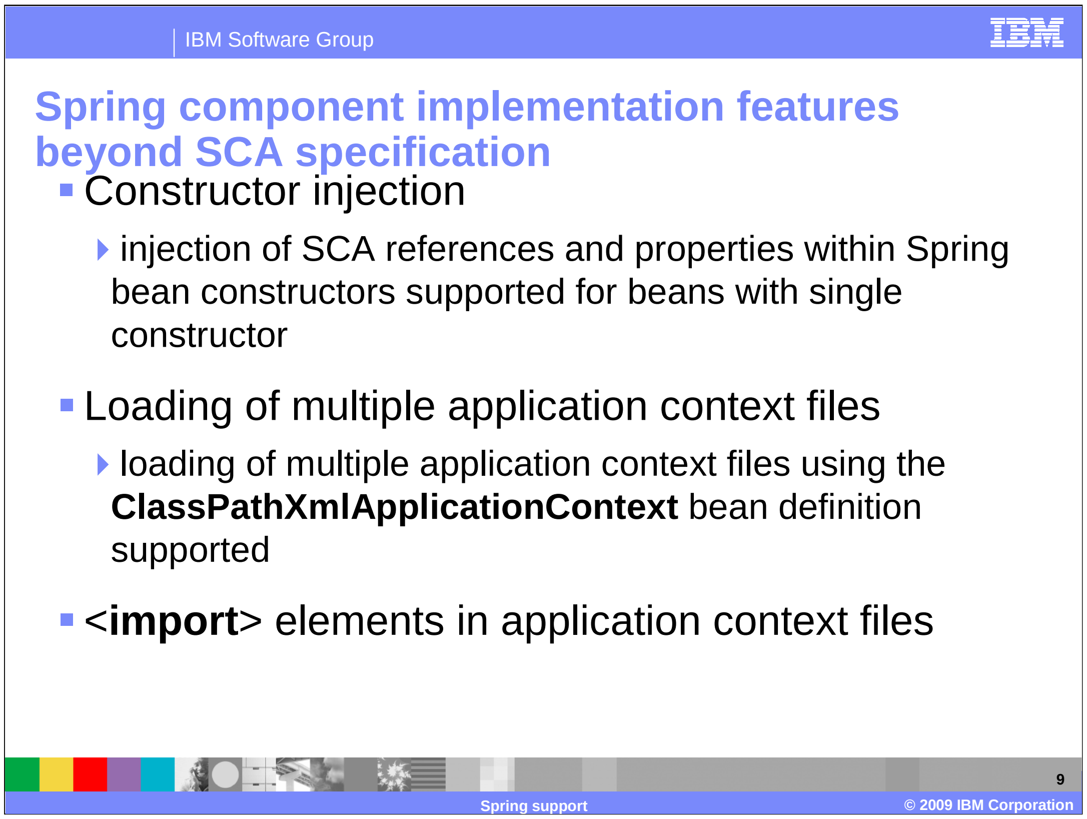


Spring component implementation

- Spring component implementation SCDL has this format:
  - <implementation.spring location="targetURI" />
- Example two ways of specifying the target URI in the location attribute
  - Fully qualified path
    - <implementation.spring location="./spring/application-context.xml" />
  - Specify a directory
    - <implementation.spring location="./spring" />

The Spring component implementation SCDL has the format shown here, where the location attribute of that element specifies the target URI of an archive file or directory or the fully qualified path that contains the Spring application context files.

An example of all three ways of specifying the target URI in the location attribute is shown in the information center under the topic “Using Spring 2.5.5 containers in Service Component Architecture applications.”



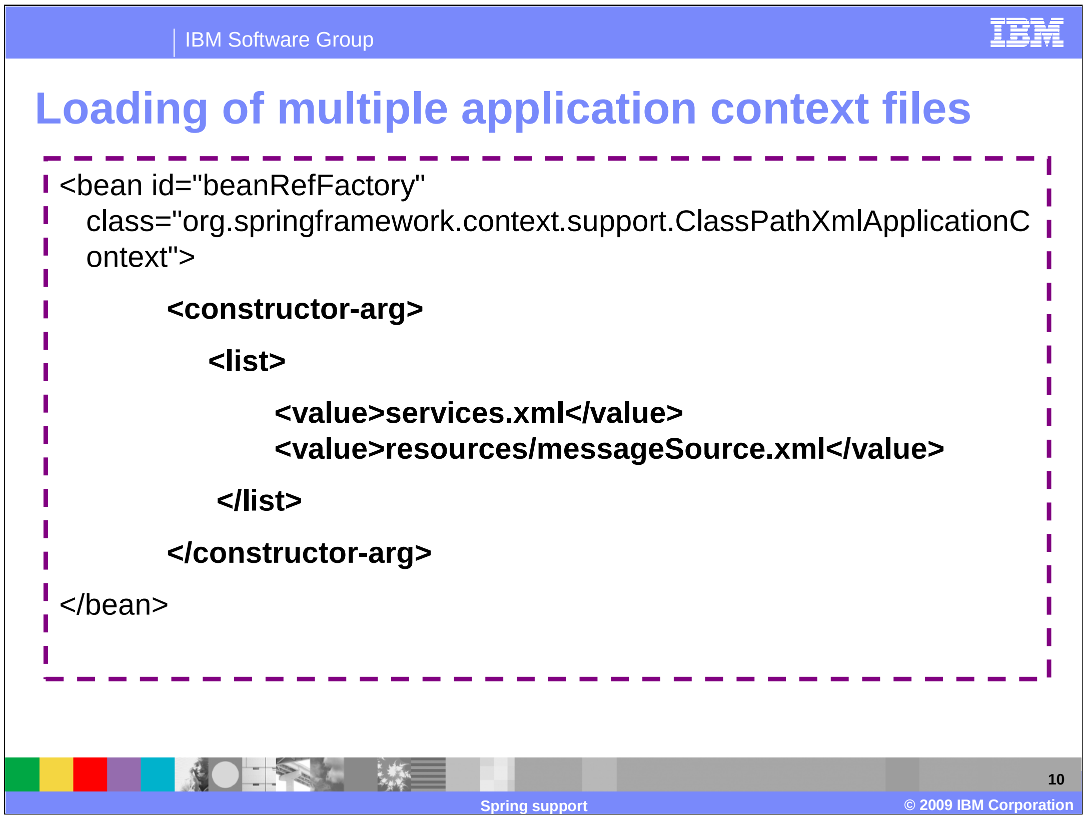
Spring component implementation features beyond SCA specification

- Constructor injection
  - injection of SCA references and properties within Spring bean constructors supported for beans with single constructor
- Loading of multiple application context files
  - loading of multiple application context files using the **ClassPathXmlApplicationContext** bean definition supported
- <**import**> elements in application context files

The feature pack supports the injection of SCA references and properties within Spring bean constructors. Define <constructor-arg> elements that specify the appropriate type of the SCA references or properties to use. If the elements do not specify the type attribute, then at least specify the index attribute. Note that currently the product does not support constructor injection for an unannotated Spring bean with multiple constructors.

The feature pack also supports loading of multiple application context files using the ClassPathXmlApplicationContext bean definition. In this case, use a list value that points to an application context XML file in the <constructor-arg> element.

The feature pack supports use of <import> elements in application context files. Each <import> element points to an application context XML file.

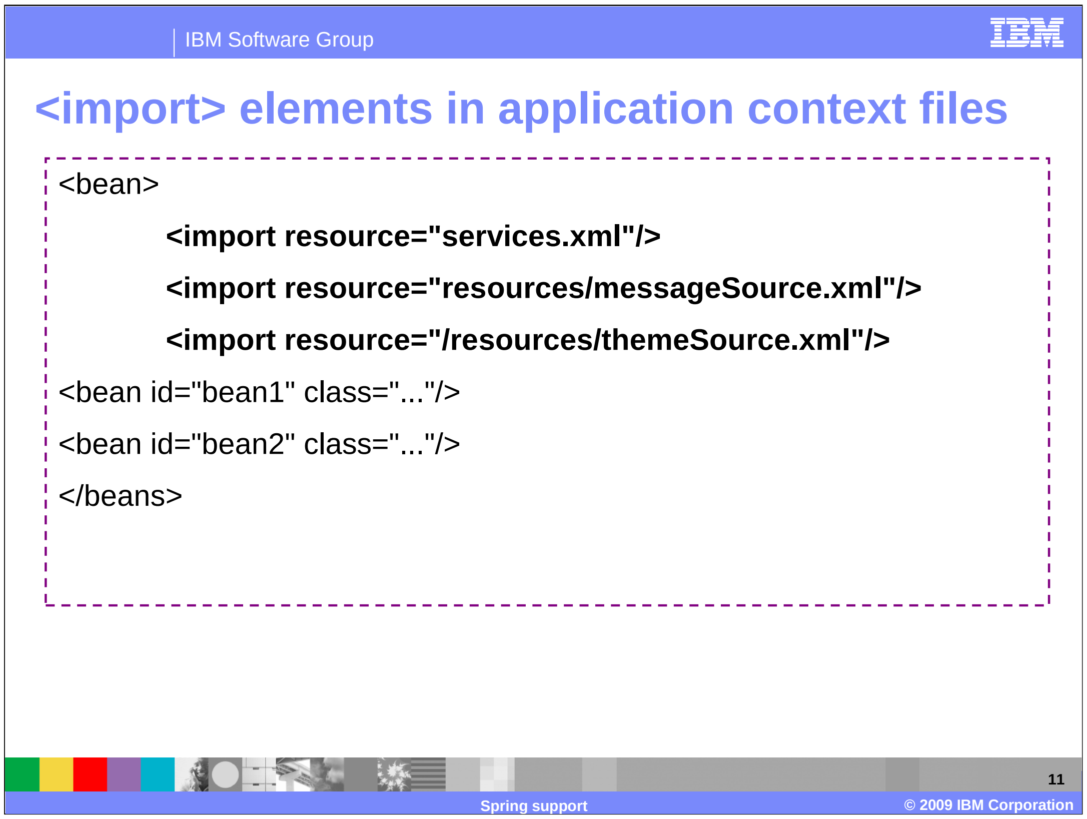


Loading of multiple application context files

```
<bean id="beanRefFactory"
   class="org.springframework.context.support.ClassPathXmlApplicationC
   ontext">
          <constructor-arg>
             <list>
                  <value>services.xml</value>
                  <value>resources/messageSource.xml</value>
              </list>
          </constructor-arg>
</bean>
```

Here is an example of loading of multiple application context files.

In this scenario, use a ***list*** value that points to an application context XML file in the <constructor-arg> element as shown here.

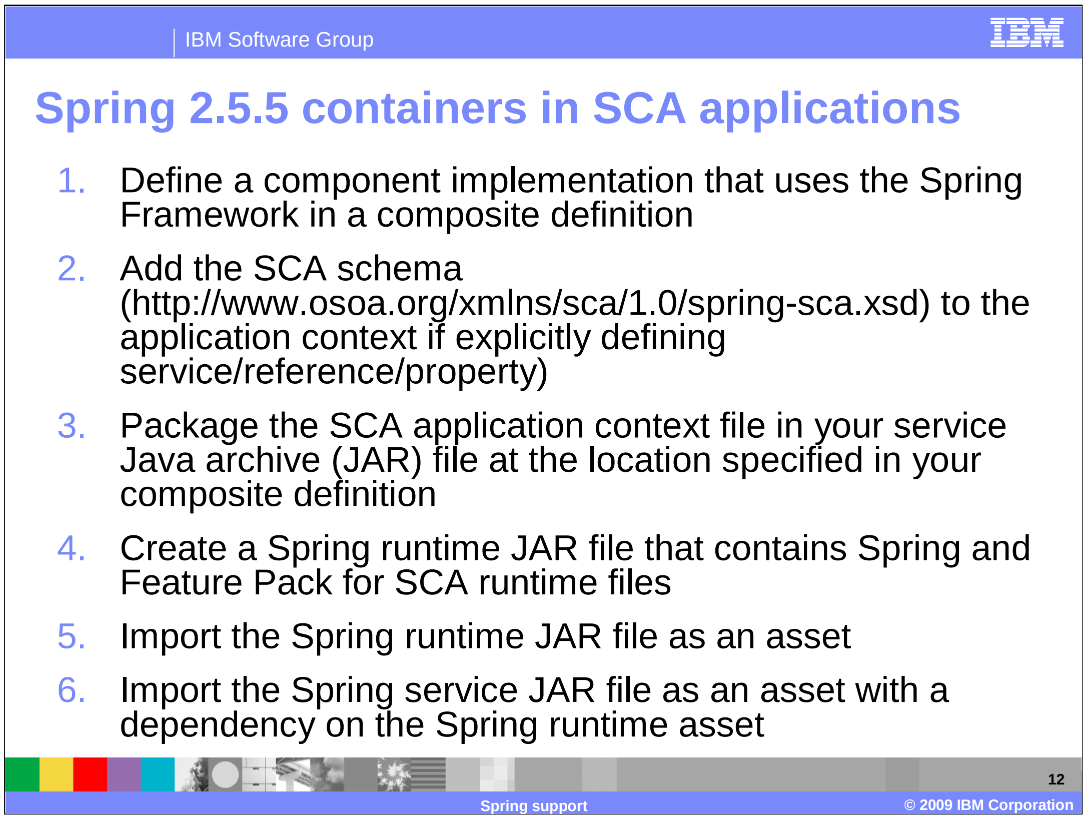


<import> elements in application context files

```
<bean>
        <import resource="services.xml"/>
        <import resource="resources/messageSource.xml"/>
        <import resource="/resources/themeSource.xml"/>
<bean id="bean1" class="..."/>
<bean id="bean2" class="..."/>
</beans>
```

Here is another example of the <import> elements in application context file.

Each **<import>** element points to an application context XML file as shown in this example.



Spring 2.5.5 containers in SCA applications

1. Define a component implementation that uses the Spring Framework in a composite definition
2. Add the SCA schema (http://www.osoa.org/xmlns/sca/1.0/spring-sca.xsd) to the application context if explicitly defining service/reference/property)
3. Package the SCA application context file in your service Java archive (JAR) file at the location specified in your composite definition
4. Create a Spring runtime JAR file that contains Spring and Feature Pack for SCA runtime files
5. Import the Spring runtime JAR file as an asset
6. Import the Spring service JAR file as an asset with a dependency on the Spring runtime asset

You can use the SCA programming model to invoke beans in a Spring 2.5.5 container. The SCA feature pack supports components implemented with Spring Framework that use <implementation.spring> in composite definitions.

Here are the steps to use Spring application context as an implementation within an SCA composite component:

1. First, define a component implementation that uses the Spring Framework in a composite definition. The Spring component implementation in a composite definition has this format:

<implementation.spring location="targetURI"/>

2. Next, add the SCA schema to the application context. Specify a Spring application context that defines the SCA schema namespace and makes the Spring application aware of the SCA-related beans.

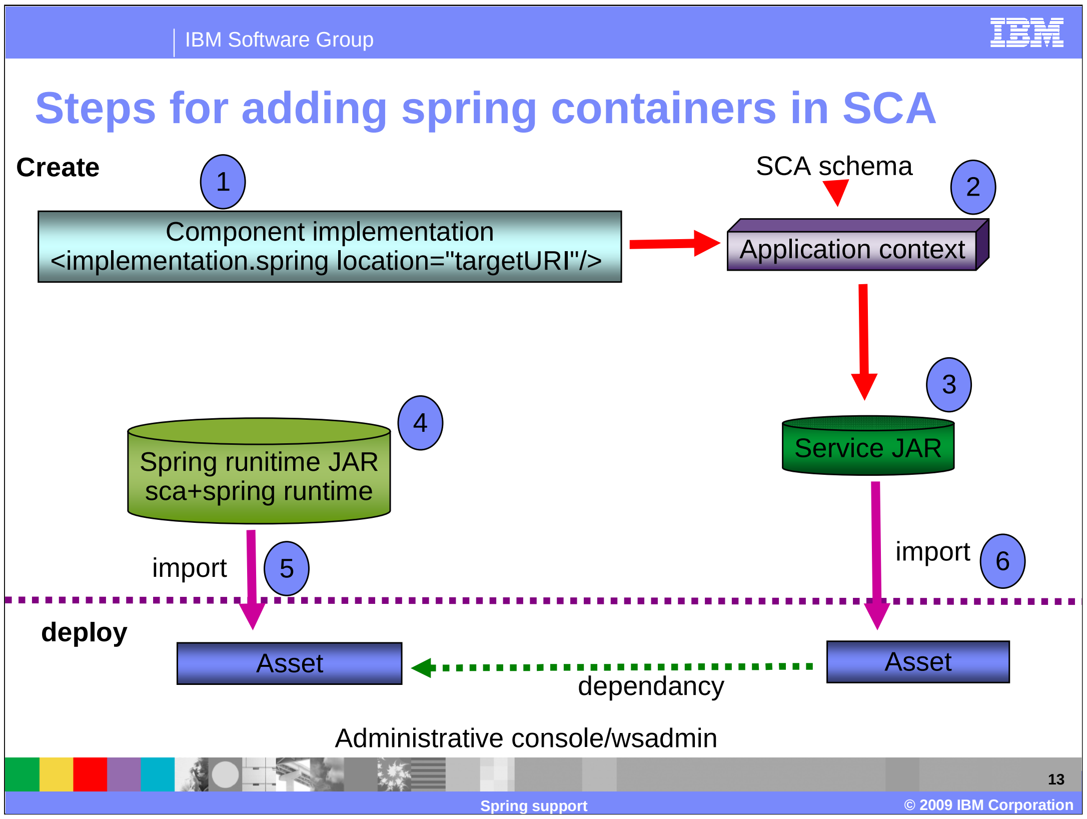
3. Package the SCA application context file in your service JAR file at the location specified in your composite definition. For example, the SCA application context file in a Spring service JAR file can be called helloworld-spring.jar.

4. Create a Spring runtime JAR file that contains Spring and Feature Pack for SCA runtime files. Because the Spring runtime binary files are not shipped with the product, you must create an asset that contains three Spring framework JAR files and one Feature Pack for SCA JAR file. See information center for more details on how to do this.

5. Import the Spring runtime JAR file as an asset using administrative console or wsadmin commands

6. Import the Spring service JAR file as an asset with a dependency on the Spring runtime asset. After you import your Spring service JAR file, create a dependency on the Spring runtime asset, springAsset.jar. The dependency enables the product to access the necessary Spring classes. The result is the Spring runtime JAR file and Spring service JAR file are imported assets available for use in a business-level application. The Spring service JAR asset has a dependency on the Spring runtime asset.

Note that the information center provides details of these steps. The next slide is a pictorial view of these steps.



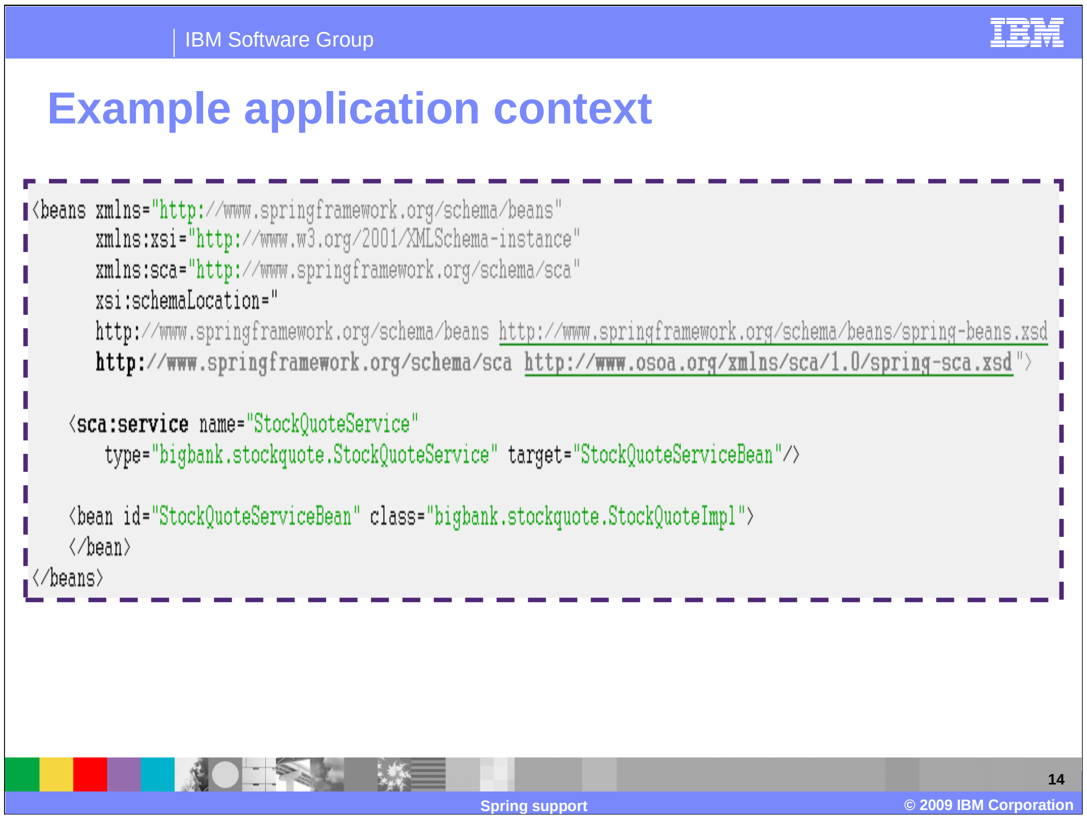
Steps for adding spring containers in SCA

Here is a picture of the main steps for adding spring containers in SCA as discussed in the previous slide. As a recap,

First, define a component implementation that uses the Spring Framework in a composite definition.

Then add the SCA schema to the application context. Package the SCA application context file in your service JAR file at the location specified in your composite definition. You then create a Spring runtime JAR file that contains Spring and Feature Pack for SCA runtime files. Import the Spring runtime JAR file as an asset in the administrative console. Finally, in step 6, import the Spring service JAR file as an asset with a dependency on the Spring runtime asset in the administrative console.

There are a couple of slides provided that show you how to import the jars as assets and to create dependancy on the administrative console.



IBM Software Group

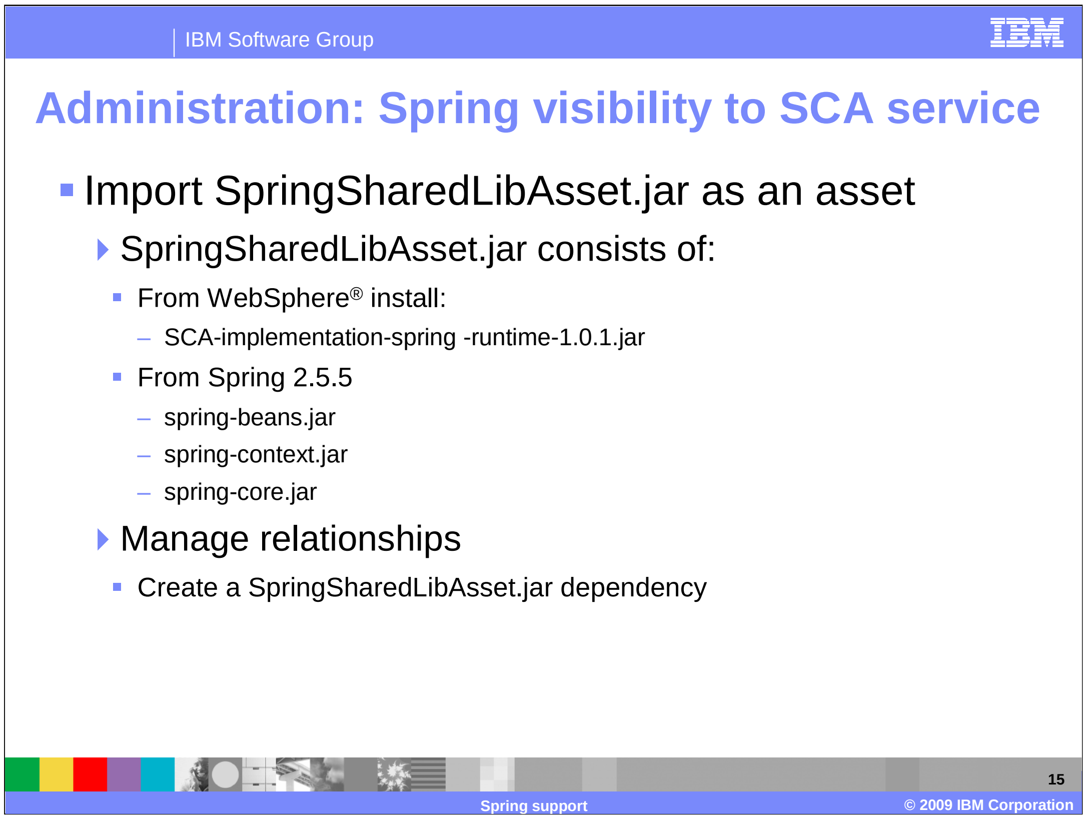
Example application context

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:sca="http://www.springframework.org/schema/sca"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/sca http://www.osoa.org/xmlns/sca/1.0/spring-sca.xsd">

    <sca:service name="StockQuoteService"
        type="bigbank.stockquote.StockQuoteService" target="StockQuoteServiceBean"/>

    <bean id="StockQuoteServiceBean" class="bigbank.stockquote.StockQuoteImpl">
    </bean>
</beans>
```

14

Spring support © 2009 IBM Corporation

Here is an example of an application context that uses the explicit mapping and with the schema added as detailed in step 2 of the previous slide.

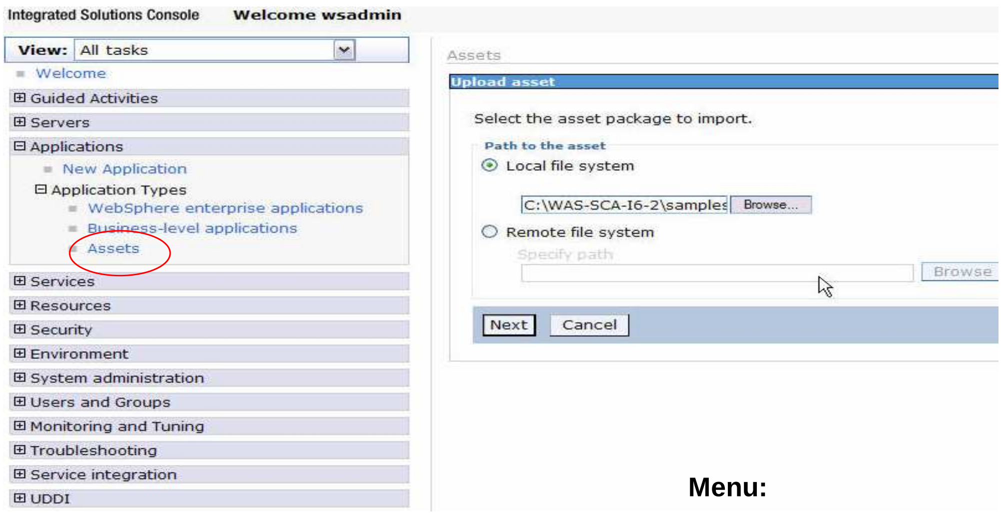


Administration: Spring visibility to SCA service

- Import SpringSharedLibAsset.jar as an asset
  - SpringSharedLibAsset.jar consists of:
    - From WebSphere® install:
      - SCA-implementation-spring -runtime-1.0.1.jar
    - From Spring 2.5.5
      - spring-beans.jar
      - spring-context.jar
      - spring-core.jar
  - Manage relationships
    - Create a SpringSharedLibAsset.jar dependency

On the administrative side, to make Spring visible to an SCA service, first you import SpringSharedLibAsset.jar as an asset using the Asset menu in the administrative console. This SpringSharedLibAsset.jar is made up of SCA-implementation-spring -runtime-1.0.1.jar from the WebSphere Application Server installation, and spring-beans.jar, spring-context.jar, and spring-core.jar from Spring 2.5.5. During the jar import, create a dependency on SpringSharedLibAsset.jar. Once you have imported the jar with the dependency, when the jar is added to a business level application, the administrative code will automatically add the SpringSharedLibAsset.jar to the business level application. For example, when you add the helloworld-spring.jar to a business level application, the administrative code will automatically pull in a copy of SpringSharedLibAsset.jar.

Import Spring SCA asset



Select the asset package to import.

Path to the asset

[✓ Local file system] C:\WAS-SCA-I6-2\samples

Remote file system

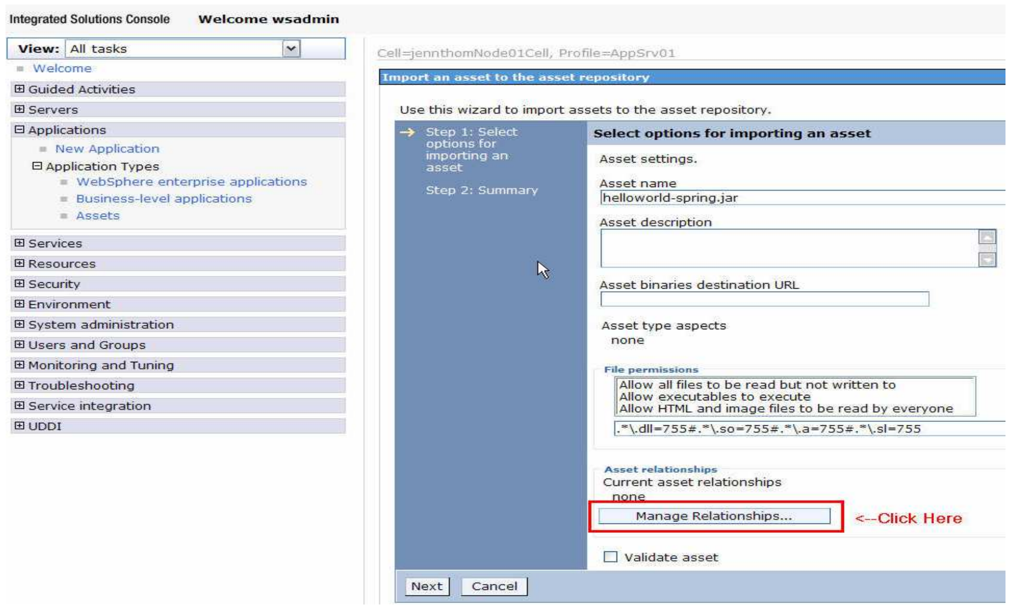
Specify path

Menu:

Applications->Application Types->Assets

On the menu click Applications->Application Types->Assets ->Import. Locate Spring SCA asset and import it, then click **Next.**

Manage relationships


Integrated Solutions Console Welcome wsadmin

View: All tasks

Cell=jennthomNode01Cell, Profile=AppSrv01

Import an asset to the asset repository

Use this wizard to import assets to the asset repository.

Step 1: Select options for importing an asset

Step 2: Summary

Select options for importing an asset

Asset settings.

Asset name: helloworld-spring.jar

Asset description

Asset binaries destination URL

Asset type aspects: none

File permissions

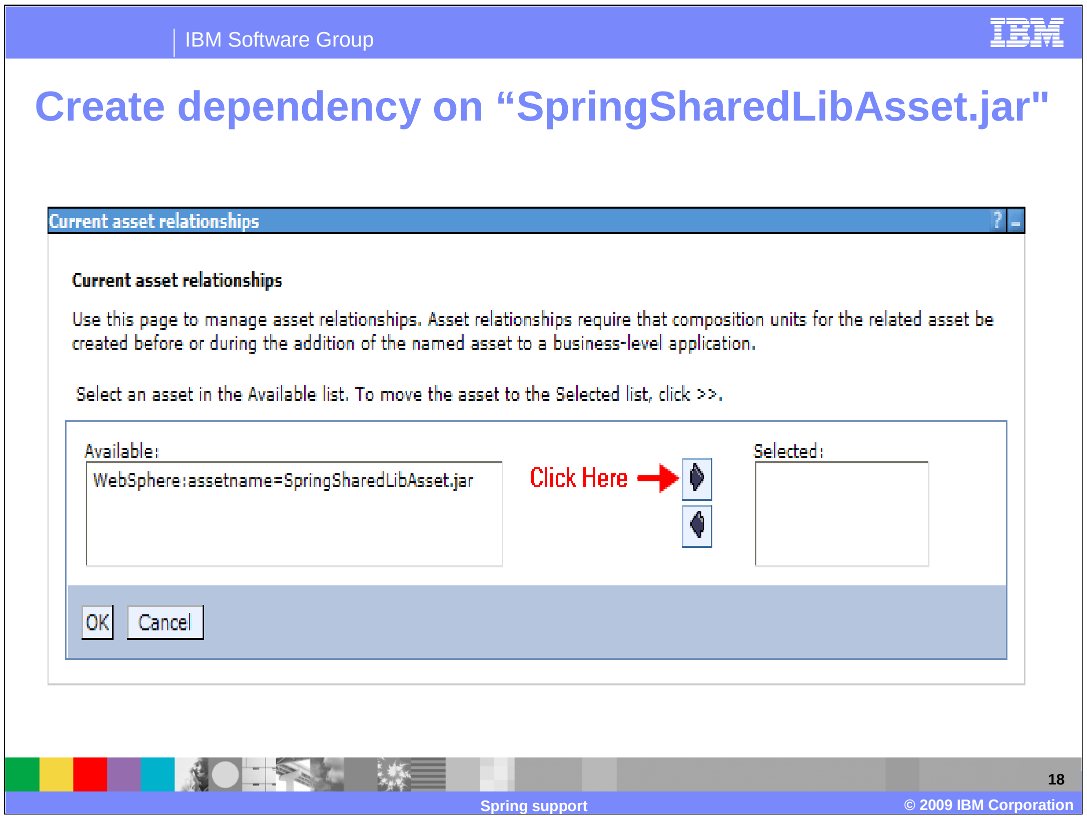
Allow all files to be read but not written to
Allow executables to execute
Allow HTML and image files to be read by everyone

.*\.dll=755#.*\.so=755#.*\.a=755#.*\.sl=755

Asset relationships

Current asset relationships: none

On the next screen click “Manage Relationships…” to access that menu option.


IBM Software Group

Create dependency on “SpringSharedLibAsset.jar"

Current asset relationships

Current asset relationships

Use this page to manage asset relationships. Asset relationships require that composition units for the related asset be created before or during the addition of the named asset to a business-level application.

Select an asset in the Available list. To move the asset to the Selected list, click >>.

Available:
WebSphere:assetname=SpringSharedLibAsset.jar

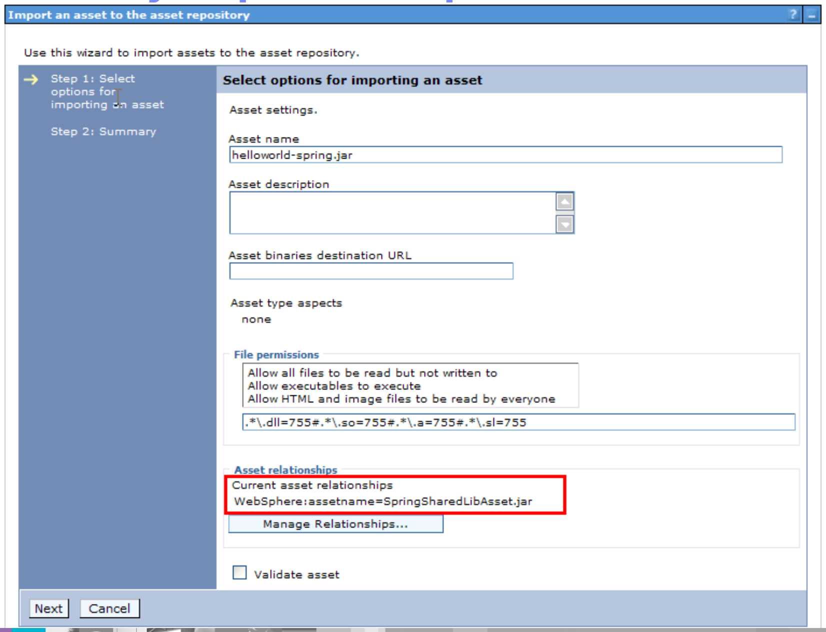
18

Spring support © 2009 IBM Corporation

On the “**Current asset relationships**” screen, create a dependency on " SpringSharedLibAsset.jar " by moving " SpringSharedLibAsset.jar " to the right then click OK.

Note that if the SpringSharedLibAsset.jar is not available, add the asset first then open the asset and then access the “Manage Relationships..” menu where you can then create the dependency.

Dependancy in place - Import asset



Import an asset to the asset repository

Use this wizard to import assets to the asset repository.

Select options for importing an asset

Asset settings.

Asset name: helloworld-spring.jar

Asset description:

Asset binaries destination URL:

Asset type aspects: none

File permissions

Allow all files to be read but not written to
Allow executables to execute
Allow HTML and image files to be read by everyone

.*\.dll=755#.*\.so=755#.*\.a=755#.*\.sl=755

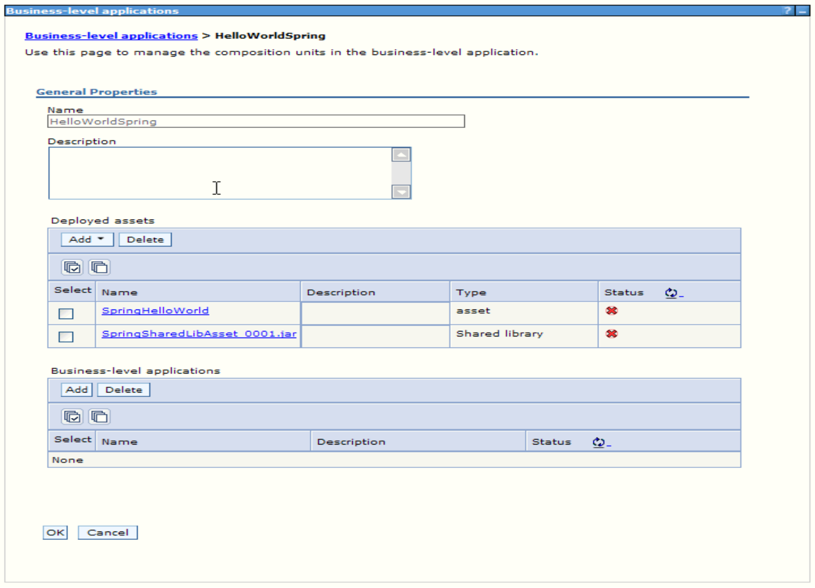
Asset relationships

Current asset relationships
WebSphere:assetname=SpringSharedLibAsset.jar

You can see the dependency has been created and you can proceed to import the asset as usual.

Once you have imported the jar with the dependency, when the jar is added to a business level application, the administrative code will automatically add the SpringSharedLibAsset.jar to the business level application. For example, when you add the helloworld-spring.jar to a business level application, the administrative code will automatically pull in a copy of SpringSharedLibAsset.jar.

Hello world example



Business-level applications

Business-level applications > HelloWorldSpring

Use this page to manage the composition units in the business-level application.

General Properties

Name: HelloWorldSpring

Description:

Deployed assets

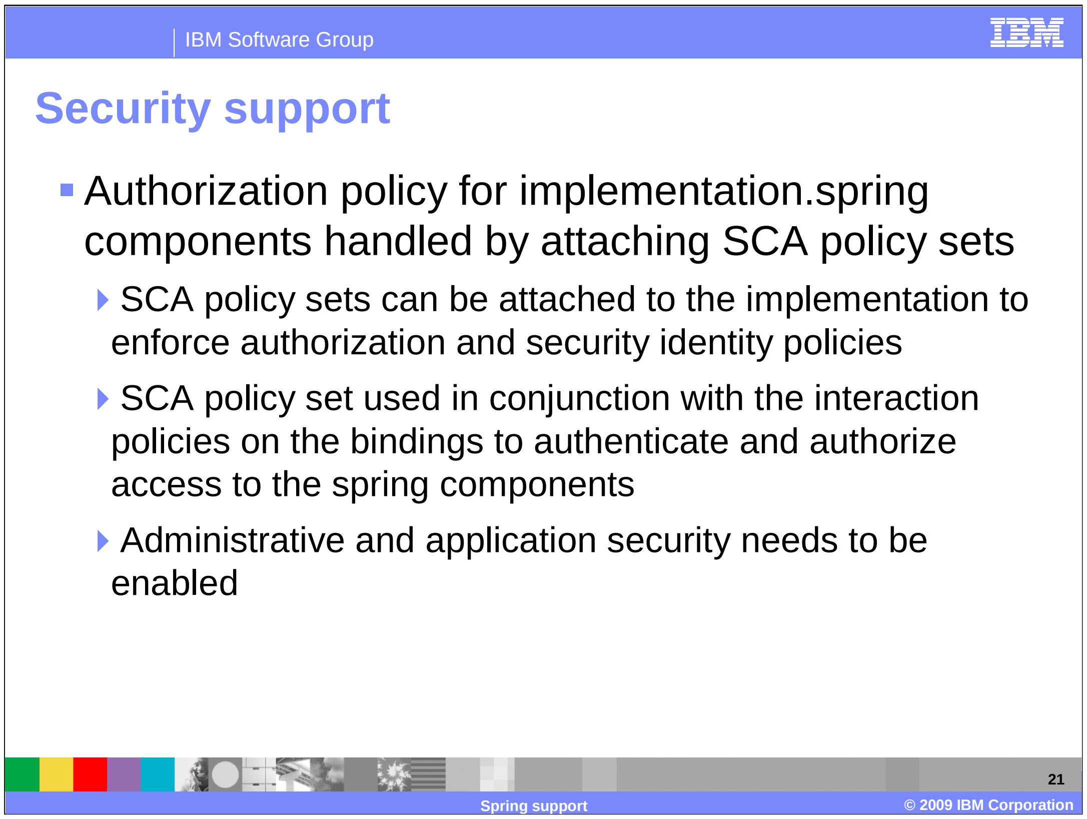
Select	Name	Description	Type	Status
☐	SpringHelloWorld		asset	✖
☐	SpringSharedLibAsset_0001.jar		Shared library	✖

Business-level applications

Select	Name	Description	Status
None			

As a follow-up example of the “hello world” mentioned in the previous slide, if you create a new HelloWorldSpring business level application and add helloworld-spring.jar asset to that application, your application will contain two assets: SpringHelloWorld and SpringSharedLibAsset_0001.jar, as shown. The SpringSharedLibAsset.jar file is the shared library you created as explained in the previous slides.

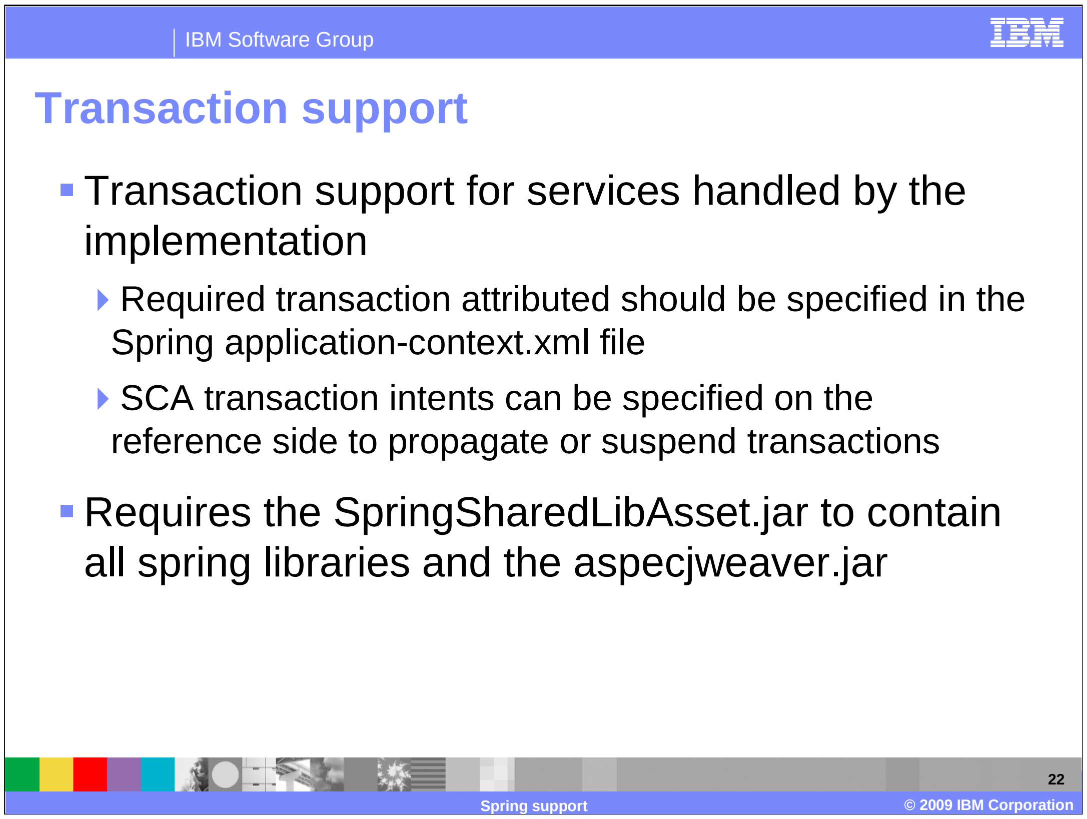
The Result is an SCA component that uses implemention.spring and is visible to the SCA runtime.



Security support

- Authorization policy for implementation.spring components handled by attaching SCA policy sets
  - SCA policy sets can be attached to the implementation to enforce authorization and security identity policies
  - SCA policy set used in conjunction with the interaction policies on the bindings to authenticate and authorize access to the spring components
  - Administrative and application security needs to be enabled

Authorization and security identity for implementation.spring components is handled by the SCA container. SCA policy sets, defined in the definitions.xml file, can be attached to the implementation to enforce authorization and security identity policies. These policy sets are used in conjunction with the interaction policies on the bindings to authenticate and authorize access to the spring components. Administrative and application security needs to be enabled in order for security roles to be enforced.


IBM Software Group

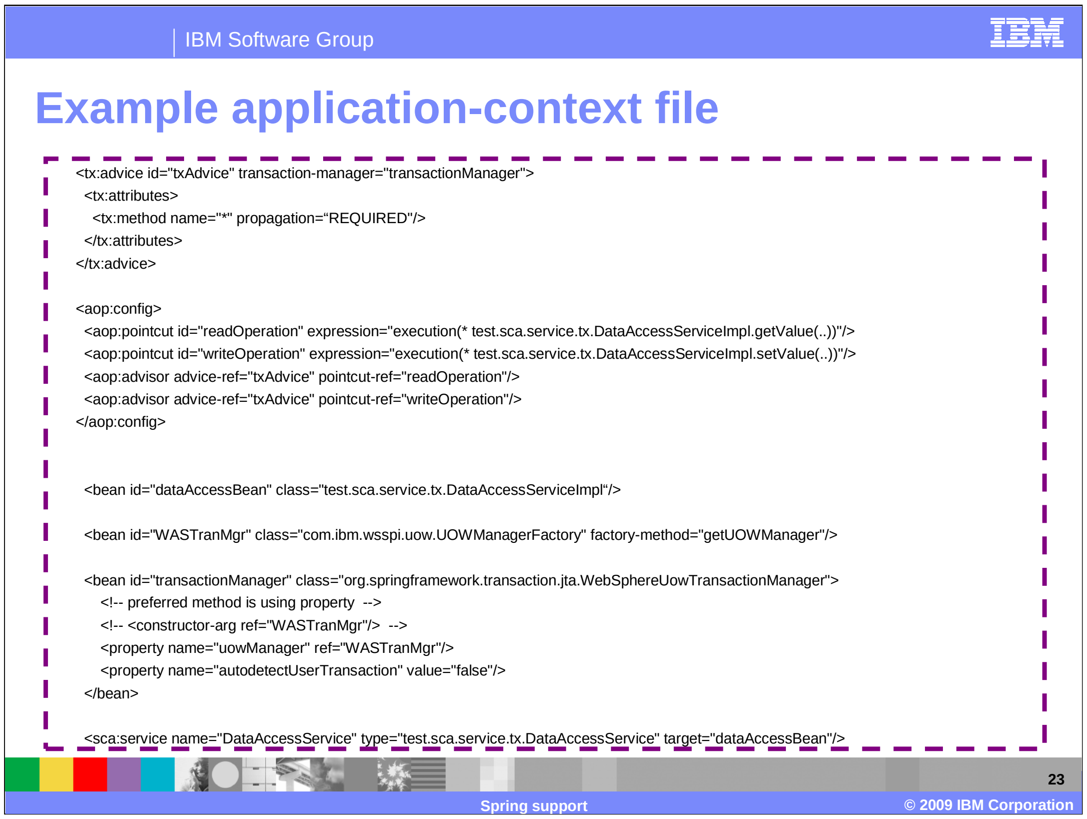
Transaction support

- Transaction support for services handled by the implementation
  - Required transaction attributed should be specified in the Spring application-context.xml file
  - SCA transaction intents can be specified on the reference side to propagate or suspend transactions
- Requires the SpringSharedLibAsset.jar to contain all spring libraries and the aspecjweaver.jar

22

Spring support © 2009 IBM Corporation

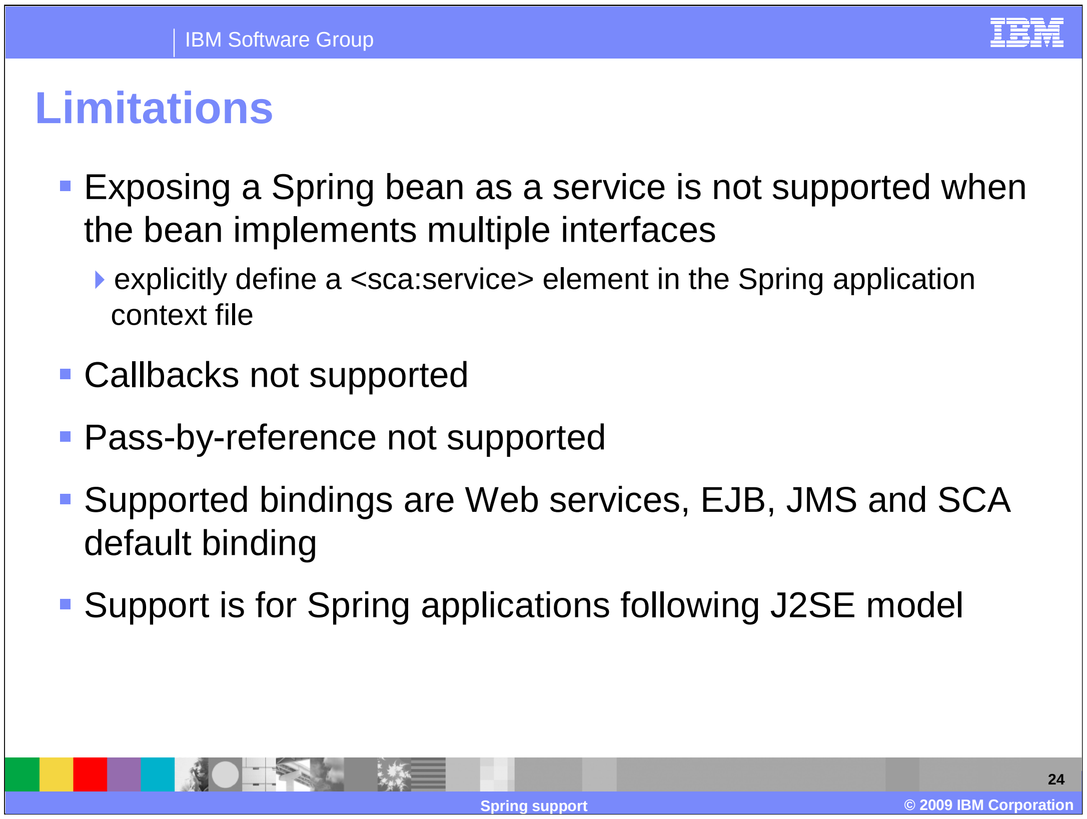
Transaction support for services are handled by the implementation. The required transaction attributed should be specified in the Spring application-context.xml file. SCA transaction intents can be specified on the reference side to propagate or suspend transactions. This requires the SpringSharedLibAsset.jar to contain all spring libraries and the aspecjweaver.jar.



Example application-context file

```
<tx:advice id="txAdvice" transaction-manager="transactionManager">
  <tx:attributes>
    <tx:method name="*" propagation="REQUIRED"/>
  </tx:attributes>
</tx:advice>

<aop:config>
  <aop:pointcut id="readOperation" expression="execution(* test.sca.service.tx.DataAccessServiceImpl.getValue(..))"/>
  <aop:pointcut id="writeOperation" expression="execution(* test.sca.service.tx.DataAccessServiceImpl.setValue(..))"/>
  <aop:advisor advice-ref="txAdvice" pointcut-ref="readOperation"/>
  <aop:advisor advice-ref="txAdvice" pointcut-ref="writeOperation"/>
</aop:config>


  <bean id="dataAccessBean" class="test.sca.service.tx.DataAccessServiceImpl"/>

  <bean id="WASTranMgr" class="com.ibm.wsspi.uow.UOWManagerFactory" factory-method="getUOWManager"/>

  <bean id="transactionManager" class="org.springframework.transaction.jta.WebSphereUowTransactionManager">
      <!-- preferred method is using property  -->
      <!-- <constructor-arg ref="WASTranMgr"/>  -->
      <property name="uowManager" ref="WASTranMgr"/>
      <property name="autodetectUserTransaction" value="false"/>
  </bean>

  <sca:service name="DataAccessService" type="test.sca.service.tx.DataAccessService" target="dataAccessBean"/>
```

Here is an example of application-context file with transaction attributes.



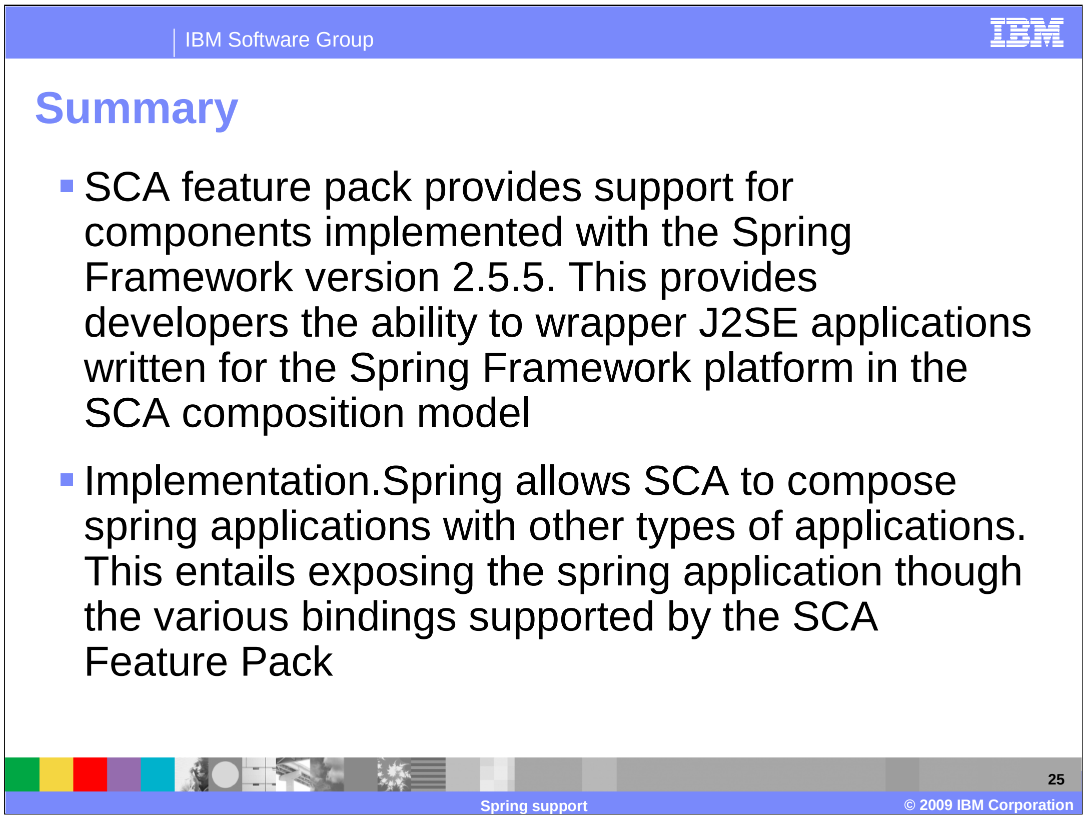
Limitations

- Exposing a Spring bean as a service is not supported when the bean implements multiple interfaces
  - explicitly define a <sca:service> element in the Spring application context file
- Callbacks not supported
- Pass-by-reference not supported
- Supported bindings are Web services, EJB, JMS and SCA default binding
- Support is for Spring applications following J2SE model

Exposing a Spring bean as a service is not supported when the bean implements multiple interfaces. To resolve this problem, explicitly define an <sca:service> element in the Spring application context file. If no explicit definition of <sca:service> is available, the problem remains by default when you expose all the beans as defined in the Spring context as services. Also, callbacks and pass-by-reference are not supported.

The only supported bindings are Web services, EJB, JMS and SCA default binding.

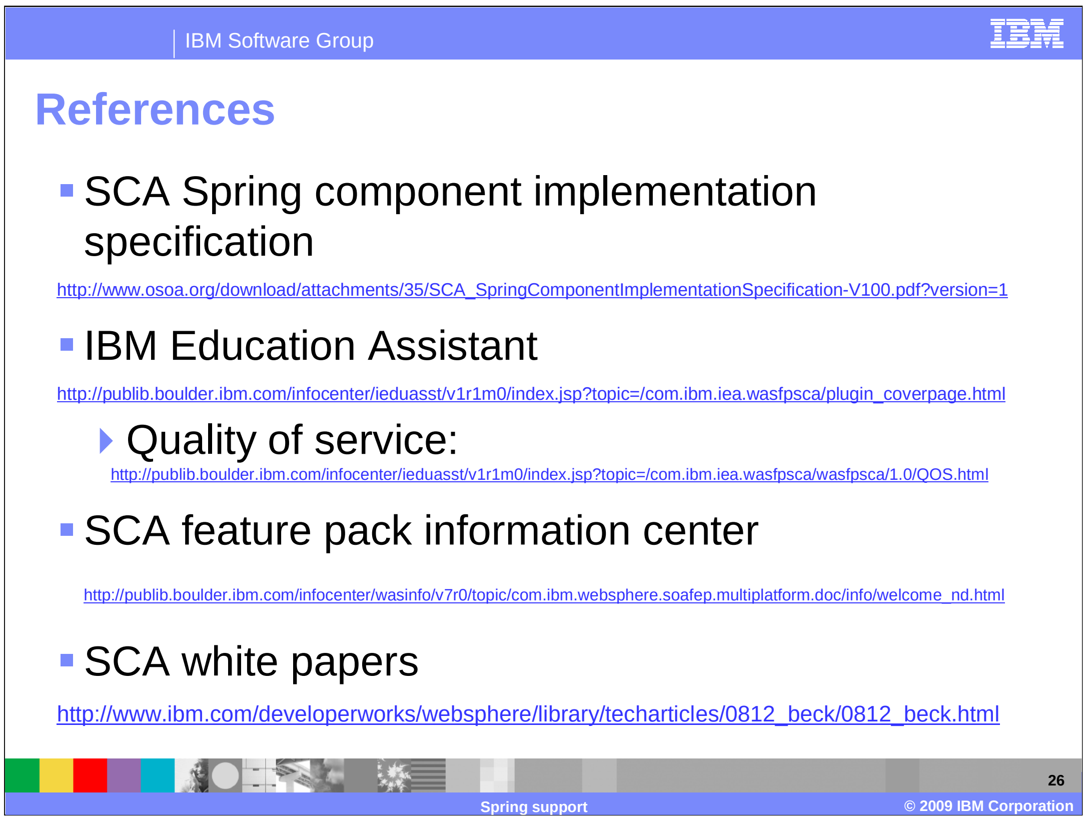
The Spring framework in SCA uses the J2SE model and, like SCA applications, does not have access to the local namespace, which is required for elements accessed through “java:comp…” . Use explicit or direct JNDI lookups instead.



Summary

- SCA feature pack provides support for components implemented with the Spring Framework version 2.5.5. This provides developers the ability to wrapper J2SE applications written for the Spring Framework platform in the SCA composition model
- Implementation.Spring allows SCA to compose spring applications with other types of applications. This entails exposing the spring application though the various bindings supported by the SCA Feature Pack

To summarize, the Spring Framework is an open source project that provides a framework for simple Java objects. It enables them to use the Java EE container through wrapper classes and XML configuration. SCA feature pack provides support for components implemented with the Spring Framework version 2.5.5. Implementation.Spring allows SCA to compose spring applications with other types of applications. It also exposes the spring application through the various bindings supported by the SCA Feature Pack.



References

- SCA Spring component implementation specification

http://www.osoa.org/download/attachments/35/SCA_SpringComponentImplementationSpecification-V100.pdf?version=1

- IBM Education Assistant

http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.wasfpsca/plugin_coverpage.html

  - Quality of service:

    http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.wasfpsca/wasfpsca/1.0/QOS.html

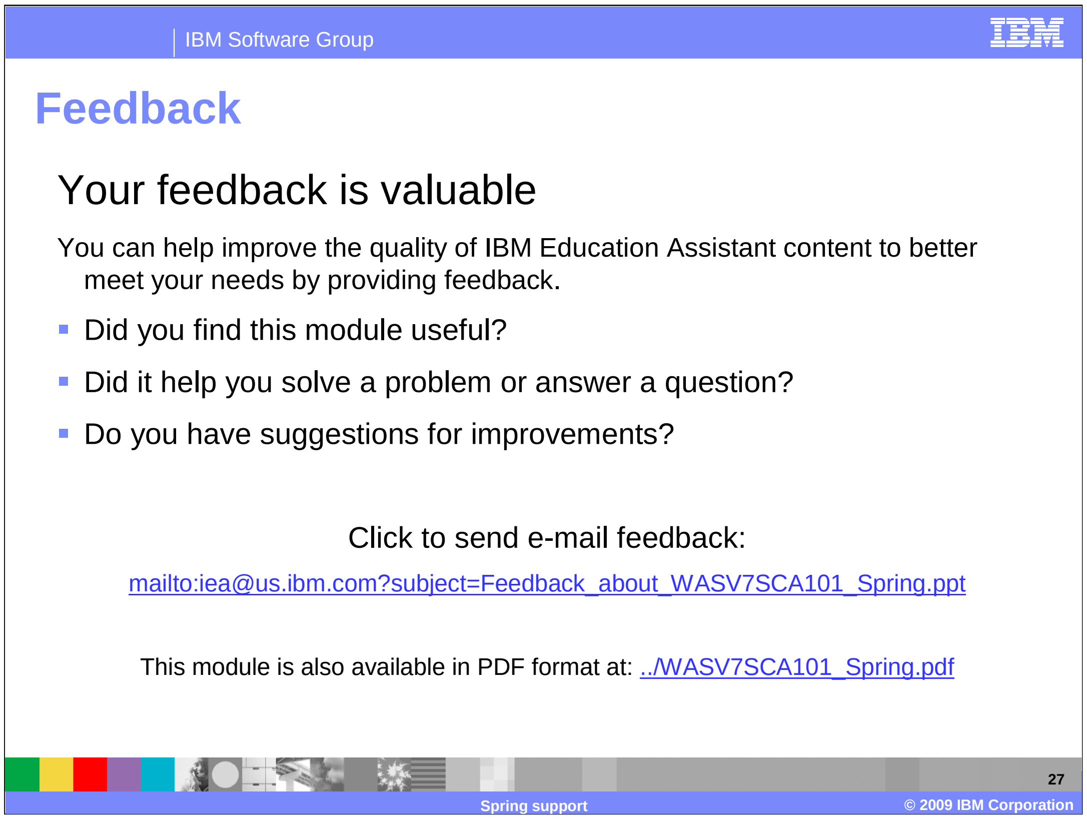
- SCA feature pack information center

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.soafep.multiplatform.doc/info/welcome_nd.html

- SCA white papers

http://www.ibm.com/developerworks/websphere/library/techarticles/0812_beck/0812_beck.html

Here is a list of useful reference links.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

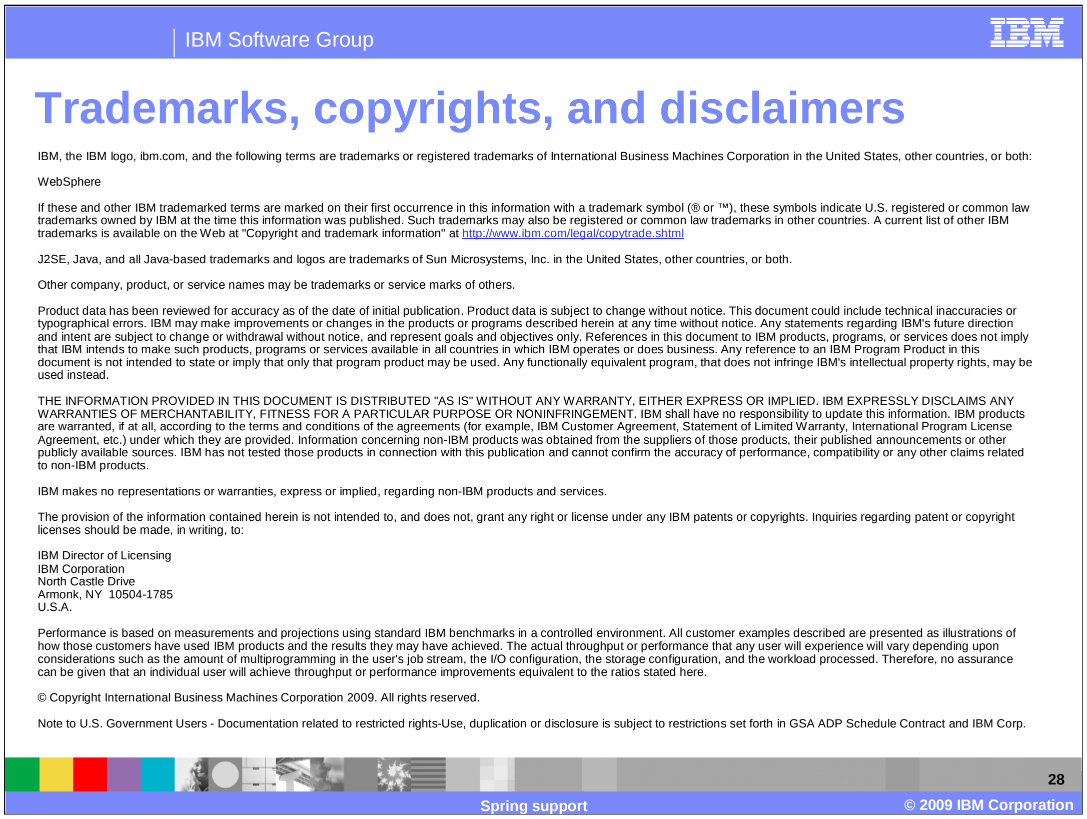
- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WASV7SCA101_Spring.ppt

This module is also available in PDF format at: ../WASV7SCA101_Spring.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.



Trademarks, copyrights, and disclaimers