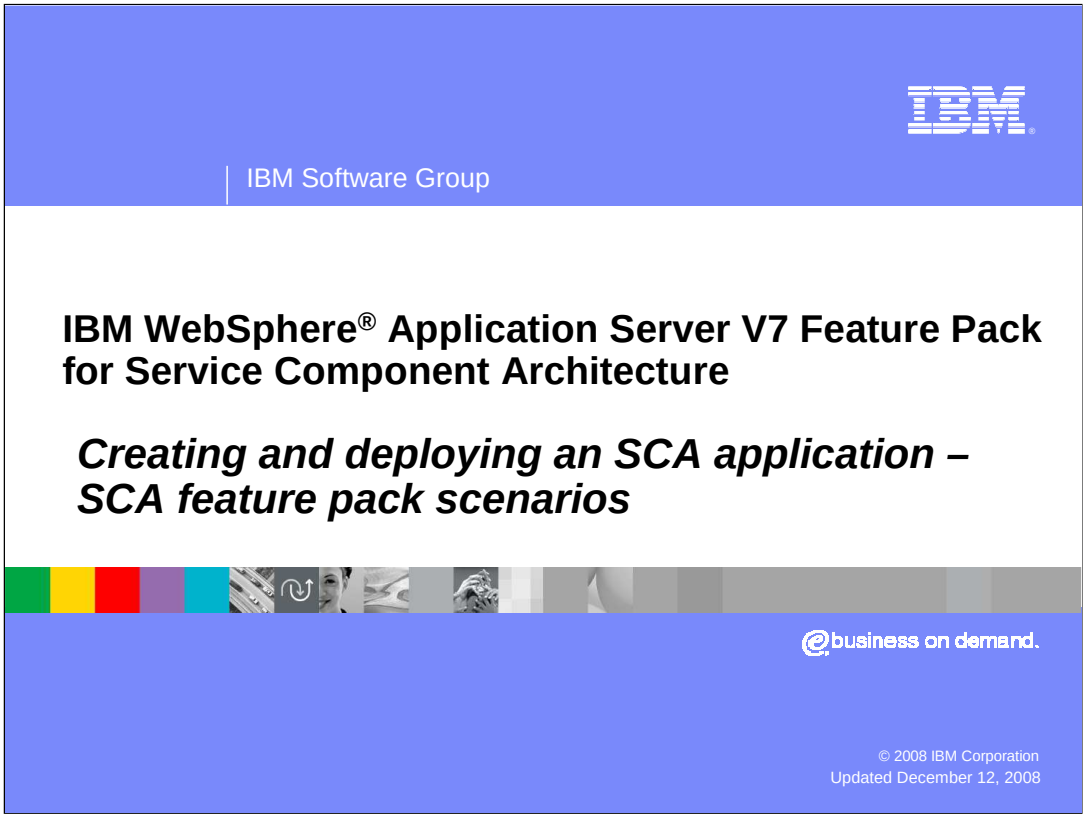


This presentation will focus on SCA feature pack scenarios.

Section

SCA feature pack scenarios

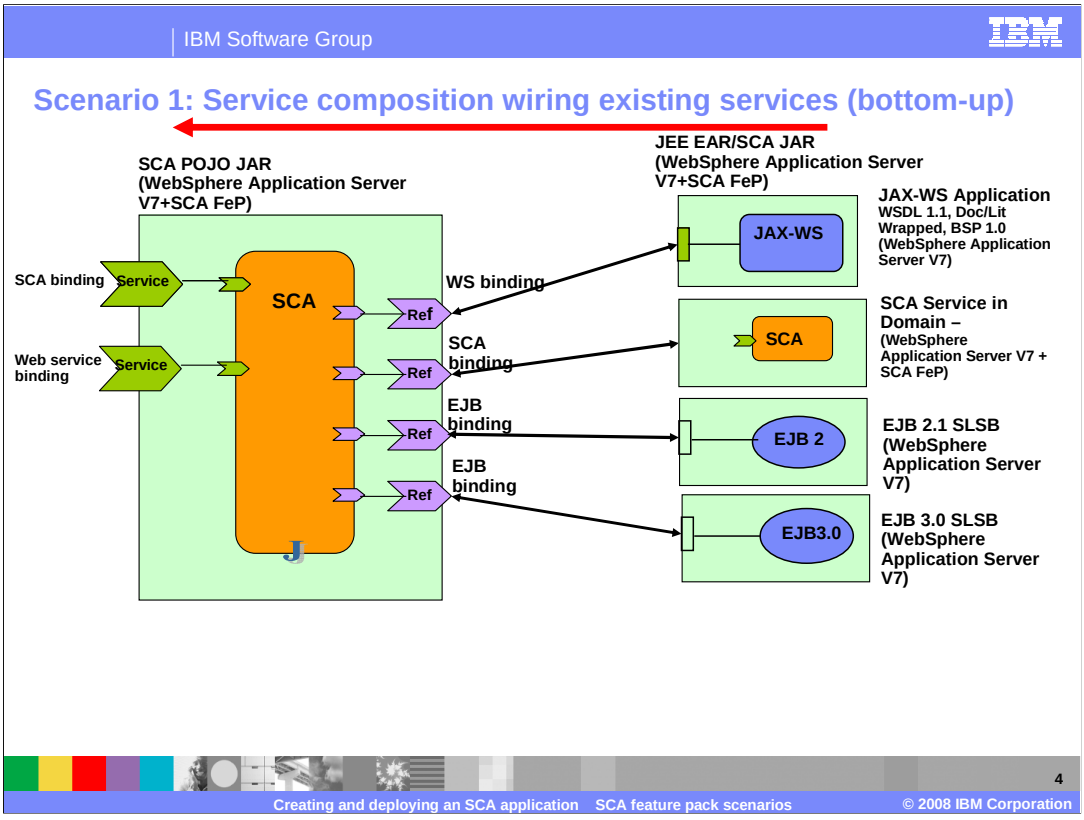
Let's look at some SCA feature pack scenarios.

SCA feature pack scenarios: overview

- There are three key scenarios that demonstrate the big picture of the functional aspects of the SCA feature pack
  - Scenario 1: bottom-up approach - service composition wiring existing services
  - Scenario 2: top-down approach - simple service development
  - Scenario 3: meet-in-the-middle approach - service evolution (flexibility and agility)

There are three key scenarios that show the big picture of the functional aspects of the SCA feature pack.

The first scenario is the bottom-up approach where existing services are wired by a service composition. In this approach, the key aspects are composition and using existing business services. The second scenario is the top-down approach. In this one the key aspects are ease of use, jar deployment, annotations, JAX-B generation and minimal administration and configuration. The third scenario is the meet-in-the-middle approach. The key aspects of this approach are demonstrating flexibility and agility as the application is extended using SCA principles, enterprise principles using industry defined schemas and defined business faults.



Let's look at the first scenario which is a bottom-up scenario. You should read this picture from **right to left**.

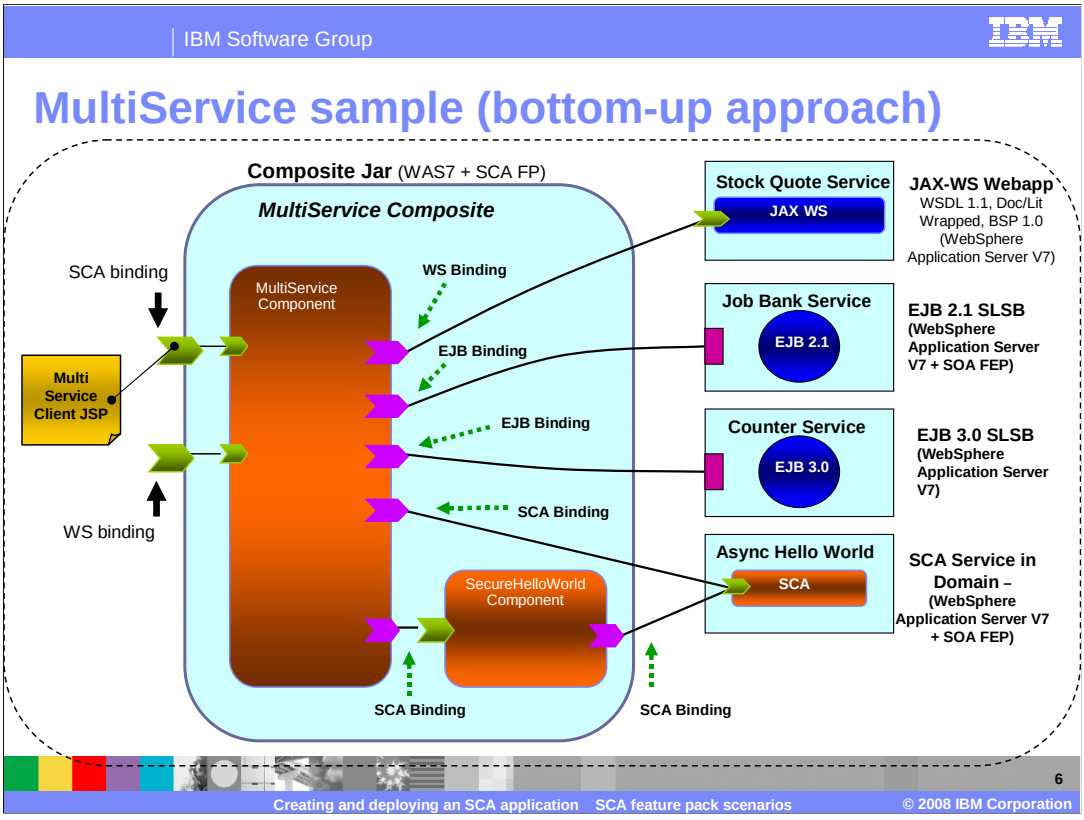
As mentioned in the previous slide, the bottom-up approach means that you are creating new services using existing interfaces defined by existing services using the technology of their existing implementation.

In this example, a Java™ POJO exposes a new service over a Web service binding and the default (SCA) binding, and is deployed into an SCA domain using JAX-B data binding. These new services are a composite of existing available services, synchronous and asynchronous requests with single sign-on, deployed on a different server across the technologies mentioned below. Web service is one of the technologies. In this scenario, Web service is implemented using Web application. The Web service is available over two ports, WSDL 1.1, SOAP 1.1 and 1.2. EJB2 is another technology. The Java POJO service can also be deployed on a different server across an existing remote method on a 2.1 stateless session bean. This will be a Synchronous request with Business Faults and Authentication. Similarly with EJB3, the POJO service can be deployed across an existing remote method on a 3.0 stateless session bean. This will also be Synchronous request with Business Faults and Authentication. Last but not least, the POJO service can be deployed across an existing deployed SCA component in the same domain. This will be both Asynchronous and Synchronous Requests with SCA service being deployed in a domain.

Example of bottom-up approach - MultiService

- Service composition using existing services
- Wires to existing apps
- Wiring to services over
  - Default binding
  - Web services binding
  - EJB binding for both EJB 2.0 and EJB 3.0 services
- Exposes one business service over the default binding and the Web service binding that returns a JAXB object
- Client JSP uses the business service exposed over the default binding

A sample that ships with the SCA feature pack is called MultiService. This sample uses a bottom-up approach. It shows service composition using existing services. The MultiService sample wires to several existing samples which includes Stock Quote, async HelloWorld, Job Bank Service, and EJB3 Counter sample. All these samples are shipped with the product. The MultiService sample demonstrates wiring to services over the default binding, Web services binding, and the EJB binding for both EJB 2.0 and EJB 3.0 services. The MultiService sample exposes one business service over the default binding and the Web service binding that returns a JAXB object. The client JSP, packaged in a WAR file, uses this business service exposed over the default binding.



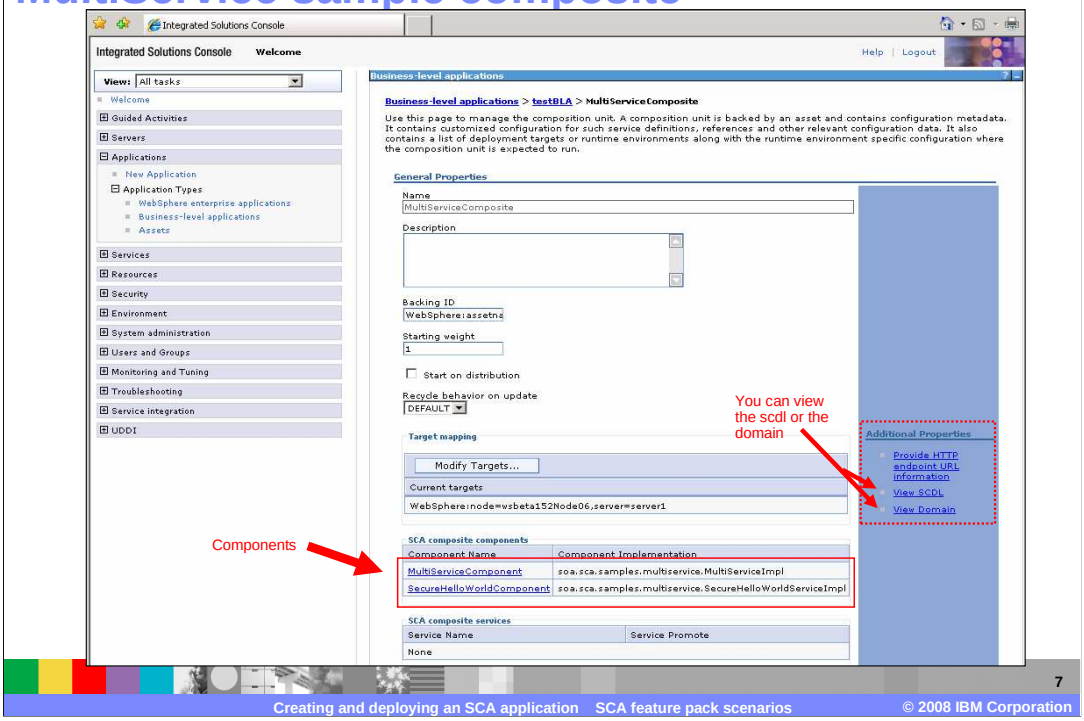
Here is a picture of the MultiService sample just to give you a glimpse of it. As you can see, it uses existing samples, namely Stock Quote, Hello World, Job Bank and Counter services.

So how is this sample created? It's created by developing and assembling an SCA Java implementation component to provide an SCA service. This is to consume Web Services through a Web services binding reference and consume SCA services through a default binding reference. It also consume a EJB 2x service through an EJB binding reference and consume an EJB 3x service through an EJB binding reference. The secure HelloWorld service is developed and assembled. After the development and assembly of these pieces, MultiService implementation is completed and a client JSP is created to test the new SCA application.

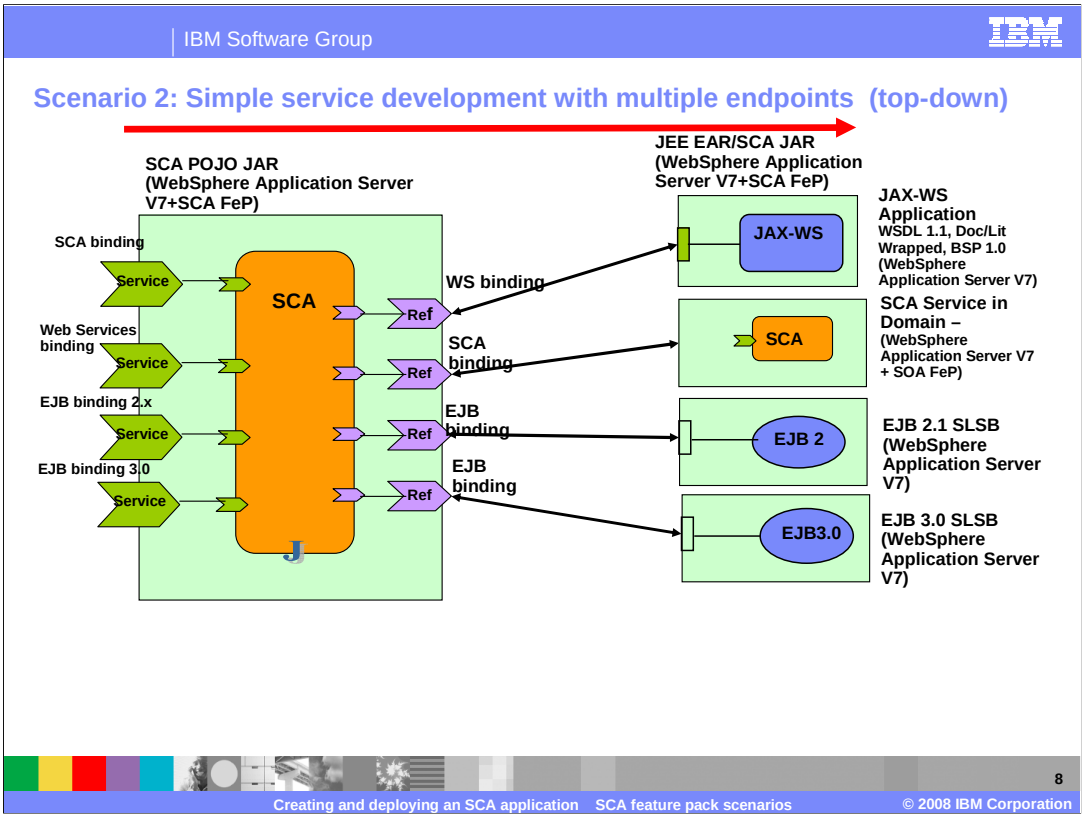
In addition, single sign-on is enabled, and the SCA Java composite with the services and references defined is deployed.

Finally, the client application is deployed and run to interact with SCA Java composite

MultiService sample composite



Once the sample is deployed, you can see the composites for this sample from the business level applications section as shown in the screen capture.



The second approach mentioned earlier was the top-down approach shown on this slide. The top-down approach is simple service development with multiple endpoints. You read this picture from **left to right**.

The service interfaces are all generated from a WSDL interface and then the implementation is generated from those generated interfaces

Features of this scenario include:

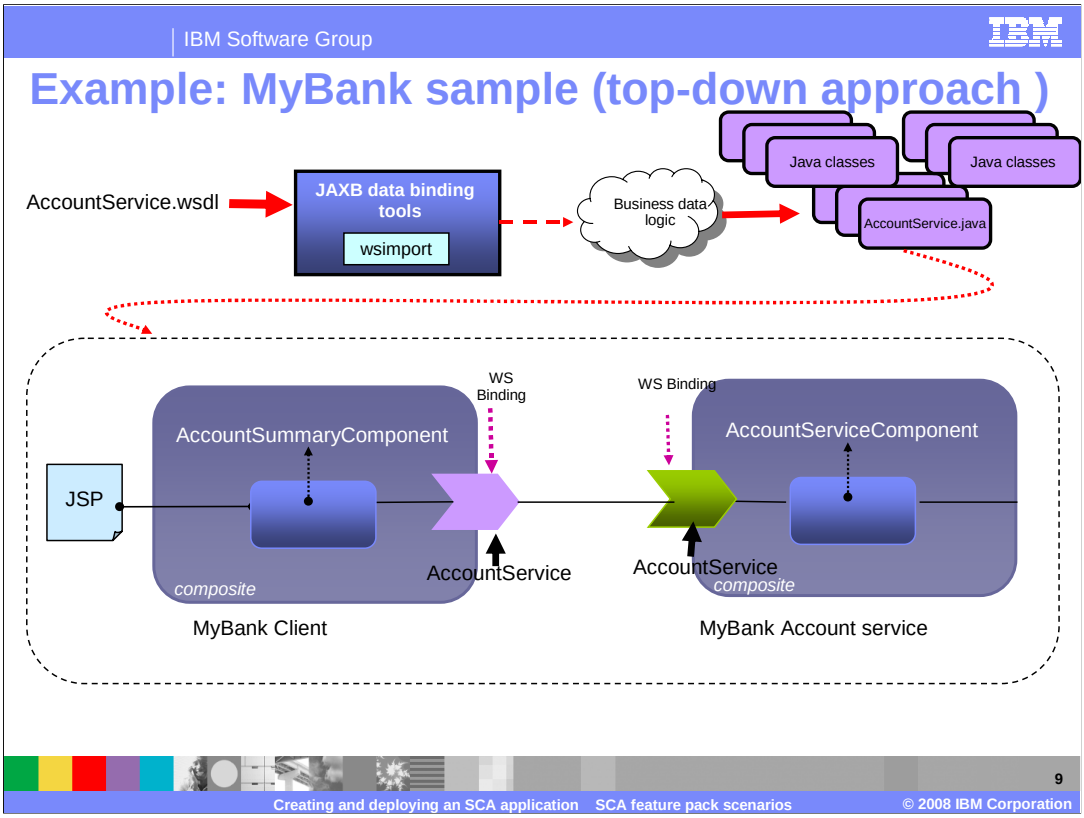
(1) Interfaces are defined through SCA design without regard to implementation details and further are captured in canonical form, for instance, WSDL and XSD schema.

(2) Wsdl2java is used to generate code used as skeleton for the implementation using JAX-B Java data bindings.

(3) Services are exposed over Web services and SCA default bindings; multiple bindings to a single service.

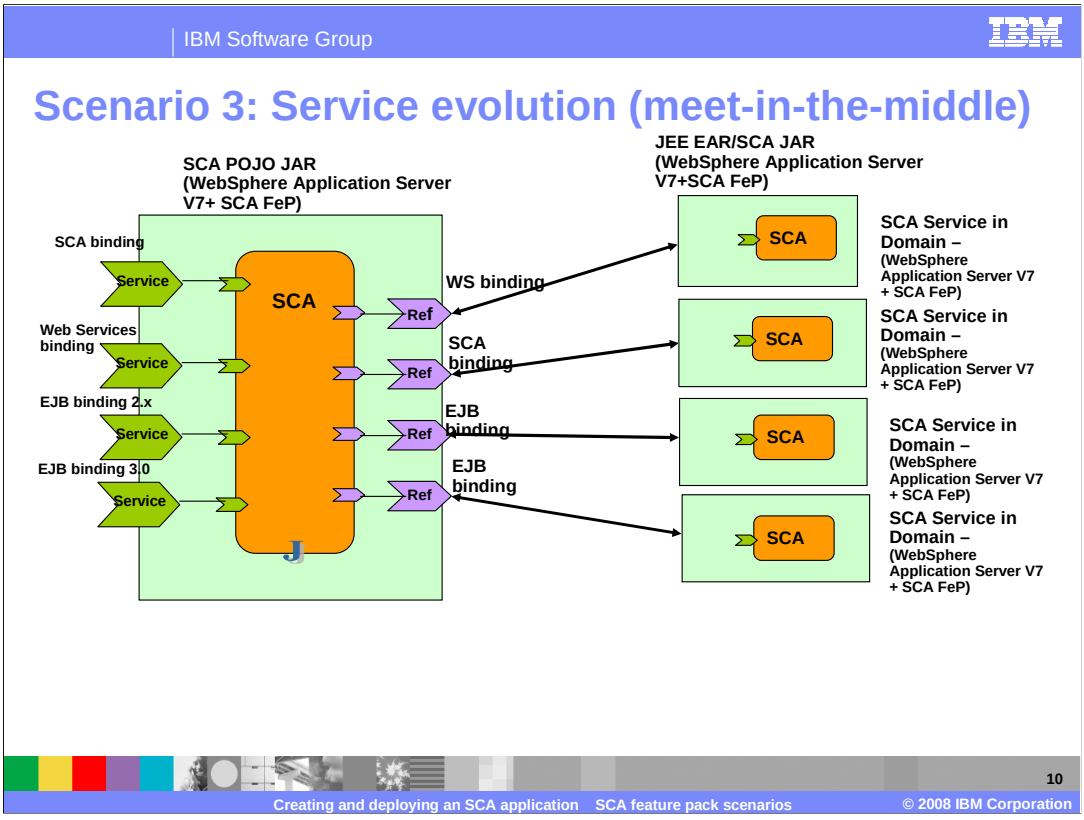
(4) Deployment is into SCA domain (ND cell) using a JAR file. There is minimal SCDL and redeployment can happen without restarting the server.

(5) The service must be secure. Authentication intent is defined on the service.



The SCA feature pack ships a top-down approach sample that is called MyBank whose base classes are created from a WSDL file, AccountService.wsdl.

This sample uses JAXB data binding and Web services binding.



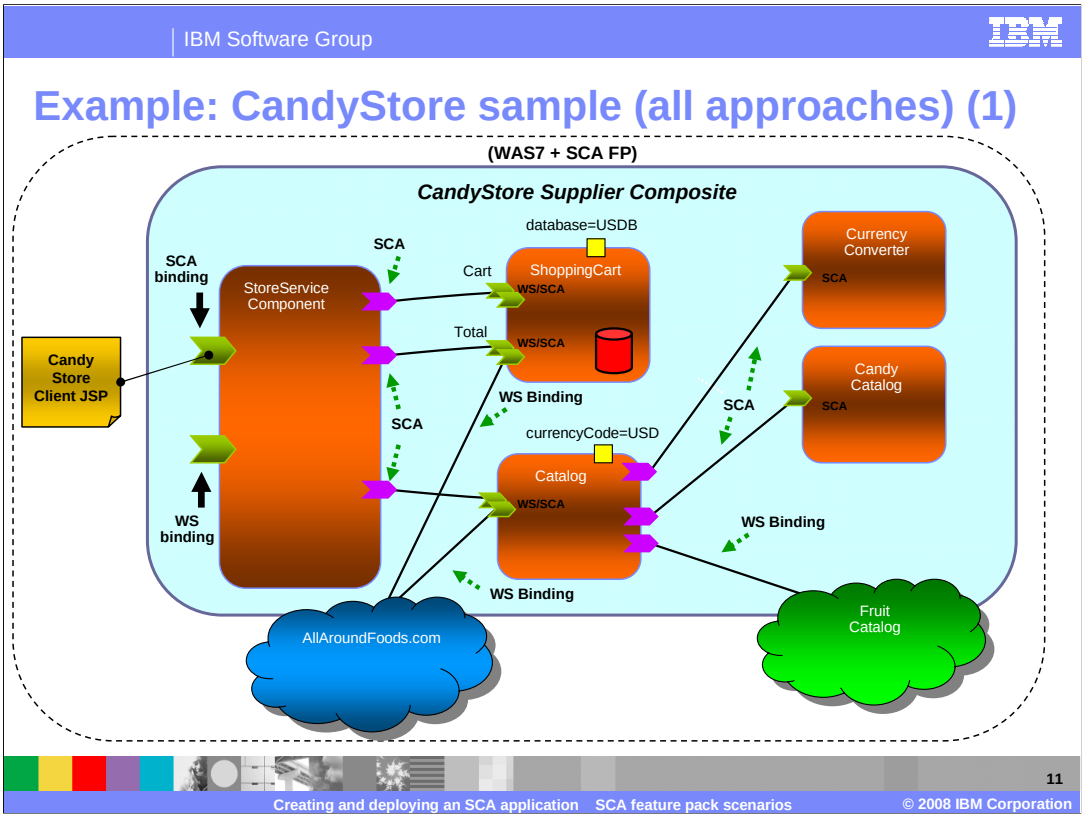
The meet-in-the-middle approach is the service evolution which is the third approach that was mentioned.

In this scenario, you start with scenario one, the bottom-up approach, where you wire existing services, then you move to the top-down approach where the service is developed before the service can evolve.

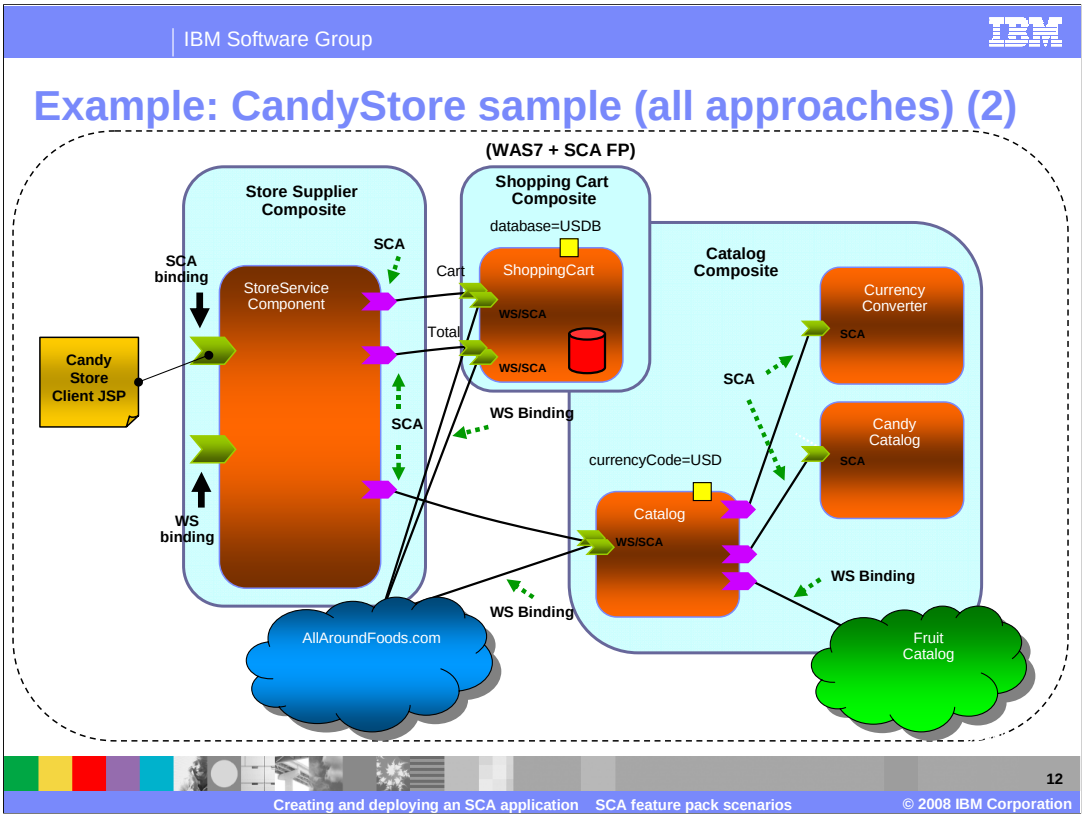
Using bottom-up, the SCA component would be moved local to a POJO. Then use the composite of the service created with the bottom-up approach as an implementation. This will promote the Web services invocation to component, and rewire to tops down approach through SCA default binding without changing the service POJO component. A redeploy is required but not a server restart.

The exposed Web service uses a virtual host (deployment-choice alternative). Then swap deployed top-down approach: SCA service implementation with an SCA Composite. Note that the Web service needs to be reliable. EJB3/JPA is invoked in global transaction demonstrating implementations which use JEE resources. Autowire for local references (implementation detail), local-wiring has to be manually turned on in composite service stubs replaced by services available in SCA domain over default bindings. Service stubs will be removed from composite during final step. Application Managed Authentication uses SCA API to get the subject from request context, and validate the subject is “authorized” from “application user registry.”

IBM currently has a sample shipped with the product called CandyStore that encompasses all these approaches.

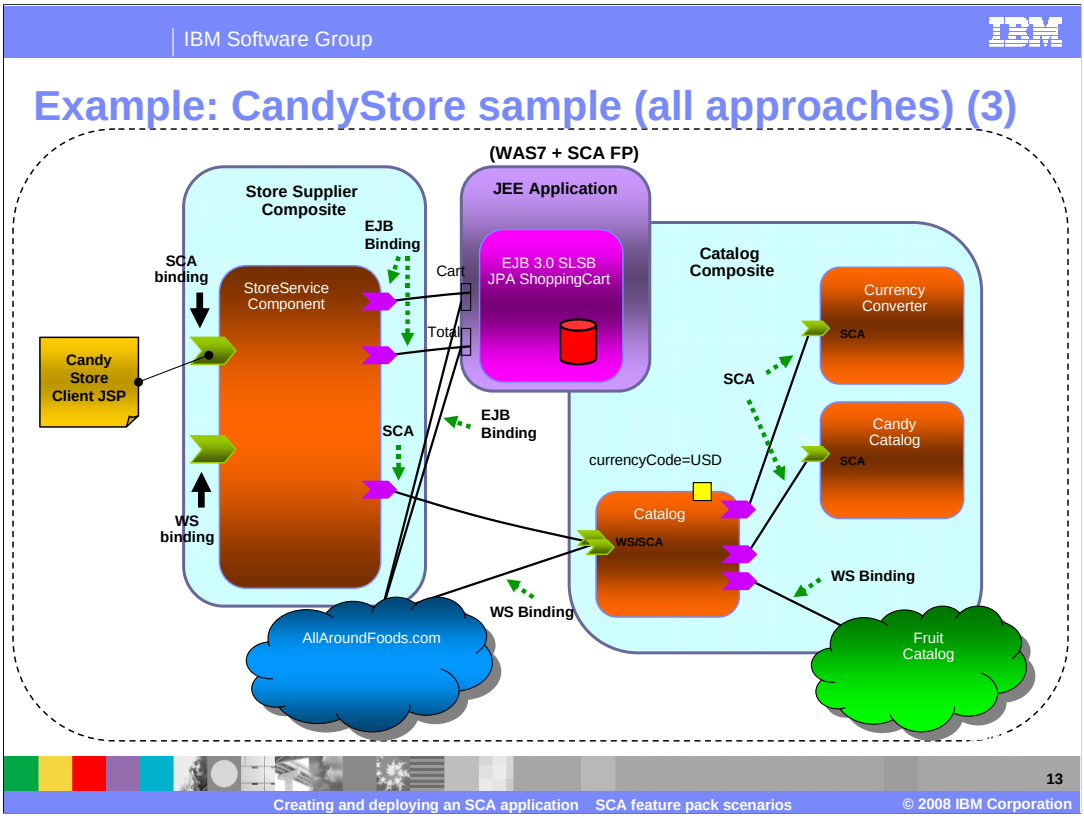


As mentioned in the previous slide, the CandyStore sample that ships with the product is a good example of an end-to-end scenario of an SCA application. It encompasses all the SCA feature pack approaches just discussed. To take a look at a few of the main composites involved in this sample, take a look at the storeSupplierComposite. The StoreSupplierComposite shows how a candy and fruit store can be merged together to form a single store and be a supplier for other online stores. In this case, the Catalog and ShoppingCart expose their services over the Web service binding for other online stores to exploit. The shopping cart in this composite is implemented using a database backend. In this sample, the client JSP is deployed in a WAR, and uses the SCA API to obtain a reference to the Store component and use the services it provides.



Here you see three composites; the storeSupplierComposite, the shoppingCart composite and the catalog compoiste.

The diagram above shows how the Store component wires to the components in the other composites. Each of the components can be deployed and run independently of each other. This allows different components to be swapped out without restarting the entire application.



This is yet another composite of the CandyStore sample. The **StoreSupplierJPATopDownComposite** shows the use of the EJB binding to wire to a shopping cart implemented using EJB 3.0 and JPA. Here you can also see how the Store component wires to the components in the other composites. The shopping cart in this scenario is implemented using a EJB 3.0 stateless session bean, and the StoreService component wires to the EJB module over the EJB binding. In this scenario you are using an EJB and the Java Persistence API (JPA) to handle any interactions with the database rather than in a direct implementation of an SCA component. Using JPA is the recommended method of database interaction. Similarly, like in the other two diagrams, in this scenario the client JSP is deployed in a WAR. It uses the SCA API to obtain a reference to the Store component and use the services it provides.

Section

Summary

In summary, this presentation looked at the SCA feature pack scenarios.

Summary

- SCA feature pack has implemented the discussed scenarios which will help you understand the features in SCA feature pack
- The shipped samples are a good start at looking at these features in preparation for creating your own SCA applications

The SCA feature pack allows you to choose from three different development approaches. A good starting point to learn about your options are the samples that are provided with the product.

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_WASv7SCA_CreatingSCApp_SCAfp_scenarios.ppt

This module is also available in PDF format at:
../WASv7SCA_CreatingSCApp_SCAfp_scenarios.pdf

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

Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

WebSphere

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of other IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

EJB, Java, JSP, and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.