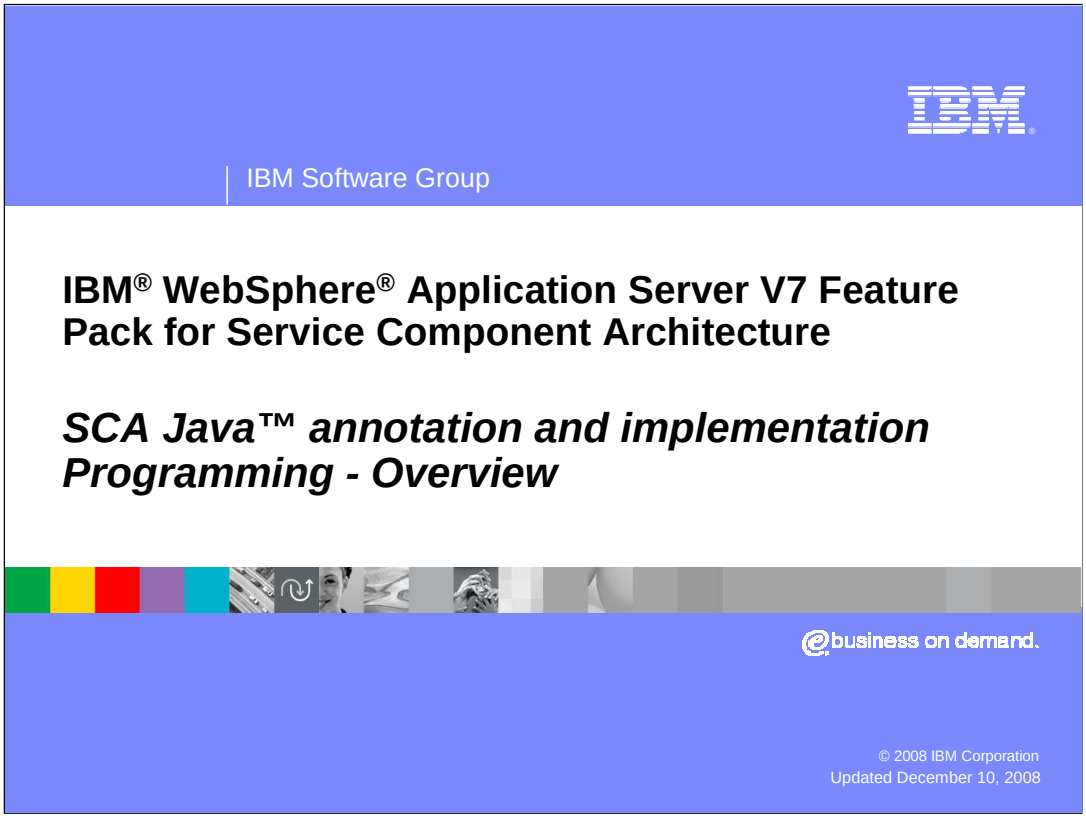
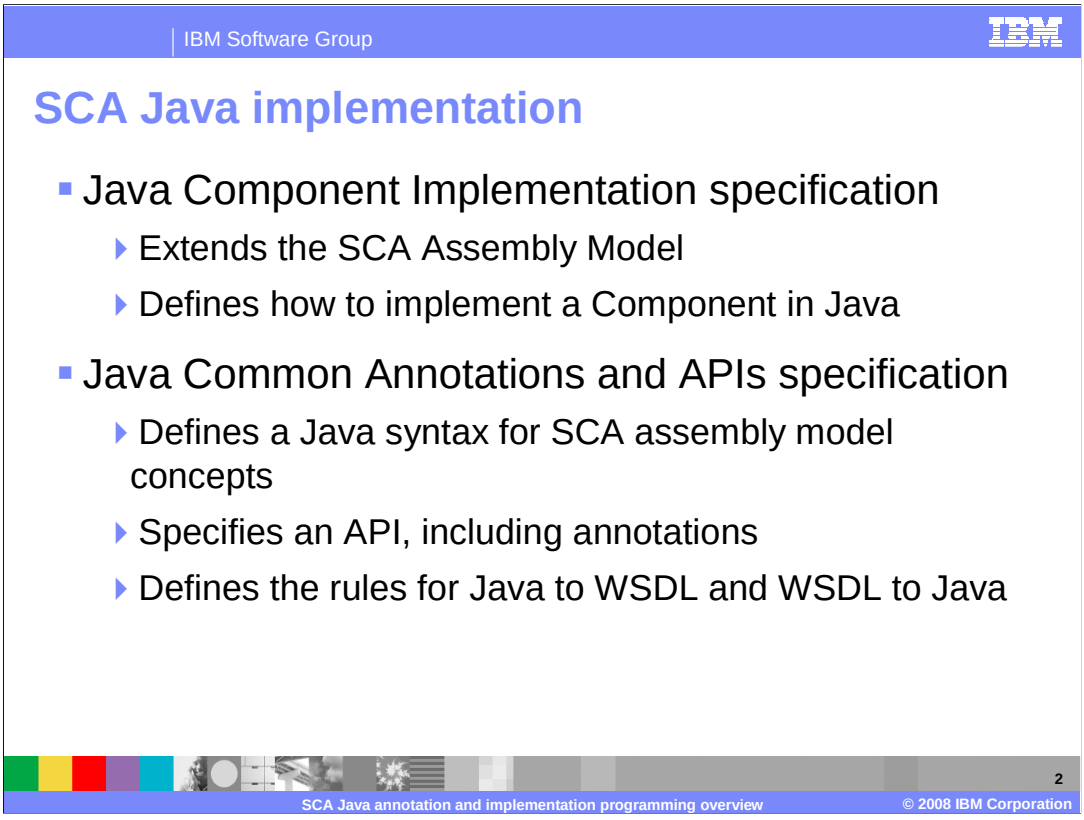


This presentation will cover the overview of the two specifications for SCA v1.0 programming model – Java annotations and Java implementation.



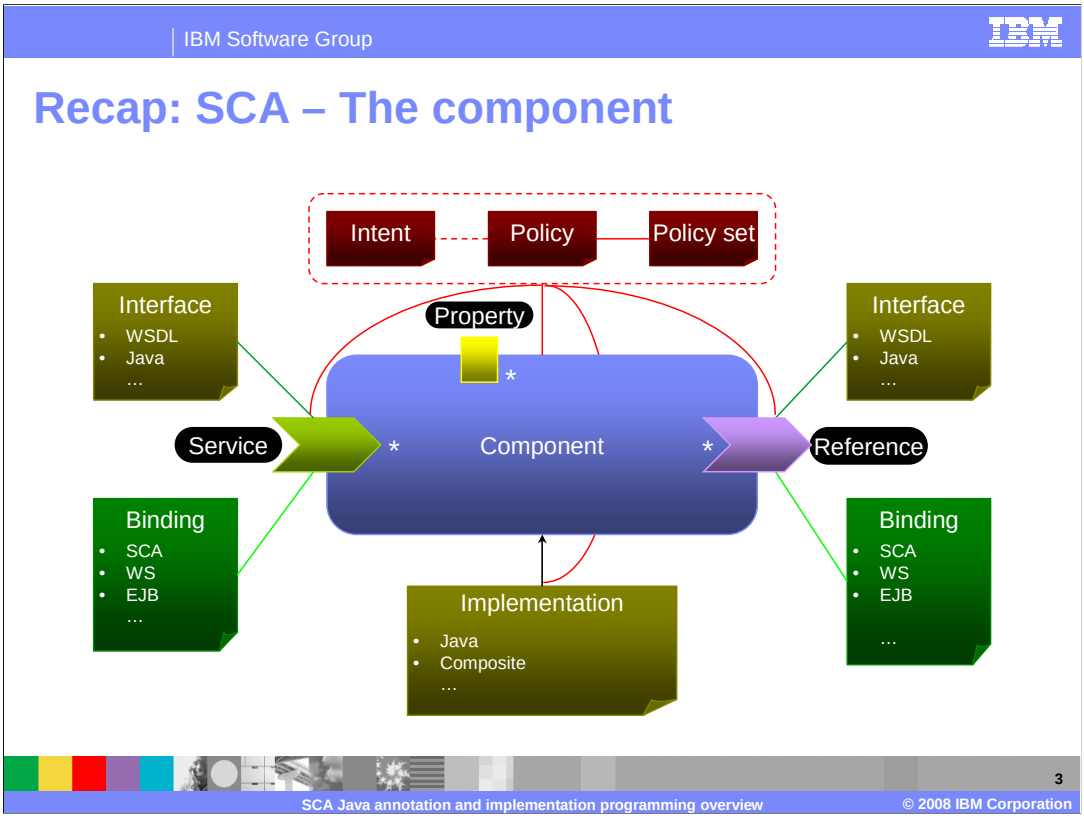
Programming model covers two specifications:

(1) Java Component Implementation specification

(2) Java Common Annotations and APIs specification

Java Component Implementation specification extends the SCA Assembly Model (the core spec) by defining how a Java class provides an implementation of an SCA component. Also be defining how that class is used in SCA as a component implementation type. This specification defines a simple Java POJO model for creating service components.

The **SCA Java Common Annotations and APIs specification** defines Java APIs and annotations that enable service components and service clients to be built in the Java programming language. There are closely related models describing how other Java-based frameworks and models can be used in the context of SCA, such as Spring and EJBs, which also use the common annotations and APIs.



Before venturing into the Java specifications where component will be mentioned several times, a recap of what the component and it's surroundings programmatically are will be helpful. As shown on the diagram, the component has very busy surroundings. Components describe the *services* they provide and the services they depend on or *reference*. Components also point at the chunk of code which provides the *implementation* of the service. Components can tailor their behavior by exposing configurable *properties*. Policy and quality of service *intents* are used to decorate services or references (called *interaction intents*), and components (called *implementation intents*) in order to configure additional semantics that are provided by the SCA runtime. Last but not least, Services and references are bound to specific protocols (such as Web services) through the usage of *bindings*.

Java implementation type

- A Java class provides an implementation of an SCA component
- Services provided by a Java-based implementation may have a:
  - Java interface
  - Java class
  - Java interface generated from a WSDL

A Java class provides an implementation of an SCA component, including its various attributes such as services, references, and properties. In addition, it details the use of metadata and the Java API in the context of a Java class used as a component implementation type

The services provided by a Java-based implementation may have an interface defined in one of these ways. Interface may be defined as a Java interface, a Java class and a Java interface generated from a Web Services Description Language (WSDL) portType.

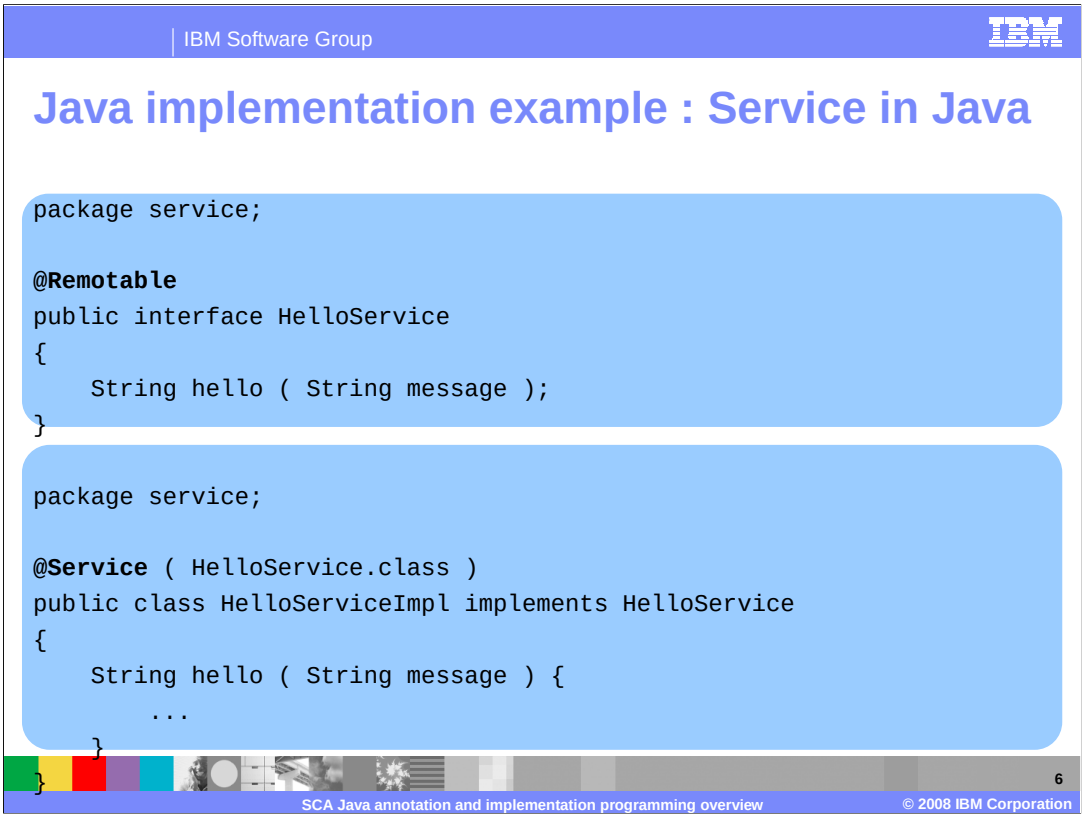
Java implementation: services

- Java implementation classes must implement all the operations defined by the service interface
- A service whose interface is defined by a Java class (as opposed to a Java interface) is not remotable
- A Java implementation type may specify the services it provides explicitly through the use of @Service

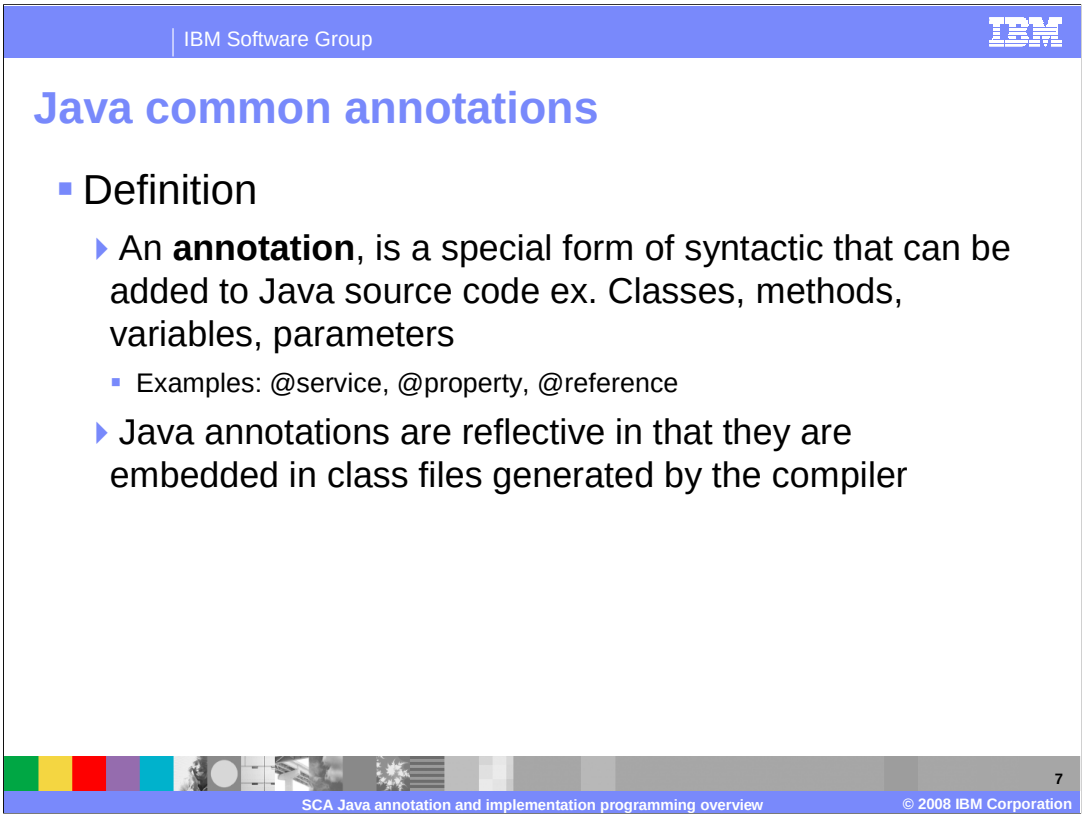
For the services, aspect, Java implementation classes must implement all the operations defined by the service interface. If the service interface is defined by a Java interface, the Java-based component can either implement that Java interface, or implement all the operations of the interface.

A service whose interface is defined by a Java class (as opposed to a Java interface) is not remotable. Java interfaces generated from WSDL portTypes are remotable

A Java implementation type may specify the services it provides explicitly through the use of @Service annotation. In certain cases, the use of @Service is not required and the services a Java implementation type offers may be inferred from the implementation class itself. @Service may be used to specify multiple services offered by an implementation.



Here is an example of a Java service interface and a Java implementation, which provides a service using that interface. Notice how the infrastructure logic is all but removed from the business logic. Java 5 annotations make service implementation, declaration and use easy.



An **annotation**, in the Java programming language is a special form of syntactic metadata that can be added to Java source code. Classes, methods, variables, parameters and packages may be annotated. Unlike Javadoc tags, Java annotations are reflective in that they are embedded in class files generated by the compiler and may be retained by the Java VM to be made retrievable at run-time.

The nice thing about annotations is that they do not interfere with your business logic.

You can write a POJO which does some business function and you would not need to worry about the middleware framework (like EJB, SCA, JAX-WS). Unlike the old programming style where annotations were not used, if say using EJB2-style, a class that extends EJBObject is loaded in a J2SE environment, there will be a NoClassDefined error

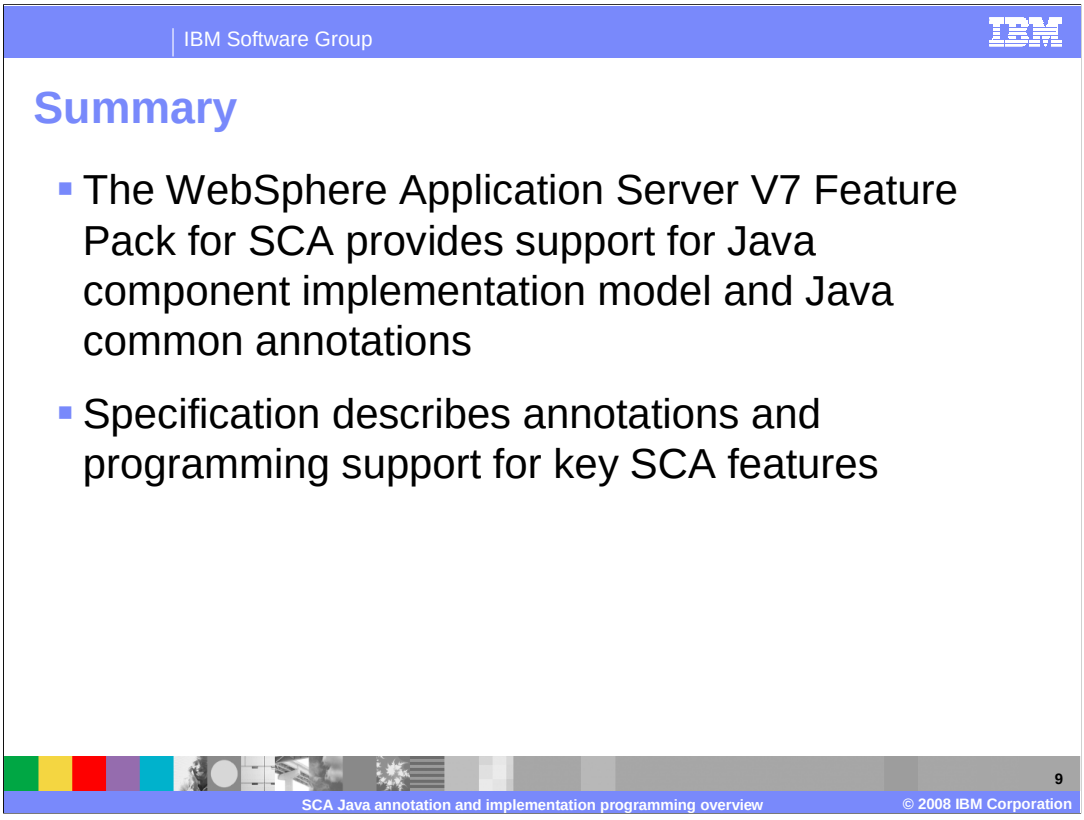
In the annotation world, that will not be an issue. That is why JEE, Java5 and others went to this model.

** Reference the spec for more examples of annotations.

** Aside note: SCA did not invent annotations. Annotations are constructs within the Java Programming Language that were added in JDK 1.5. SCA provides a set of SCA specific annotations within its programming model.

Section

Summary and references



So there are many features of Java implementation model supported in the SCA feature pack.

In summary, Unlike the older JEE technologies, SCA's Java implementation model relies on annotations rather than API calls. This approach makes creating a basic service quite easy.

Reference

- Annotations specification

http://www.osoa.org/download/attachments/35/SCA_JavaAnnotationsAndAPIs_V100.pdf?version=1

- Component implementation specification

http://www.osoa.org/download/attachments/35/SCA_JavaComponentImplementation_V100.pdf?version=1

Here is a reference to the specs.

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_JavaProgrammingModel_overview.ppt

This module is also available in PDF format at:
../WASv7SCA_JavaProgrammingModel_overview.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, J2SE, Java, Javadoc, JDK, 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.