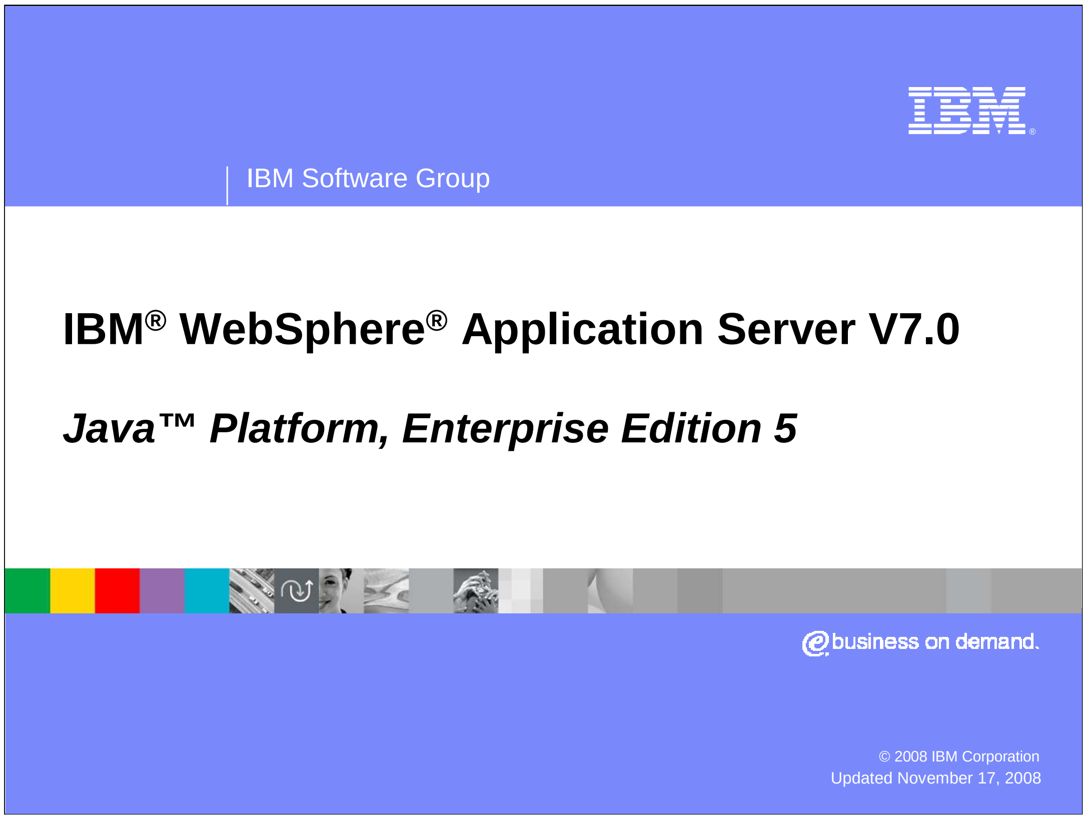


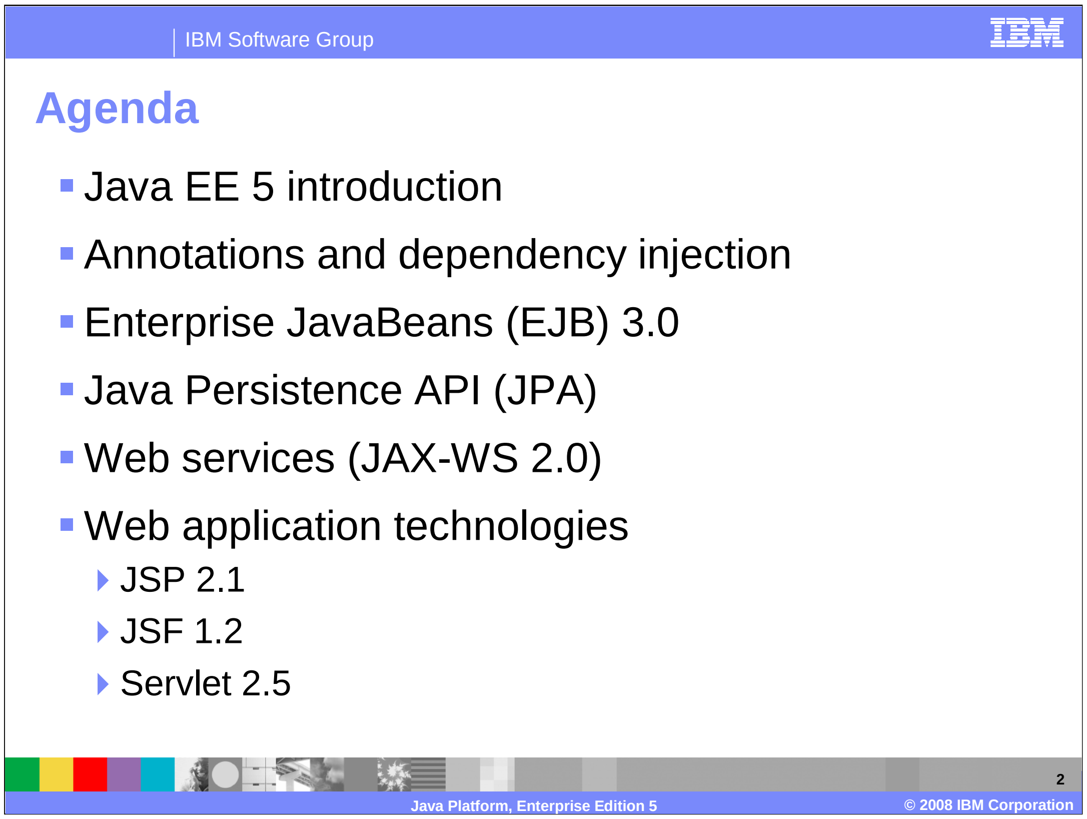
IBM Software Group

IBM® WebSphere® Application Server V7.0

Java™ Platform, Enterprise Edition 5

business on demand.

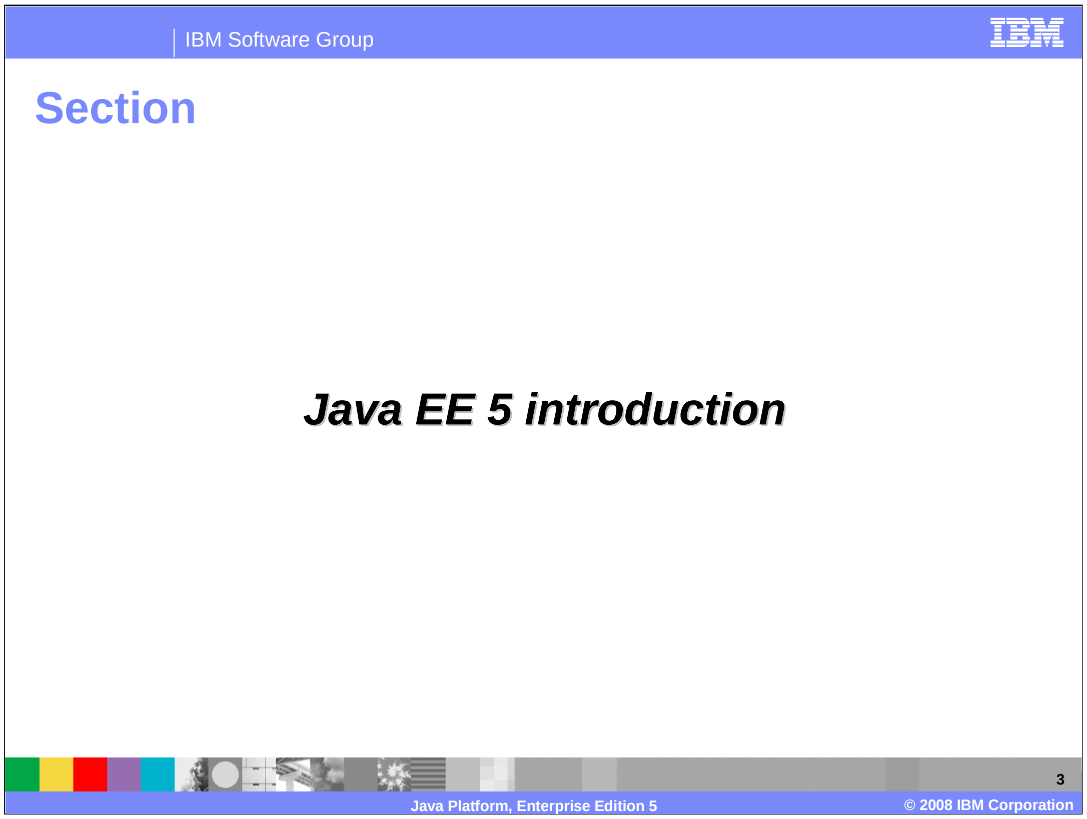
This presentation will introduce you to Java EE 5.



Agenda

- Java EE 5 introduction
- Annotations and dependency injection
- Enterprise JavaBeans (EJB) 3.0
- Java Persistence API (JPA)
- Web services (JAX-WS 2.0)
- Web application technologies
  - JSP 2.1
  - JSF 1.2
  - Servlet 2.5

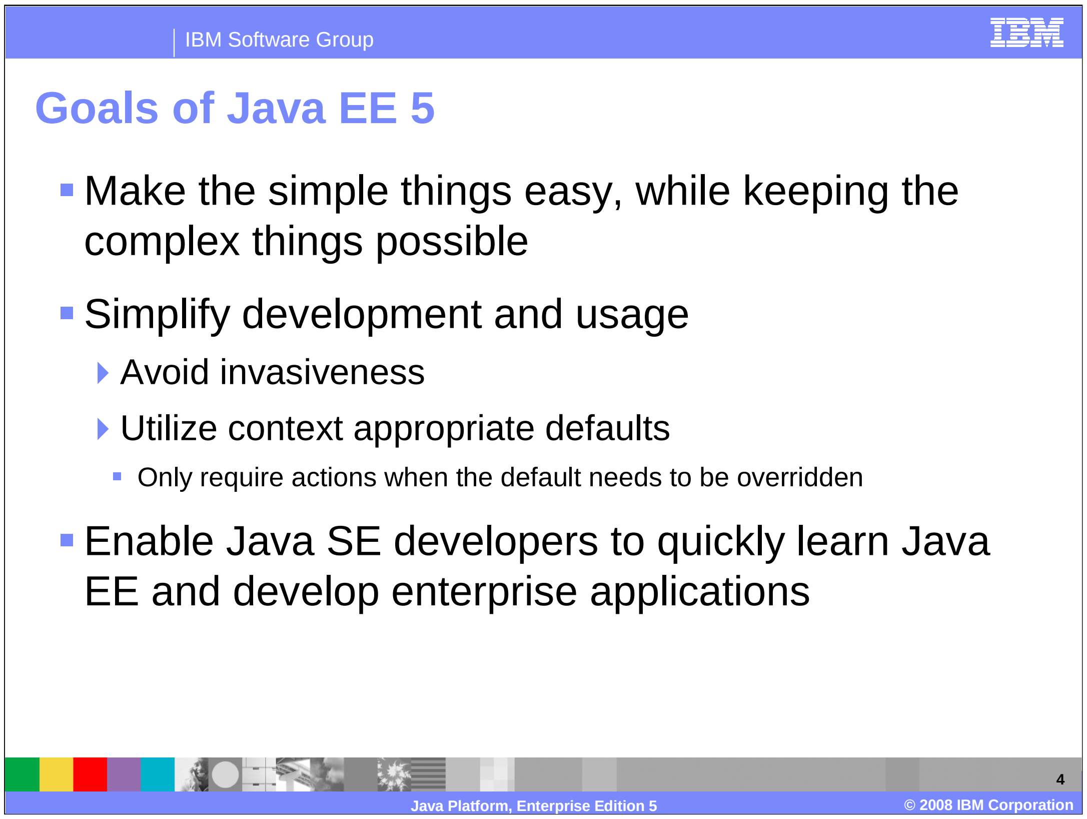
This presentation will introduce Java EE 5 by discussing several of the major enhancements to the Java EE specification. New language features, such as annotations and dependency injection will be covered first, followed by a look at some of the major specification upgrades, including EJB 3.0, JPA, Web services enhancements, and updated Web technologies.



Section

Java EE 5 introduction

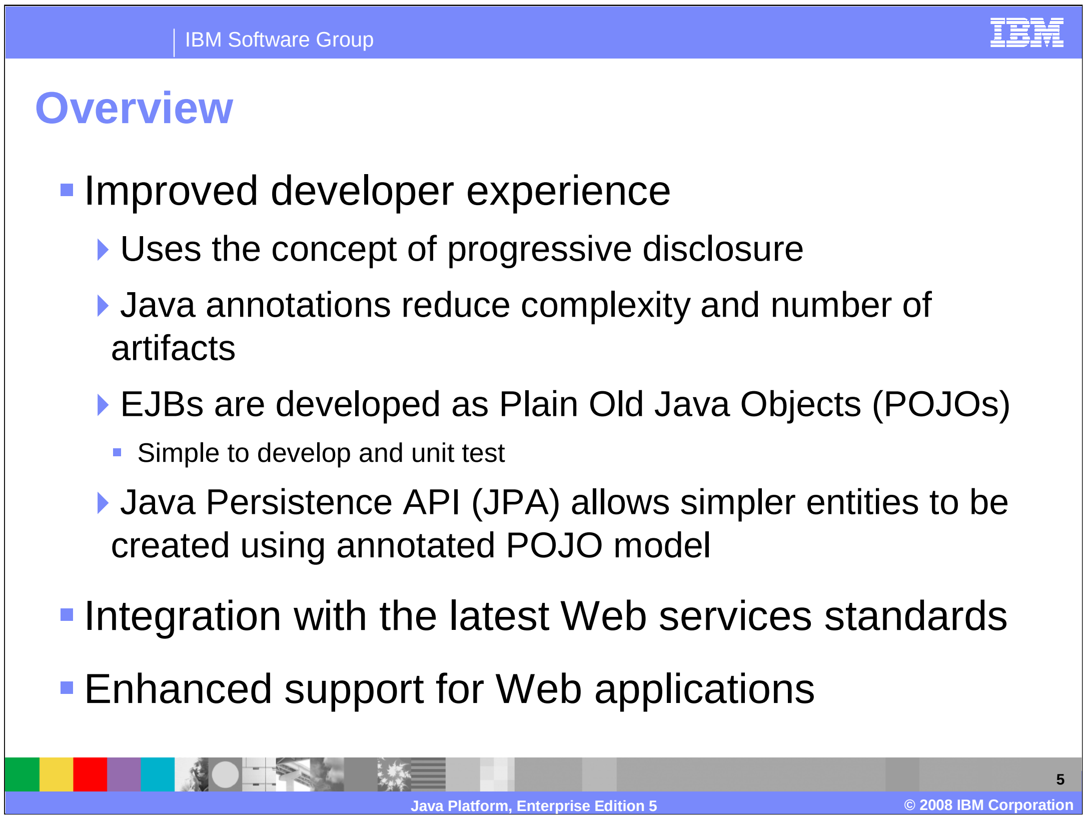
This section will discuss the main themes of Java EE 5 and new language features.



Goals of Java EE 5

- Make the simple things easy, while keeping the complex things possible
- Simplify development and usage
  - Avoid invasiveness
  - Utilize context appropriate defaults
    - Only require actions when the default needs to be overridden
- Enable Java SE developers to quickly learn Java EE and develop enterprise applications

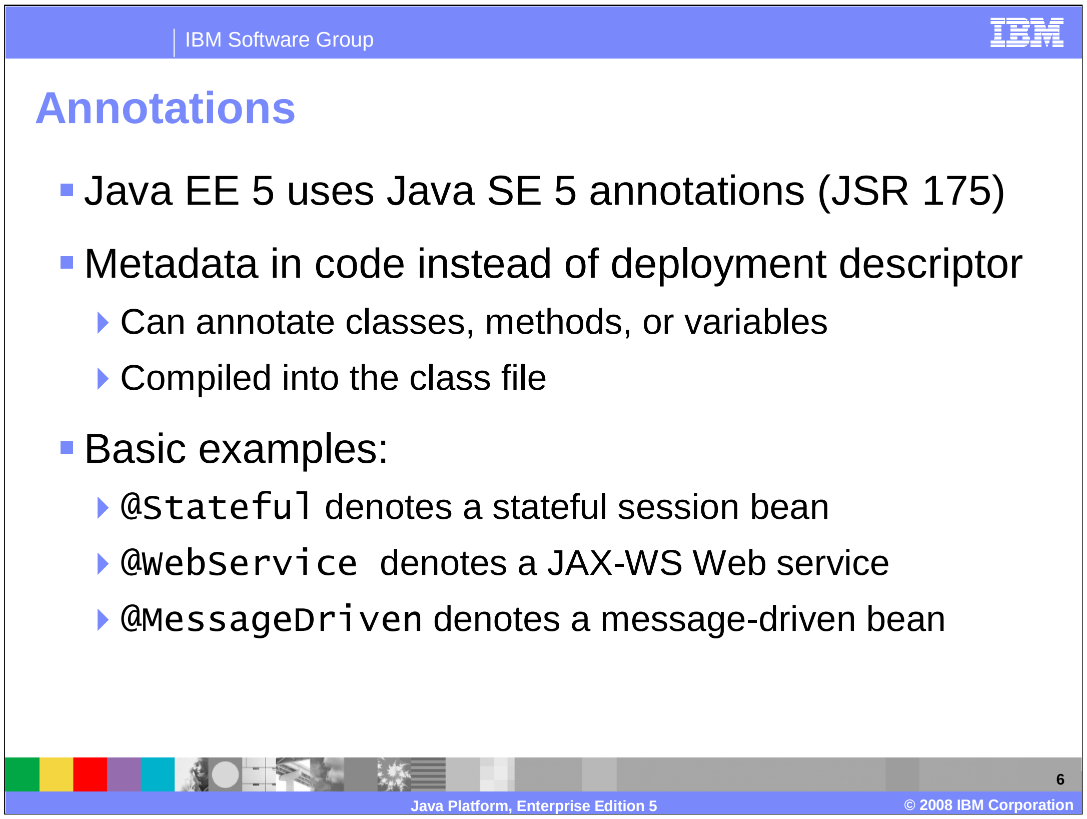
The main goal of Java EE 5 is to simplify the programming model. The new specification aims to make it as easy as possible to implement simple things, while keeping complex things possible. In many cases this is accomplished by utilizing contextually appropriate default values, allowing you to override the defaults when needed. When it is possible for the container to figure something out, it will do so, rather than requiring a developer to provide unnecessary information. With this simplification, regular Java developers should be more able to make the transition to developing enterprise Java applications.



Overview

- Improved developer experience
  - Uses the concept of progressive disclosure
  - Java annotations reduce complexity and number of artifacts
  - EJBs are developed as Plain Old Java Objects (POJOs)
    - Simple to develop and unit test
  - Java Persistence API (JPA) allows simpler entities to be created using annotated POJO model
- Integration with the latest Web services standards
- Enhanced support for Web applications

Many changes have been made in support of programming model simplification. The concept of “progressive disclosure” is employed heavily by Java EE 5, which means that default values have been chosen to account for the most common scenarios, and developers will only need to specify information or implement particular methods if they want an application to behave differently than the default. Again, this ensures that simple applications are far easier to develop than in previous releases, while retaining the capability to create more complex applications. The EJB specification has been heavily revised, and is now a plain-old Java object based programming model. This approach extends even to data persistence, with the introduction of JPA, the Java Persistence API. Java EE 5 also incorporates the latest Java Web services standards, such as JAX-WS, the Java API for XML Web services. Web application technologies, such as Servlets, JavaServer Pages, and JavaServer Faces have also been revised in this specification.


IBM Software Group

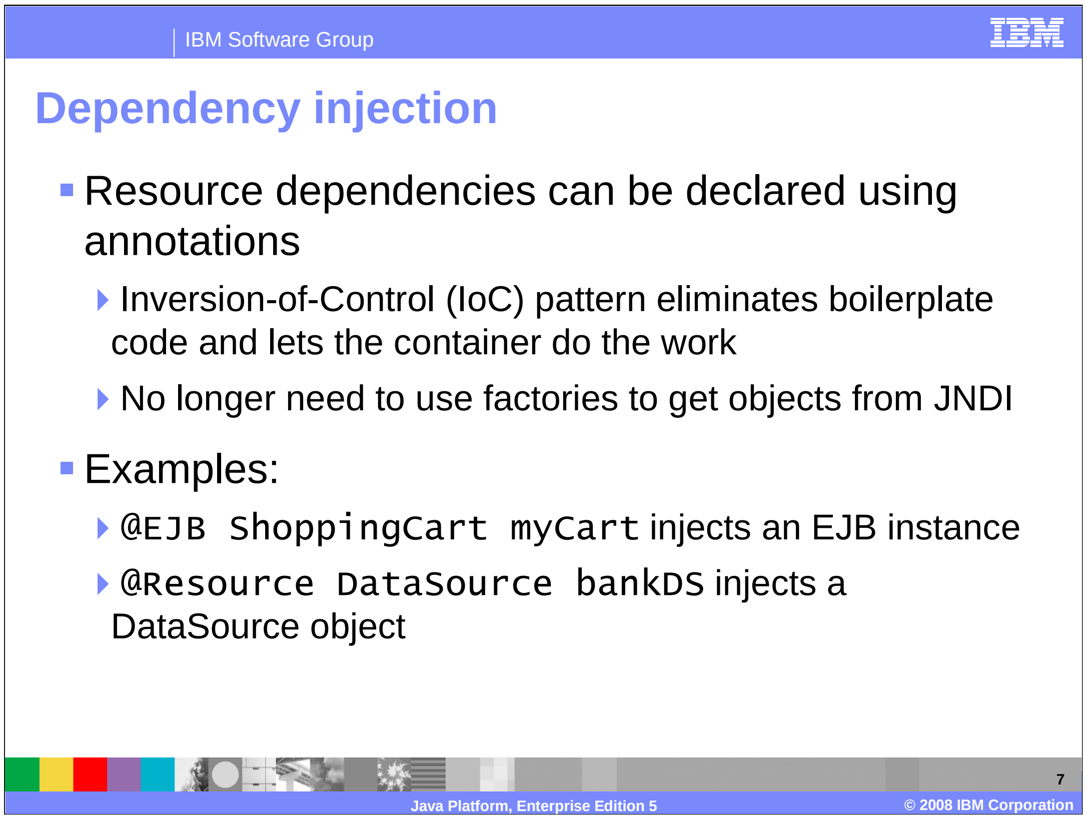
Annotations

- Java EE 5 uses Java SE 5 annotations (JSR 175)
- Metadata in code instead of deployment descriptor
  - Can annotate classes, methods, or variables
  - Compiled into the class file
- Basic examples:
  - `@Stateful` denotes a stateful session bean
  - `@WebService` denotes a JAX-WS Web service
  - `@MessageDriven` denotes a message-driven bean

6

Java Platform, Enterprise Edition 5 © 2008 IBM Corporation

Java annotations were introduced in Java SE 5, under JSR-175, and are widely used in EJB 3.0. Annotations are used to specify metadata directly in code, rather than in an XML deployment descriptor, and are compiled directly into your class file. For example, adding "@Stateful" before the name of a class defines it as a stateful session bean, and that no longer needs to be specified in a deployment descriptor. Similar annotations exist for Stateless Session Beans, Message-Driven beans, and JPA Entities. Annotations can be used to provide more complex information as well, as you can see in the presentation titled "EJB 3.0 code examples".


IBM Software Group

Dependency injection

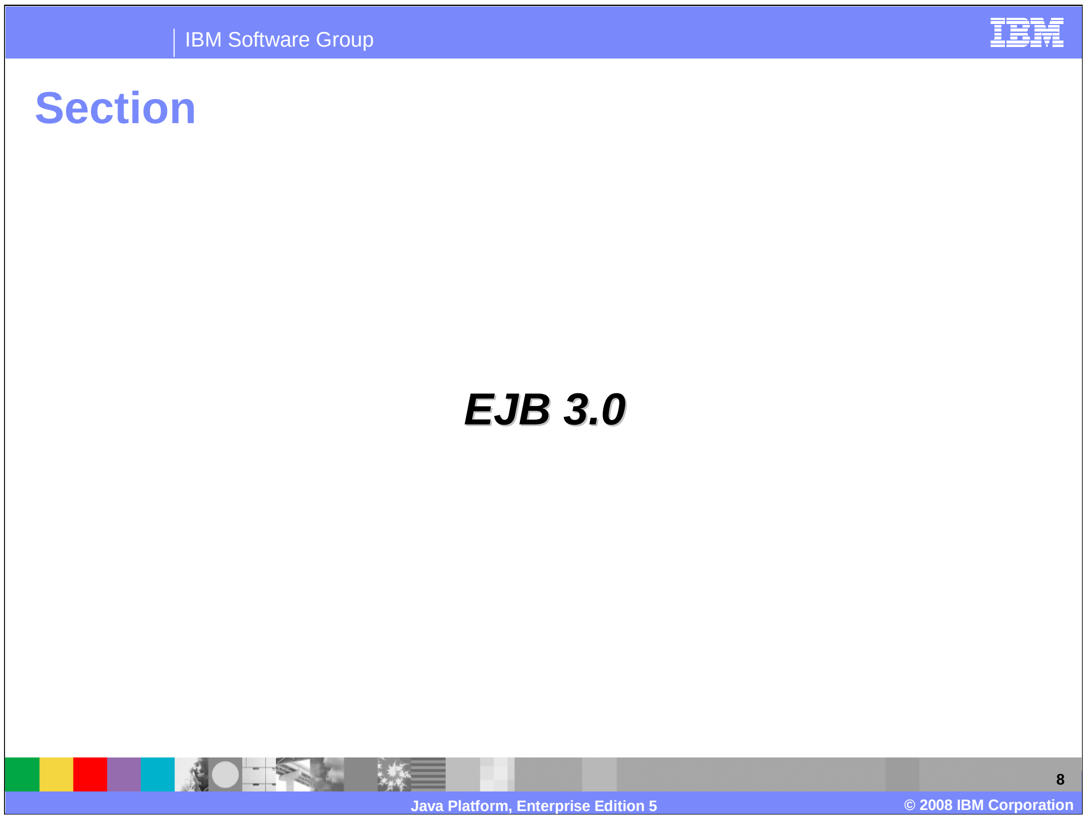
- Resource dependencies can be declared using annotations
  - Inversion-of-Control (IoC) pattern eliminates boilerplate code and lets the container do the work
  - No longer need to use factories to get objects from JNDI
- Examples:
  - `@EJB ShoppingCart myCart` injects an EJB instance
  - `@Resource DataSource bankDS` injects a DataSource object

7

Java Platform, Enterprise Edition 5

© 2008 IBM Corporation

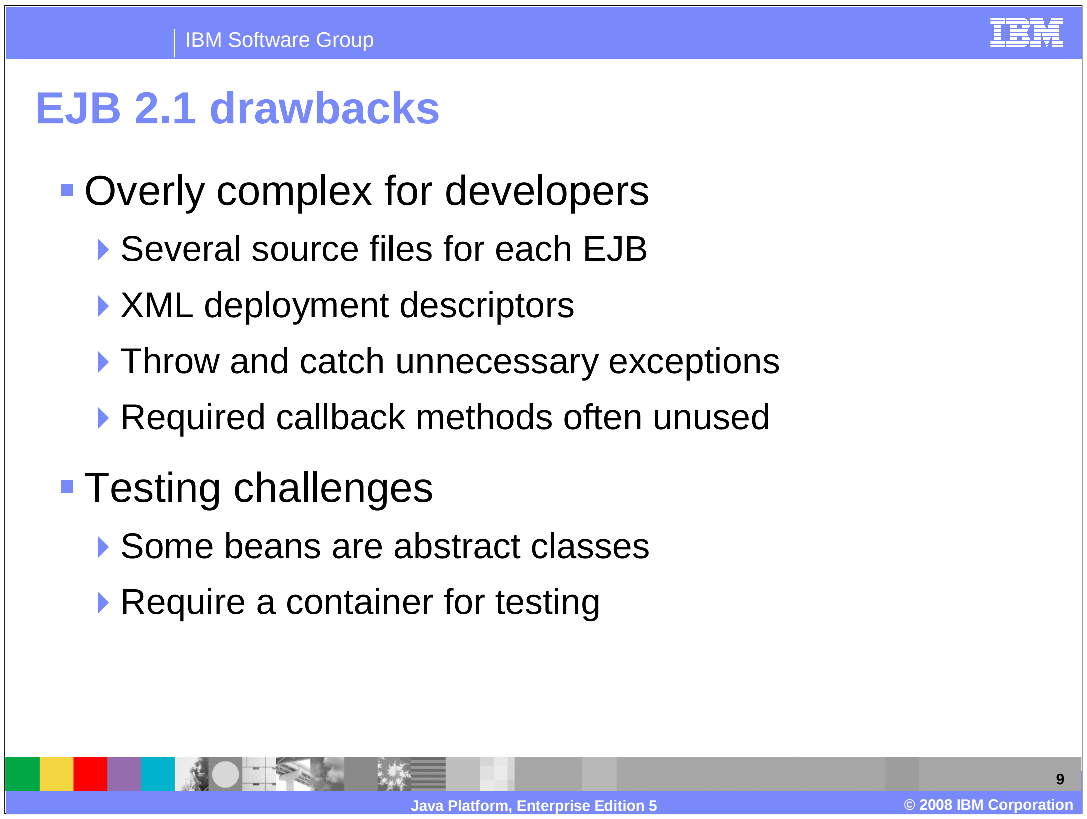
EJB 3.0 also introduces annotations for injecting resource dependencies using the "Inversion of control" pattern. Rather than implement boilerplate code, and use a factory to get an object, look it up in JNDI, and then cast or narrow it to make it useable, you now need only to use the @Resource annotation to inject a dependency. The examples shown on this slide illustrate how to inject an instance of an EJB using the @EJB annotation, and a data source using the @Resource annotation.



Section

EJB 3.0

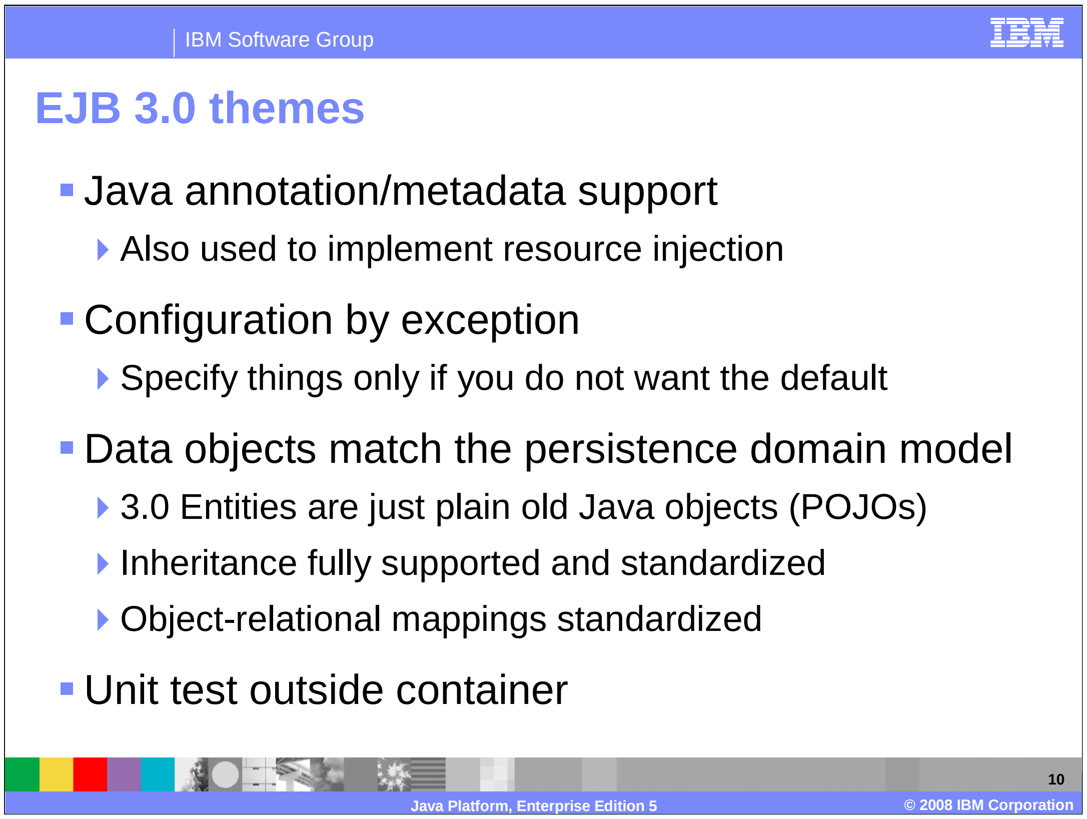
This section will introduce the EJB 3.0 specification.



EJB 2.1 drawbacks

- Overly complex for developers
  - Several source files for each EJB
  - XML deployment descriptors
  - Throw and catch unnecessary exceptions
  - Required callback methods often unused
- Testing challenges
  - Some beans are abstract classes
  - Require a container for testing

The EJB specification has often been criticized for being overly complex. For example, new developers are often confused about why it is necessary to create several different source files for each EJB. Similarly, you are required to implement several callback methods even if you do not use them, and handle exceptions that may be unnecessary. Deployment descriptors can also be difficult to understand, and can be a bottleneck in a team development environment, since only one person can be updating the deployment descriptor at a given time. EJB 2.1 applications also cannot be tested outside of a container, since some beans are abstract classes that are implemented by the container at runtime.


IBM Software Group

EJB 3.0 themes

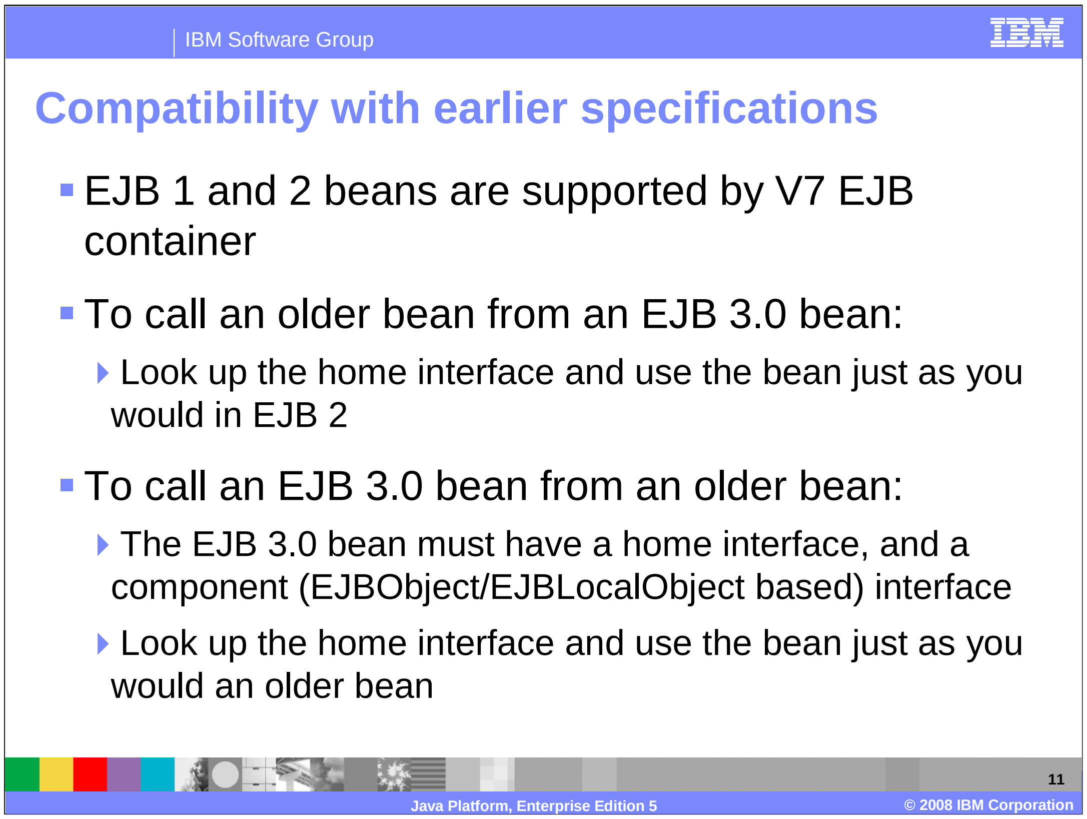
- Java annotation/metadata support
  - Also used to implement resource injection
- Configuration by exception
  - Specify things only if you do not want the default
- Data objects match the persistence domain model
  - 3.0 Entities are just plain old Java objects (POJOs)
  - Inheritance fully supported and standardized
  - Object-relational mappings standardized
- Unit test outside container

10

Java Platform, Enterprise Edition 5

© 2008 IBM Corporation

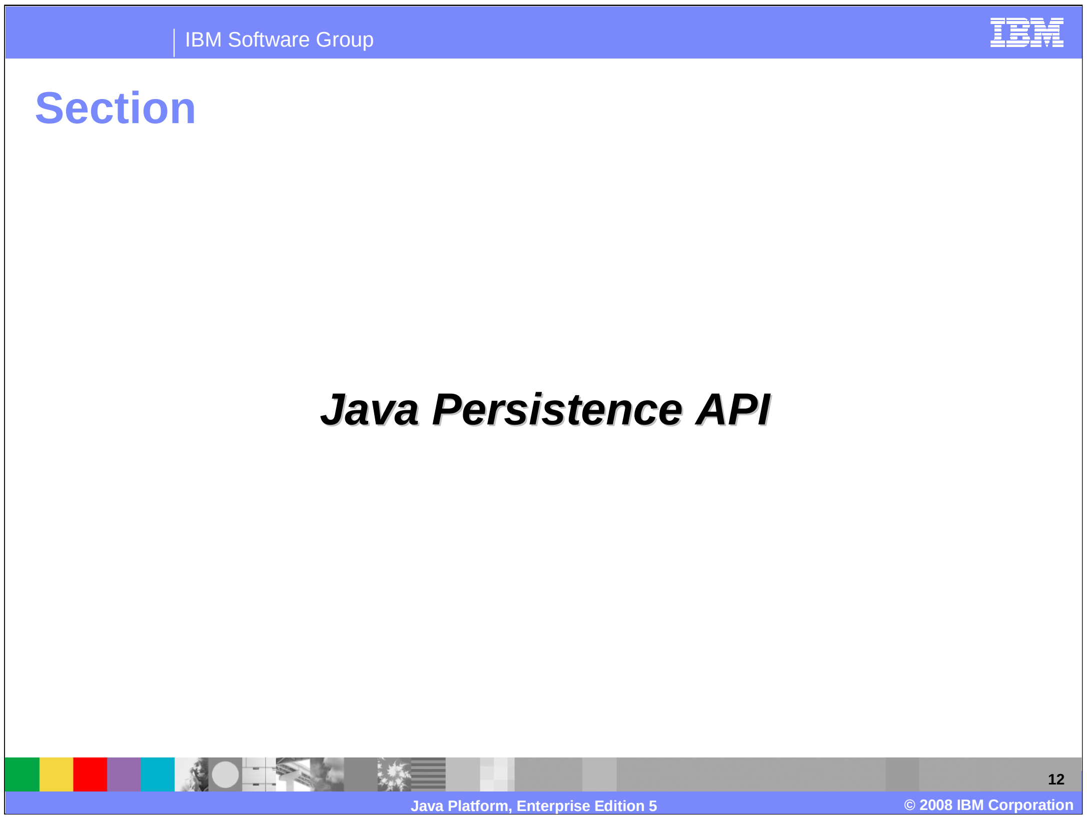
Like Java EE 5 in general, the EJB 3.0 specification aims to simplify enterprise application development. EJB 3.0 simplifies the EJB development model by eliminating a lot of the less-desirable requirements of EJB 2.1. Java annotations are used to provide metadata directly in code, rather than requiring an XML deployment descriptor, and to support resource injection, as was described earlier. The specification also defines intelligent default behaviors, so that you only need to specify information or implement functions when you want them to behave differently than the default. This greatly simplifies the process of developing simple EJBs. Object persistence is also greatly simplified in EJB 3.0, since the Java Persistence API defines a new standard framework for persistence and object-relational mapping that is based on plain-old Java objects rather than Entity Beans. In addition, unit testing is much easier, because the specification no longer requires abstract classes that depend on an application server to implement them.



Compatibility with earlier specifications

- EJB 1 and 2 beans are supported by V7 EJB container
- To call an older bean from an EJB 3.0 bean:
  - Look up the home interface and use the bean just as you would in EJB 2
- To call an EJB 3.0 bean from an older bean:
  - The EJB 3.0 bean must have a home interface, and a component (EJBObject/EJBLocalObject based) interface
  - Look up the home interface and use the bean just as you would an older bean

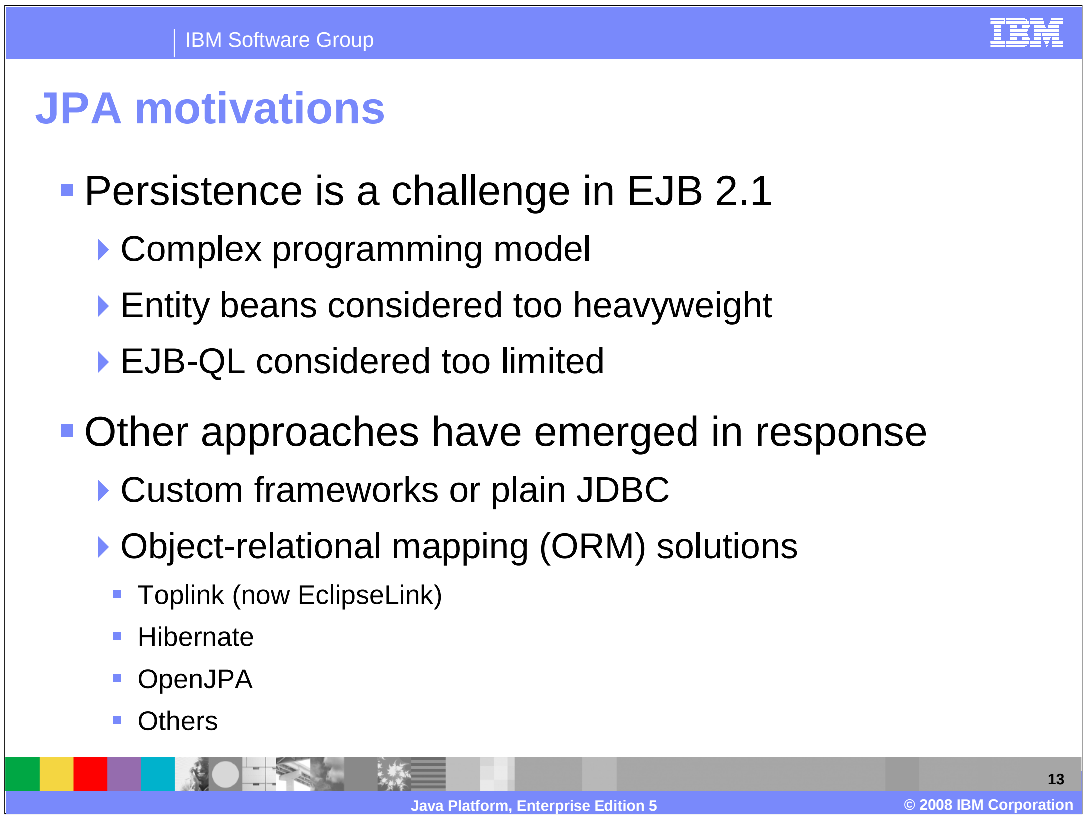
In addition to supporting EJB 3.0, WebSphere Application Server V7 is compatible with EJB 1 and 2 beans. Older EJBs can run independently, or they can interoperate with new EJB 3.0 artifacts. If you want to call an older EJB from an EJB 3.0 bean, you only need to look up the bean's home interface, and use it exactly as you would in an EJB 2 bean. To call an EJB 3.0 bean from an older EJB, you need to create a home interface and a component interface for the EJB 3.0 bean, since they are expected by the older bean. Once you have created those interfaces, you can look up the home interface and call it exactly as you would call an older bean.



Section

Java Persistence API

This section will introduce the Java Persistence API, known as JPA.



IBM Software Group

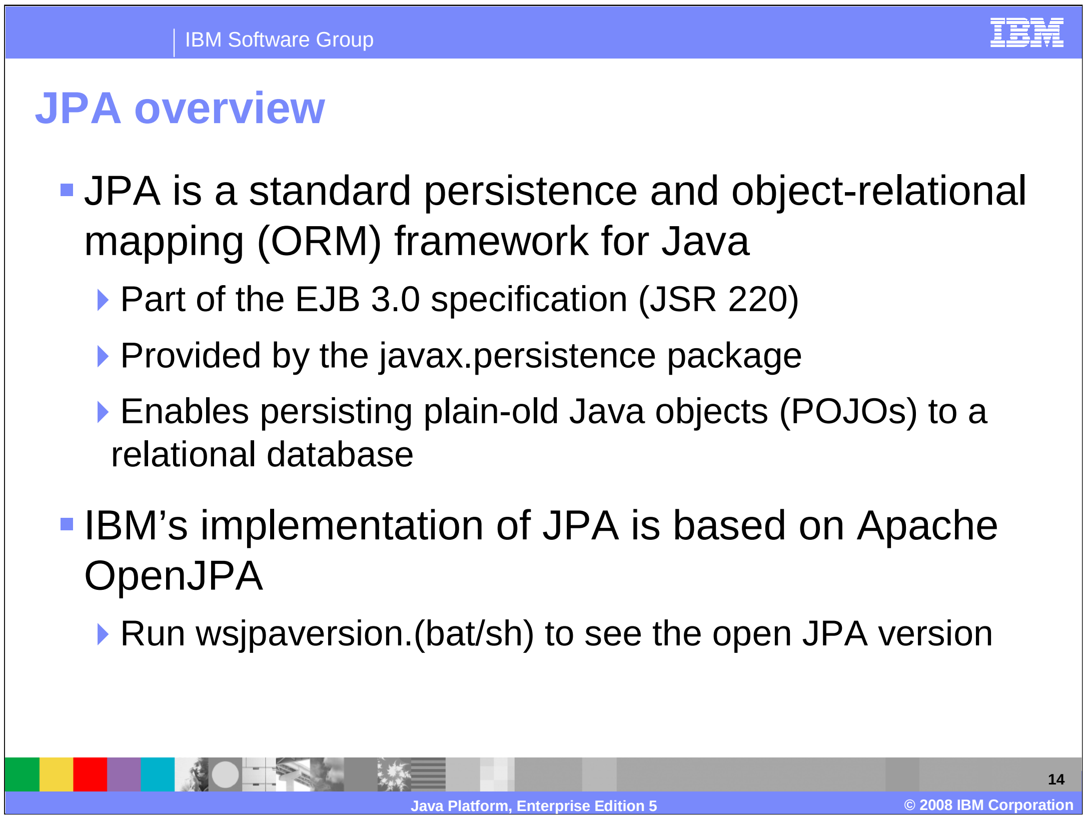
JPA motivations

- Persistence is a challenge in EJB 2.1
  - Complex programming model
  - Entity beans considered too heavyweight
  - EJB-QL considered too limited
- Other approaches have emerged in response
  - Custom frameworks or plain JDBC
  - Object-relational mapping (ORM) solutions
    - Toplink (now EclipseLink)
    - Hibernate
    - OpenJPA
    - Others

13

Java Platform, Enterprise Edition 5 © 2008 IBM Corporation

Persisting data with EJBs has come to be a sore spot for many developers. Entity beans can be complicated and time consuming to write, even with the support of vendor-supplied tools. The heavyweight programming model can be less than ideal for many applications, particularly smaller scale applications. Many Java developers use custom persistence frameworks or plain JDBC to better meet their needs. As a result, several alternatives have emerged, offering lightweight and straightforward solutions for persisting Java objects to a relational database. EclipseLink and Hibernate are two popular solutions. The EJB 3.0 specification introduced JPA in response to these challenges, offering a lightweight, Plain-old Java object (POJO) based persistence framework.


IBM Software Group

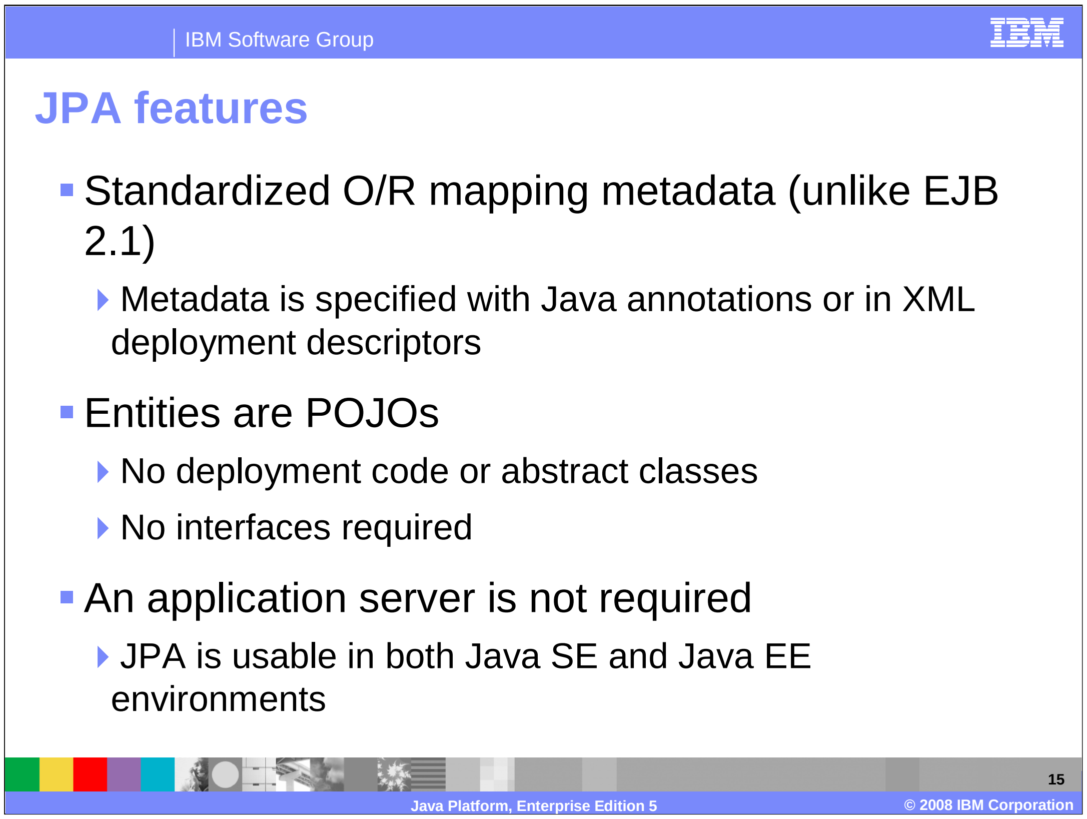
JPA overview

- JPA is a standard persistence and object-relational mapping (ORM) framework for Java
  - Part of the EJB 3.0 specification (JSR 220)
  - Provided by the javax.persistence package
  - Enables persisting plain-old Java objects (POJOs) to a relational database
- IBM's implementation of JPA is based on Apache OpenJPA
  - Run wsjpaversion.(bat/sh) to see the open JPA version

14

Java Platform, Enterprise Edition 5 © 2008 IBM Corporation

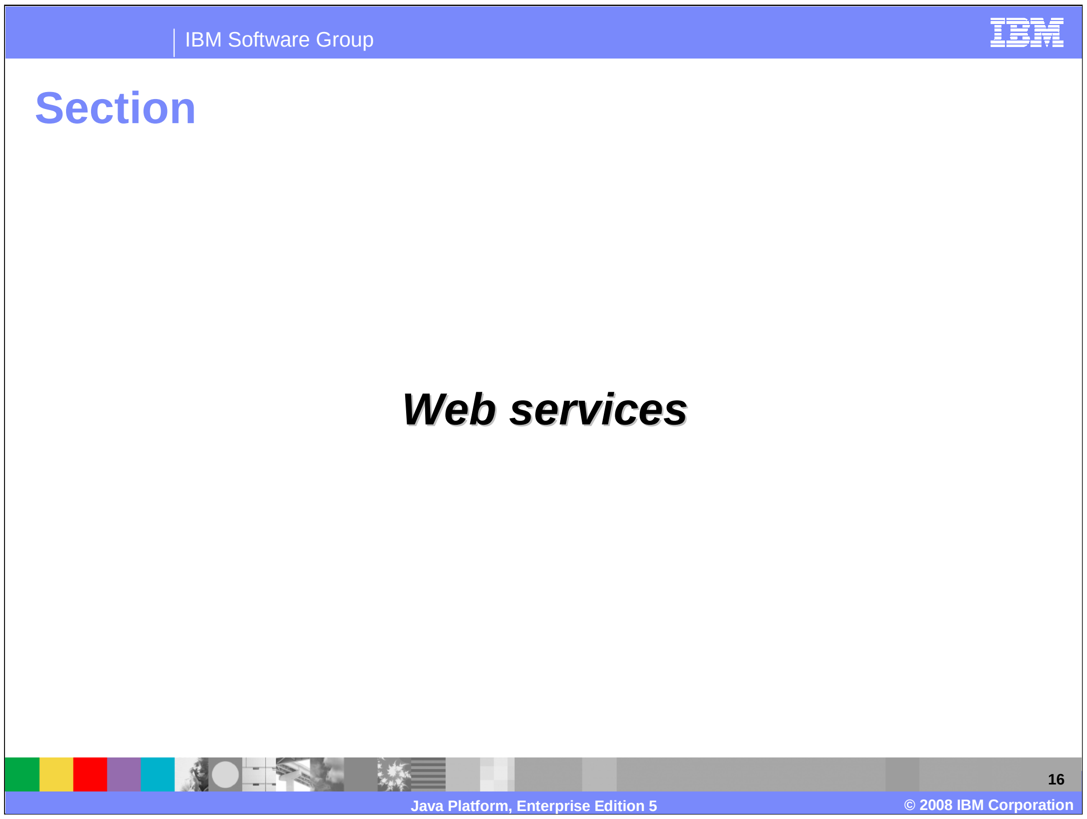
The Java Persistence API is a standard framework that provides persistence and object-relational mapping as a part of the EJB 3.0 specification. It is a Java standard that provides many of the benefits of alternative persistence frameworks, with the added benefit of portability to any EJB 3.0 compliant container, without having to install any extra libraries. It allows you to persist plain-old Java objects to a relational database – a much simpler approach than using container-managed persistence (CMP) in EJB 2.1. The JPA implementation in WebSphere Application Server is based on the open source Apache OpenJPA project.



JPA features

- Standardized O/R mapping metadata (unlike EJB 2.1)
  - Metadata is specified with Java annotations or in XML deployment descriptors
- Entities are POJOs
  - No deployment code or abstract classes
  - No interfaces required
- An application server is not required
  - JPA is usable in both Java SE and Java EE environments

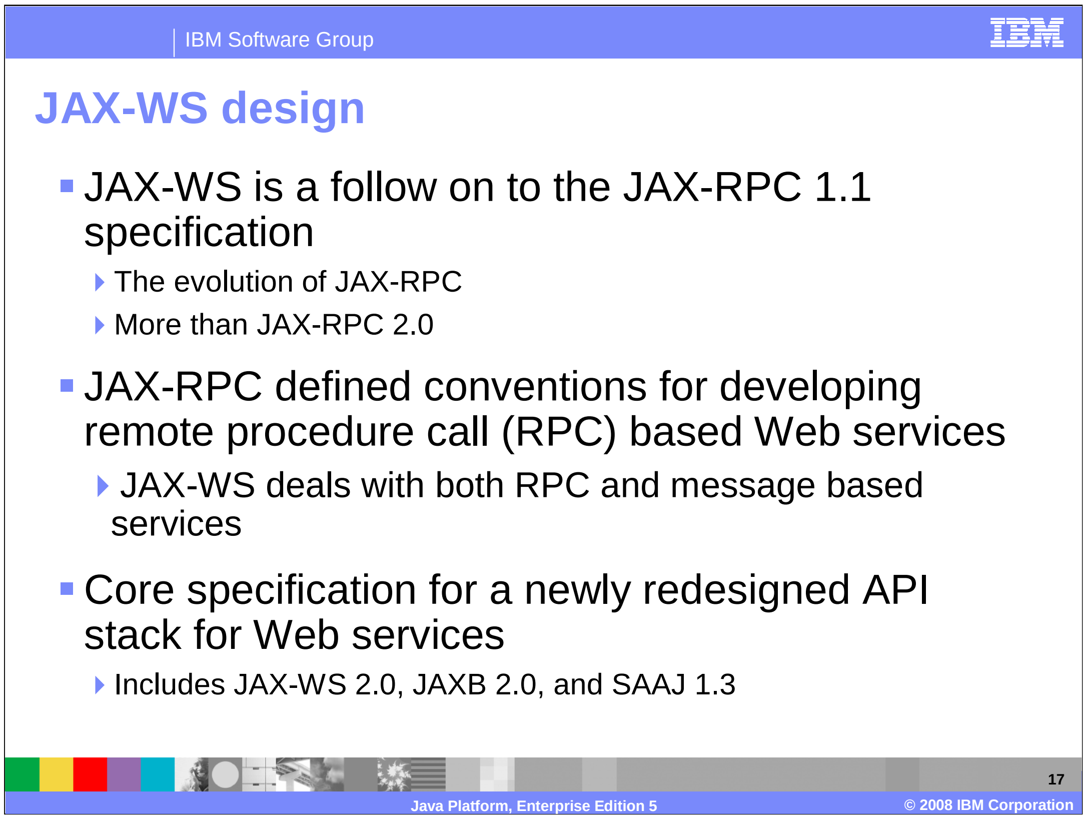
In JPA, Entities are plain-old Java objects, just like other components in EJB 3.0. An entity typically represents a table in a relational database, and each instance of an entity is a row. Entities are concrete classes, not abstract classes like Entity Beans in EJB 2.1. You do not need to generate deployment code, which speeds deployment, and you do not have to implement any particular interfaces. Another benefit is that all object-relational mapping information is specified in a standard fashion, using Java annotations or XML files. In earlier versions of the EJB specification, there was not a standard way to provide this information, which meant each vendor's implementation was different, and led to a reliance on vendor-specific tools. JPA can also be used without an EJB container, that is, in a Java Standard Edition (SE) environment.



Section

Web services

This section will highlight the new Web services features of Java EE 5.


IBM Software Group

JAX-WS design

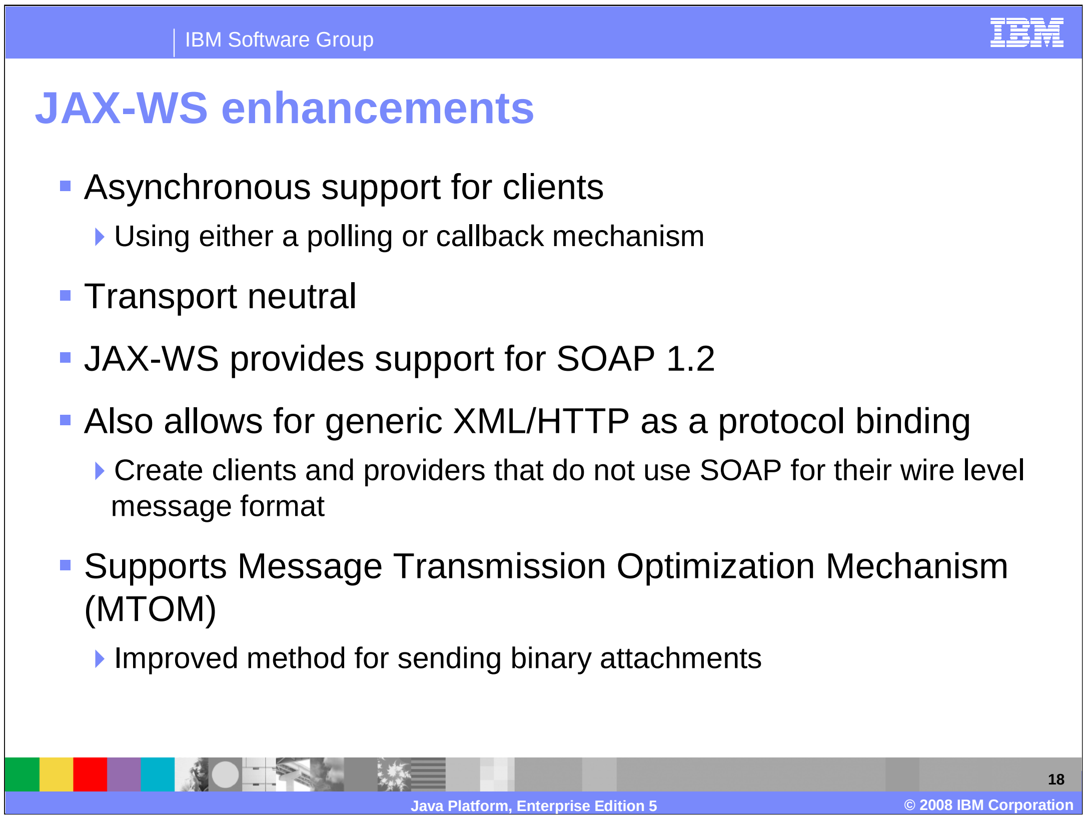
- JAX-WS is a follow on to the JAX-RPC 1.1 specification
  - The evolution of JAX-RPC
  - More than JAX-RPC 2.0
- JAX-RPC defined conventions for developing remote procedure call (RPC) based Web services
  - JAX-WS deals with both RPC and message based services
- Core specification for a newly redesigned API stack for Web services
  - Includes JAX-WS 2.0, JAXB 2.0, and SAAJ 1.3

17

Java Platform, Enterprise Edition 5 © 2008 IBM Corporation

The Java API for XML Web Services, or JAX-WS, is the centerpiece of a new programming model for Web services, this new model includes JAX-WS 2.0, JAXB 2.0, and SAAJ 1.3. JAX-WS is designed to eventually take the place of JAX-RPC in Web services and Web applications, and began as a new version of the JAX-RPC specification before becoming a separate specification.

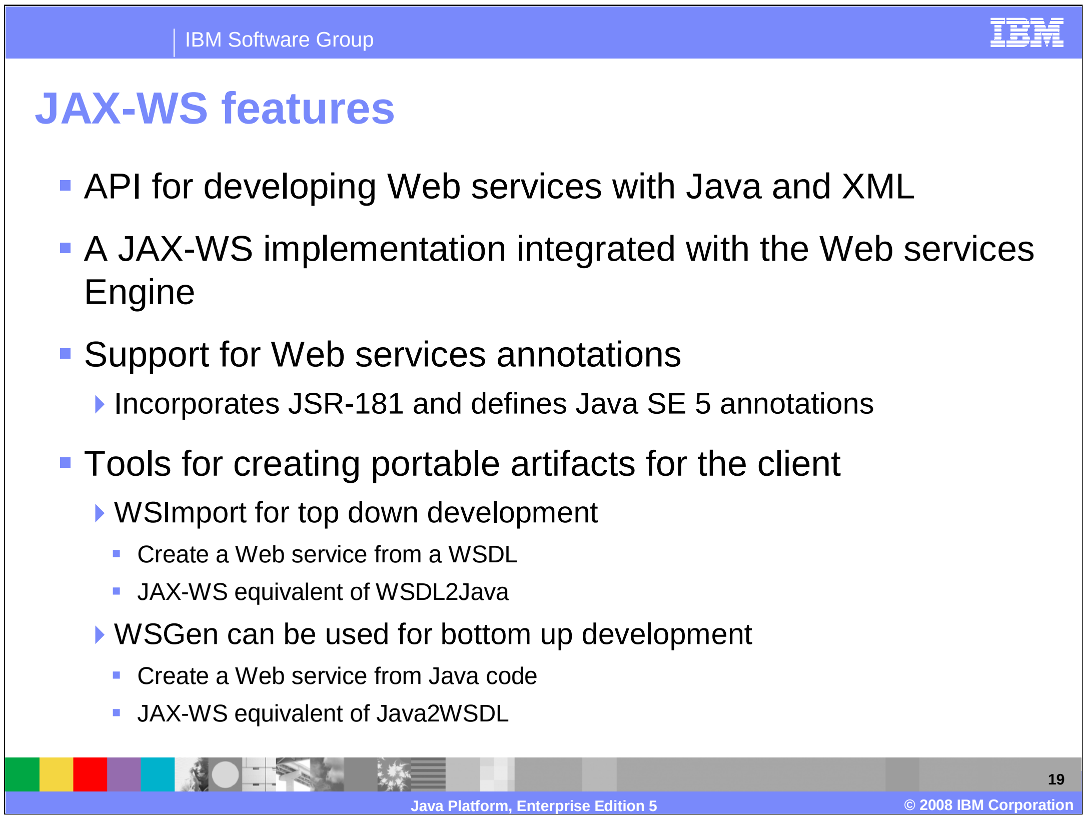
In the previous Web services development stack there was considerable overlap of data binding functionality between JAX-RPC and JAXB APIs. This is because JAX-RPC originally included basic data binding functionality. When JAXB later emerged, the need to separate the Web services definition and data binding components became clearer. The result is an easier-to-understand architecture for Web services development.



JAX-WS enhancements

- Asynchronous support for clients
  - Using either a polling or callback mechanism
- Transport neutral
- JAX-WS provides support for SOAP 1.2
- Also allows for generic XML/HTTP as a protocol binding
  - Create clients and providers that do not use SOAP for their wire level message format
- Supports Message Transmission Optimization Mechanism (MTOM)
  - Improved method for sending binary attachments

JAX-WS includes several important enhancements for developing Web services. This includes an asynchronous callback mechanism for clients. The asynchronous support is available using either a polling or callback mechanism. Another significant change is that the JAX-WS specification is transport neutral, not being linked to any particular transport mechanisms, such as HTTP. This allows JAX-WS applications to be more flexible in supporting additional transports such as JMS. JAX-WS also provides support for using SOAP version 1.2, though JAX-WS does not require SOAP to be used. It supports using pure XML over HTTP as a protocol binding, allowing developers to create services that do not use SOAP for the wire level format of their messages. JAX-WS also includes support for the Message Transmission Optimization Mechanism, otherwise known as MTOM, which is a better method to send binary attachments.


IBM Software Group

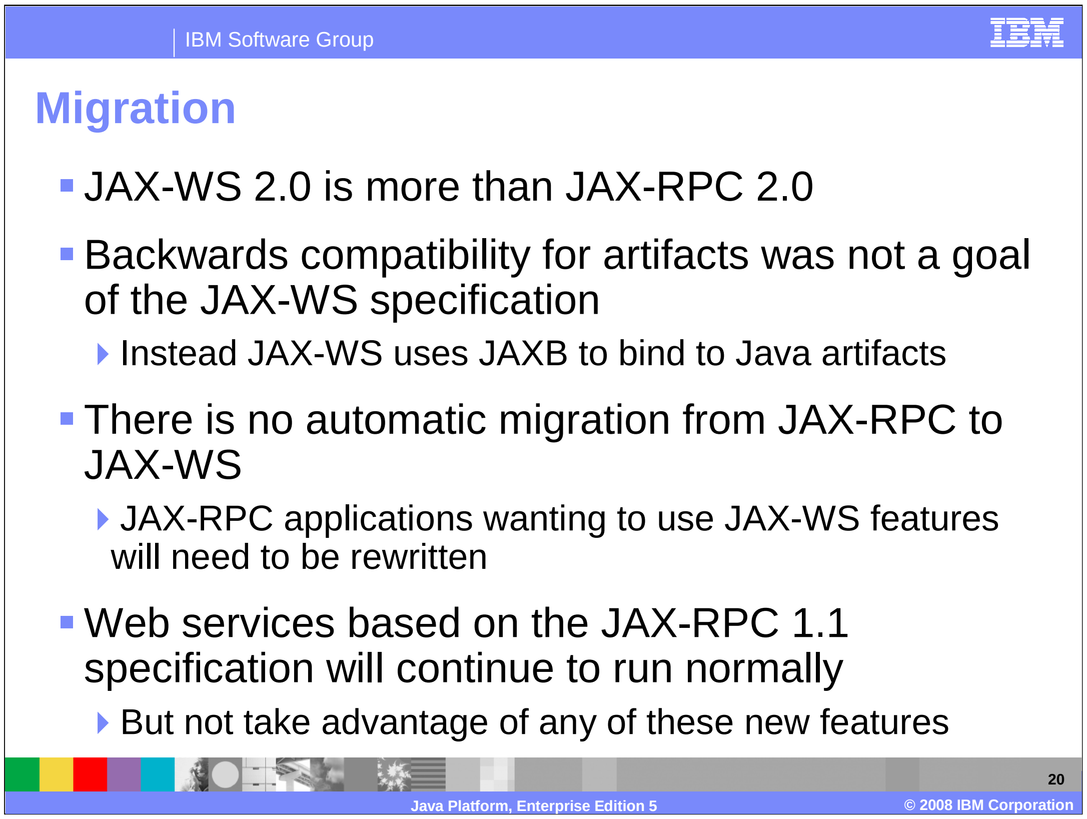
JAX-WS features

- API for developing Web services with Java and XML
- A JAX-WS implementation integrated with the Web services Engine
- Support for Web services annotations
  - Incorporates JSR-181 and defines Java SE 5 annotations
- Tools for creating portable artifacts for the client
  - WSImport for top down development
    - Create a Web service from a WSDL
    - JAX-WS equivalent of WSDL2Java
  - WSGen can be used for bottom up development
    - Create a Web service from Java code
    - JAX-WS equivalent of Java2WSDL

19

Java Platform, Enterprise Edition 5 © 2008 IBM Corporation

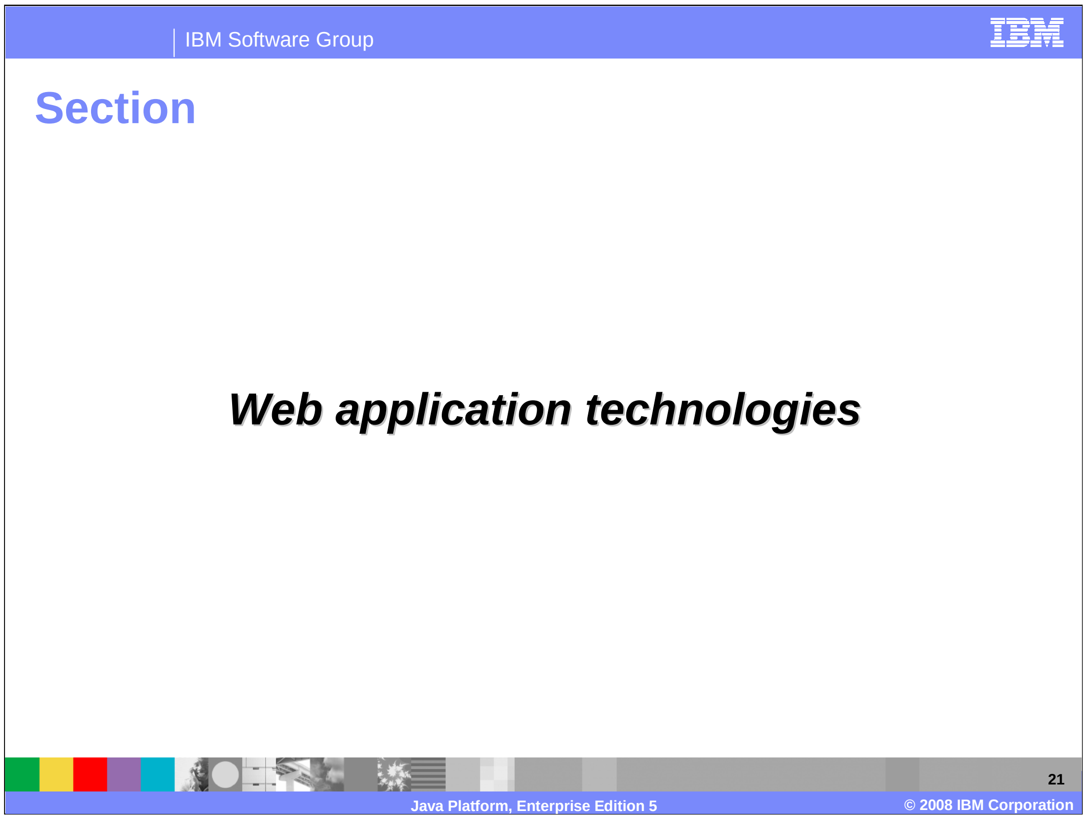
JAX-WS includes the standard implementation of the specification along with tools for building Web services, such as annotations. JAX-WS features an API for creating Web services with Java and XML. The JAX-WS implementation is integrated with the Web services engine, and support annotations based on the JAX-WS specification and JSR 181. There are also two command line tools for generating the necessary Java artifacts: WSImport for top down development from a WSDL file, and WSGen for bottom up development from Java code.



Migration

- JAX-WS 2.0 is more than JAX-RPC 2.0
- Backwards compatibility for artifacts was not a goal of the JAX-WS specification
  - Instead JAX-WS uses JAXB to bind to Java artifacts
- There is no automatic migration from JAX-RPC to JAX-WS
  - JAX-RPC applications wanting to use JAX-WS features will need to be rewritten
- Web services based on the JAX-RPC 1.1 specification will continue to run normally
  - But not take advantage of any of these new features

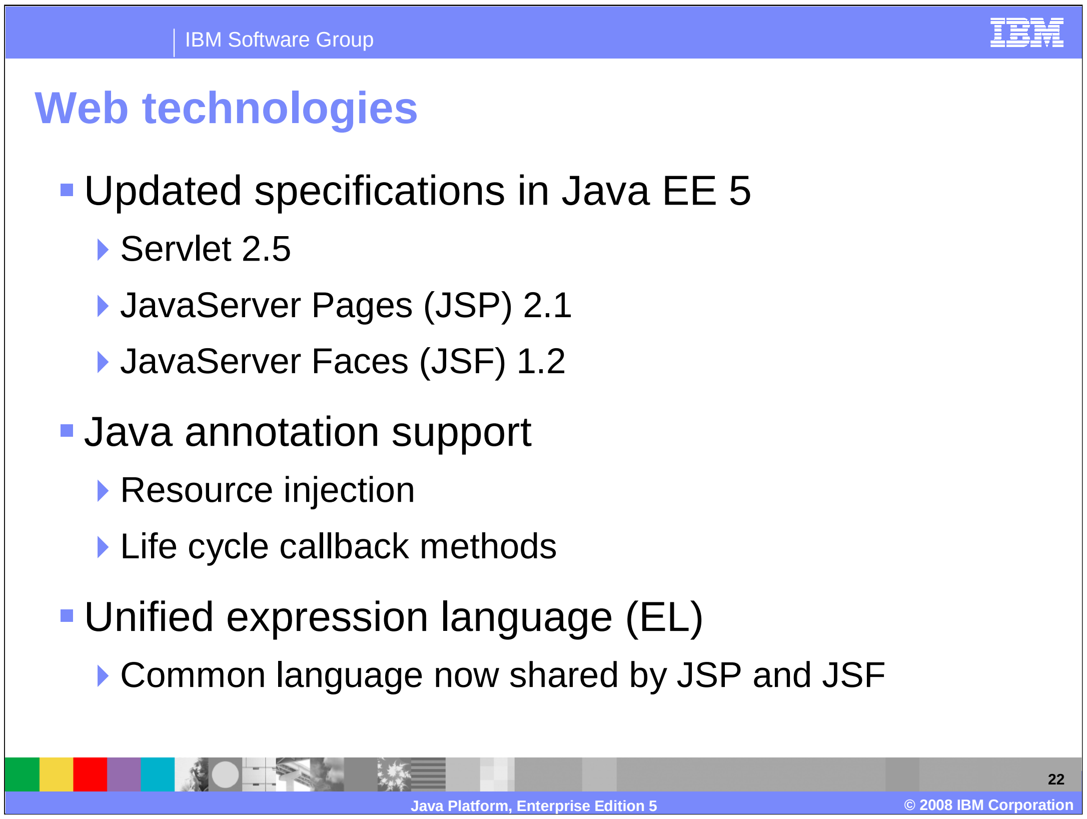
The JAX-WS specification was not designed with compatibility with earlier versions of artifacts as a goal. Due to this, there is no automatic or simple migration from JAX-RPC to JAX-WS. JAX-RPC applications that need to use these new technologies will have to be rewritten based on the JAX-WS specification. JAX-RPC applications will still function normally, but cannot take advantage of any of the new JAX-WS based features.



Section

Web application technologies

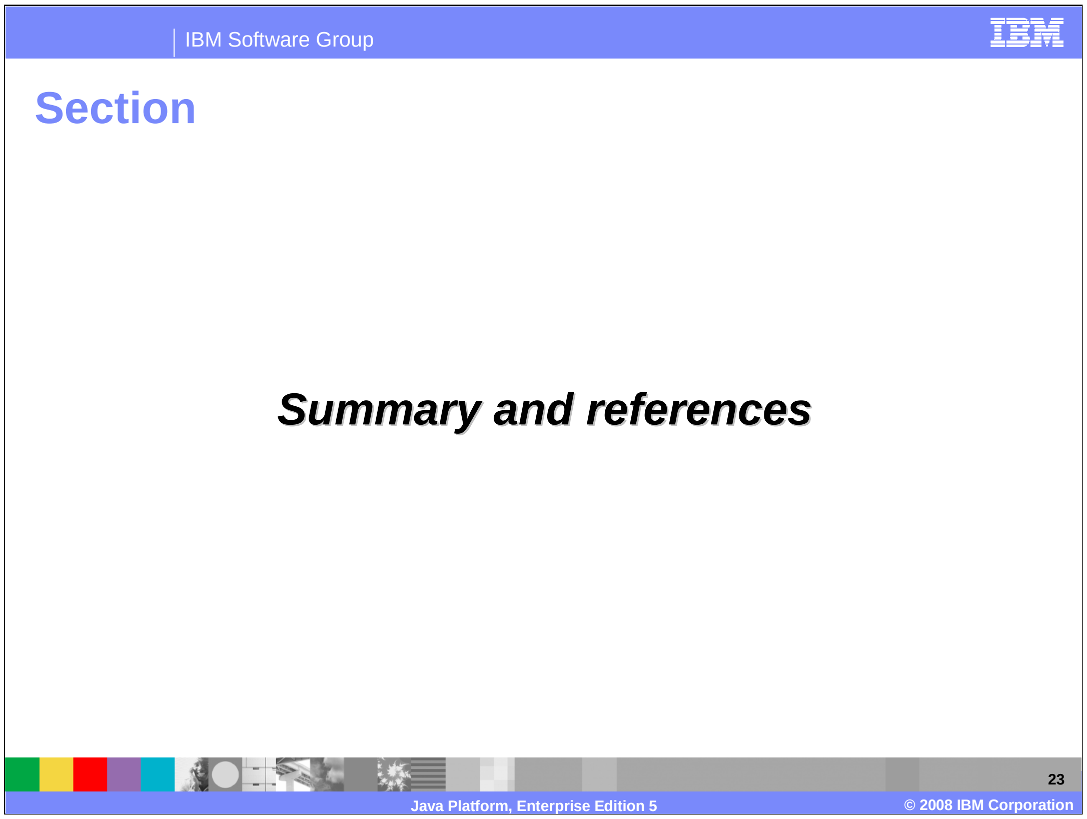
This section will discuss new Web application technologies in Java EE 5.



Web technologies

- Updated specifications in Java EE 5
  - Servlet 2.5
  - JavaServer Pages (JSP) 2.1
  - JavaServer Faces (JSF) 1.2
- Java annotation support
  - Resource injection
  - Life cycle callback methods
- Unified expression language (EL)
  - Common language now shared by JSP and JSF

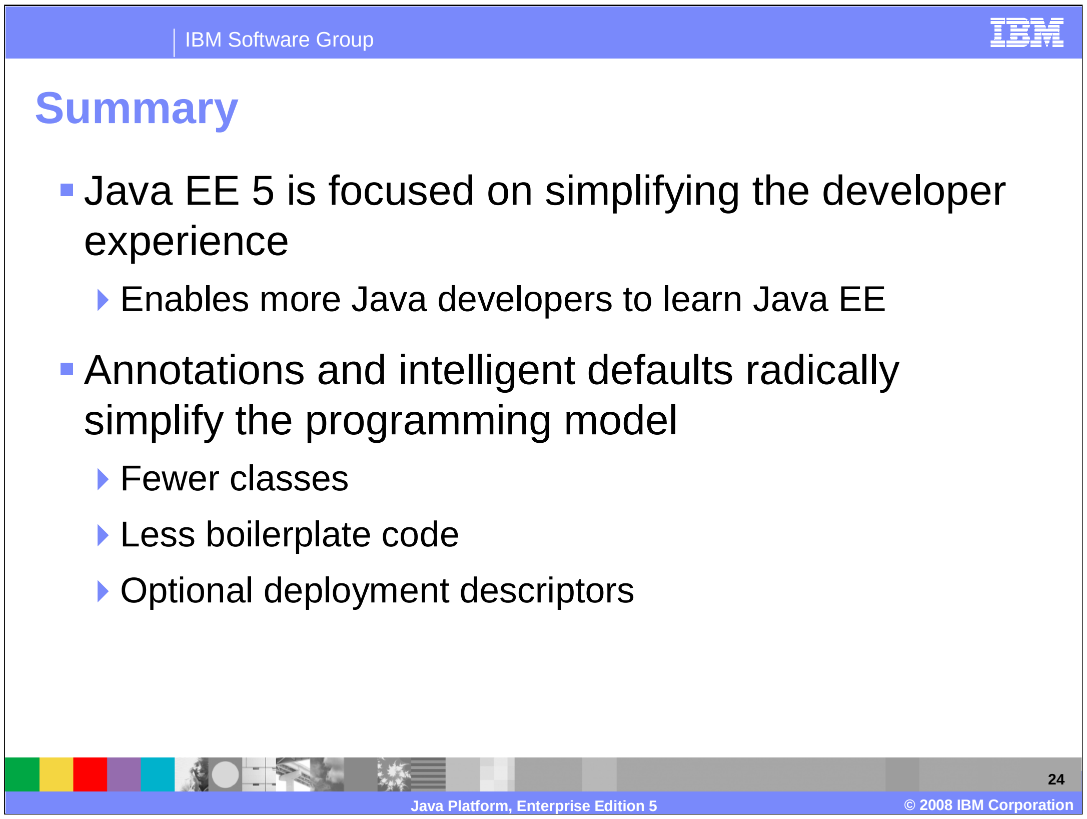
Several Web-related technologies have been revised in Java EE 5. The Servlet, JavaServer Pages, and JavaServer Faces specifications have all seen minor revisions in this release, and now support the use of Java annotations for things like resource injection and life cycle callback methods. If a WAR file contains only JSPs, it need not include an XML deployment descriptor. If it contains any servlets, a deployment descriptor must still be used. Also in this revision, the expression languages used by JavaServer Pages and JavaServer Faces have been unified, so that both types of pages now used the same language.



Section

Summary and references

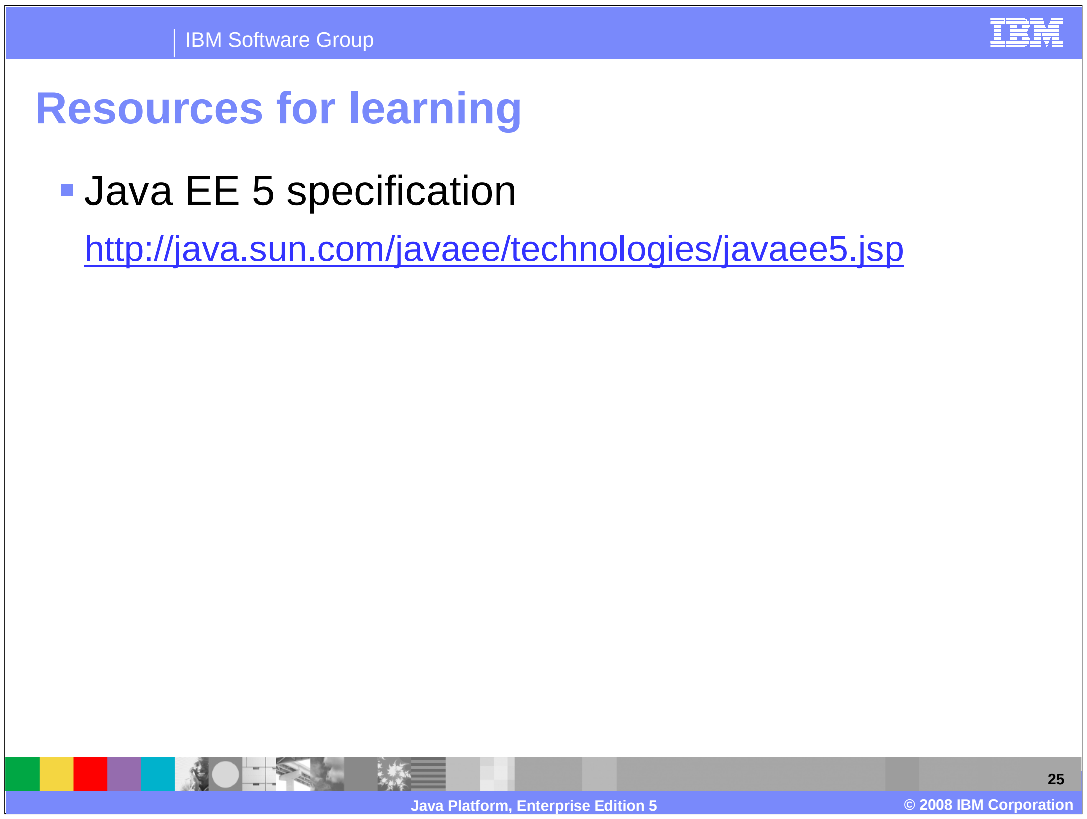
This section will summarize the presentation.



Summary

- Java EE 5 is focused on simplifying the developer experience
  - Enables more Java developers to learn Java EE
- Annotations and intelligent defaults radically simplify the programming model
  - Fewer classes
  - Less boilerplate code
  - Optional deployment descriptors

The Java EE 5 specification aims to enable more Java developers to become enterprise Java developers by reducing the learning curve for Java EE. Java annotations provide the ability to specify metadata directly in application code, rather than using an XML deployment descriptor in many cases. The heavy use of default values greatly reduces the amount of boilerplate code that is required to create a basic enterprise application. Fewer artifacts are involved in creating Java EE 5 applications as well, since many interfaces are no longer required, and others can be generated automatically. The key new specifications in Java EE 5 are EJB 3.0, JPA, JAX-WS 2.0, Servlet 2.5, JSP 2.1, and JSF 1.2.

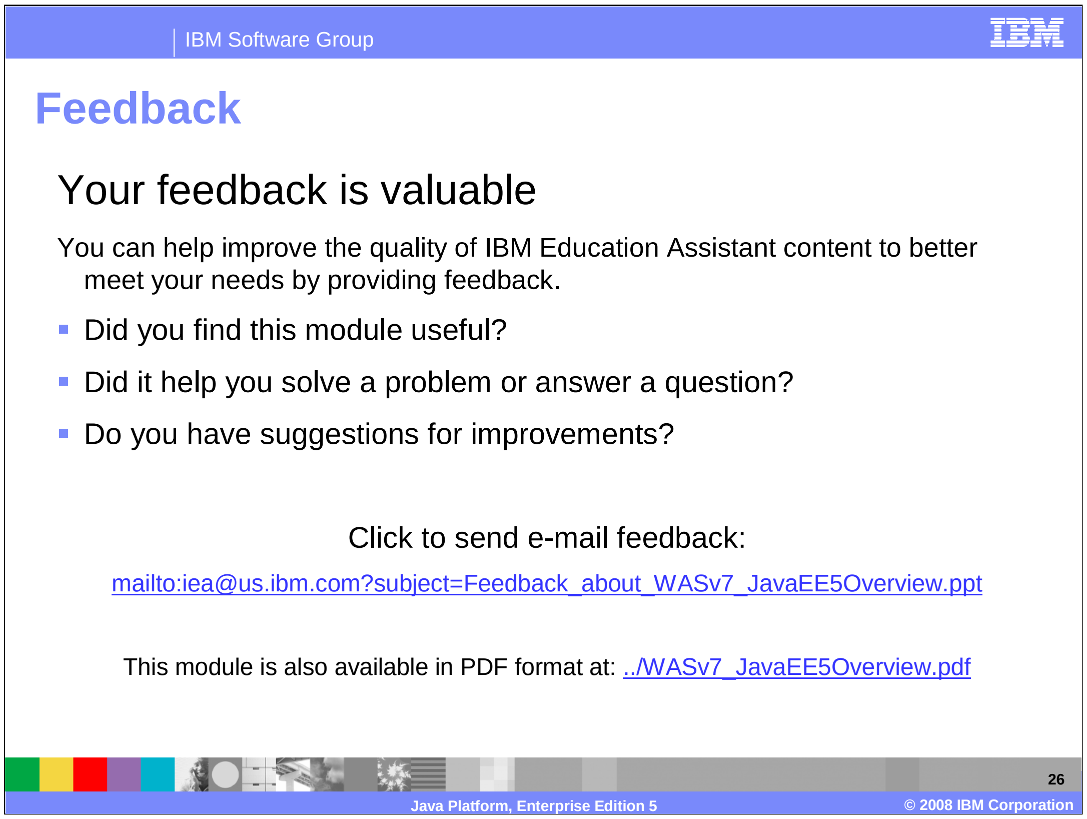


Resources for learning

- Java EE 5 specification

  http://java.sun.com/javaee/technologies/javaee5.jsp

For more details on the Java EE 5 specification, consult the online documentation for the specification directly, using the link shown here.



Feedback

Your feedback is valuable

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

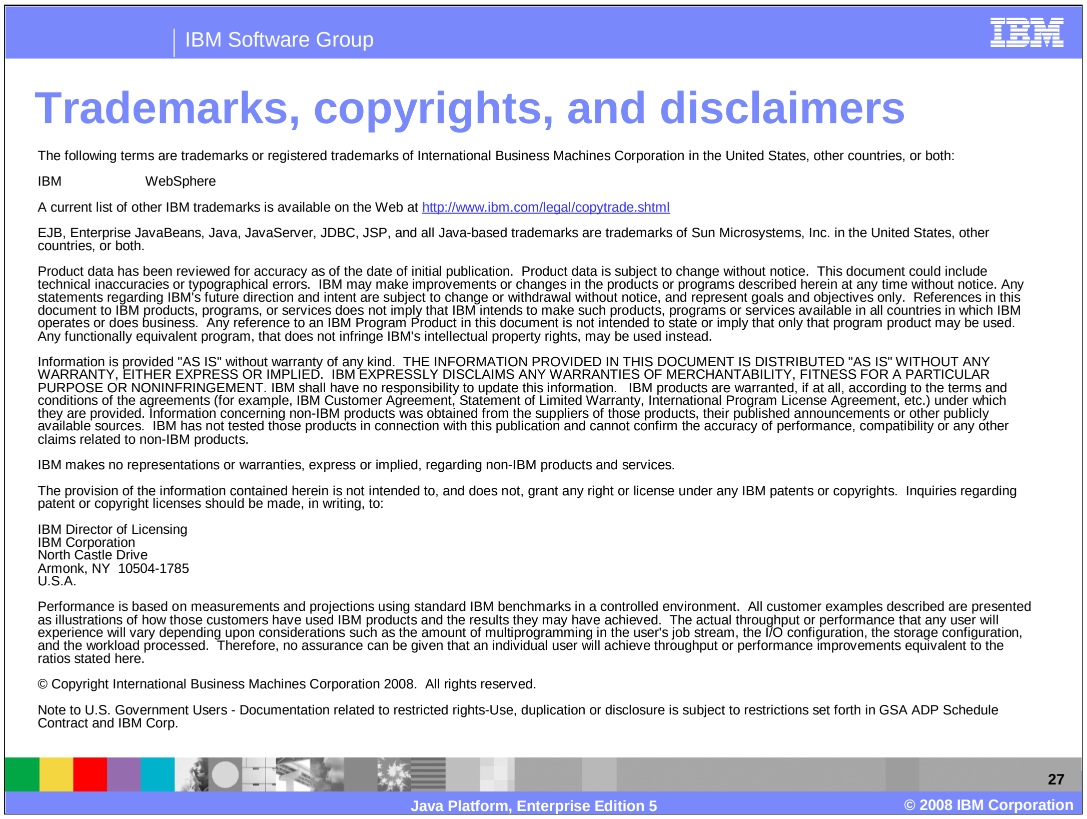
- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WASv7_JavaEE5Overview.ppt

This module is also available in PDF format at: ../WASv7_JavaEE5Overview.pdf

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



Trademarks, copyrights, and disclaimers