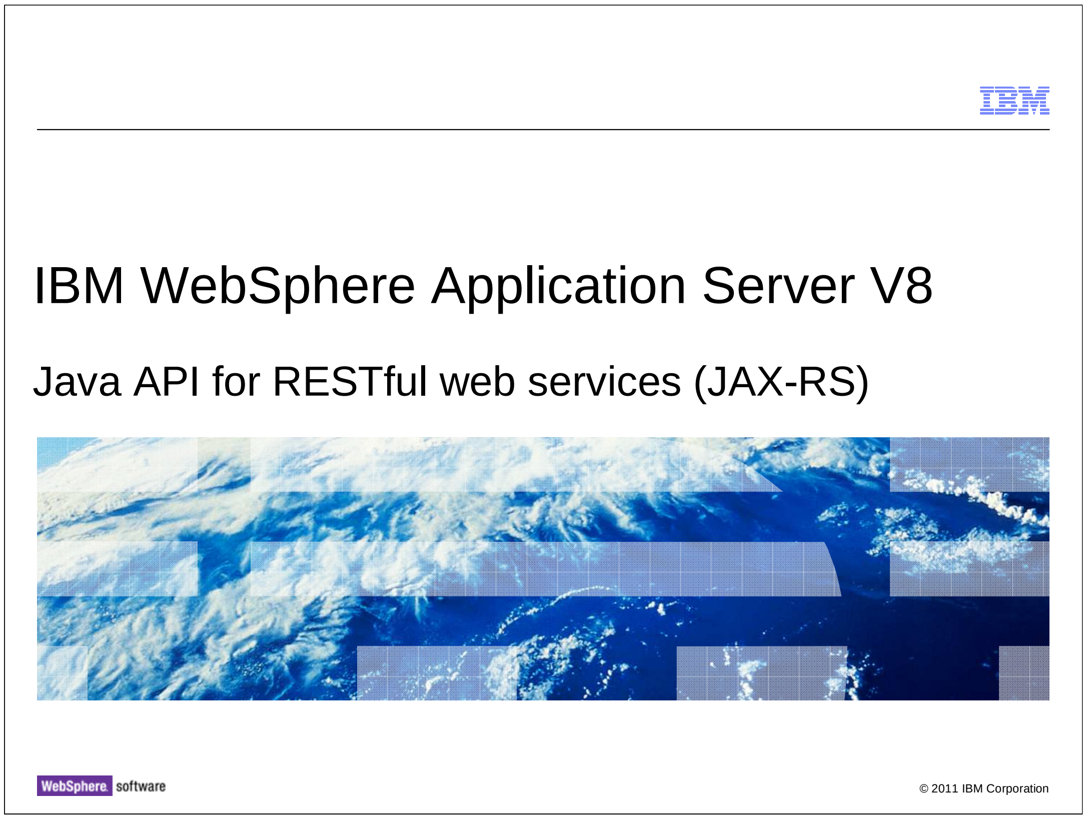
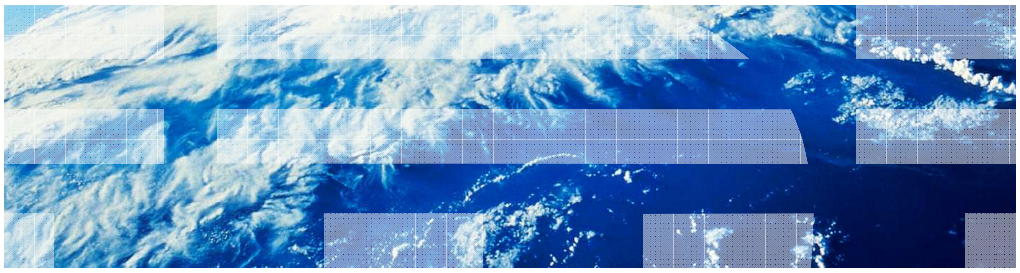


IBM WebSphere Application Server V8

Java API for RESTful web services (JAX-RS)



This presentation describes support for Java API for RESTful web Services (JAX-RS) included in IBM WebSphere Application Server V8.

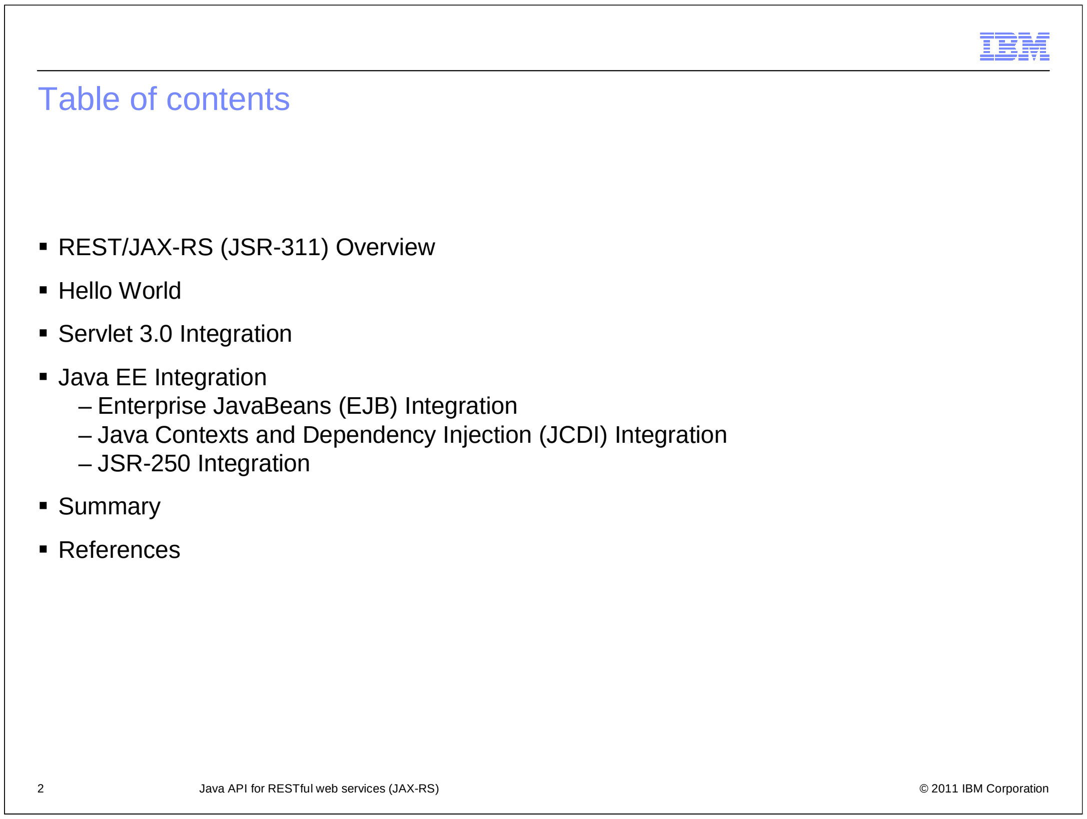
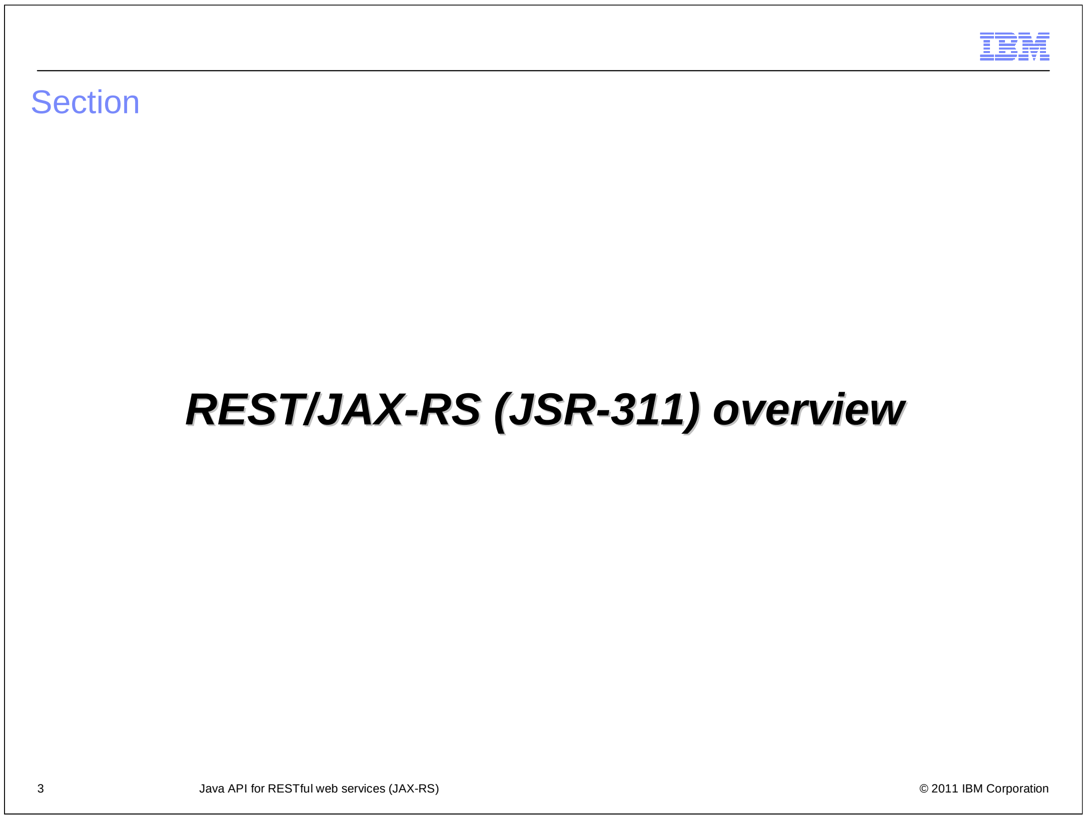


Table of contents

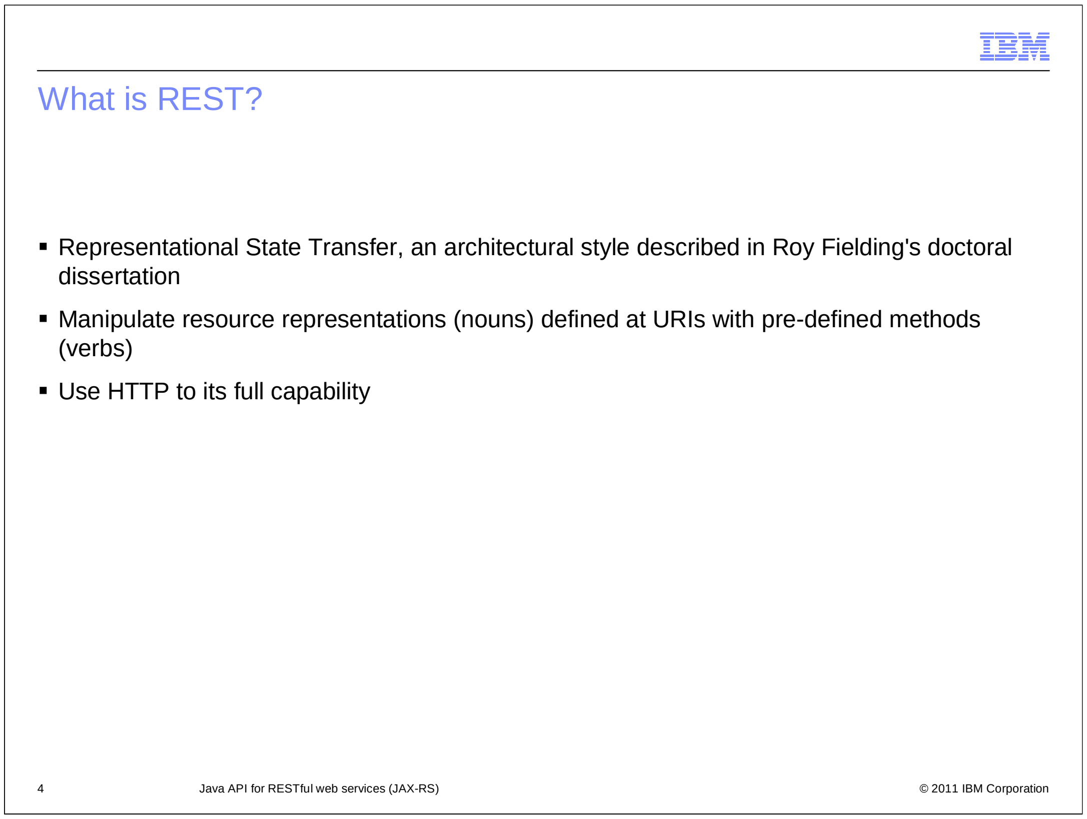
This presentation provides an overview of the Java API for RESTful web Services (JAX-RS) support in WebSphere Application Server V8.



Section

REST/JAX-RS (JSR-311) overview

What will JAX-RS do for you?

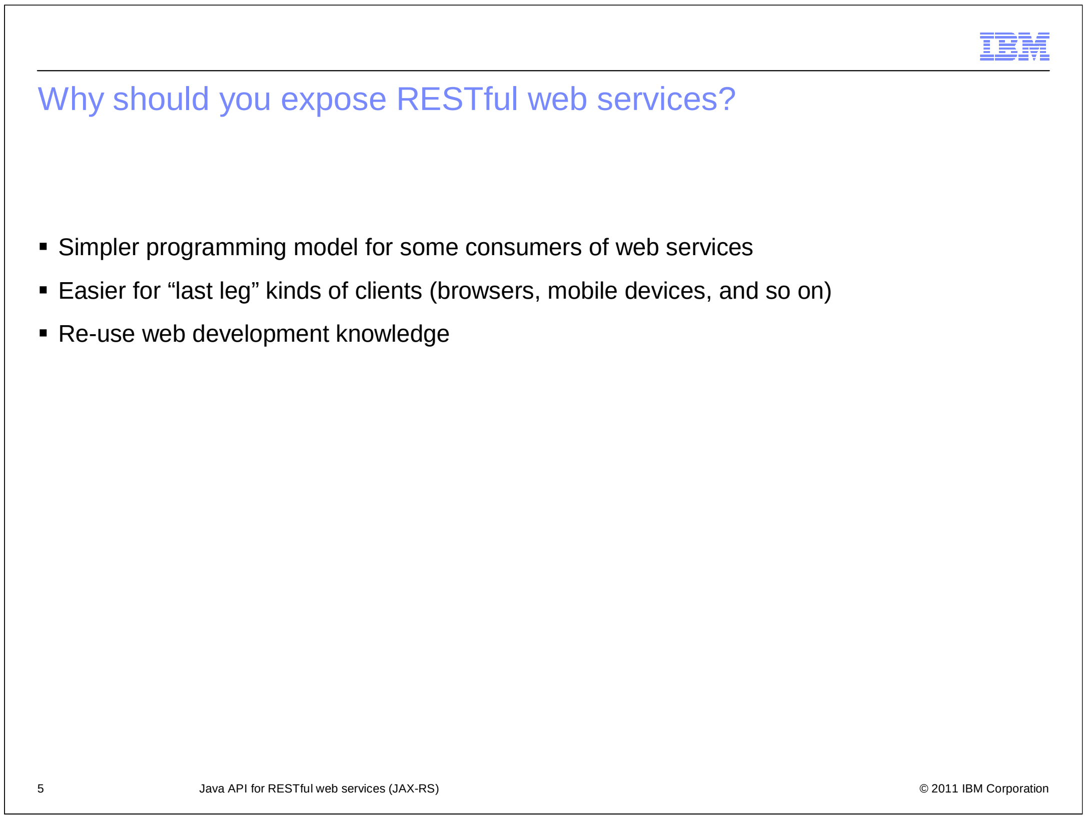


What is REST?

- Representational State Transfer, an architectural style described in Roy Fielding's doctoral dissertation
- Manipulate resource representations (nouns) defined at URIs with pre-defined methods (verbs)
- Use HTTP to its full capability

4 Java API for RESTful web services (JAX-RS) © 2011 IBM Corporation

REST stands for Representational State Transfer. The acronym was coined by Roy Fielding in his Ph.D. dissertation, in which he lays out the guiding principles for REST-style or RESTful web services. REST is an architectural style in which resources are identified by URIs and interconnected through hyperlinks. Not only does REST take advantage of the transport properties of HTTP, but it also uses HTTP verbs to invoke operations on resources. Each HTTP request for a resource includes a verb to indicate what operation should be performed on the resource. The standard HTTP verbs are POST, GET, PUT, and DELETE to represent the standard database operations of create, read, update, and delete, respectively. The HTTP response to a request is typically a standard response code, such as 404 for resource not found or 200 for success, or a MIME-typed representation of the resource, such as text/html.

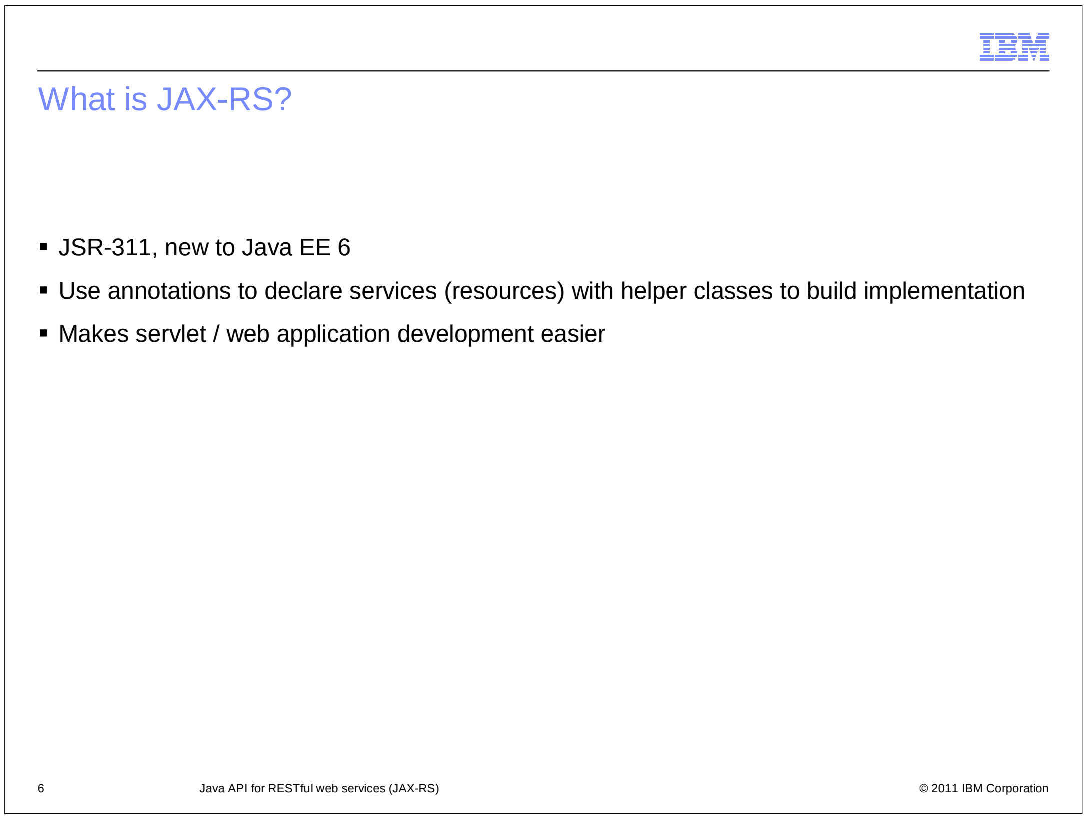


Why should you expose RESTful web services?

- Simpler programming model for some consumers of web services
- Easier for “last leg” kinds of clients (browsers, mobile devices, and so on)
- Re-use web development knowledge

5 Java API for RESTful web services (JAX-RS) © 2011 IBM Corporation

The RESTful approach tries to simplify web services by using capabilities that are already built into HTTP, including uniform operations, resources that are accessible using URIs, and resources that are represented by media types. The transport layer with RESTful web services is less complex than with SOAP-based web services. With REST, each resource is transmitted in the body of an HTTP message, whereas a SOAP message is encapsulated as the body of a transport message. REST services are stateless and cacheable, and a client only needs to provide a starting URL to invoke a web service. REST can work on any platform that has an HTTP client, including web browsers and mobile devices.

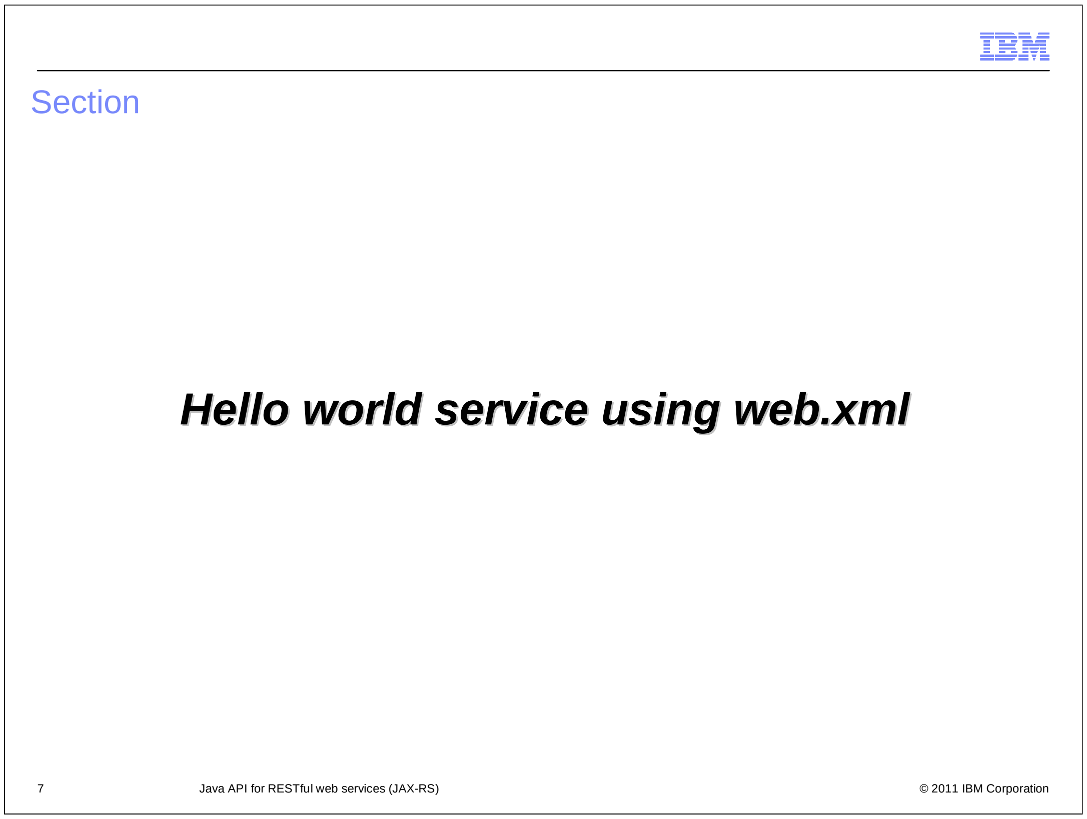


What is JAX-RS?

- JSR-311, new to Java EE 6
- Use annotations to declare services (resources) with helper classes to build implementation
- Makes servlet / web application development easier

6 Java API for RESTful web services (JAX-RS) © 2011 IBM Corporation

The Java API for RESTful web Services, or JAX-RS, is an annotations-based framework that is defined in JSR 311. JAX-RS is a Java API designed to make it easy to develop applications that use the REST architecture. The JAX-RS API uses Java programming language annotations to simplify the development of RESTful web services. Developers use annotations to define resources and the actions that can be performed on those resources. The JAX-RS runtime will generate the helper classes and artifacts for the resource, making web application development easier.

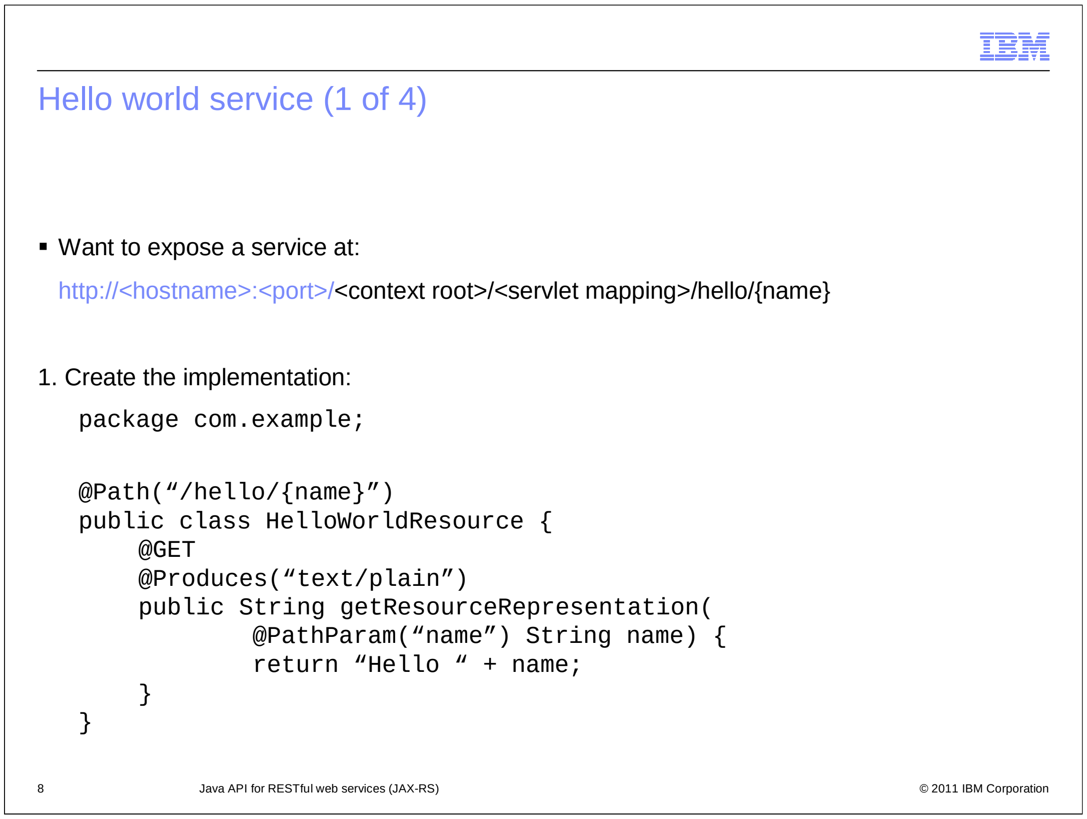


Section

Hello world service using web.xml

7 Java API for RESTful web services (JAX-RS) © 2011 IBM Corporation

The following charts will show an example of creating a Hello Word service using a web.xml file.



Hello world service (1 of 4)

- Want to expose a service at:

  http://<hostname>:<port>/<context root>/<servlet mapping>/hello/{name}

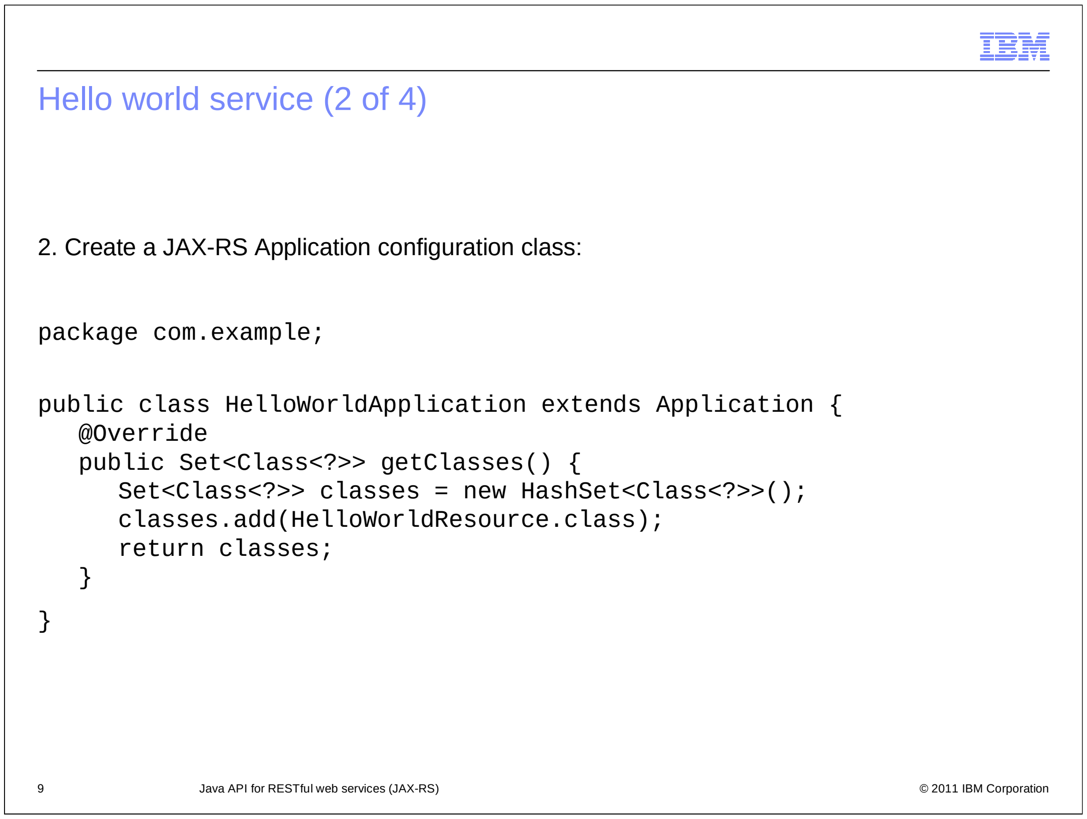
1. Create the implementation:

```
package com.example;

@Path(“/hello/{name}”)
public class HelloWorldResource {
    @GET
    @Produces(“text/plain”)
    public String getResourceRepresentation(
            @PathParam(“name”) String name) {
            return “Hello “ + name;
    }
}
```

8 Java API for RESTful web services (JAX-RS) © 2011 IBM Corporation

The first step is to create the implementation class for the HelloWorldResource. The @Path annotation specifies the relative URI path for the service, and the @GET and @Produces annotations indicate that the service will process HTTP GET requests and return a media type of “text/plain”.



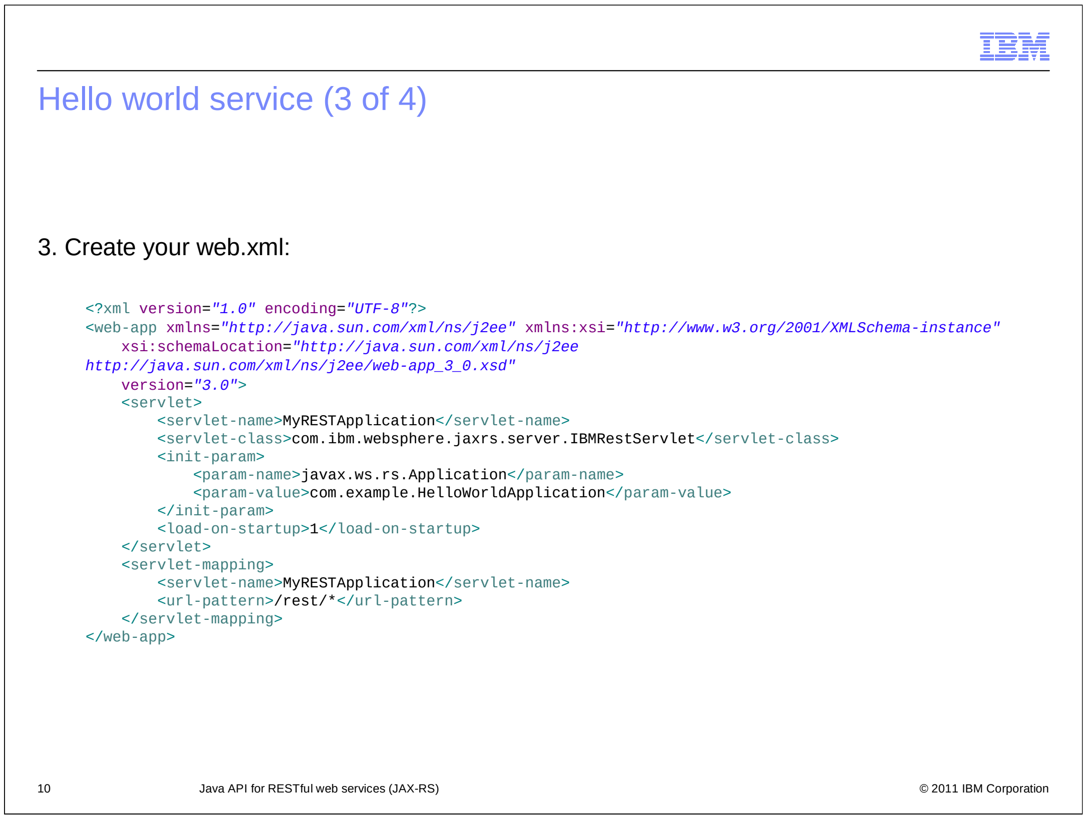
Hello world service (2 of 4)

2. Create a JAX-RS Application configuration class:

```
package com.example;

public class HelloWorldApplication extends Application {
   @Override
   public Set<Class<?>> getClasses() {
      Set<Class<?>> classes = new HashSet<Class<?>>();
      classes.add(HelloWorldResource.class);
      return classes;
   }
}
```

The second step is to create a JAX-RS application configuration class, named HelloWorldApplication. This class implements the getClasses method to return the set of resources and providers.



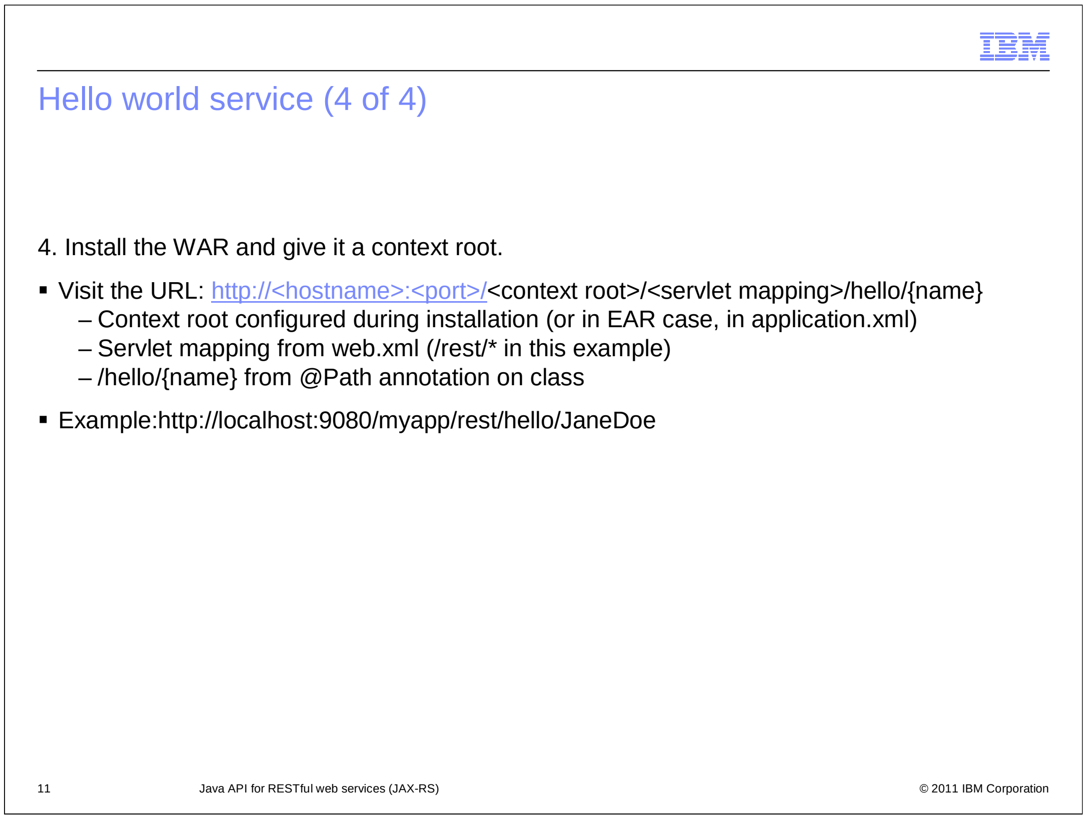
Hello world service (3 of 4)

3. Create your web.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_3_0.xsd"
    version="3.0">
    <servlet>
        <servlet-name>MyRESTApplication</servlet-name>
        <servlet-class>com.ibm.websphere.jaxrs.server.IBMRestServlet</servlet-class>
        <init-param>
            <param-name>javax.ws.rs.Application</param-name>
            <param-value>com.example.HelloWorldApplication</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>MyRESTApplication</servlet-name>
        <url-pattern>/rest/*</url-pattern>
    </servlet-mapping>
</web-app>
```

10 Java API for RESTful web services (JAX-RS) © 2011 IBM Corporation

The third step is to create the web.xml, which specifies the servlet name and mapping information.



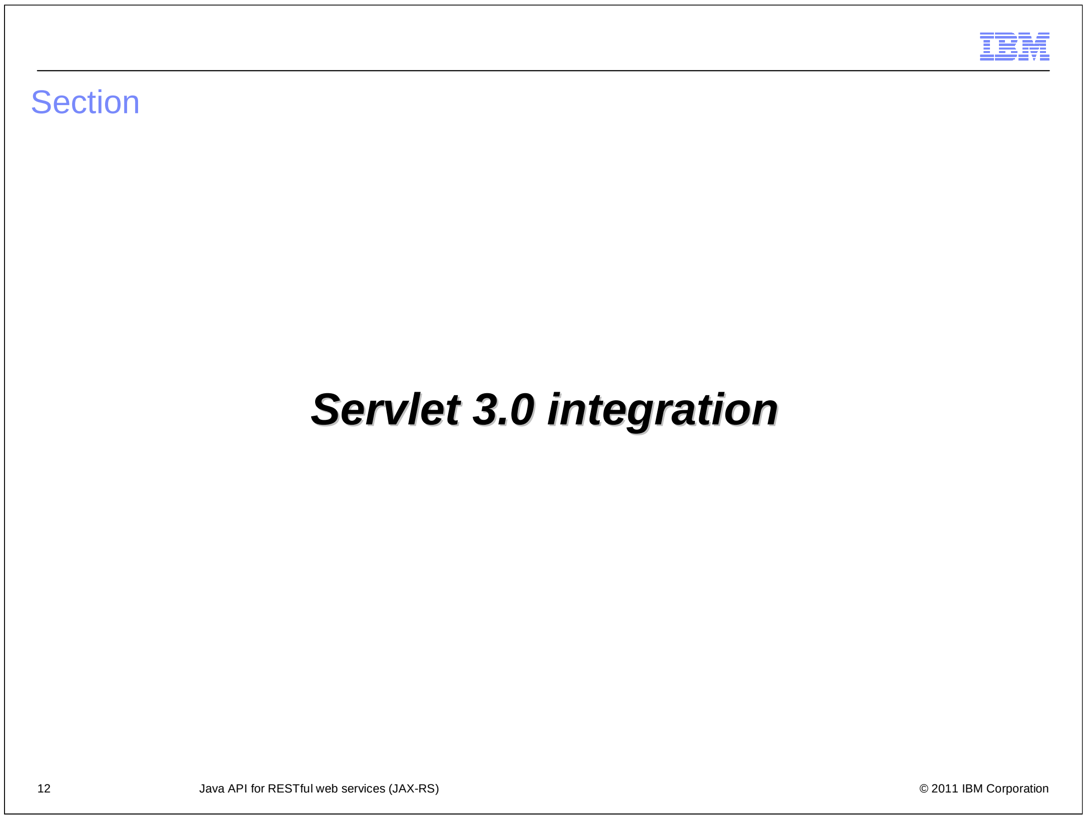
Hello world service (4 of 4)

4. Install the WAR and give it a context root.

- Visit the URL: http://<hostname>:<port>/<context root>/<servlet mapping>/hello/{name}
  - Context root configured during installation (or in EAR case, in application.xml)
  - Servlet mapping from web.xml (/rest/* in this example)
  - /hello/{name} from @Path annotation on class
- Example:http://localhost:9080/myapp/rest/hello/JaneDoe

11 Java API for RESTful web services (JAX-RS) © 2011 IBM Corporation

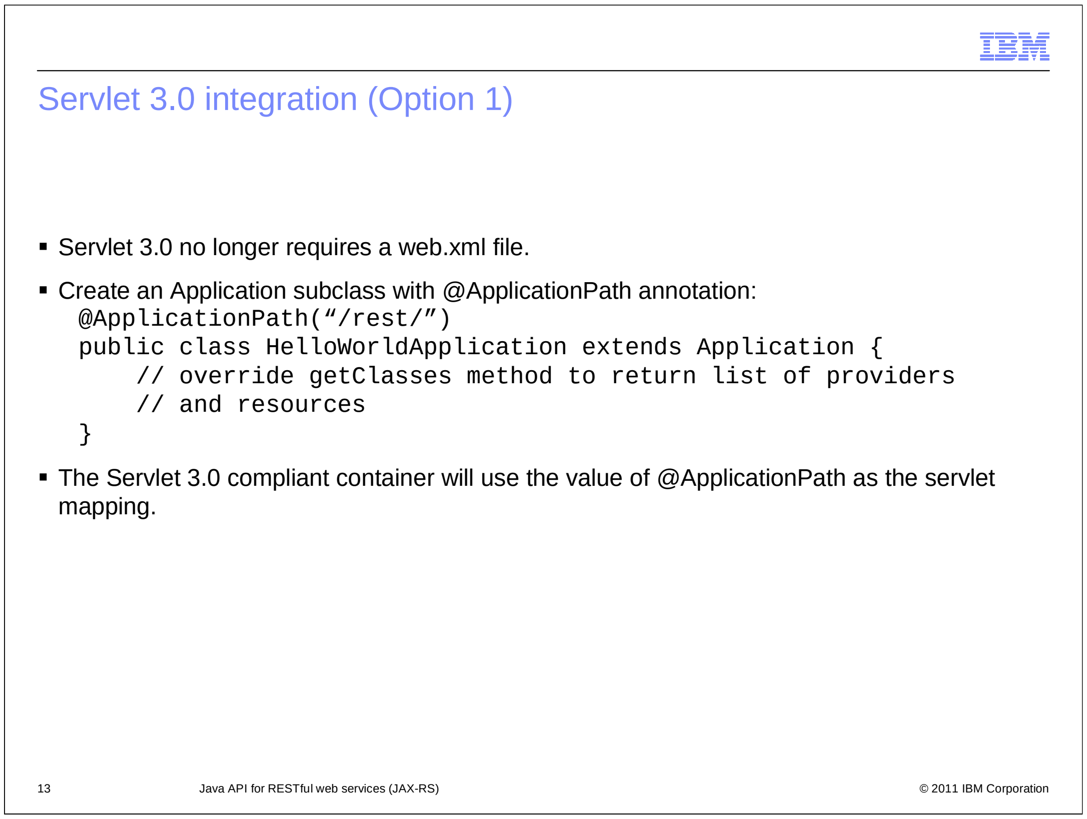
In Step 4, you install the WAR and provide the context root.



Section

Servlet 3.0 integration

The next section discusses Servlet 3.0 integration.



Servlet 3.0 integration (Option 1)

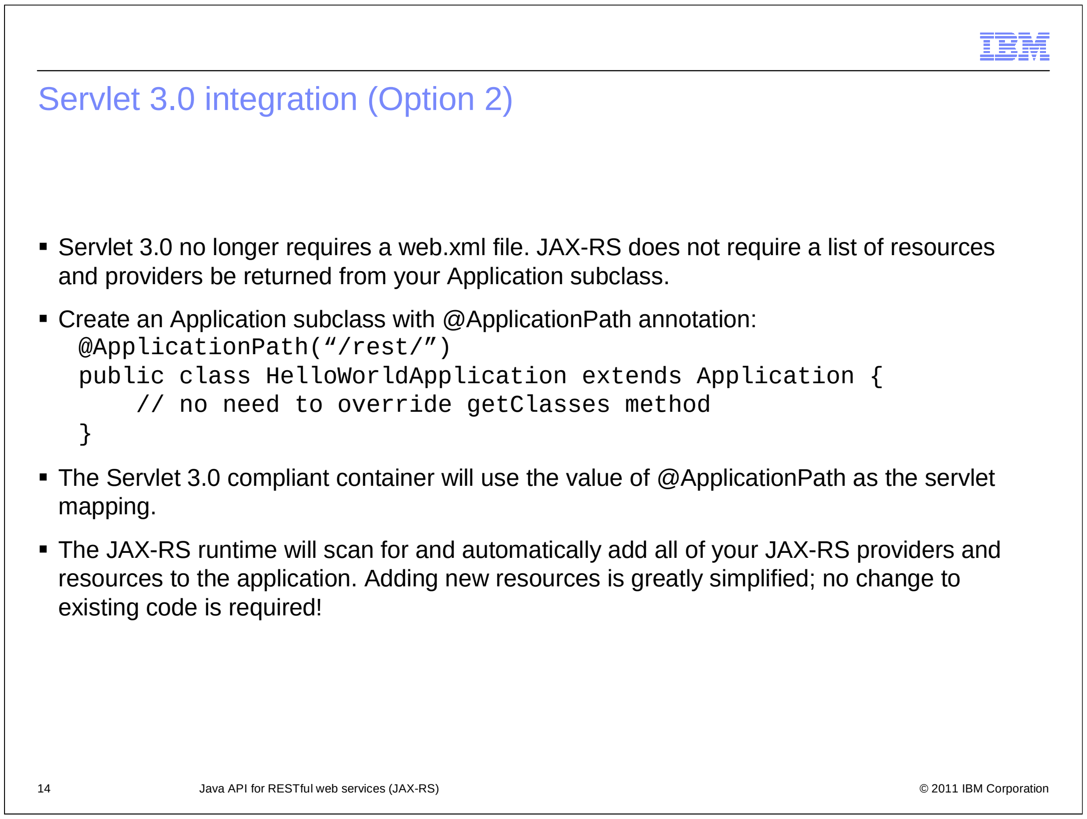
- Servlet 3.0 no longer requires a web.xml file.
- Create an Application subclass with @ApplicationPath annotation:

```
@ApplicationPath(“/rest/”)
public class HelloWorldApplication extends Application {
    // override getClasses method to return list of providers
    // and resources
}
```

- The Servlet 3.0 compliant container will use the value of @ApplicationPath as the servlet mapping.

13 Java API for RESTful web services (JAX-RS) © 2011 IBM Corporation

The Servlet 3.0 specification no longer requires a web.xml file for a web application. If the application subclass contains an @ApplicationPath annotation, the Servlet 3.0-compliant container will use the value of @ApplicationPath for the servlet mapping.



Servlet 3.0 integration (Option 2)

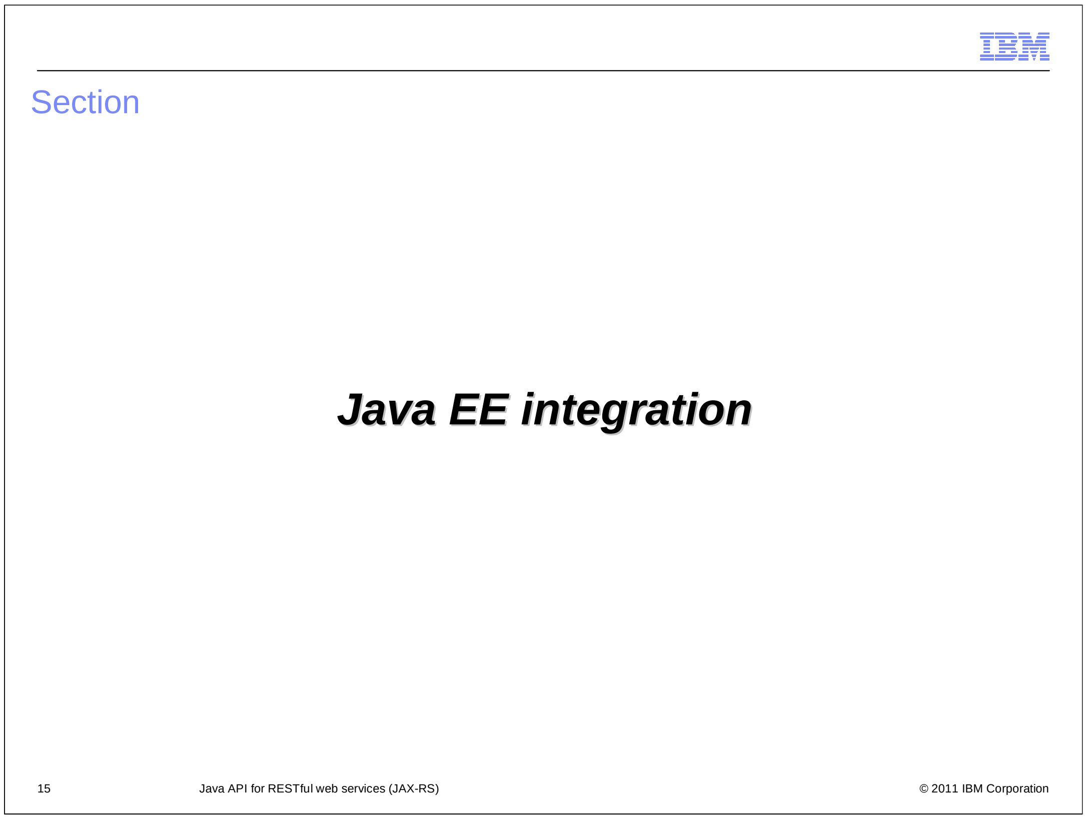
- Servlet 3.0 no longer requires a web.xml file. JAX-RS does not require a list of resources and providers be returned from your Application subclass.
- Create an Application subclass with @ApplicationPath annotation:

```
@ApplicationPath(“/rest/”)
public class HelloWorldApplication extends Application {
    // no need to override getClasses method
}
```

- The Servlet 3.0 compliant container will use the value of @ApplicationPath as the servlet mapping.
- The JAX-RS runtime will scan for and automatically add all of your JAX-RS providers and resources to the application. Adding new resources is greatly simplified; no change to existing code is required!

14 Java API for RESTful web services (JAX-RS) © 2011 IBM Corporation

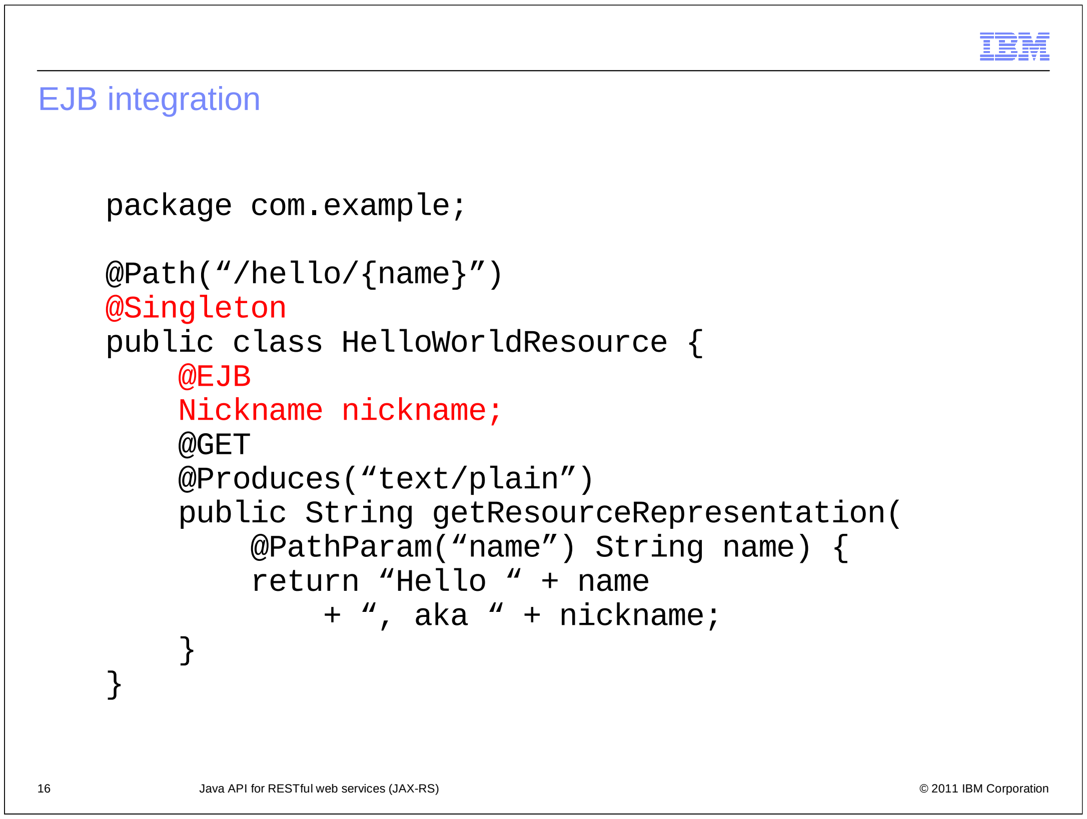
In addition, JAX-RS no longer requires the Application subclass to override the getClasses method to return the list of resources and providers. The JAX-RS runtime will automatically scan for annotations and add the appropriate JAX-RS providers and resources to the application.



Section

Java EE integration

This section discusses Java EE integration.

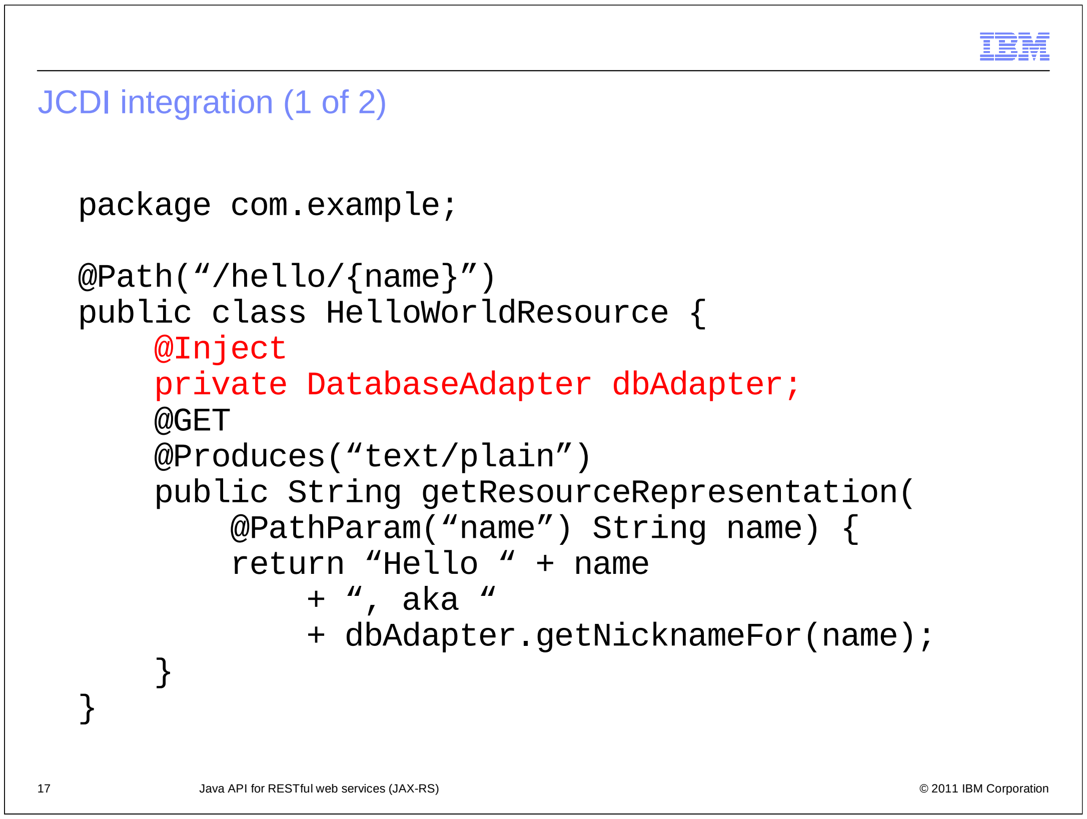


EJB integration

```
package com.example;

@Path(“/hello/{name}”)
@Singleton
public class HelloWorldResource {
    @EJB
    Nickname nickname;
    @GET
    @Produces(“text/plain”)
    public String getResourceRepresentation(
        @PathParam(“name”) String name) {
        return “Hello “ + name
            + “, aka “ + nickname;
    }
}
```

The singleton EJB is a new feature in Java EE 6. The container will maintain a single shared instance of an EJB Singleton. In this example, the singleton Nickname will persist the EJB state.



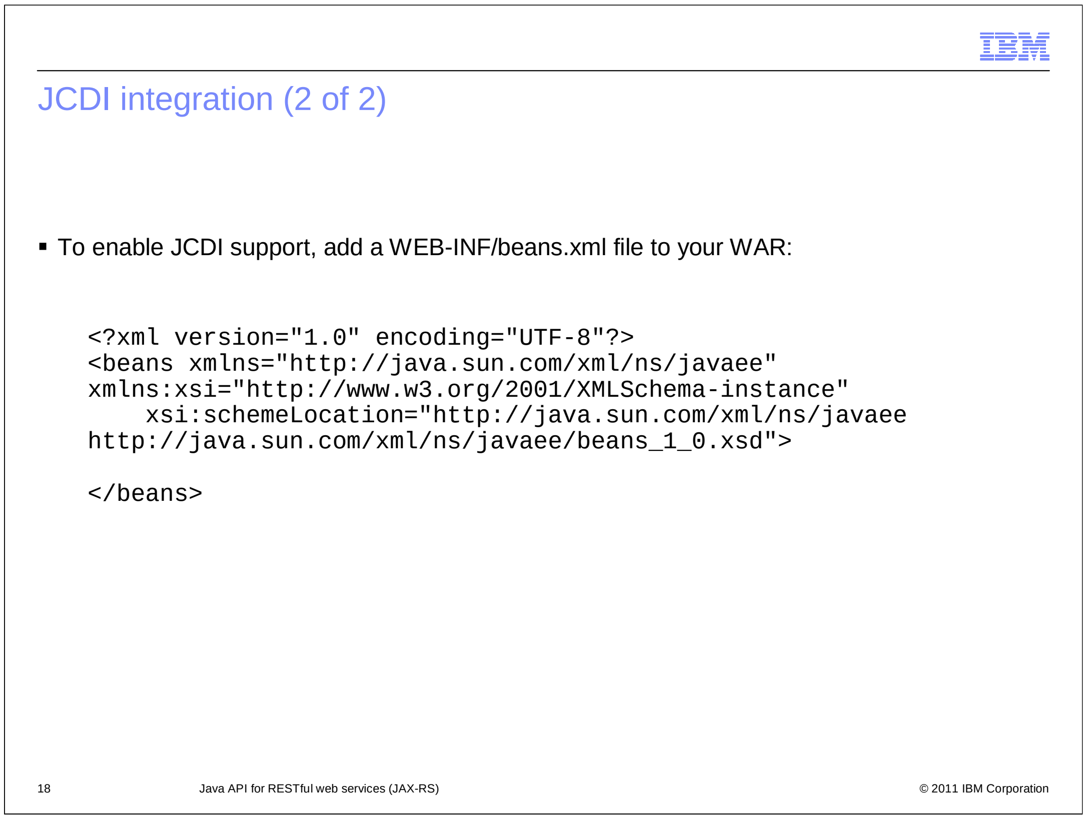
JCDI integration (1 of 2)

```
package com.example;

@Path(“/hello/{name}”)
public class HelloWorldResource {
    @Inject
    private DatabaseAdapter dbAdapter;
    @GET
    @Produces(“text/plain”)
    public String getResourceRepresentation(
        @PathParam(“name”) String name) {
        return “Hello “ + name
            + “, aka “
            + dbAdapter.getNicknameFor(name);
    }
}
```

17 Java API for RESTful web services (JAX-RS) © 2011 IBM Corporation

Java Contexts and Dependency Injection (JCDI) supports dependency injection in managed beans. In this example, an instance of the Database adapter type is injected into the dbAdapter field during runtime.



JCDI integration (2 of 2)

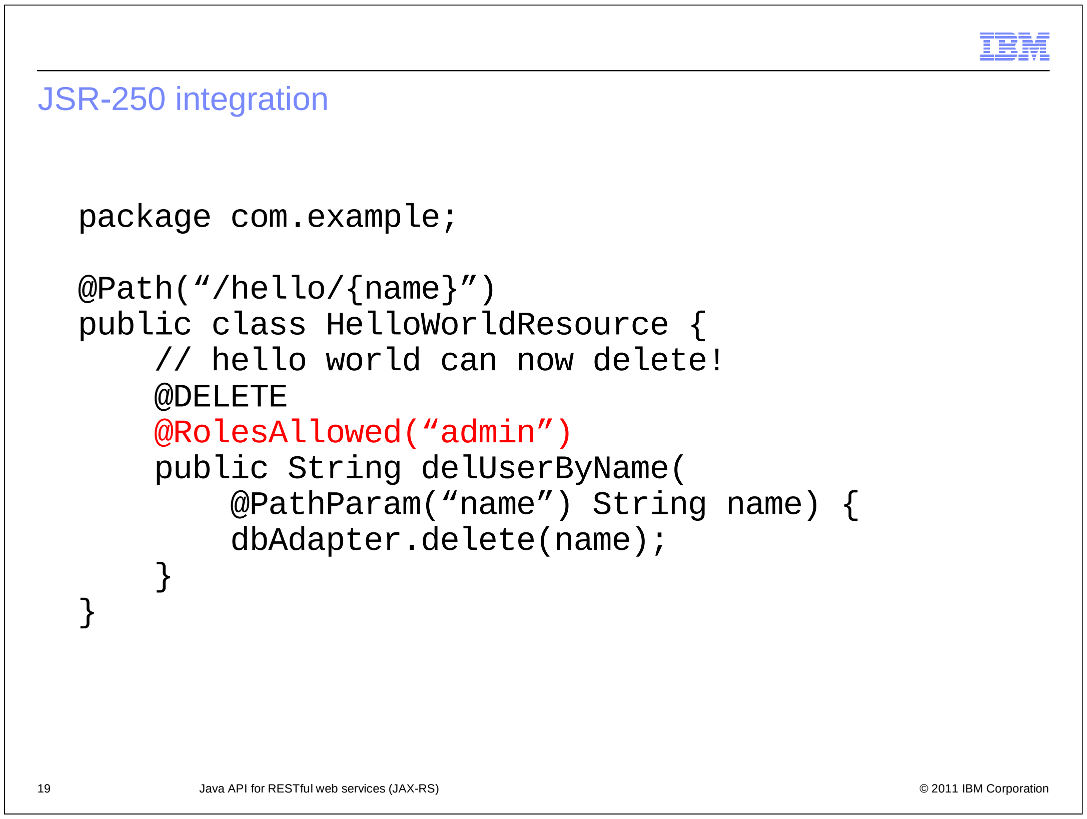
- To enable JCDI support, add a WEB-INF/beans.xml file to your WAR:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemeLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/beans_1_0.xsd">

</beans>
```

18 Java API for RESTful web services (JAX-RS) © 2011 IBM Corporation

EJB support in a WAR is automatically enabled through annotation scanning, but JCDI support requires the beans.xml file in order to start the web beans engine for the WAR module.



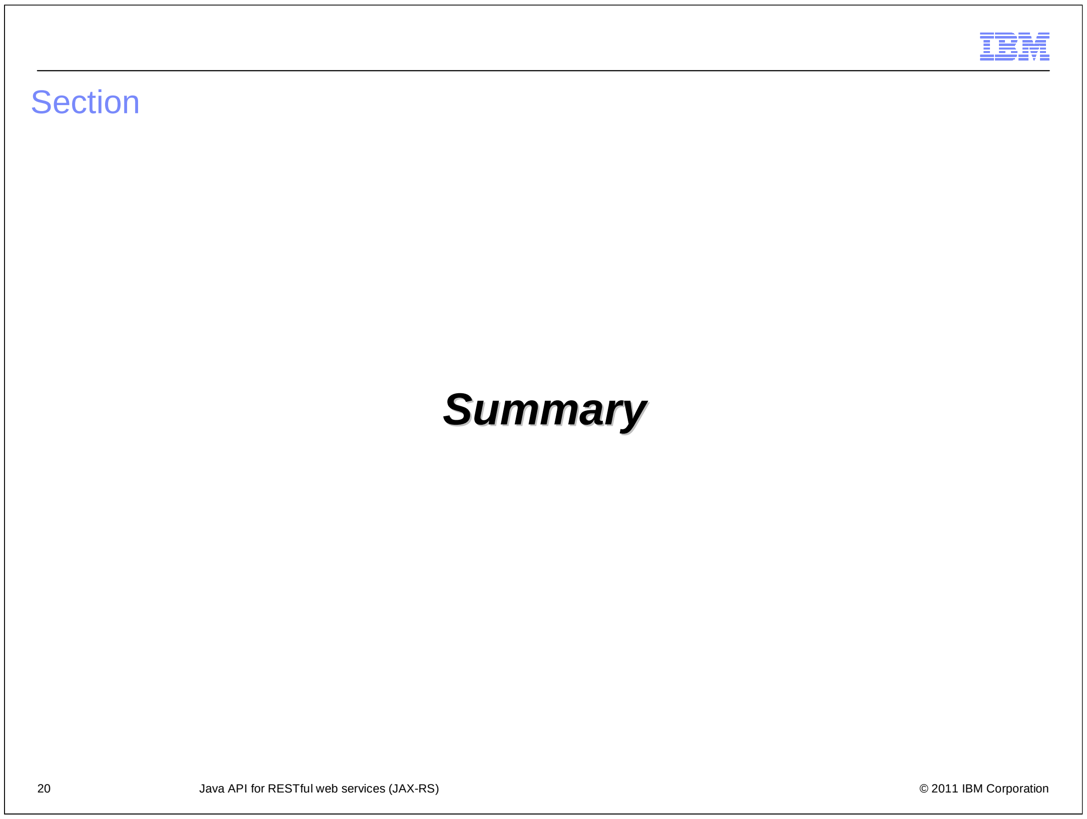
JSR-250 integration

```
package com.example;

@Path(“/hello/{name}”)
public class HelloWorldResource {
    // hello world can now delete!
    @DELETE
    @RolesAllowed(“admin”)
    public String delUserByName(
        @PathParam(“name”) String name) {
        dbAdapter.delete(name);
    }
}
```

19 Java API for RESTful web services (JAX-RS) © 2011 IBM Corporation

The objective of the Java Specification Request (JSR) 250 is to define a set of annotations that address common semantic concepts and therefore can be used by many Java EE components. One example of a common annotation is the @RolesAllowed annotation, which indicates that the specified method or all methods in the EJB can be accessed by users associated with the specified roles. This example indicates that the delUserByName method can only be accessed by users with an administrator role.

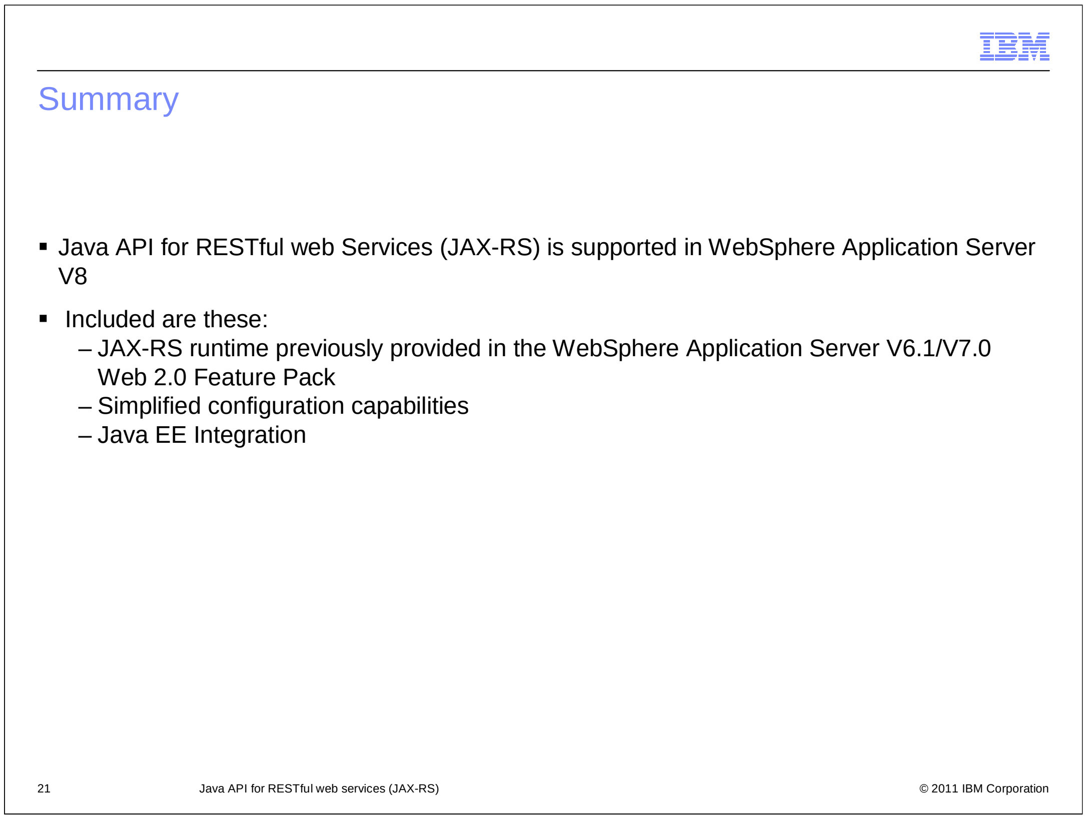


Section

Summary

20 Java API for RESTful web services (JAX-RS) © 2011 IBM Corporation

This section provides a summary of what you have learned in this presentation.

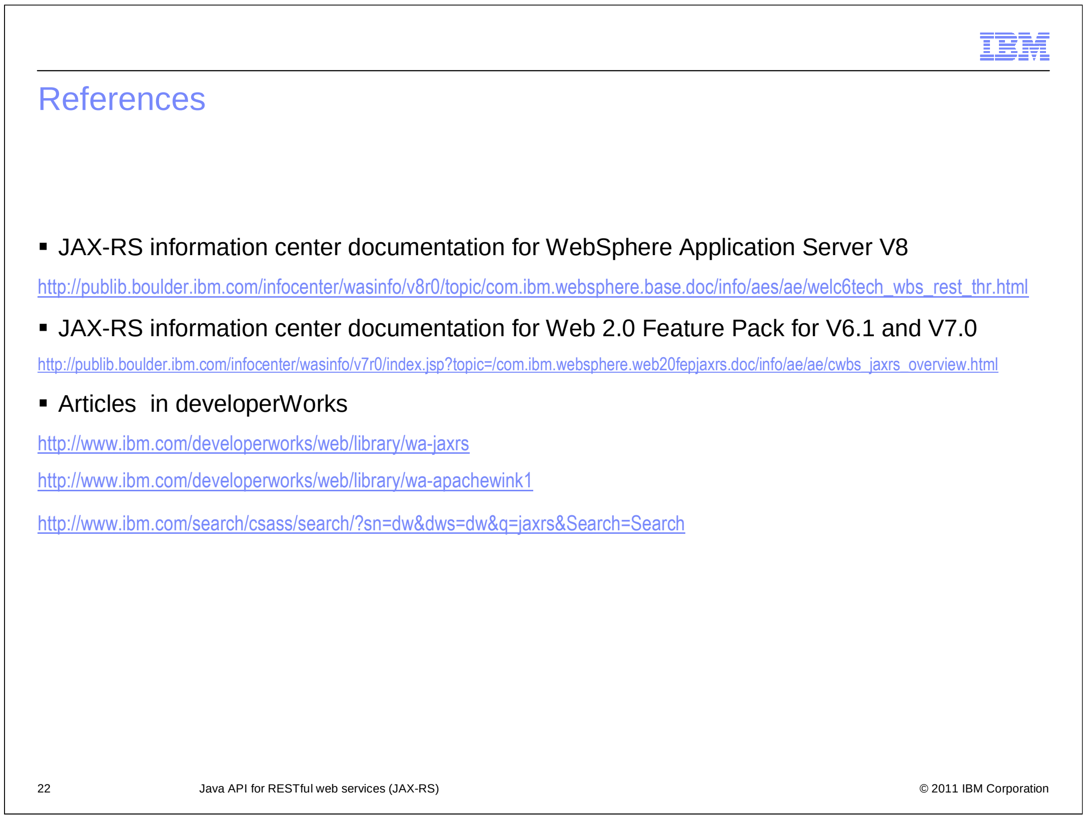


Summary

- Java API for RESTful web Services (JAX-RS) is supported in WebSphere Application Server V8
- Included are these:
  - JAX-RS runtime previously provided in the WebSphere Application Server V6.1/V7.0 Web 2.0 Feature Pack
  - Simplified configuration capabilities
  - Java EE Integration

21 Java API for RESTful web services (JAX-RS) © 2011 IBM Corporation

The acronym JAX-RS stands for Java API for RESTful web Services. Included in WebSphere Application Server V8 are the JAX-RS runtime, simplified configuration capabilities, and Java EE integration.



References

- JAX-RS information center documentation for WebSphere Application Server V8

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/topic/com.ibm.websphere.base.doc/info/aes/ae/welc6tech_wbs_rest_thr.html

- JAX-RS information center documentation for Web 2.0 Feature Pack for V6.1 and V7.0

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.web20fepjaxrs.doc/info/ae/ae/cwbs_jaxrs_overview.html

- Articles in developerWorks

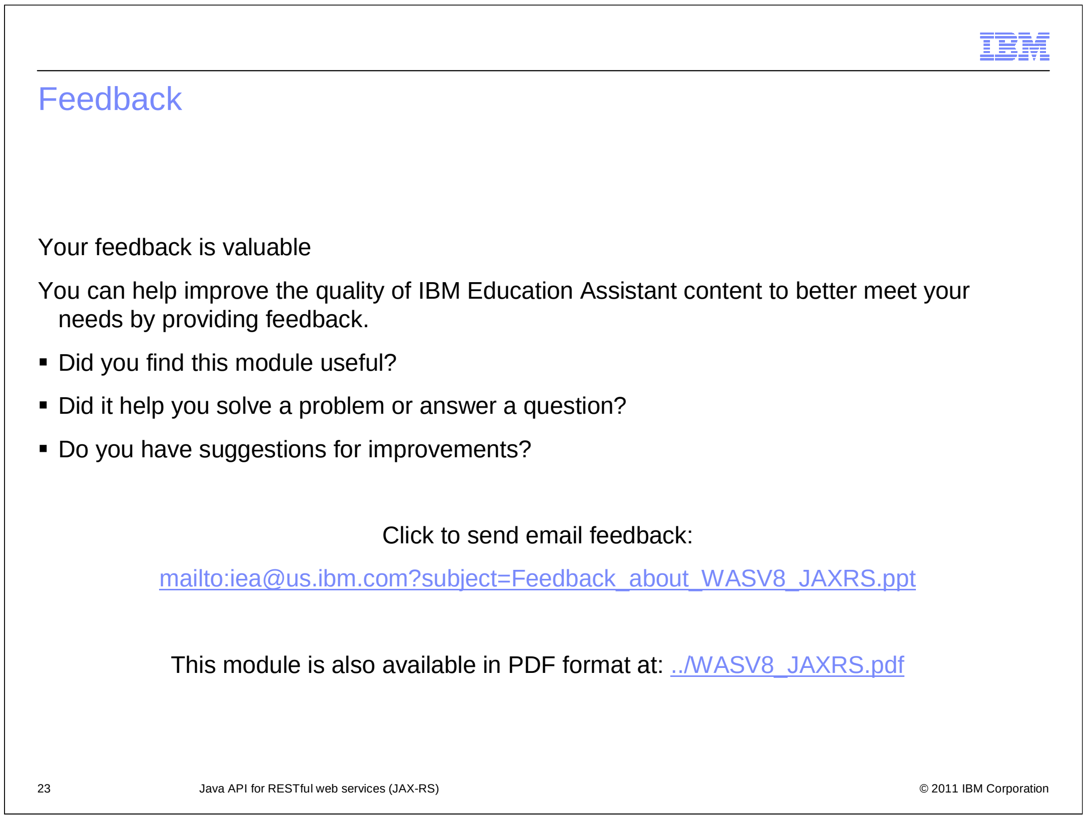
http://www.ibm.com/developerworks/web/library/wa-jaxrs

http://www.ibm.com/developerworks/web/library/wa-apachewink1

http://www.ibm.com/search/csass/search/?sn=dw&dws=dw&q=jaxrs&Search=Search

22 Java API for RESTful web services (JAX-RS) © 2011 IBM Corporation

This slide contains links to useful information.



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 email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WASV8_JAXRS.ppt

This module is also available in PDF format at: ../WASV8_JAXRS.pdf

23 Java API for RESTful web services (JAX-RS) © 2011 IBM Corporation

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

Trademarks, disclaimer, and copyright information