IBM Software Group

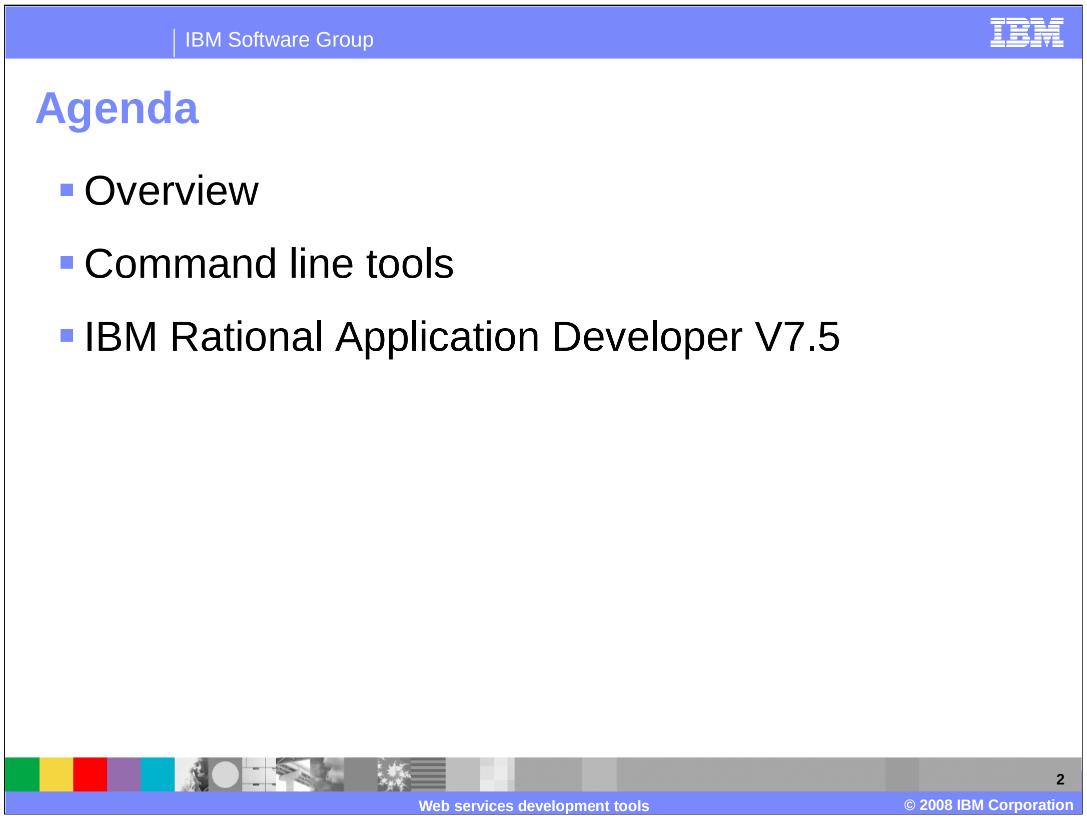
IBM® Rational® Application Developer V7.5

Web services development tools

business on demand.

Updated November 25, 2008

This presentation will cover the Web services development tools for WebSphere® Application Server that are available in Rational Application Developer V7.5.



IBM Software Group

Agenda

- Overview
- Command line tools
- IBM Rational Application Developer V7.5

2

Web services development tools

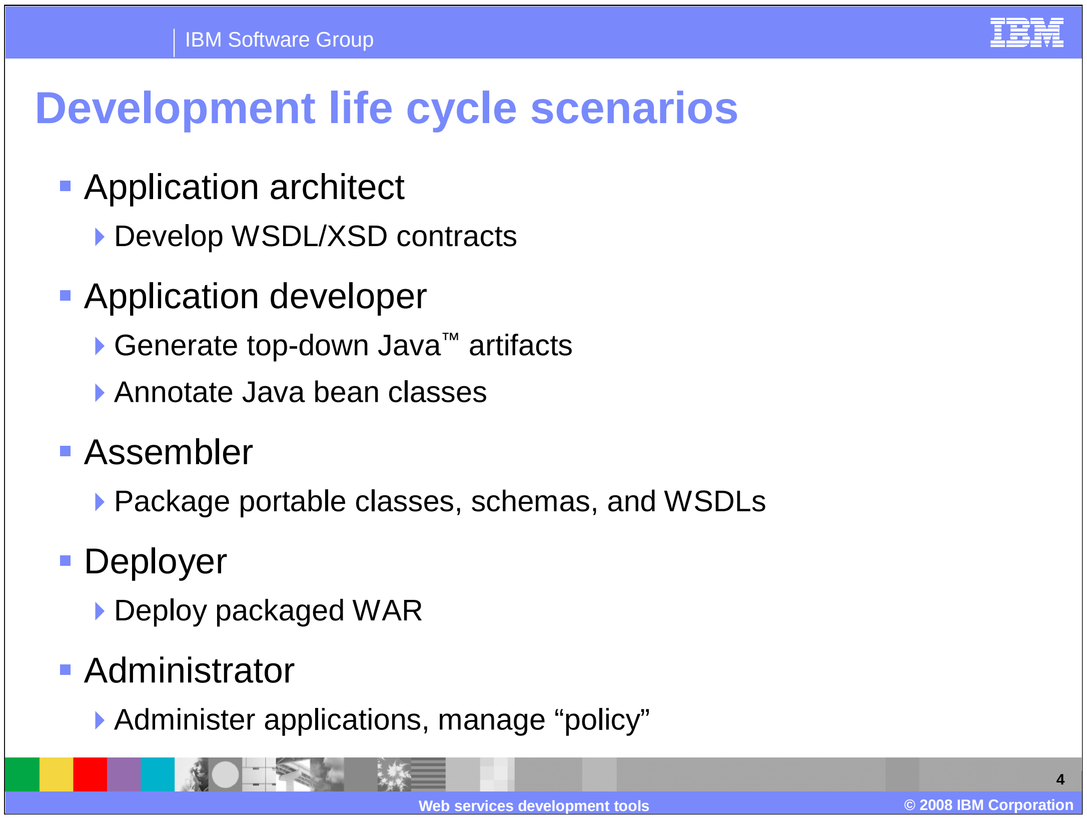
© 2008 IBM Corporation

This presentation will begin with an overview of the tools options available for developing Web services applications for WebSphere Application Server V7. It will then discuss each of those options in more detail. Starting with command line tools, or scripts that can be used to generate Web services and related artifacts, then it will explain how the IBM Rational Application Developer V7.5 can be used to develop Web services.

Section

Overview

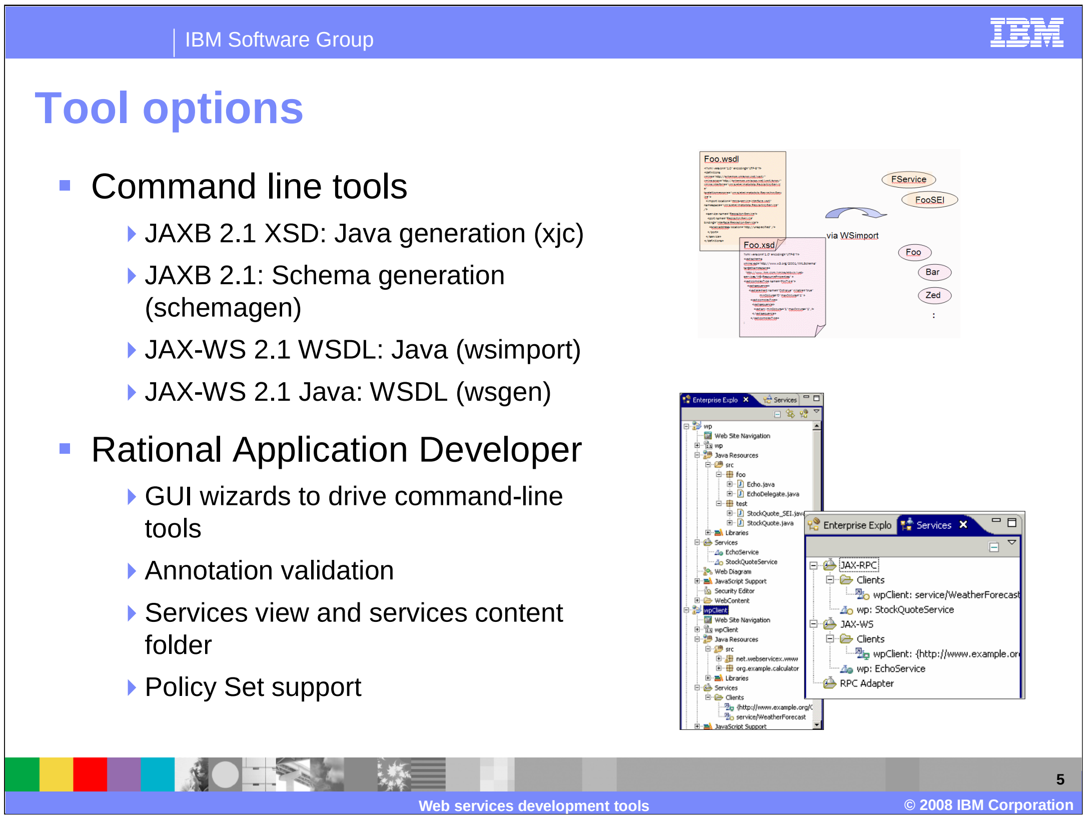
This section will provide an overview of the tools for the feature pack for Web services.



Development life cycle scenarios

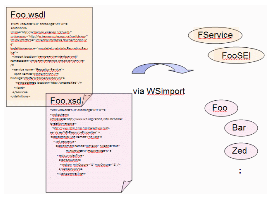
- Application architect
  - Develop WSDL/XSD contracts
- Application developer
  - Generate top-down Java™ artifacts
  - Annotate Java bean classes
- Assembler
  - Package portable classes, schemas, and WSDLs
- Deployer
  - Deploy packaged WAR
- Administrator
  - Administer applications, manage "policy"

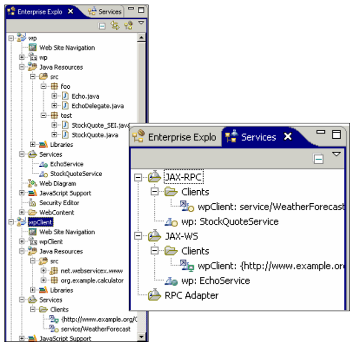
The roles listed above have been identified as key to development and management of Web services in WebSphere Application Server. In many cases, the same people play one or more of these roles. For the architect role, developing WSDL and XSD files, their role actually gets simplified. Since JAXB 2.0 is used as the mapping language, and handles all of XSD schema, it is easier for architects to develop services that use the full breadth of XSD. The application developer also has several new tools available to create JAX-WS based Web services. Administrators have also had their roles improved and simplified through the introduction of policies.



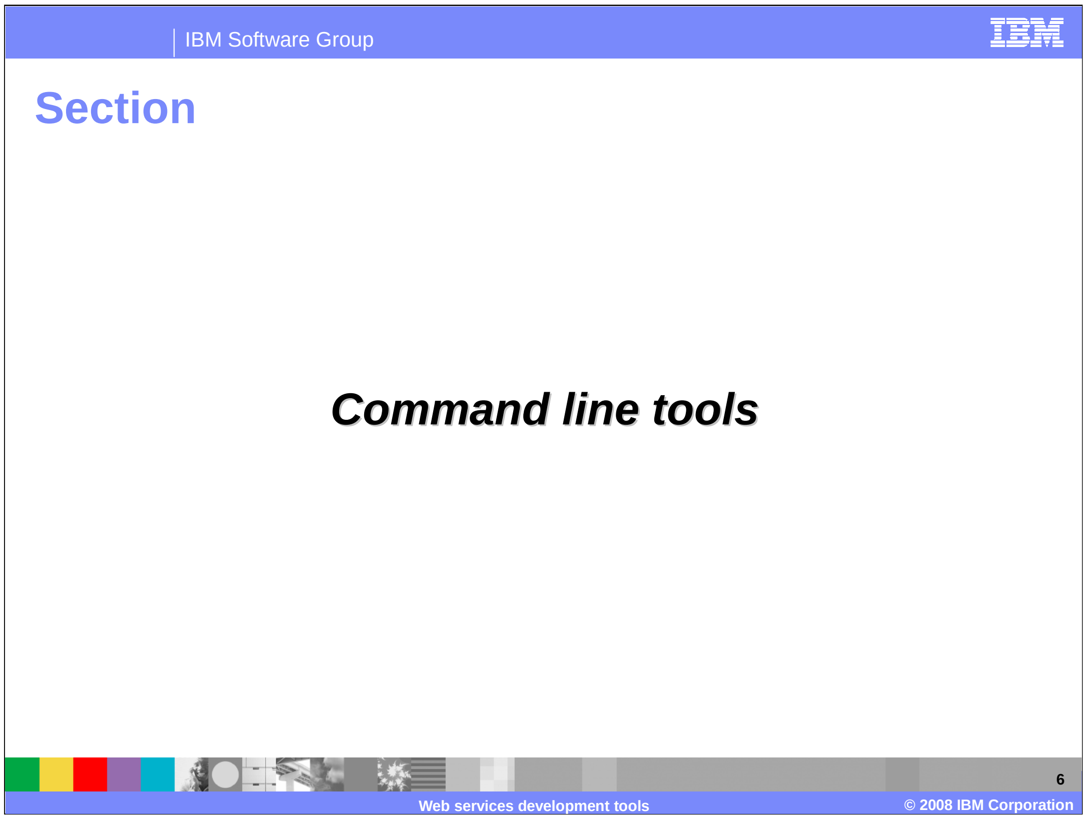
Tool options

- Command line tools
  - JAXB 2.1 XSD: Java generation (xjc)
  - JAXB 2.1: Schema generation (schemagen)
  - JAX-WS 2.1 WSDL: Java (wsimport)
  - JAX-WS 2.1 Java: WSDL (wsgen)
- Rational Application Developer
  - GUI wizards to drive command-line tools
  - Annotation validation
  - Services view and services content folder
  - Policy Set support





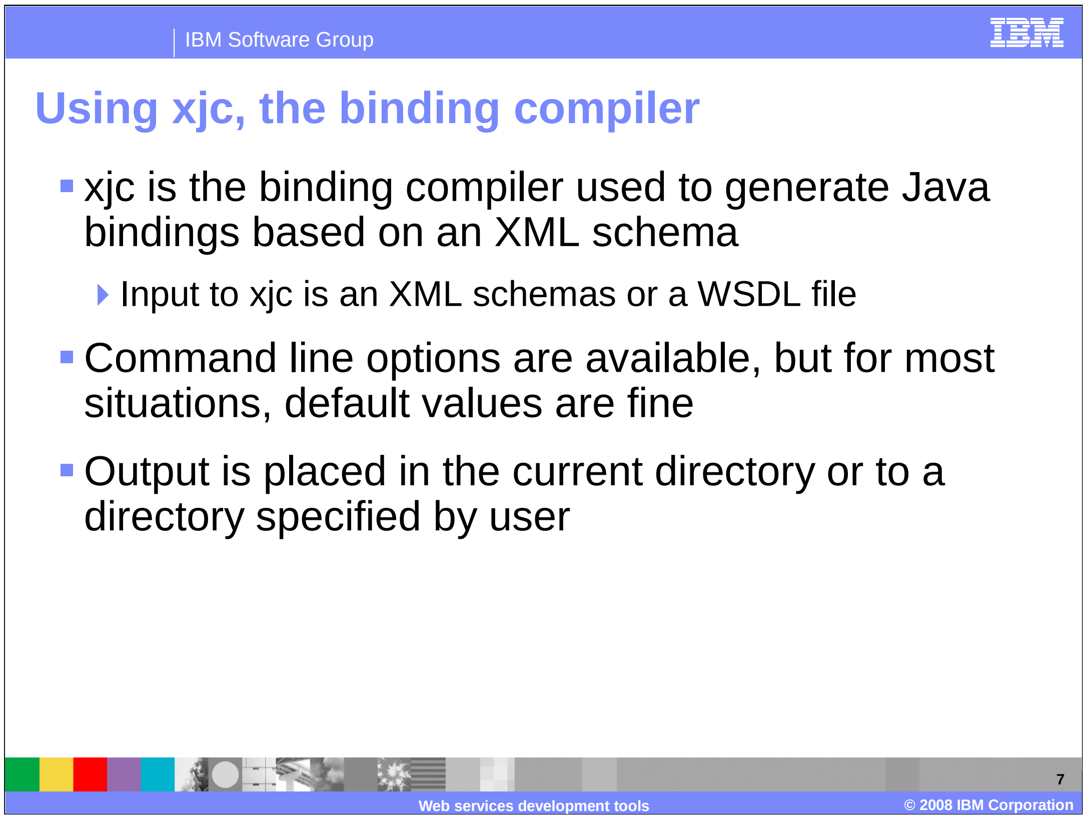
For the application developer, WebSphere Application Server V7 provides a set of tools to help make development easier. At the most basic level there is command line tools to generate various artifacts. An XJC command can be used to generate Java artifacts based on a JAXB 2.1 XSD definition or also from a WSDL file. There are also WSIMPORT and WSGEN commands for top down and bottom up Web services development. There are also updates to the Rational application developer, with additions to wizards for the creation of Web services. These have been extended to support JAX-WS and JAXB based Web services with; annotation validation, graphical wizards, publishing tools for developed Web services and a Jython debugger for scripting.



Section

Command line tools

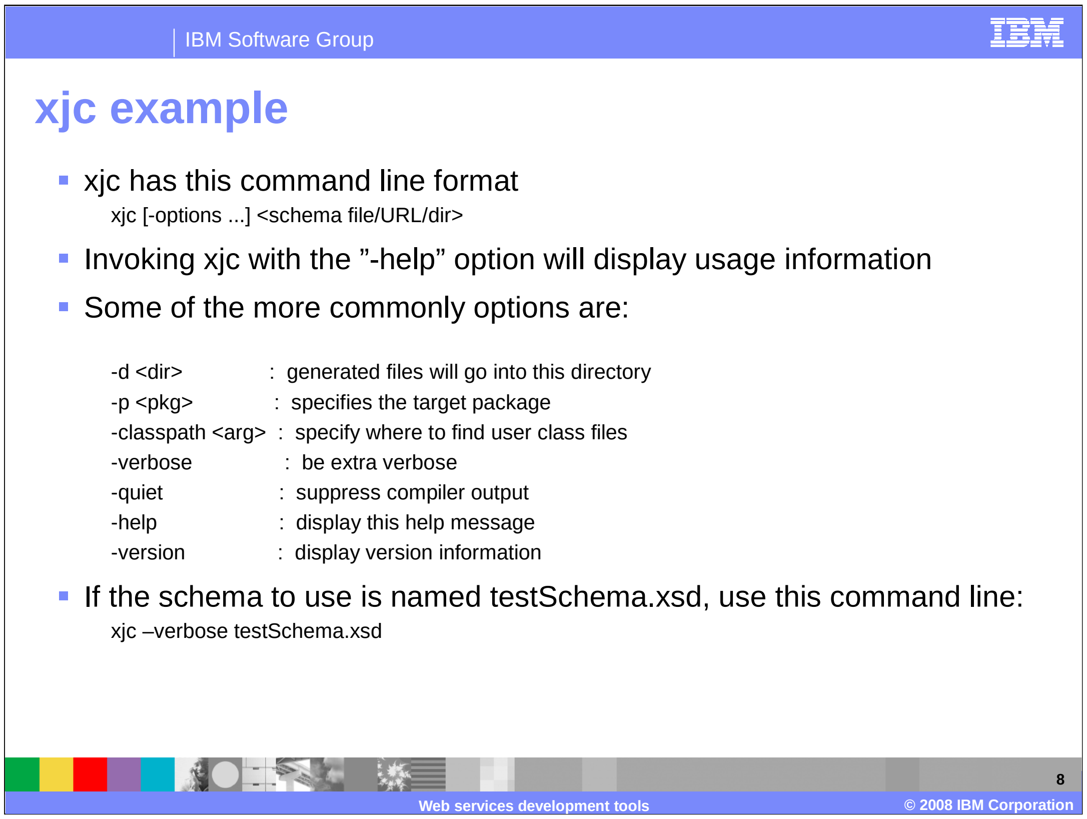
The next section discusses the Web services command line tools for WebSphere Application Server V7.



Using xjc, the binding compiler

- xjc is the binding compiler used to generate Java bindings based on an XML schema
  - Input to xjc is an XML schemas or a WSDL file
- Command line options are available, but for most situations, default values are fine
- Output is placed in the current directory or to a directory specified by user

The XJC command can be used to compile the Java bindings based on an XML schema. XML schemas describe the data elements and relationships in an XML document. After a data mapping or binding exists, XML documents can be converted to and from Java objects. Generate fully annotated Java classes from an XML schema file by using the XJC command-line tool. The schema compiler is located in the bin directory under the application server root. The output will be placed in the current directory or one specified by the user.



xjc example

- xjc has this command line format

```
xjc [-options ...] <schema file/URL/dir>
```

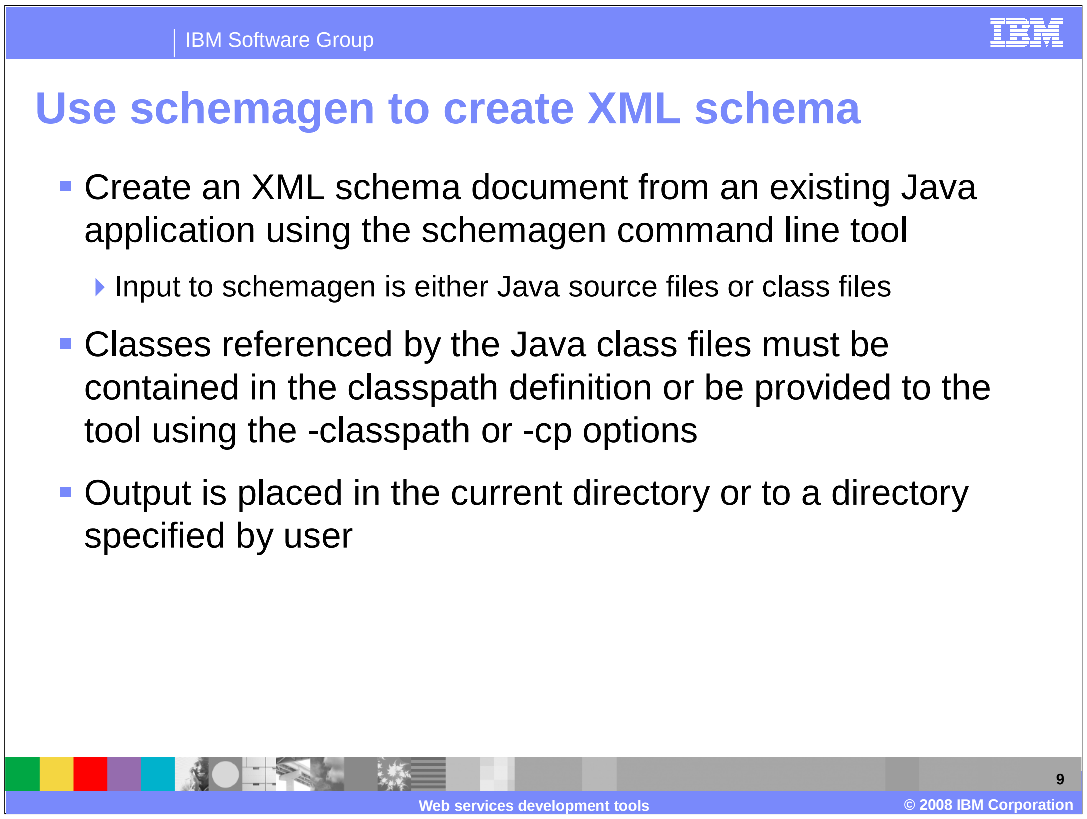
- Invoking xjc with the "-help" option will display usage information
- Some of the more commonly options are:

```
-d <dir>            :  generated files will go into this directory
-p <pkg>            :  specifies the target package
-classpath <arg>  :  specify where to find user class files
-verbose              :  be extra verbose
-quiet                 :  suppress compiler output
-help                  :  display this help message
-version              :  display version information
```

- If the schema to use is named testSchema.xsd, use this command line:

```
xjc –verbose testSchema.xsd
```

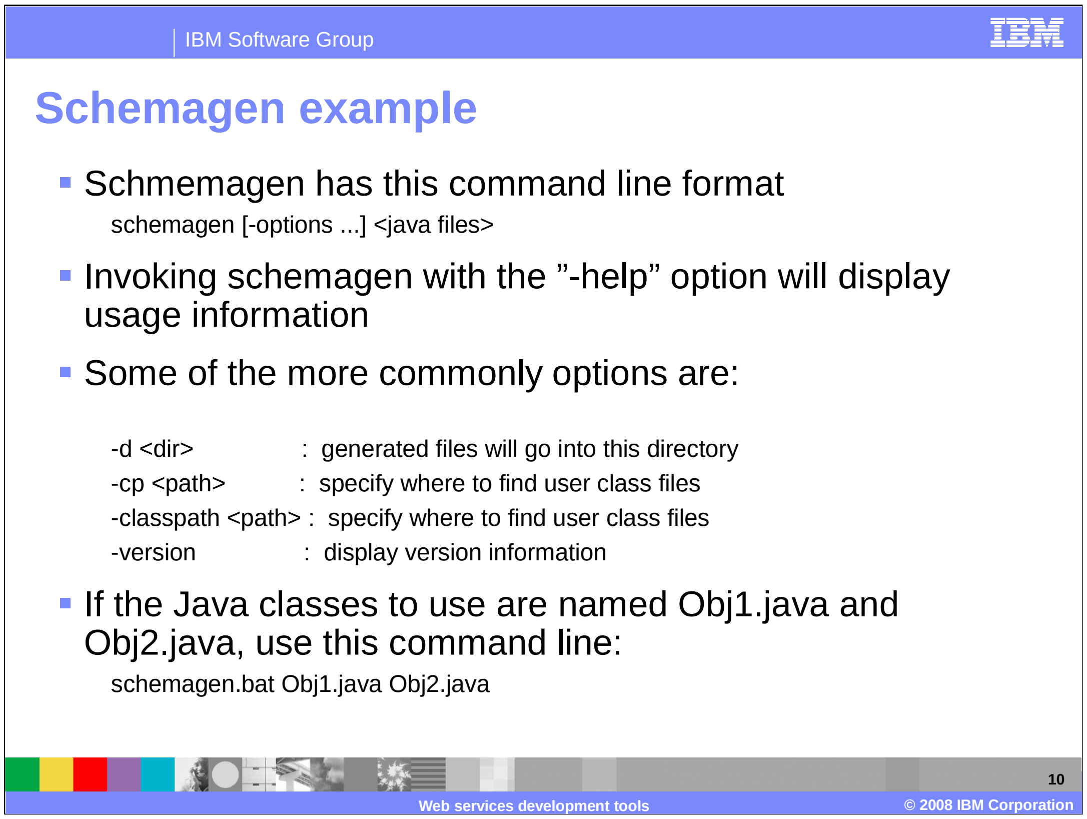
The above information shows how to use the XJC command. Use the –help option to display usage information on the command line. An example is given with a sample test schema.



Use schemagen to create XML schema

- Create an XML schema document from an existing Java application using the schemagen command line tool
  - Input to schemagen is either Java source files or class files
- Classes referenced by the Java class files must be contained in the classpath definition or be provided to the tool using the -classpath or -cp options
- Output is placed in the current directory or to a directory specified by user

An XML schema can be documented from existing Java classes, which represent the data elements of an application, using the JAXB schema generator or schemagen command-line tool. The JAXB schema generator processes either Java source files or class files. Annotations within the Java classes provide the capability to customize the default mappings from existing Java classes to the generated schema components. The XML schema file and the annotated Java class files contain all the necessary information that JAXB requires to parse the XML documents for serialization and deserialization.



Schemagen example

- Schmemagen has this command line format

  schemagen [-options ...] <java files>

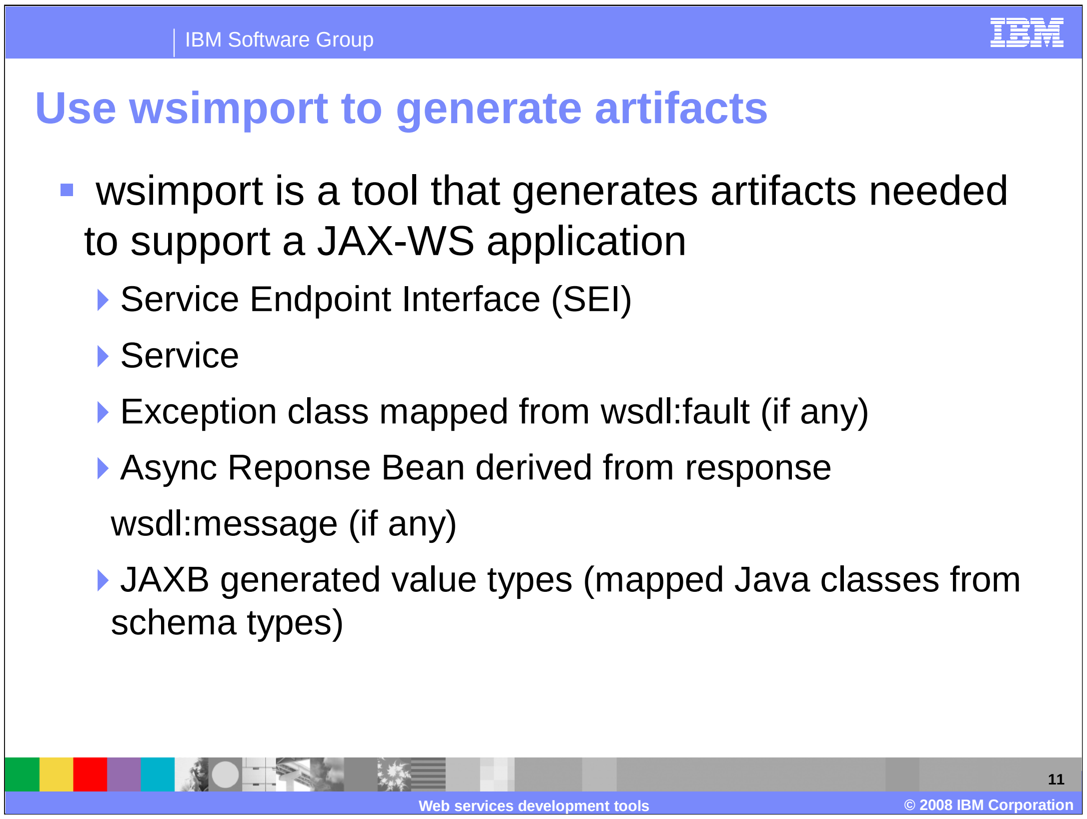
- Invoking schemagen with the ”-help” option will display usage information
- Some of the more commonly options are:

```
-d <dir>              :  generated files will go into this directory
-cp <path>            :  specify where to find user class files
-classpath <path> :  specify where to find user class files
-version              :  display version information
```

- If the Java classes to use are named Obj1.java and Obj2.java, use this command line:

  schemagen.bat Obj1.java Obj2.java

The above information shows how to use the schemagen command. Use the –help option to display usage information on the command line. An example is given with sample test files.

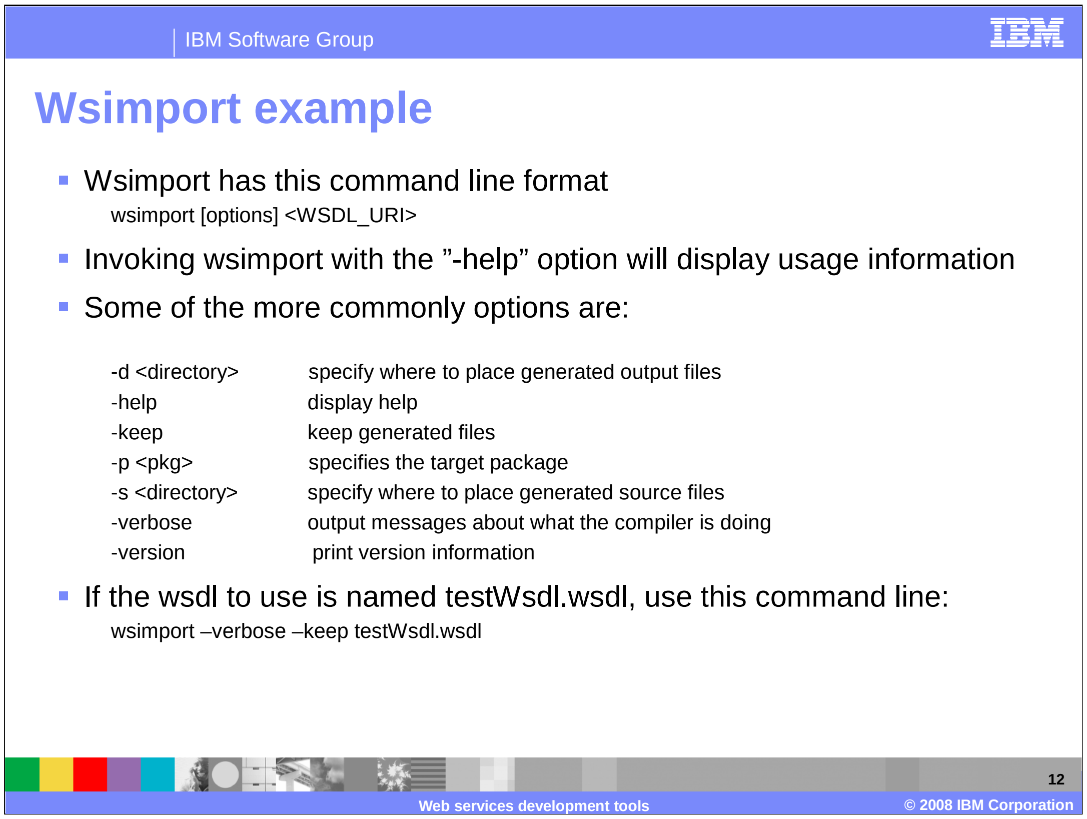


Use wsimport to generate artifacts

- wsimport is a tool that generates artifacts needed to support a JAX-WS application
  - Service Endpoint Interface (SEI)
  - Service
  - Exception class mapped from wsdl:fault (if any)
  - Async Reponse Bean derived from response

    wsdl:message (if any)
  - JAXB generated value types (mapped Java classes from schema types)

The wsimport command-line tool processes an existing Web Services Description Language (WSDL) file and creates the required portable artifacts for developing JAX-WS based Web service applications. The wsimport command-line tool supports the top-down approach to developing JAX-WS Web services, when a WSDL is used to generate the various artifacts, including the service endpoint interface, the service class, an exception class defined by the WSDL fault element, an asynchronous response bean based on the WSDL message element, and the JAXB generated types.



Wsimport example

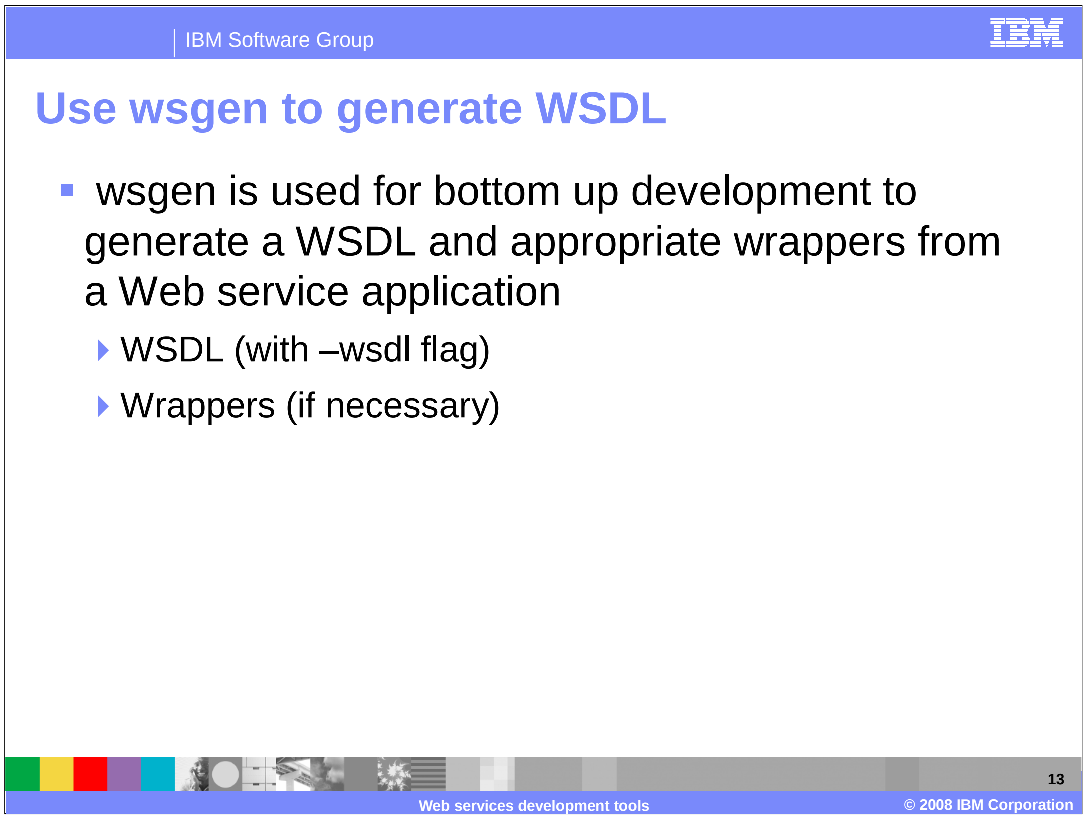
- Wsimport has this command line format
  wsimport [options] <WSDL_URI>
- Invoking wsimport with the ”-help” option will display usage information
- Some of the more commonly options are:

-d <directory>	specify where to place generated output files
-help	display help
-keep	keep generated files
-p <pkg>	specifies the target package
-s <directory>	specify where to place generated source files
-verbose	output messages about what the compiler is doing
-version	print version information

- If the wsdl to use is named testWsdl.wsdl, use this command line:

```
wsimport –verbose –keep testWsdl.wsdl
```

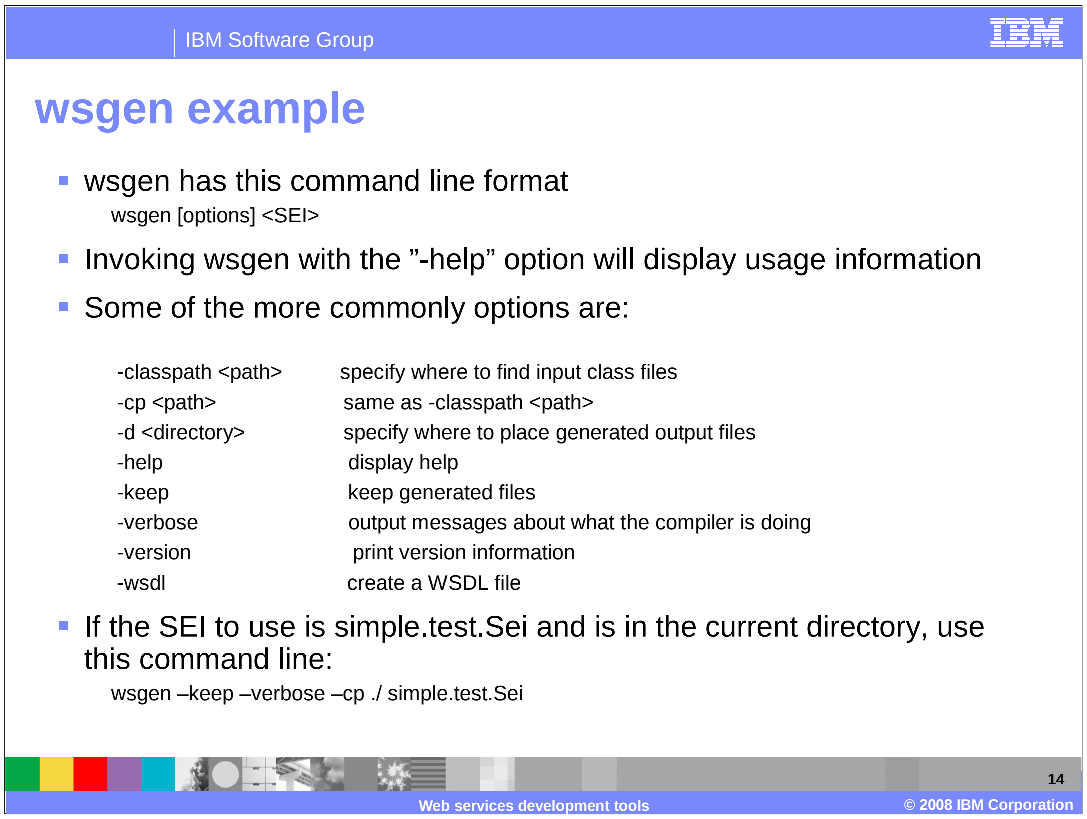
The above information shows how to use the wsimport command. Use the –help option to display usage information on the command line. An example is given with a sample WSDL, notice the use of the –keep option to retain the generated files.



Use wsgen to generate WSDL

- wsgen is used for bottom up development to generate a WSDL and appropriate wrappers from a Web service application
  - WSDL (with –wsdl flag)
  - Wrappers (if necessary)

The wsgen command-line tool generates the necessary portable artifacts required for JAX-WS applications when starting from Java code. This tool will generate a WSDL file only when specified. When using a bottom-up approach to develop JAX-WS Web services, creating a Web service from a service endpoint implementation, use the **wsgen** tool to generate the required artifacts. The wsgen tool accepts a properly annotated service endpoint implementation using the @WebService annotation as input and generates artifacts for the WSDL and the JAXB wrappers that may be necessary.



wsgen example

- wsgen has this command line format

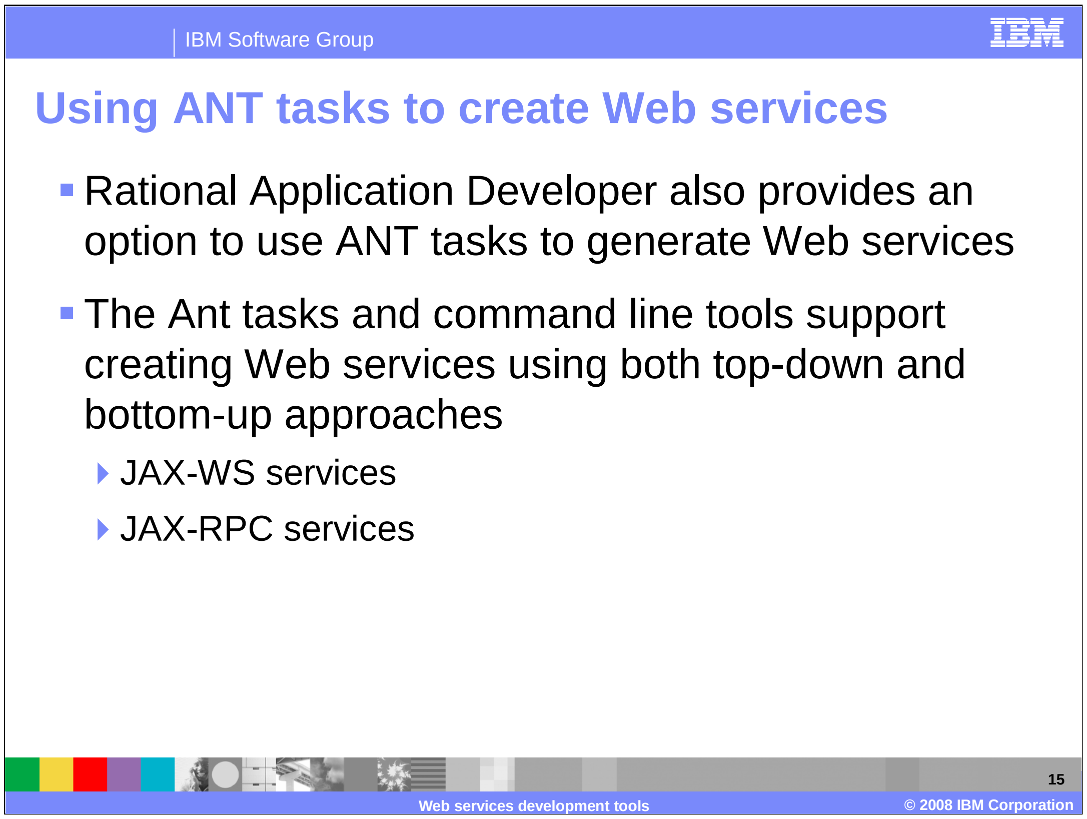
  wsgen [options] <SEI>
- Invoking wsgen with the ”-help” option will display usage information
- Some of the more commonly options are:

-classpath <path>	specify where to find input class files
-cp <path>	same as -classpath <path>
-d <directory>	specify where to place generated output files
-help	display help
-keep	keep generated files
-verbose	output messages about what the compiler is doing
-version	print version information
-wsdl	create a WSDL file

- If the SEI to use is simple.test.Sei and is in the current directory, use this command line:

  wsgen –keep –verbose –cp ./ simple.test.Sei

The above information shows how to use the wsgen command. Use the –help option to display usage information on the command line. An example is given with a sample service endpoint interface, notice the use of the –keep option to retain the generated files.

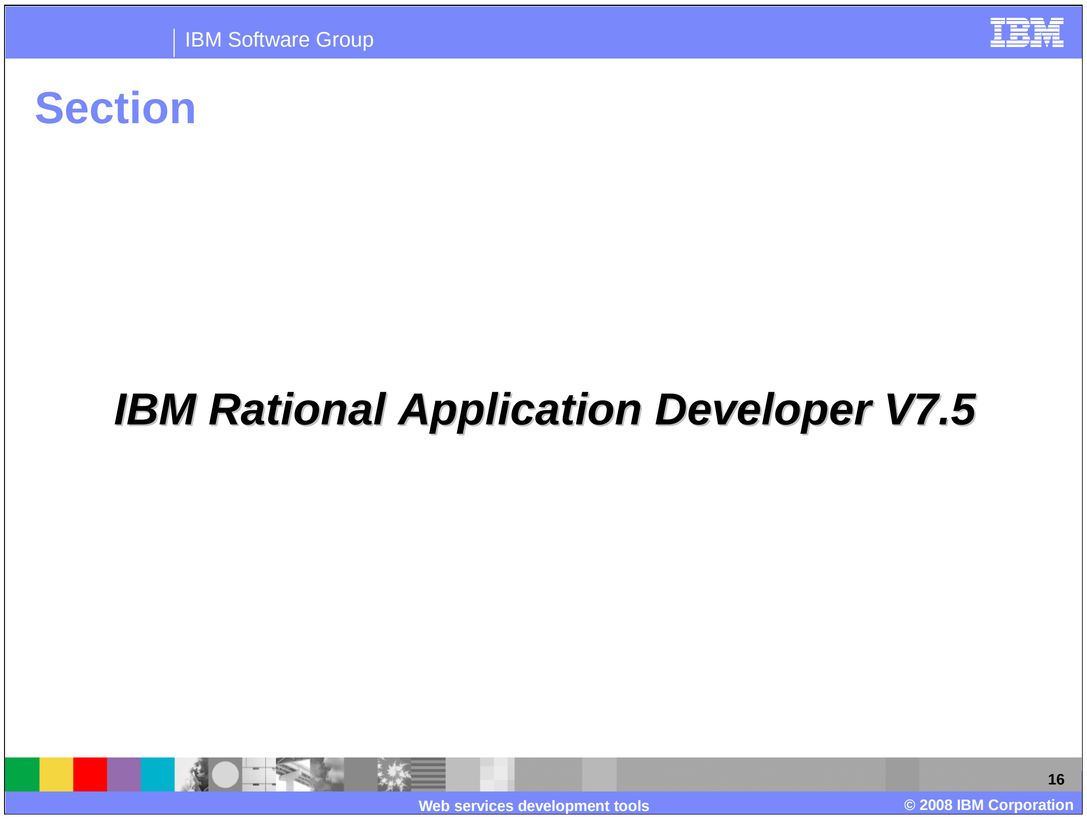


Using ANT tasks to create Web services

- Rational Application Developer also provides an option to use ANT tasks to generate Web services
- The Ant tasks and command line tools support creating Web services using both top-down and bottom-up approaches
  - JAX-WS services
  - JAX-RPC services

If you prefer not to use the Web service wizards, you can use Ant tasks or command line tools to create Web services using the IBM® WebSphere® runtime environments or Axis runtime environment.

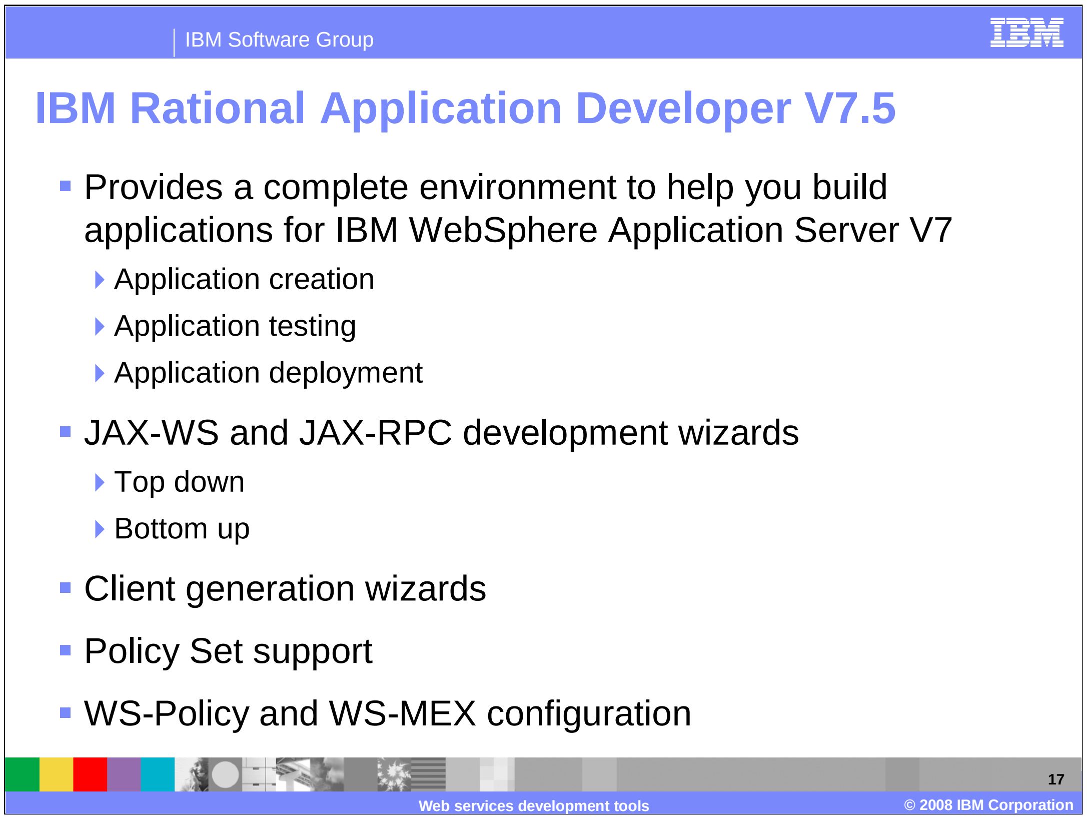
The Ant tasks and command line tools support creating Web services using both top-down and bottom-up approaches. Once you have created your Web service, you can then deploy it to a server, test it, and publish it as a business entity or business service.



Section

IBM Rational Application Developer V7.5

The next section explains the addition in IBM Rational Application Developer V7.5.


IBM Software Group

IBM Rational Application Developer V7.5

- Provides a complete environment to help you build applications for IBM WebSphere Application Server V7
  - Application creation
  - Application testing
  - Application deployment
- JAX-WS and JAX-RPC development wizards
  - Top down
  - Bottom up
- Client generation wizards
- Policy Set support
- WS-Policy and WS-MEX configuration

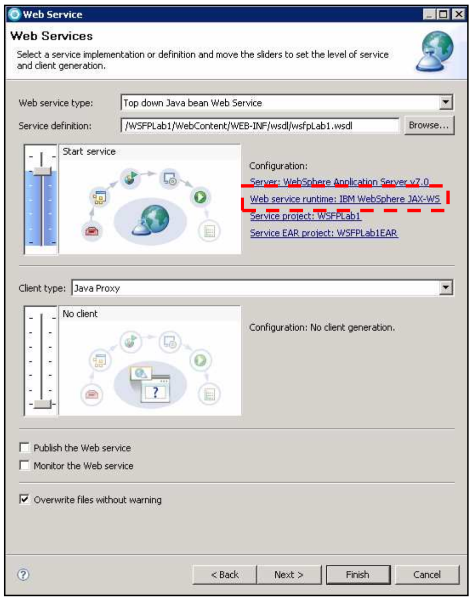
17

Web services development tools © 2008 IBM Corporation

IBM Rational Application Developer version 7.5 offers a complete environment to develop and build applications for WebSphere Application Server version 7; including the ability to create, test and deploy applications. Development wizards are provided to develop both JAX-WS and JAX-RPC applications from WSDL documents or from Java implementations. Client generation wizards are also provided, as are tools to work with policy set configurations in Rational Application Developer. Also tools to configure WS-Policy and WS-Metadata Exchange are provided in this release.

Web services creation wizard

- Select the Web service runtime
  - IBM WebSphere JAX-RPC
  - IBM WebSphere JAX-WS
- Configure the client that should be created



Select a service implementation or definition and move the sliders to set the level of service and client generation.

Web service type: Top down Java bean Web Service

Service definition: /WSFPLab1/WebContent/WEB-INF/wsdl/wsfpLab1.wsdl

Configuration:
Server: WebSphere Application Server v7.0
Web service runtime: IBM WebSphere JAX-WS
Service project: WSFPLab1
Service EAR project: WSFPLab1EAR

Client type: Java Proxy

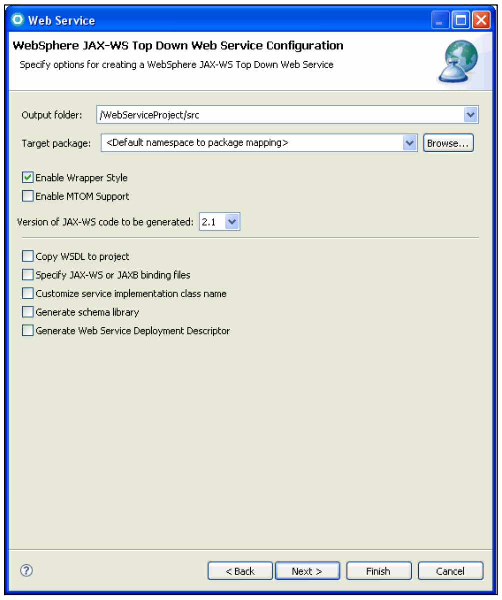
Configuration: No client generation.

☐ Publish the Web service
☐ Monitor the Web service
☑ Overwrite files without warning

When creating a new Web service using the Web services creation wizard, new options have been added. One of these is for the Web Service runtime to use, this can be changed between IBM WebSphere JAX-RPC and IBM WebSphere JAX-WS. The type of client that should be created can also be chosen.

Web service creation wizard

- Select options related to the Web service
- Option to generate the Web service deployment descriptor



Web Service

WebSphere JAX-WS Top Down Web Service Configuration

Specify options for creating a WebSphere JAX-WS Top Down Web Service

Output folder: /WebServiceProject/src

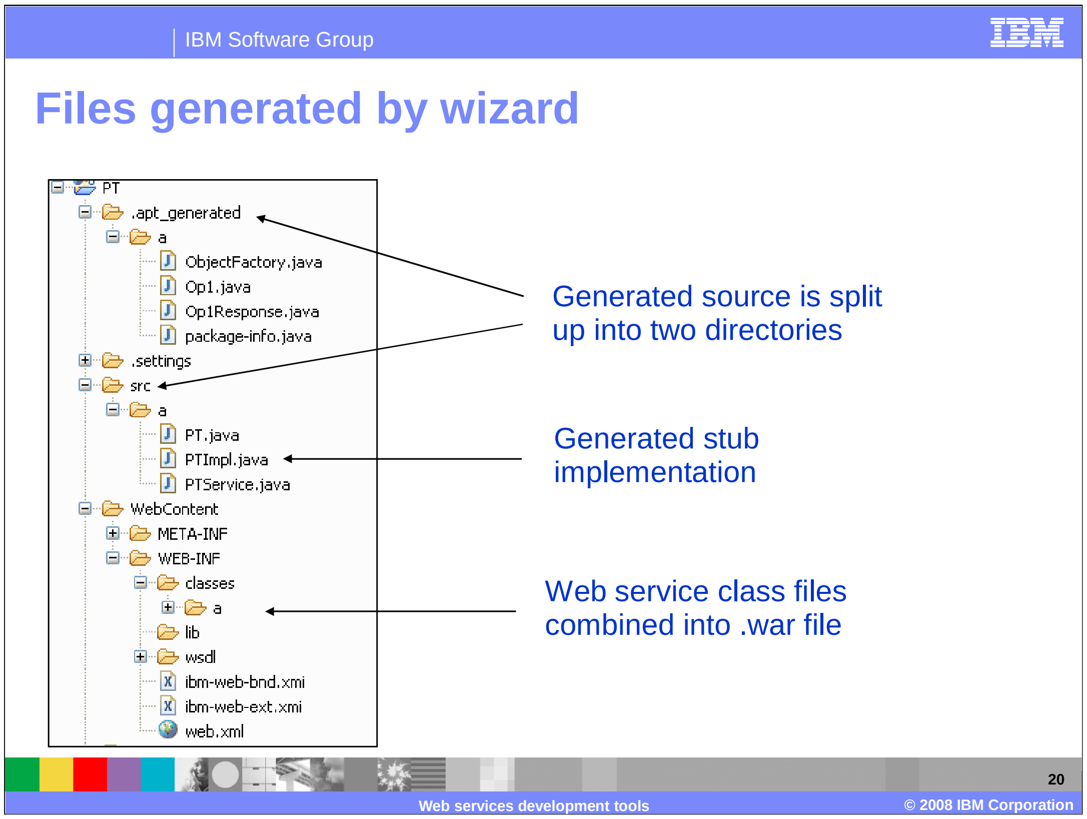
Target package: <Default namespace to package mapping>

[✓] Enable Wrapper Style
[ ] Enable MTOM Support

Version of JAX-WS code to be generated: 2.1

[ ] Copy WSDL to project
[ ] Specify JAX-WS or JAXB binding files
[ ] Customize service implementation class name
[ ] Generate schema library
[ ] Generate Web Service Deployment Descriptor

Further options can be configured for a JAX-WS Web service. These include enabling the wrapper style for the Web Service, copying the WSDL to the project, and specifying JAX-WS or JAX-B binding files. The option to generate the schema library or the Web service deployment descriptor have also been added in Rational Application Developer V7.5.



IBM Software Group

Files generated by wizard

```
PT
  .apt_generated
    a
      ObjectFactory.java
      Op1.java
      Op1Response.java
      package-info.java
  .settings
  src
    a
      PT.java
      PTImpl.java
      PTService.java
  WebContent
    META-INF
    WEB-INF
      classes
        a
      lib
      wsdl
      ibm-web-bnd.xmi
      ibm-web-ext.xmi
      web.xml
```

Generated source is split up into two directories

Generated stub implementation

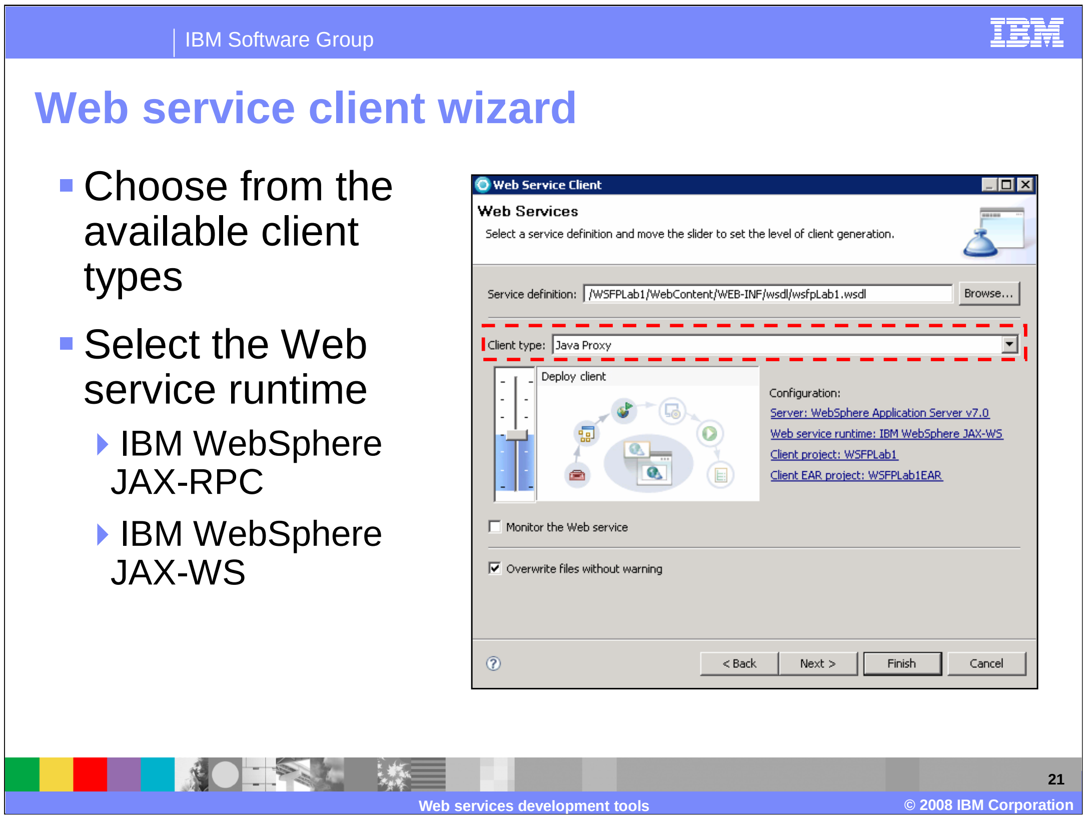
Web service class files combined into .war file

20

Web services development tools

© 2008 IBM Corporation

This picture illustrates the location of various files when generated by the tools for a bottom up Web service. The source files are split between the SRC folder and a folder for portable Java artifacts for the Web service. The stub implementation is placed in the SRC folder. The class files are combined into a single .WAR file that can be exported and deployed.



Web service client wizard

- Choose from the available client types
- Select the Web service runtime
  - IBM WebSphere JAX-RPC
  - IBM WebSphere JAX-WS

Web Service Client

Web Services

Select a service definition and move the slider to set the level of client generation.

Service definition: /WSFPLab1/WebContent/WEB-INF/wsdl/wsfpLab1.wsdl

Client type: Java Proxy

Deploy client

Configuration:

Server: WebSphere Application Server v7.0

Web service runtime: IBM WebSphere JAX-WS

Client project: WSFPLab1

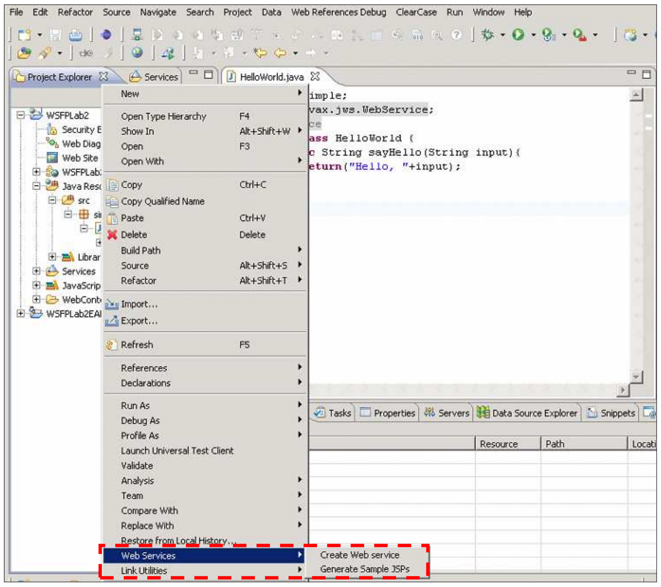
Client EAR project: WSFPLab1EAR

[ ] Monitor the Web service

[✓] Overwrite files without warning

There is also a client generation wizard for IBM WebSphere JAX-WS and IBM WebSphere JAX-RPC clients. The additional options allow a user to enable asynchronous invocation of the client, to specify JAX-WS or JAX-B binding files, and to customize the client proxy class name.

Bottom up mapping and JSP wizards

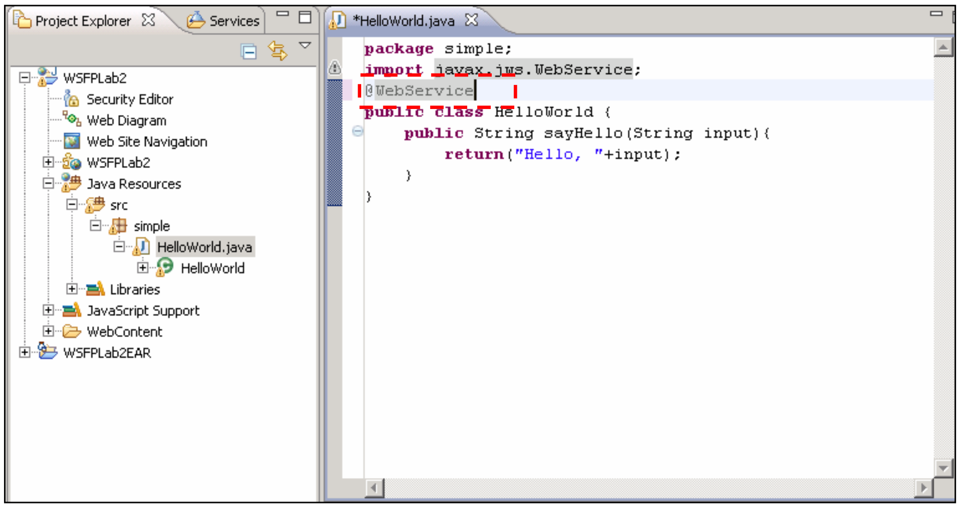


```
imple;
vax.jws.WebService;
ce
ass HelloWorld {
c String sayHello(String input){
eturn("Hello, "+input);
```

There are also tools provided for bottom up development as well. Right click a class to be able to create a Web service from that class, or to generate sample JSPs to test that class.

Annotation validation

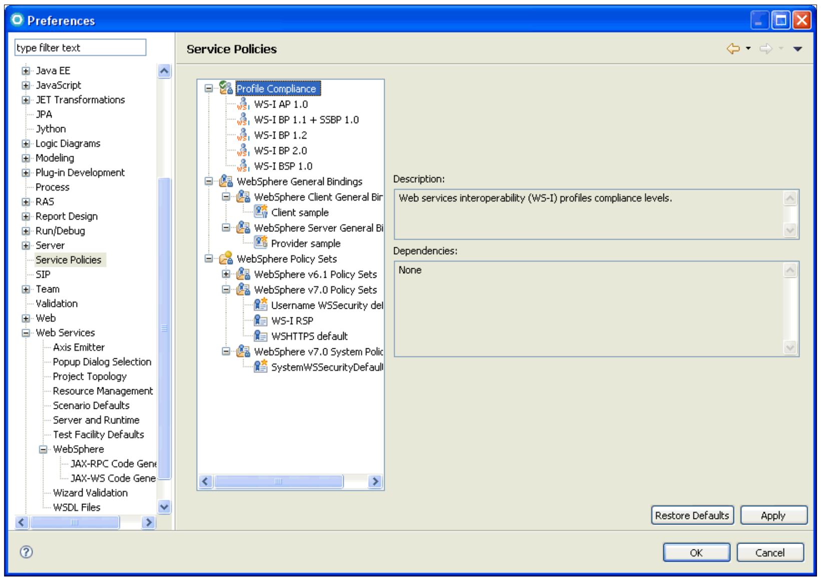
- On-the-fly annotation validation



```
package simple;
import javax.jws.WebService;
@WebService
public class HelloWorld {
    public String sayHello(String input){
        return("Hello, "+input);
    }
}
```

As annotations are added to the source, the AST will validate those annotations as they are typed. The file doesn't have to be saved to be validated. Tips will be provided as the annotations are typed as well.

Policy set support

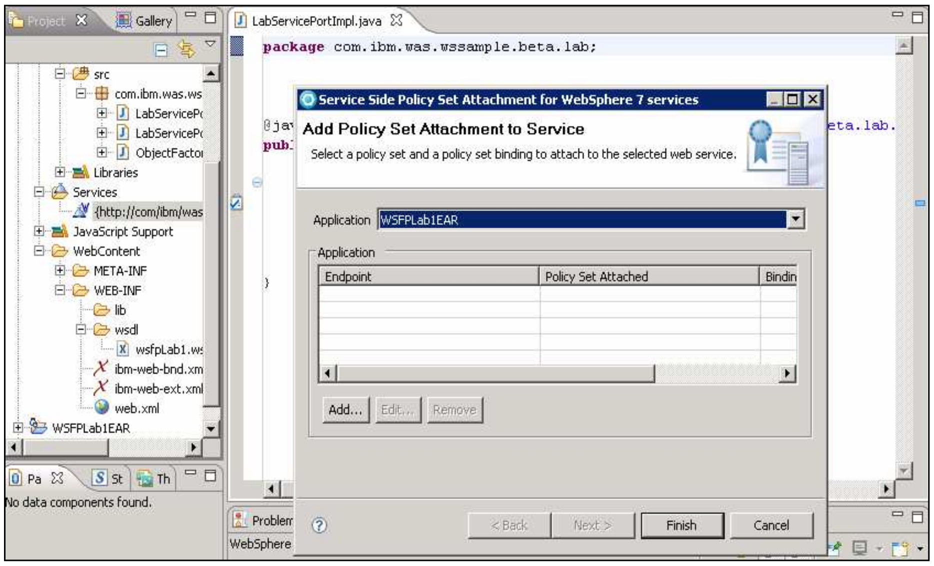


Description:
Web services interoperability (WS-I) profiles compliance levels.

Dependencies:
None

Rational Application Developer also has the abilities to work with policy sets. Policy sets can be imported in the preferences view under quality of services. This allows a developer to work with the same policy sets as are used in the production environment.

Policy set support


Service Side Policy Set Attachment for WebSphere 7 services

Add Policy Set Attachment to Service

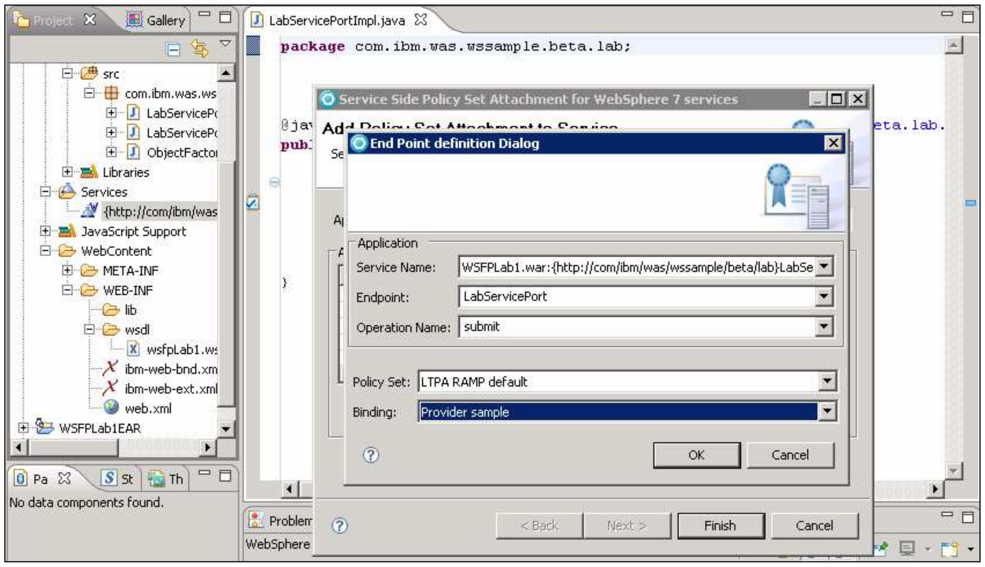
Select a policy set and a policy set binding to attach to the selected web service.

Application: WSFPLab1EAR

Endpoint	Policy Set Attached	Bindin

Policy sets can be attached to Web services using a wizard. All policy sets attached to the service will be listed, and more can be added or removed.

Policy set support



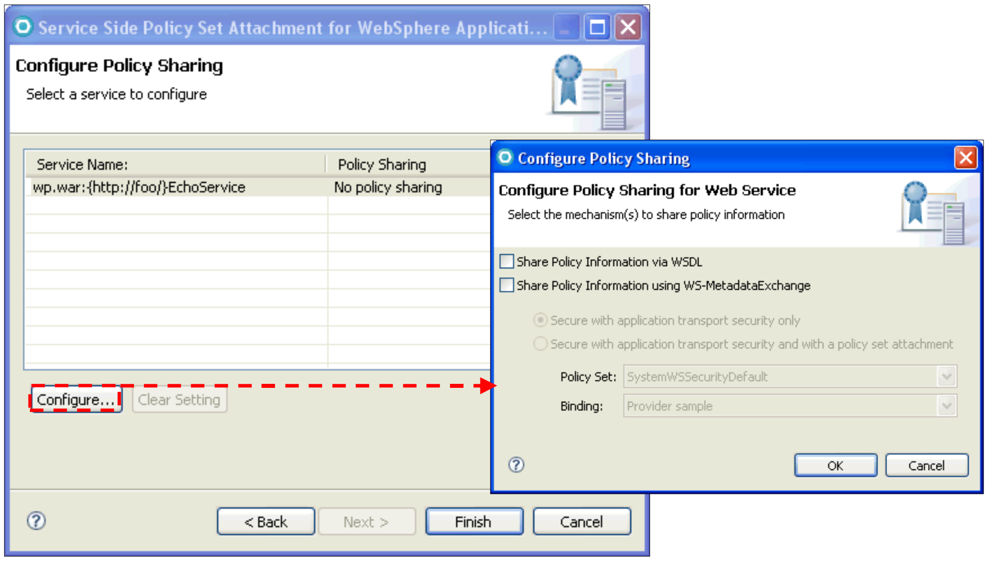
```
package com.ibm.was.wssample.beta.lab;
```

Application

Service Name:	WSFPLab1.war:{http://com/ibm/was/wssample/beta/lab}LabSe
Endpoint:	LabServicePort
Operation Name:	submit
Policy Set:	LTPA RAMP default
Binding:	Provider sample

Policy sets can be modified as well, through the wizard shown here. All policy sets attached to the service will be listed and each can be modified. Customized policy sets can then be exported to be imported into a WebSphere Application Server environment.

Ws-policy configuration



Service Side Policy Set Attachment for WebSphere Applicati...

Configure Policy Sharing

Select a service to configure

Service Name:	Policy Sharing
wp.war:{http://foo/}EchoService	No policy sharing

Configure Policy Sharing

Configure Policy Sharing for Web Service

Select the mechanism(s) to share policy information

- [ ] Share Policy Information via WSDL
- [ ] Share Policy Information using WS-MetadataExchange

[✓ Secure with application transport security only]

Secure with application transport security and with a policy set attachment

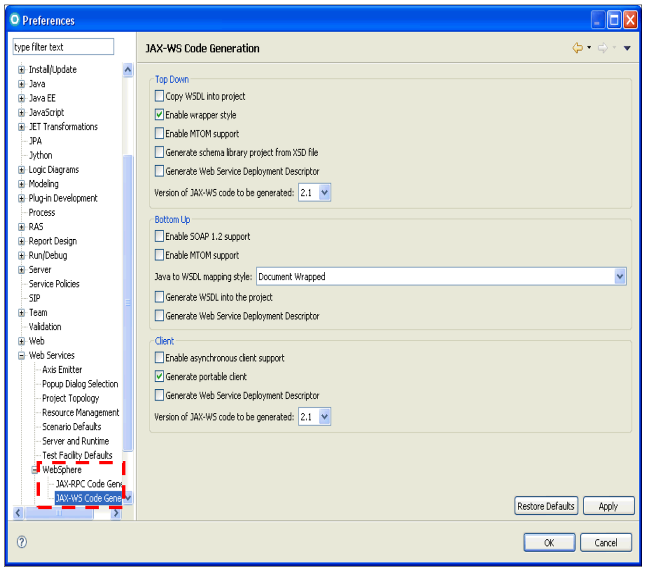
Policy Set: SystemWSSecurityDefault

Binding: Provider sample

Services can also be configured for WS-Policy and WS-Metadata Exchange support as well. Developers can select an option to share this information in the WSDL based on WS-Policy or using the WS-Metadata Exchange specifications.

Code generation preferences

- Set code defaults for the project
- Separate code generation preferences for JAX-WS and JAX-RPC applications



JAX-WS Code Generation

Top Down

- [ ] Copy WSDL into project
- [✓] Enable wrapper style
- [ ] Enable MTOM support
- [ ] Generate schema library project from XSD file
- [ ] Generate Web Service Deployment Descriptor

Version of JAX-WS code to be generated: 2.1

Bottom Up

- [ ] Enable SOAP 1.2 support
- [ ] Enable MTOM support

Java to WSDL mapping style: Document Wrapped

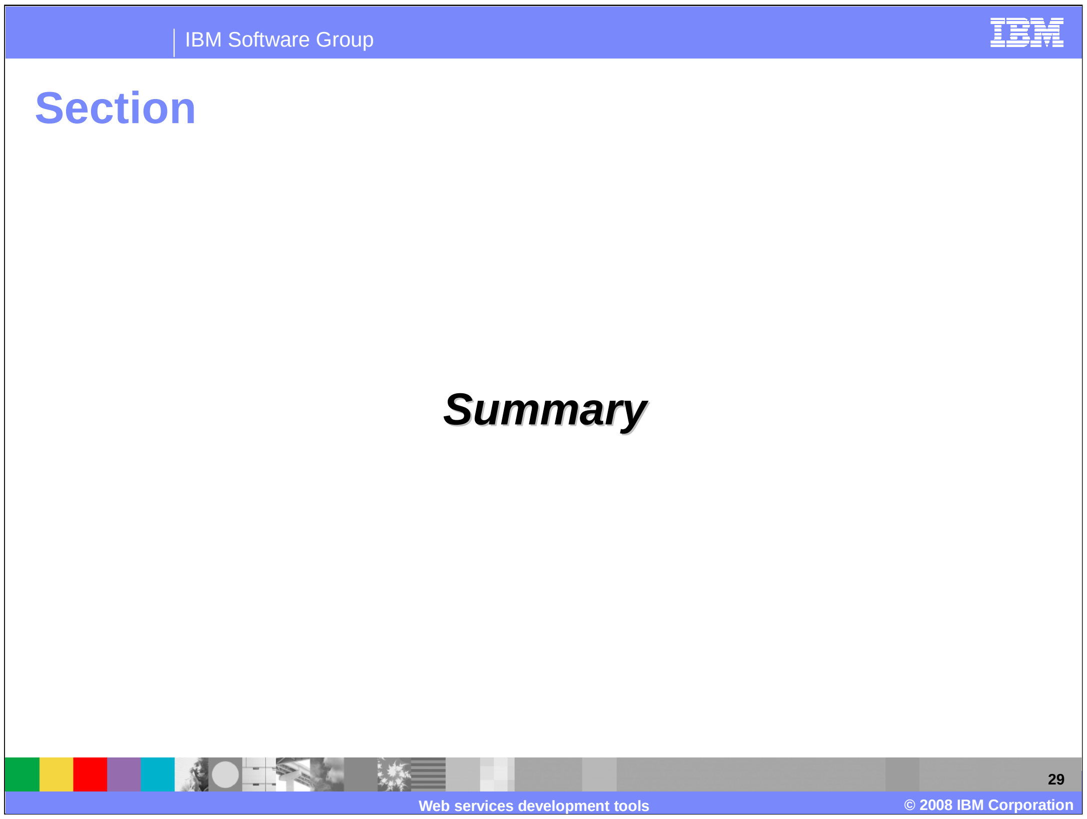
- [ ] Generate WSDL into the project
- [ ] Generate Web Service Deployment Descriptor

Client

- [ ] Enable asynchronous client support
- [✓] Generate portable client
- [ ] Generate Web Service Deployment Descriptor

Version of JAX-WS code to be generated: 2.1

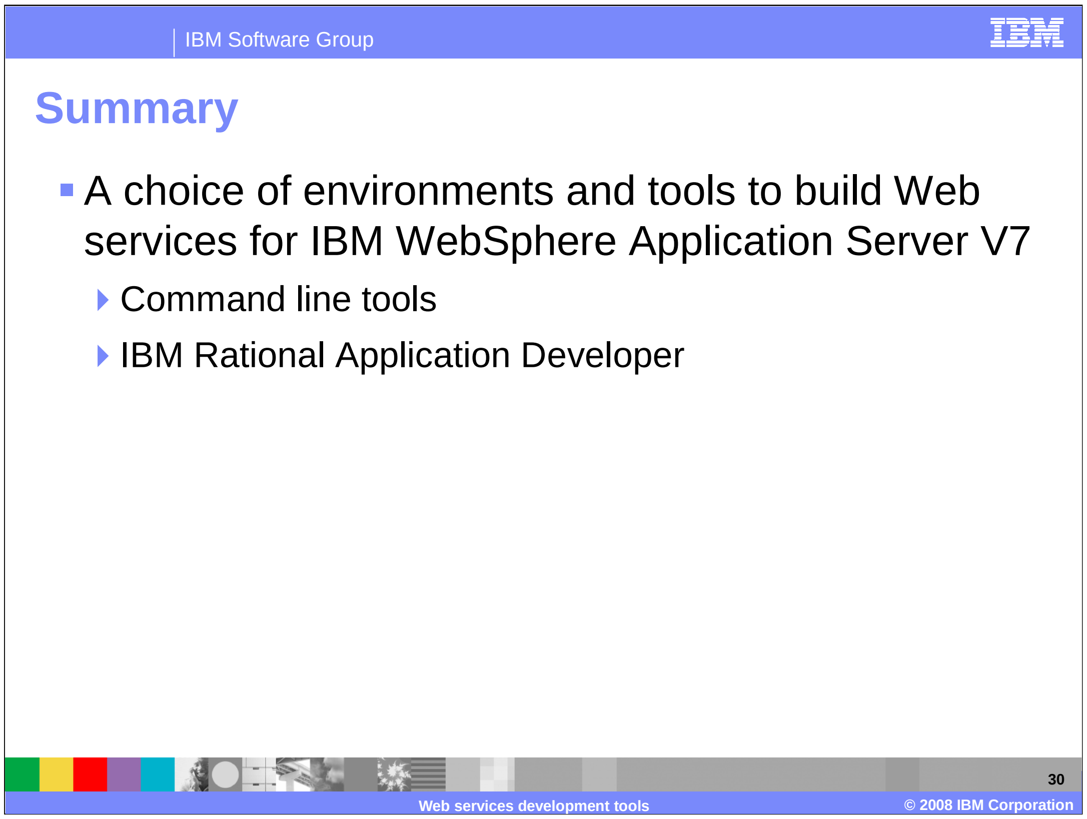
The defaults used by the tools for JAX-WS and JAX-RPC Web services can be set in the preferences, as shown above. Items such as enabling SOAP 1.2 support, enabling MTOM support and others can enabled by default on all JAX-WS Web services that are created.



Section

Summary

The next section will summarize the materials.


IBM Software Group

Summary

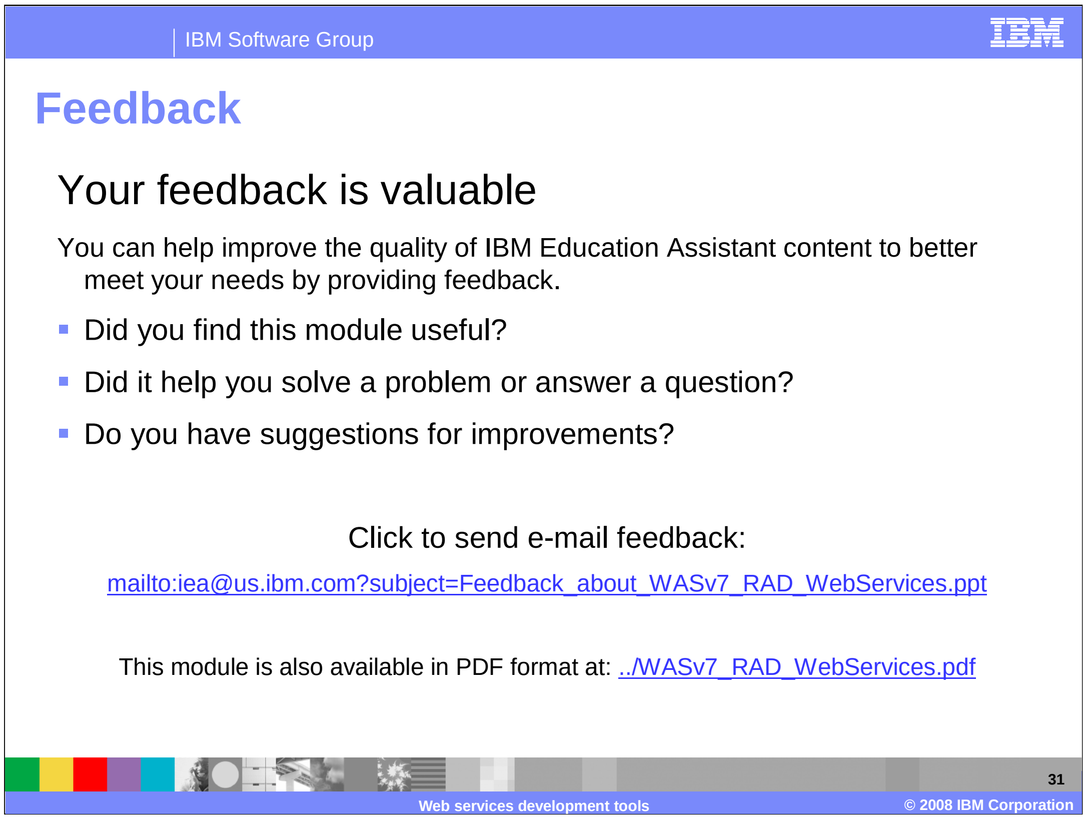
- A choice of environments and tools to build Web services for IBM WebSphere Application Server V7
  - Command line tools
  - IBM Rational Application Developer

30

Web services development tools

© 2008 IBM Corporation

Developers have several choices when creating JAX-WS applications with WebSphere Application Server V7. Command line tools and the IBM Rational Application Developer version 7.5 are available.



Feedback

Your feedback is valuable

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

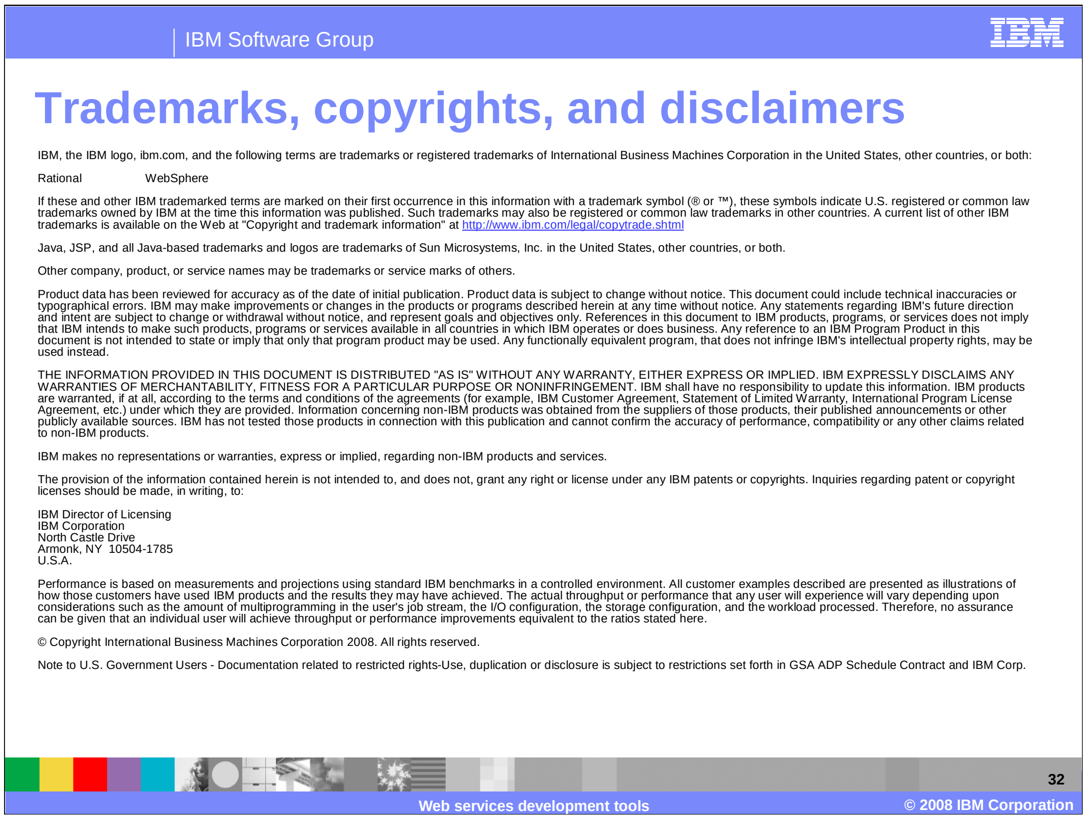
- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WASv7_RAD_WebServices.ppt

This module is also available in PDF format at: ../WASv7_RAD_WebServices.pdf

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



Trademarks, copyrights, and disclaimers