Design Guidelines and Best Practices

RESTful Service WAS v8.x Migration Guide

Revision 1.0

01/31/2013

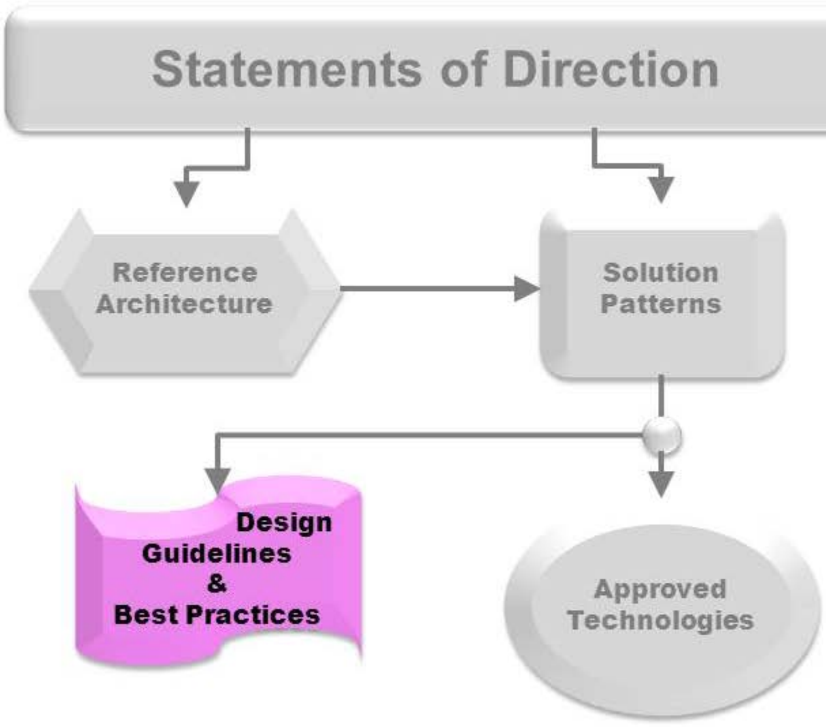


Table of Contents

List of Figures

1.0 Scope

The document describes the use of IBM provided IDE RAD v8.0.4 and its tools to develop JAX-RS v1.1 compliance RESTful services. The topics of coding RESTful services and securing RESTful services are covered in other separate documents.

2.0 Purpose

All RESTful services are JAX-RS v1.1 compliant, including code and deployment settings. All of the services can be deployed to any JAX-RS v1.1 compliant container without code change or minimum deployment setting changes depending on which JAX-RS v1.1 deployment options the container supports.

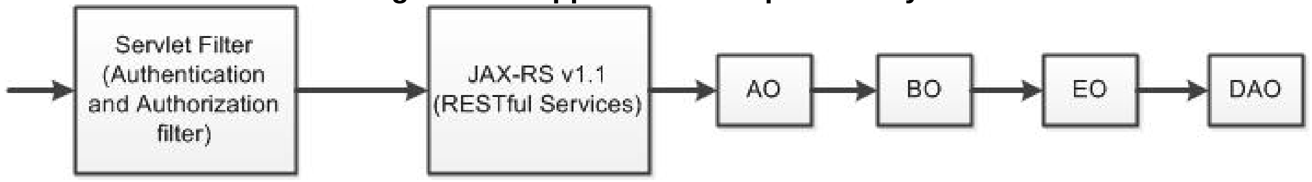
3.0 Requirements for Developers

- Understand RESTful philosophy
- Understand specification JAX-RS v1.1
- Understand JEE 6 and servlet specification v3.0
- Familiar with RAD v8.x IDE

4.0 RESTful Architecture

All RESTful services are the facades of application functions and serve to front end applications, such as user UIs or mobile applications. In general, the RESTful services are deployed in a Java EE web container; Figure 4-1 shows the application components in a layered structure.

Figure 4-1: Application Component Layer


Servlet Filter (Authentication and Authorization filter)
JAX-RS v1.1 (RESTful Services)
AO
BO
EO
DAO

Each RESTful service application must implement interface **javax.ws.rs.core.Application** and implement at least **getClasses()** method of the interface. The **getClasses()** method must return all implementation classes of RESTful service resources of the application except for JAX-RS providers. See appendix A for examples.

All of RESTful services must be coded and deployed in accordance with JAX-RS v1.1. The JAX-RS engine of the selected application servers (including WAS v8.x, or other application server, e.g. JBoss) must support JAX-RS v1.1 or higher.

5.0 Migrate RESTful Service from WAS v6.x to WAS v8.x

5.1. Pre-Conditions

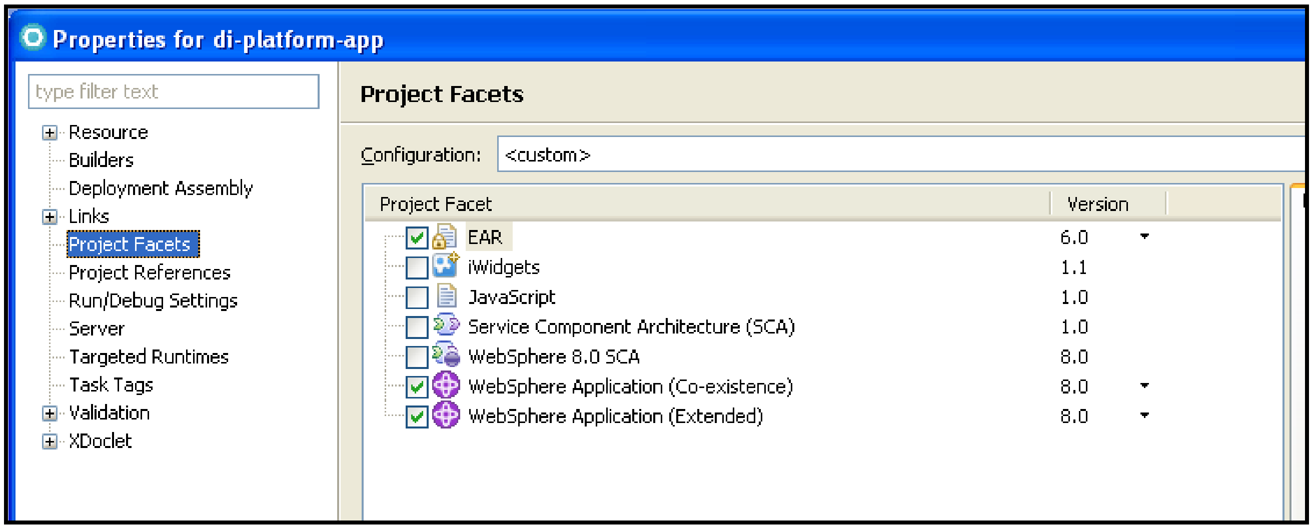
- The existing WAS v6.x RESTful service web project has been converted from IBM RAS v7.0.x/v7.5.x IDE to IBM RAD v8.x IDE.
- RESTful services are coded in compliance with JAX-RS v1.1 except for the service deployment descriptors.
- Every RESTful service application has implemented interface **javax.ws.rs.core.Application**.

5.2. Migration Steps

1. Migrate the enterprise application project to WAS v8.0. If enterprise application project hosting the web projects of RESTful services is already migrated to WAS v8.0, then ignore the step.
    - Change targeted runtimes to Websphere Application sever v8.0. (Refer to Appendix B for these procedures)
    - Run Java EE Specification Upgrade Wizard to upgrade this Java EE application to Java EE specification version 6. (Refer to Appendix B for these procedures)

The enterprise application project' facets and its included modules (WAR & EJB modules)' project facets will be migrated to Java EE specification version 6.

The expected project facets look like:



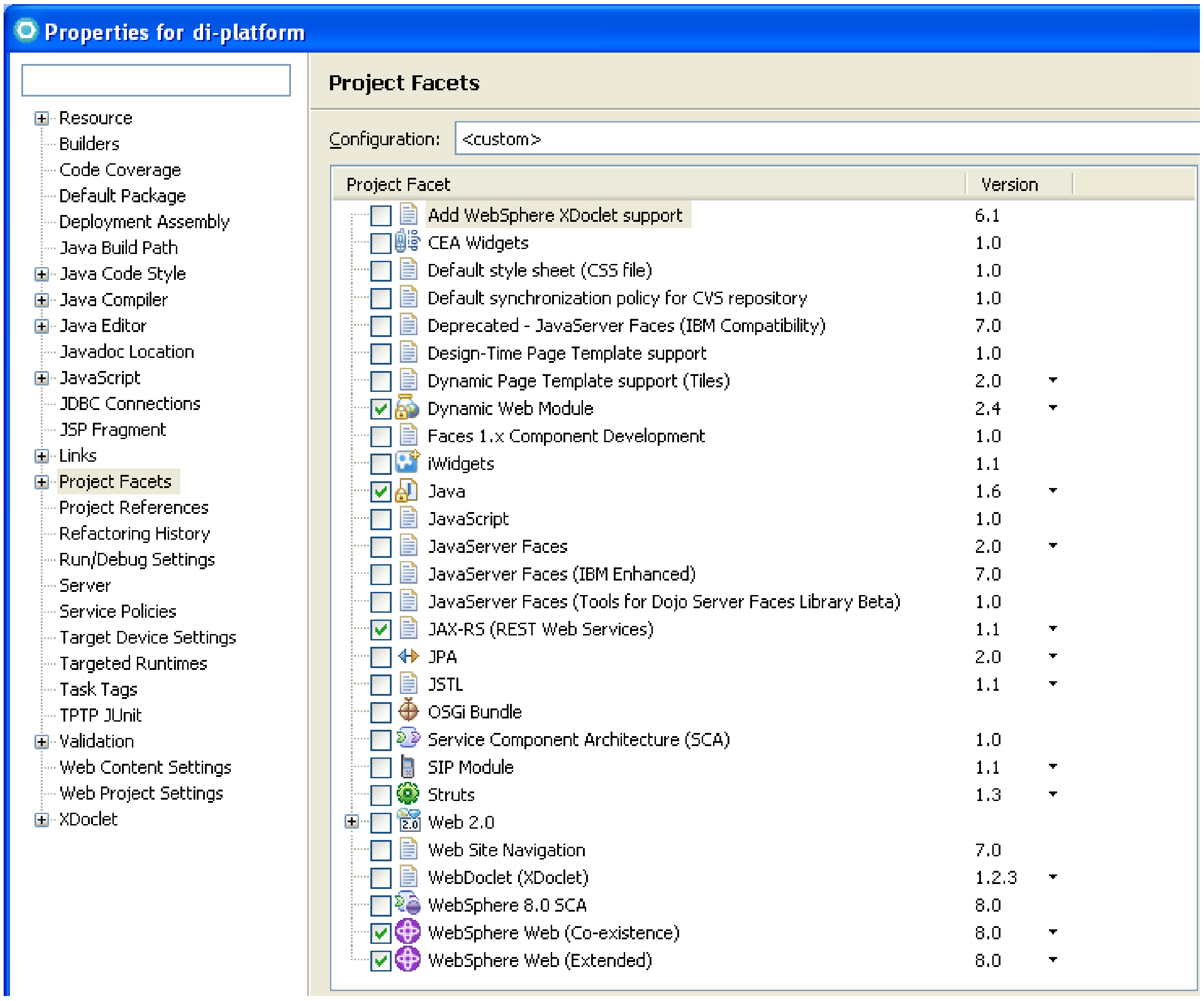
Configuration: <custom>

Project Facet	Version
✓ EAR	6.0
iWidgets	1.1
JavaScript	1.0
Service Component Architecture (SCA)	1.0
WebSphere 8.0 SCA	8.0
✓ WebSphere Application (Co-existence)	8.0
✓ WebSphere Application (Extended)	8.0

2. Migrate the web project of RESTful services to WAS v8.0. Change the web project facets of the RESTful services to meet java EE specification stacks supported by WAS v8.x (Refer to Appendix B for these procedures), that includes:

- Targeted Runtimes – Websphere Application Server v8.0 (including Websphere Web co-existence & extended)
- Dynamic Web Module – Servlet Specification v3.0
- Java – Java specification v1.6 (Java Compiler – v1.6 too)
- JAX-RS (RESTful Web Services) – JAX-RS v1.1

The following is a sample of the Project Facets:



Properties for di-platform

- Resource
- Builders
- Code Coverage
- Default Package
- Deployment Assembly
- Java Build Path
- Java Code Style
- Java Compiler
- Java Editor
- Javadoc Location
- JavaScript
- JDBC Connections
- JSP Fragment
- Links
- Project Facets
- Project References
- Refactoring History
- Run/Debug Settings
- Server
- Service Policies
- Target Device Settings
- Targeted Runtimes
- Task Tags
- TPTP JUnit
- Validation
- Web Content Settings
- Web Project Settings
- XDoclet

Project Facets

Configuration: <custom>

	Project Facet	Version
☐	Add WebSphere XDoclet support	6.1
☐	CEA Widgets	1.0
☐	Default style sheet (CSS file)	1.0
☐	Default synchronization policy for CVS repository	1.0
☐	Deprecated - JavaServer Faces (IBM Compatibility)	7.0
☐	Design-Time Page Template support	1.0
☐	Dynamic Page Template support (Tiles)	2.0
✓	Dynamic Web Module	2.4
☐	Faces 1.x Component Development	1.0
☐	iWidgets	1.1
✓	Java	1.6
☐	JavaScript	1.0
☐	JavaServer Faces	2.0
☐	JavaServer Faces (IBM Enhanced)	7.0
☐	JavaServer Faces (Tools for Dojo Server Faces Library Beta)	1.0
✓	JAX-RS (REST Web Services)	1.1
☐	JPA	2.0
☐	JSTL	1.1
☐	OSGi Bundle	
☐	Service Component Architecture (SCA)	1.0
☐	SIP Module	1.1
☐	Struts	1.3
☐	Web 2.0	
☐	Web Site Navigation	7.0
☐	WebDoclet (XDoclet)	1.2.3
☐	WebSphere 8.0 SCA	8.0
✓	WebSphere Web (Co-existence)	8.0
✓	WebSphere Web (Extended)	8.0

3. Remove jersey libraries, e.g. asm-3.1.jar, jersey-core-1.0.3.1.jar, jersey-server-1.0.3.1.jar, jsr311-api-1.0.jar, and etc.

4. Modify **web.xml** file. Remove vendor implemented specific JAX-RS servlets.

   - For jersey, remove the settings of servlet ***com.sun.jersey.spi.container.servlet.ServletContainer***. For example, remove the following sections from web.xml:

```
      <servlet>
            <servlet-name>Jersey REST Service</servlet-name>
            <servlet-
class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
            <init-param>
                  <param-
name>com.sun.jersey.spi.container.ContainerRequestFilters</param-name>
                  <param-value></param-value>
                  <!--
                  <param-
value>com.dfs.di.platform.auth.rs.filters.AuthenticationFilter</param-value>
                  <param-
value>com.dfs.di.platform.auth.rs.filters.AuthenticationFilter;com.sun.jersey.api
.container.filter.LoggingFilter</param-value>
                  -->
            </init-param>
            <init-param>
                  <param-
name>com.sun.jersey.spi.container.ContainerResponseFilters</param-name>
                  <param-value></param-value>
                  <!--
                  <param-
value>com.dfs.di.platform.auth.rs.filters.ErrorResponseFilter</param-value>
                  <param-
value>com.dfs.di.platform.auth.rs.filters.ErrorResponseFilter;com.sun.jersey.api.
container.filter.LoggingFilter</param-value>
                  -->
            </init-param>
            <init-param>
            <param-name>javax.ws.rs.Application</param-name>
            <param-
value>com.dfs.di.platform.auth.rs.services.DIApplication</param-value>
        </init-param>
            <load-on-startup>1</load-on-startup>
      </servlet>
      <servlet-mapping>
            <servlet-name>Jersey REST Service</servlet-name>
            <url-pattern>/*</url-pattern>
      </servlet-mapping>
```

   - For IBM WAS JAX-RS implementation, if exists, remove the settings of servlet com.ibm.websphere.jaxrs.server.IBMRestServlet. For example, remove the following sections `from` **`web.xml`**

```
      <servlet>
```

```
            <description>
            JAX-RS Tools Generated - Do not modify</description>
            <servlet-name>JAX-RS Servlet</servlet-name>
            <servlet-
class>com.ibm.websphere.jaxrs.server.IBMRestServlet</servlet-class>
            <load-on-startup>1</load-on-startup>
      </servlet>
      <servlet-mapping>
            <servlet-name>JAX-RS Servlet</servlet-name>
            <url-pattern>/*</url-pattern>
      </servlet-mapping>
```

5. Add JAX-RS compliance RESTful application settings. For every implementation of interface javax.ws.rs.core.Application , add the following section

```
<servlet>
      <servlet-name>Fully Qualified Class name of interface
javax.ws.rs.core.Application implementation</servlet-name>
</servlet>
<servlet-mapping>
      <servlet-name>Fully Qualified Class name of interface
javax.ws.rs.core.Application implementation </servlet-name>
      <url-pattern>path of RESTful service resources</url-pattern>
</servlet-mapping>
```

For example,

```
<servlet>
      <servlet-name>com.dfs.di.platform.auth.rs.services.DIApplication</servlet-
name>
</servlet>
<servlet-mapping>
      <servlet-name>com.dfs.di.platform.auth.rs.services.DIApplication</servlet-
name>
      <url-pattern>/*</url-pattern>
</servlet-mapping>
```

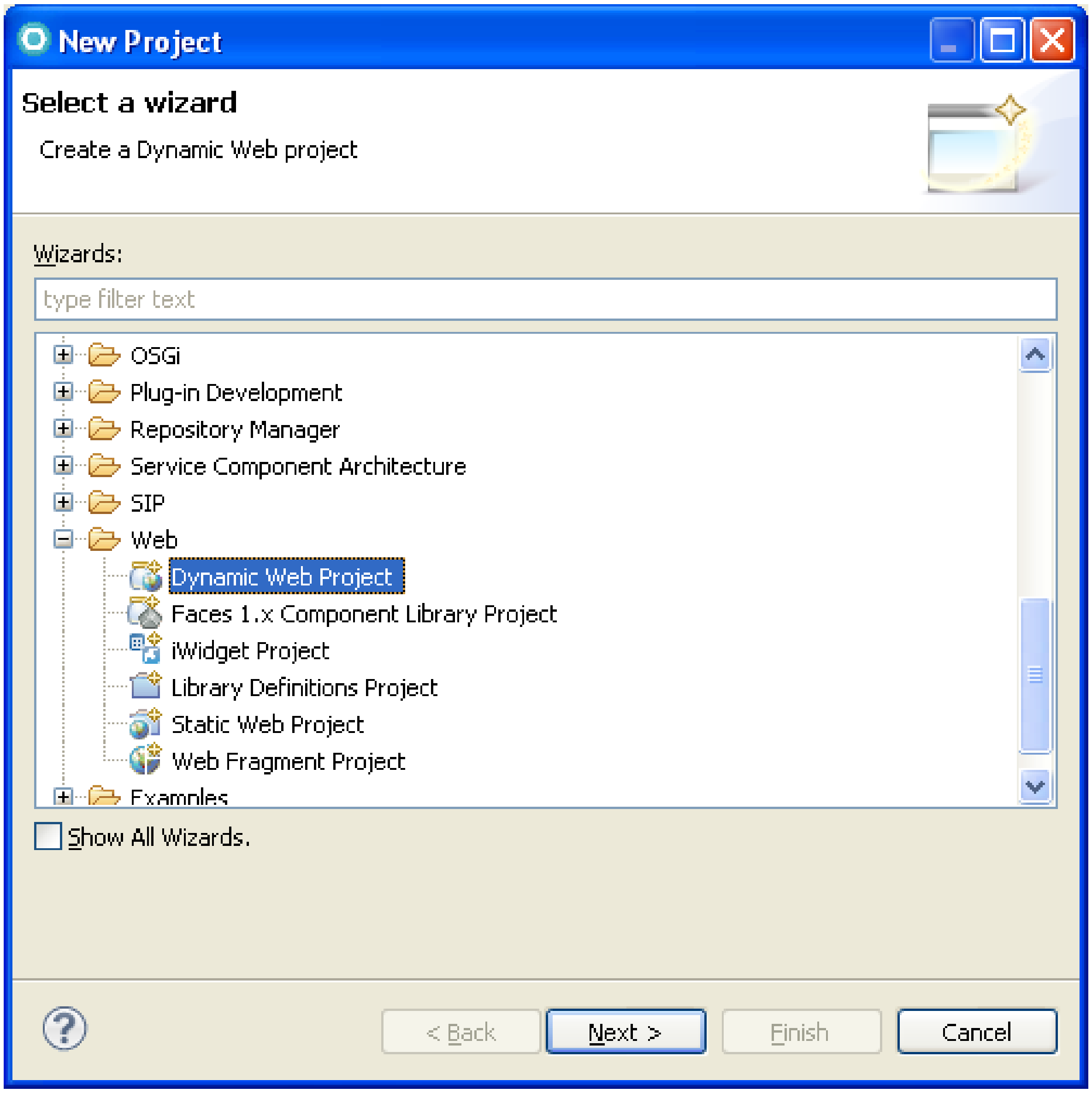
Refer to Appendix A for a complete **`web.xml`** example which is JAX-RS v1.1 web deployment compliance descriptor.

6. Remove **`ibm-web-ext-pme.xmi`** file if it exists.

6.0 Create new RESTful Service in WAS v8.x

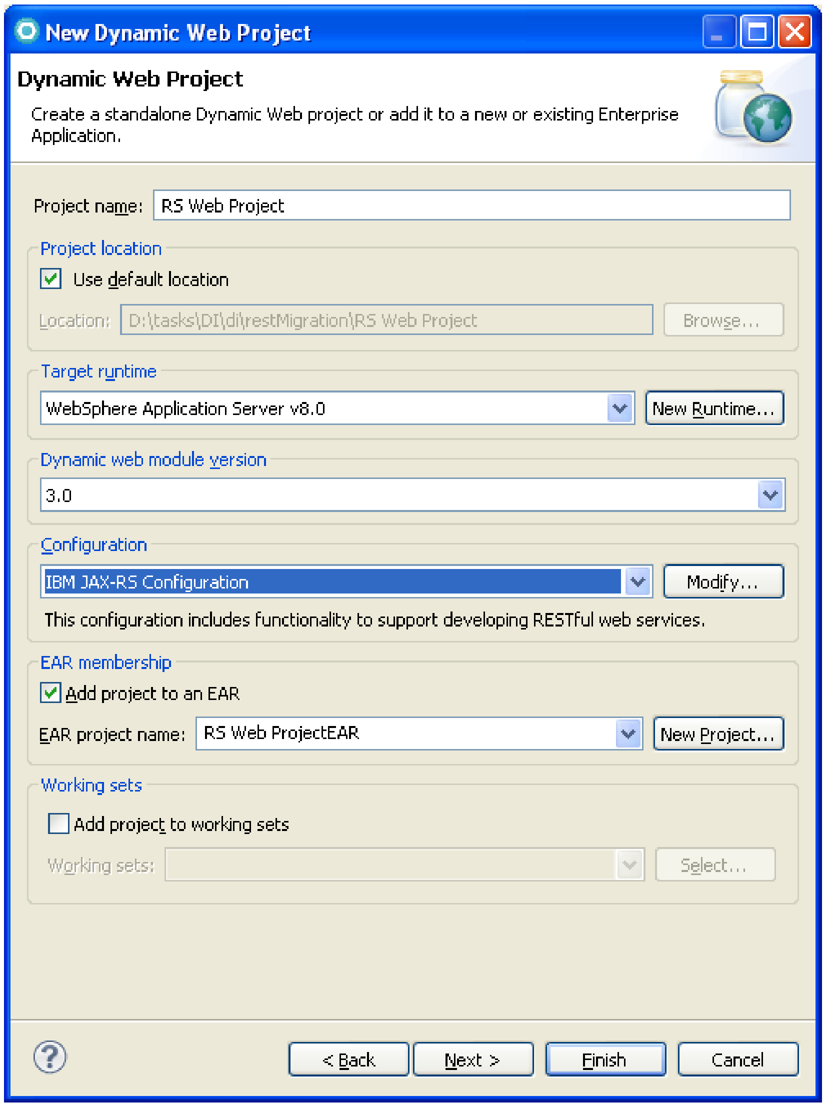
Note: While creating a RESTful service, do not forget to use IBM JAX-RS Configuration when create the dynamic web project for the new RESTful service. Within RAD v8 IDE, conduct the following steps:

1. Create a new dynamic web project. Select **File**->**New**->**Projects.** Select **Dynamic Web Project** from the Web folder.



2. Click **Next**

3. Enter the name of the web project, make sure the following fields are selected:
   - Target runtime – WebSphere Application Server v8.0
   - Dynamic web module version – 3.0
   - Configuration – IBM JAX-RS Configuration



Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: RS Web Project

Project location
☑ Use default location
Location: D:\tasks\DI\di\restMigration\RS Web Project

Target runtime
WebSphere Application Server v8.0

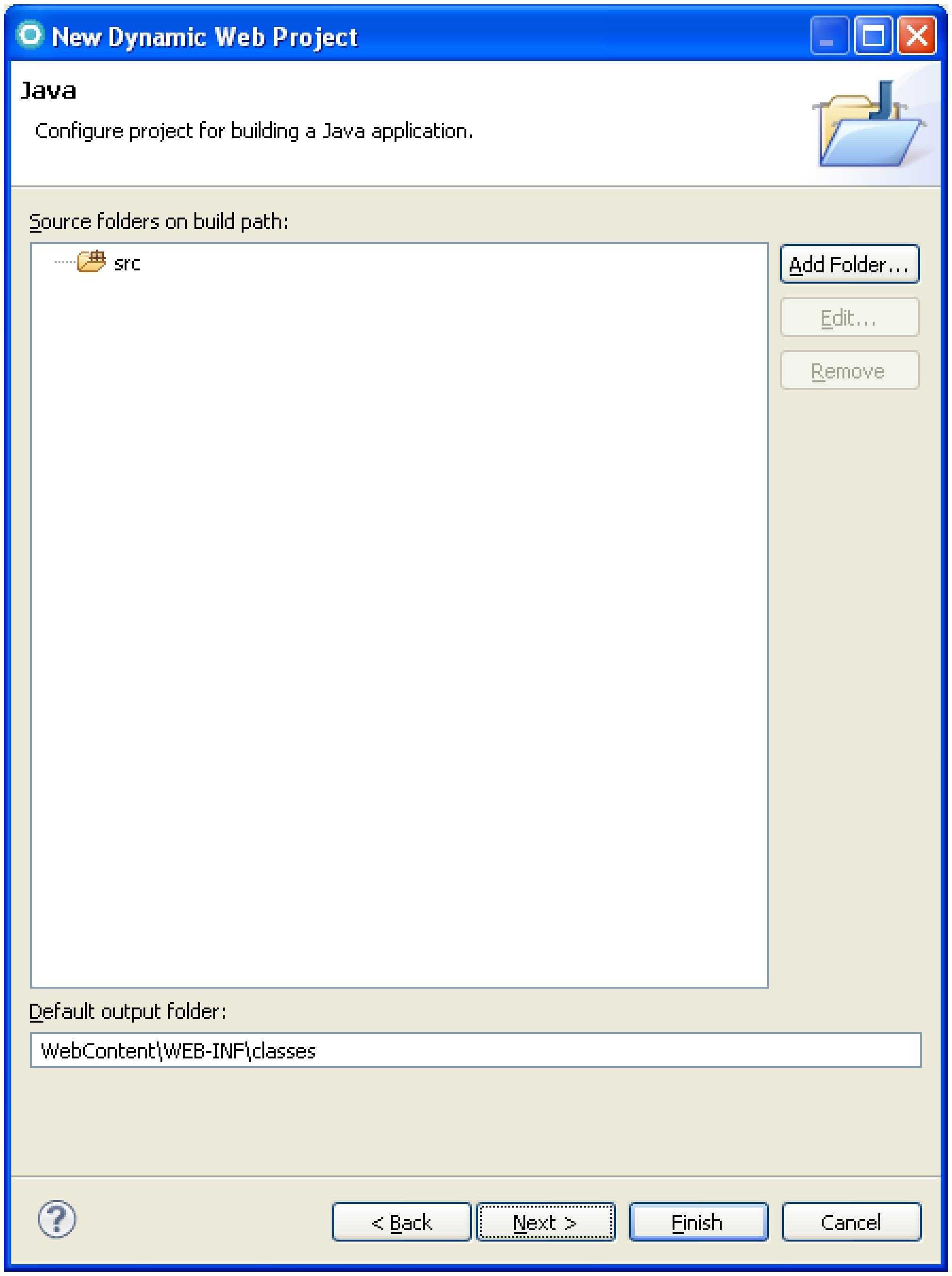
Dynamic web module version
3.0

Configuration
IBM JAX-RS Configuration
This configuration includes functionality to support developing RESTful web services.

EAR membership
☑ Add project to an EAR
EAR project name: RS Web ProjectEAR

Working sets
☐ Add project to working sets
Working sets:

4. Click **Next**


New Dynamic Web Project

Java

Configure project for building a Java application.

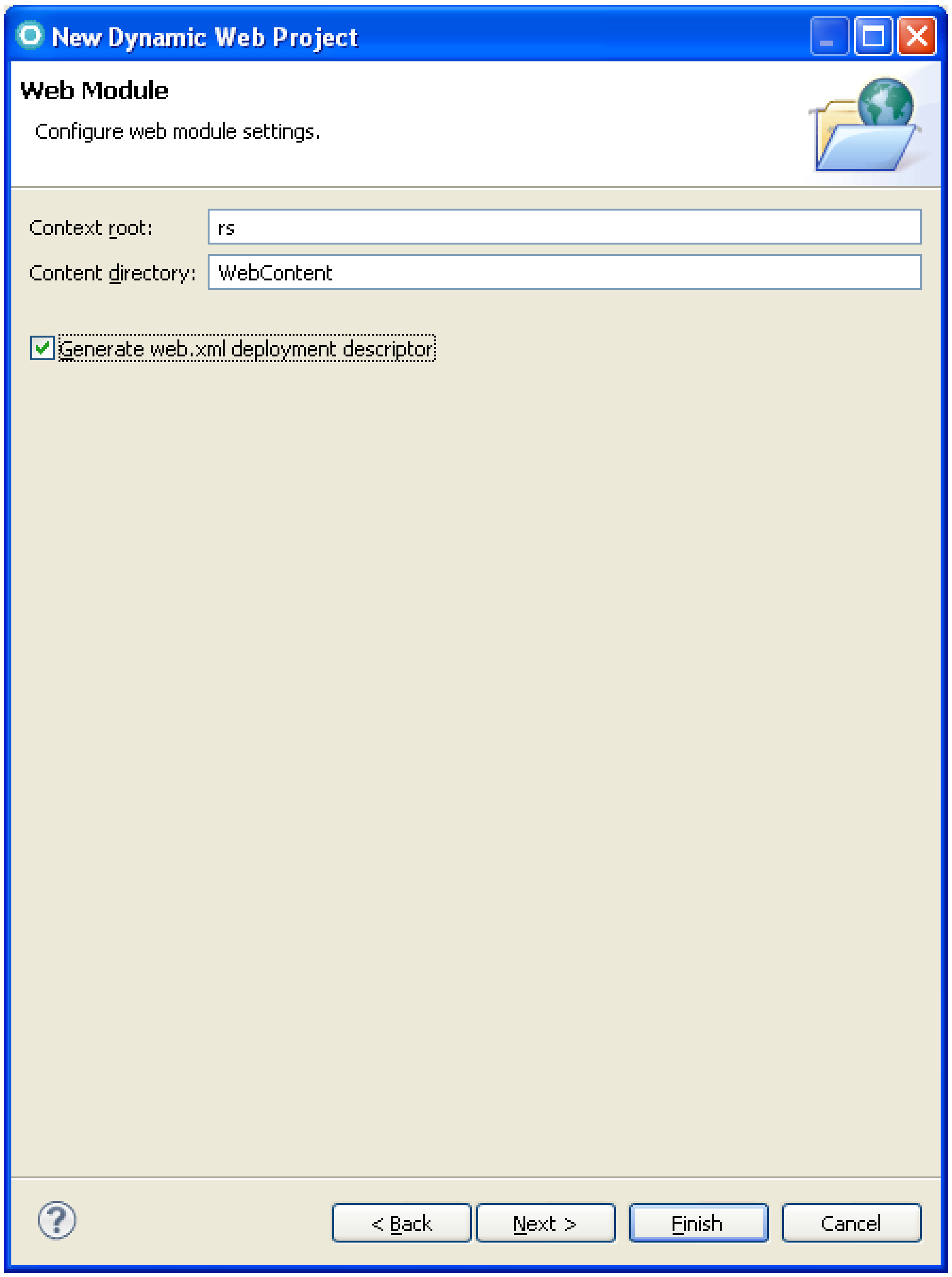
Source folders on build path:

src

Default output folder:

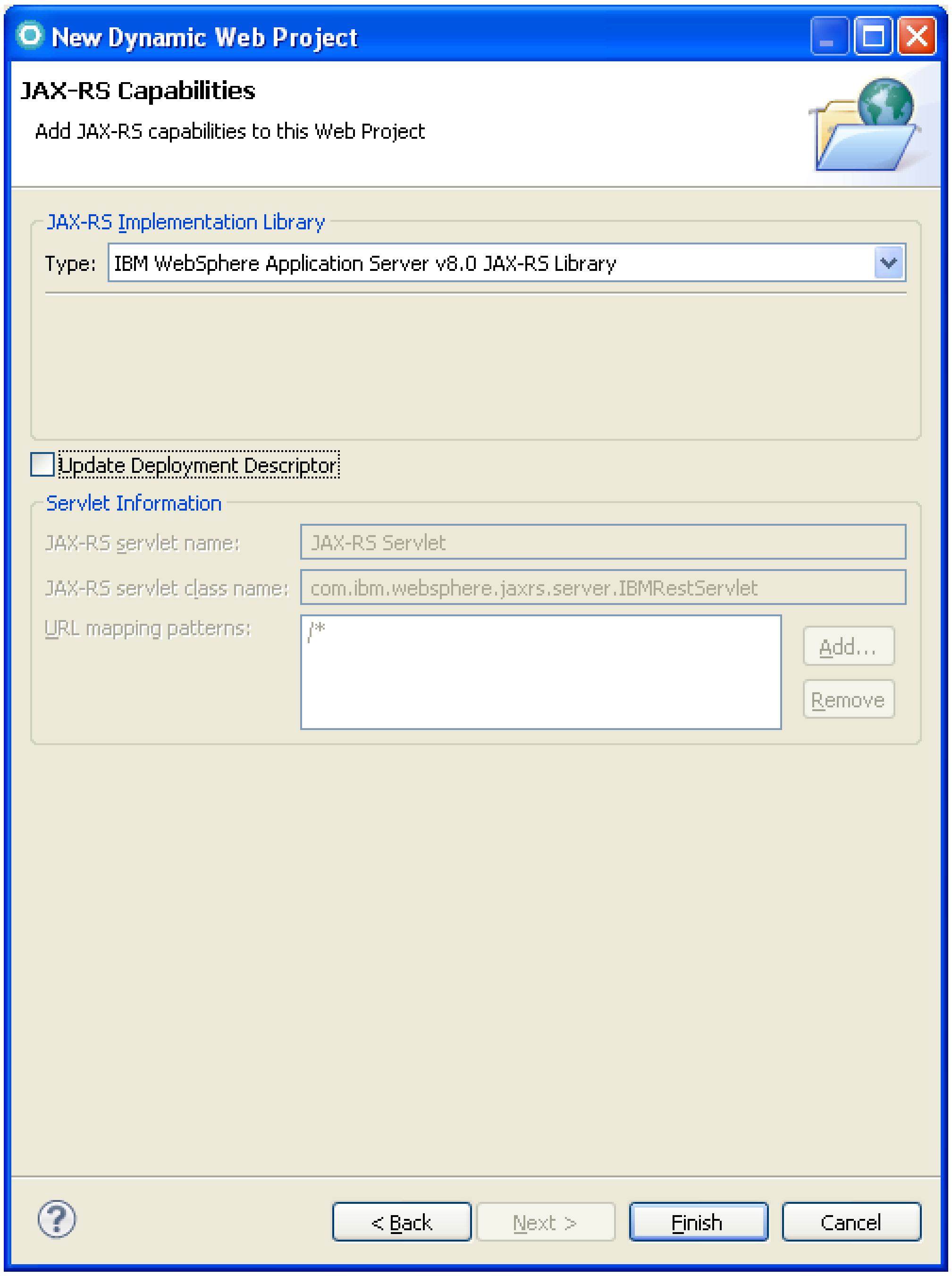
WebContent\WEB-INF\classes

5. Click **Next**



Context root: rs
Content directory: WebContent
☑ Generate web.xml deployment descriptor

6. Enter context root in **Context Root** field and check the **Generate web.xml deployment descriptor** box. This box must be checked regardless of using **web.xml deployment descriptor**.



Type: IBM WebSphere Application Server v8.0 JAX-RS Library

☐ Update Deployment Descriptor

JAX-RS servlet name: JAX-RS Servlet

JAX-RS servlet class name: com.ibm.websphere.jaxrs.server.IBMRestServlet

URL mapping patterns: /*

7. Uncheck **Update Deployment Descriptor** option (do not set IBM specific JAX-RS servlet – **IBMRestServlet**.)
8. Click **Finish**

9. Define deployment settings. There are 2 ways to define deployment settings: use **web.xml** – option a or option b (use annotations in code directly).
    a. In step 4 of section Migration Steps describes how to use web.xml to define deployment settings.
    b. Use annotation @ApplicationPath in the implementation class of interface javax.ws.rs.core.Application, e.g. adding @ApplicationPath (“/*”) on top of the implementation class. See an example in Appendix A.

7.0 Glossary

Note: Also refer to the DFS Glossary

Appendix A: Interface jaxax.ws.rs.core.Application

1. Example of interface jaxax.ws.rs.core.Application implementation

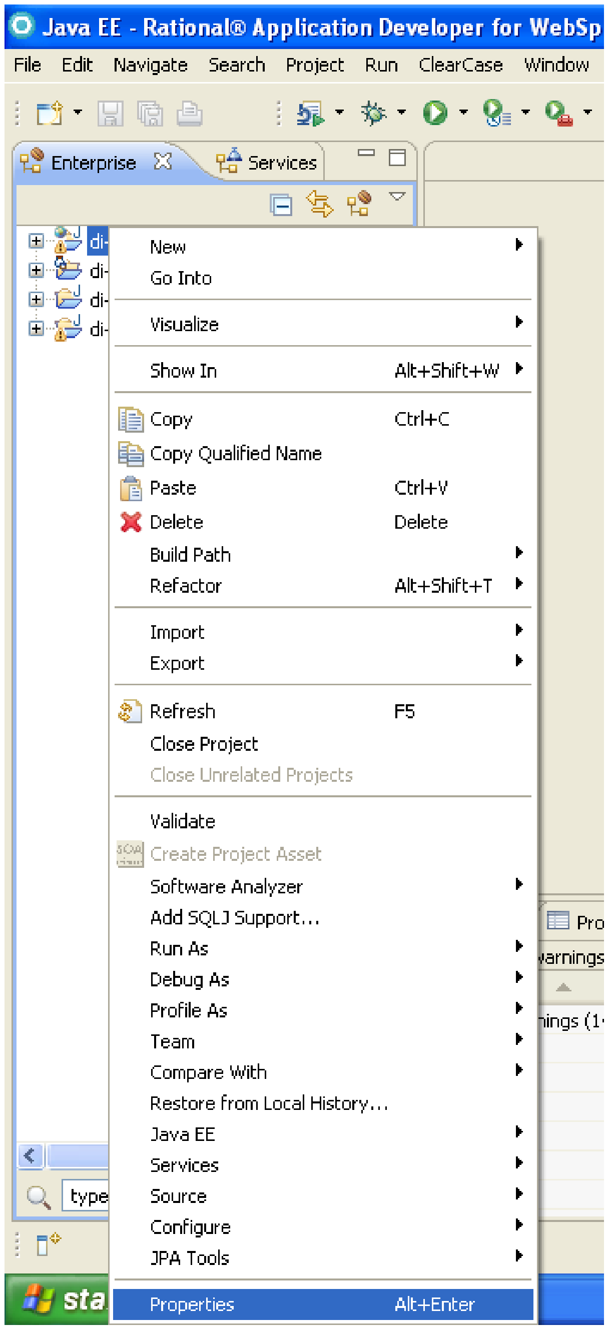
```
package com.dfs.rs.demo;

import java.util.HashSet;
import java.util.Set;

import javax.ws.rs.core.Application;

@ApplicationPath("/demo/*")
public class DemoApplication extends Application {

      private Set<Class<?>> classes = new HashSet<Class<?>>();
      private Set<Object> singletons = new HashSet<Object>();

      public DemoApplication() {
            classes.add(DemoBaseResource.class);
            classes.add(DemoResource.class);
            classes.add(DemoAuthUser.class);
            classes.add(DemoUsers.class);

            singletons add(new ContextResolver());
      }

      @Override
      public Set<Class<?>> getClasses() {
            return classes;
      }

      @Override
      public Set<Object> getSingletons() {
            return singletons;
      }

}
```

- A web.xml example which is JAX-RS v1.1 web deployment compliance descriptor

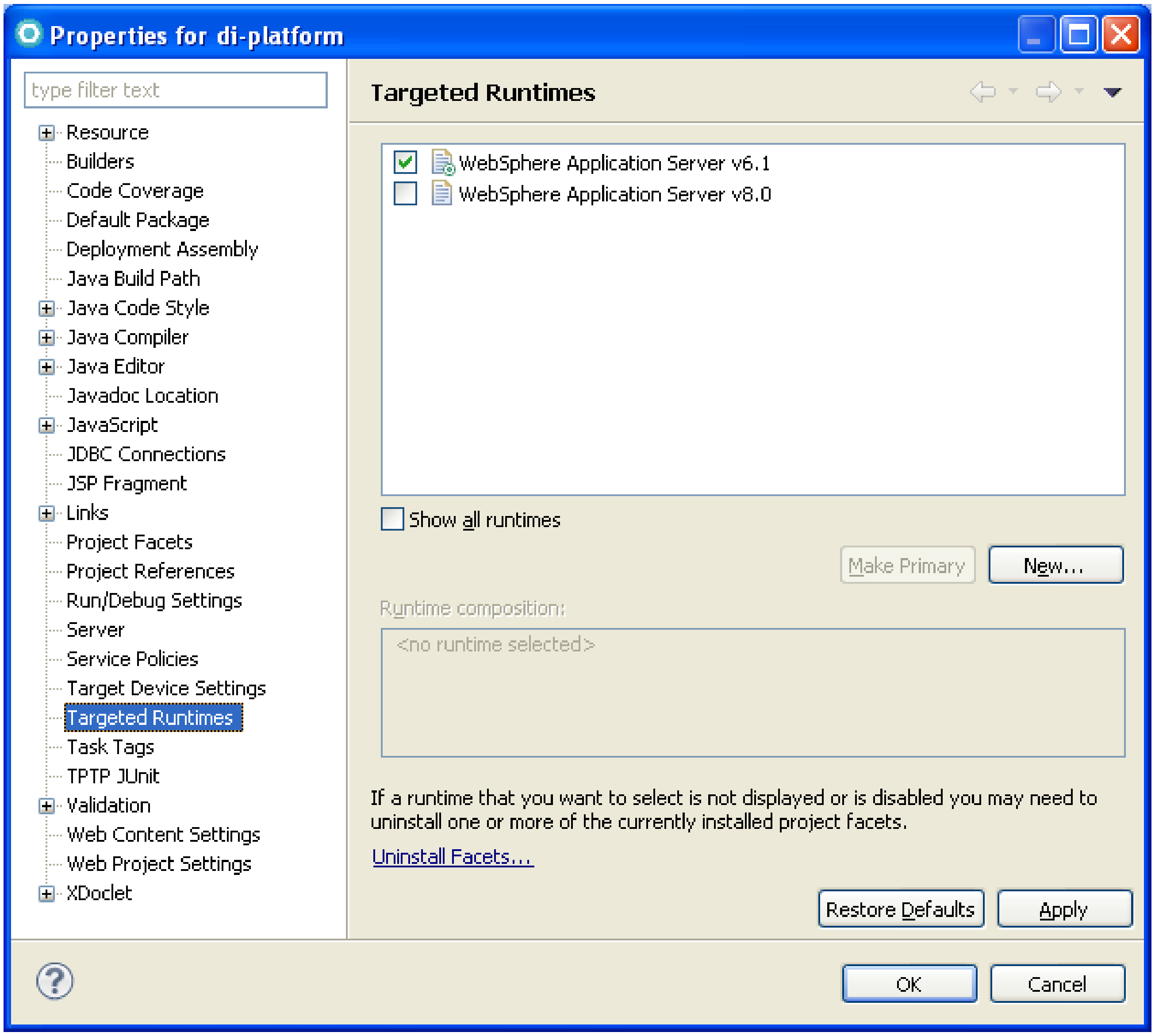
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="3.0"
      xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
      <display-name>di-platform</display-name>
      <filter>
            <description>
            </description>
            <display-name>DIAuthenticationFilter</display-name>
            <filter-name>DIAuthenticationFilter</filter-name>
            <filter-
class>com.dfs.di.platform.auth.servlet.filters.DIAuthenticationFilter</filter-
class>
      </filter>
      <filter-mapping>
            <filter-name>DIAuthenticationFilter</filter-name>
            <url-pattern>/*</url-pattern>
      </filter-mapping>
      <servlet>
            <servlet-
name>com.dfs.di.platform.auth.rs.services.DIApplication</servlet-name>
      </servlet>
      <servlet-mapping>
            <servlet-
name>com.dfs.di.platform.auth.rs.services.DIApplication</servlet-name>
            <url-pattern>/*</url-pattern>
      </servlet-mapping>
      <servlet>
            <description>
            </description>
            <display-name>ConfigLoader</display-name>
            <servlet-name>ConfigLoader</servlet-name>
            <servlet-class>
            com.dfs.di.platform.auth.servlet.config.ConfigLoader</servlet-class>
            <load-on-startup>0</load-on-startup>
      </servlet>
      <servlet-mapping>
            <servlet-name>ConfigLoader</servlet-name>
            <url-pattern>/ConfigLoader</url-pattern>
      </servlet-mapping>
      <welcome-file-list>
            <welcome-file>index.html</welcome-file>
            <welcome-file>index.htm</welcome-file>
            <welcome-file>index.jsp</welcome-file>
            <welcome-file>default.html</welcome-file>
            <welcome-file>default.htm</welcome-file>
            <welcome-file>default.jsp</welcome-file>
      </welcome-file-list>
      <error-page>
            <error-code>406</error-code>
            <location>/error406.html</location>
      </error-page>
      <error-page>
            <error-code>400</error-code>
            <location>/error400.html</location>
      </error-page>
</web-app>
```

Appendix B: Manually Change Targeted Runtimes

1. Right click on the web project, select properties



2. Select Targeted Runtimes


Properties for di-platform

type filter text

- Resource
- Builders
- Code Coverage
- Default Package
- Deployment Assembly
- Java Build Path
- Java Code Style
- Java Compiler
- Java Editor
- Javadoc Location
- JavaScript
- JDBC Connections
- JSP Fragment
- Links
- Project Facets
- Project References
- Run/Debug Settings
- Server
- Service Policies
- Target Device Settings
- Targeted Runtimes
- Task Tags
- TPTP JUnit
- Validation
- Web Content Settings
- Web Project Settings
- XDoclet

Targeted Runtimes

[✓] WebSphere Application Server v6.1
[ ] WebSphere Application Server v8.0

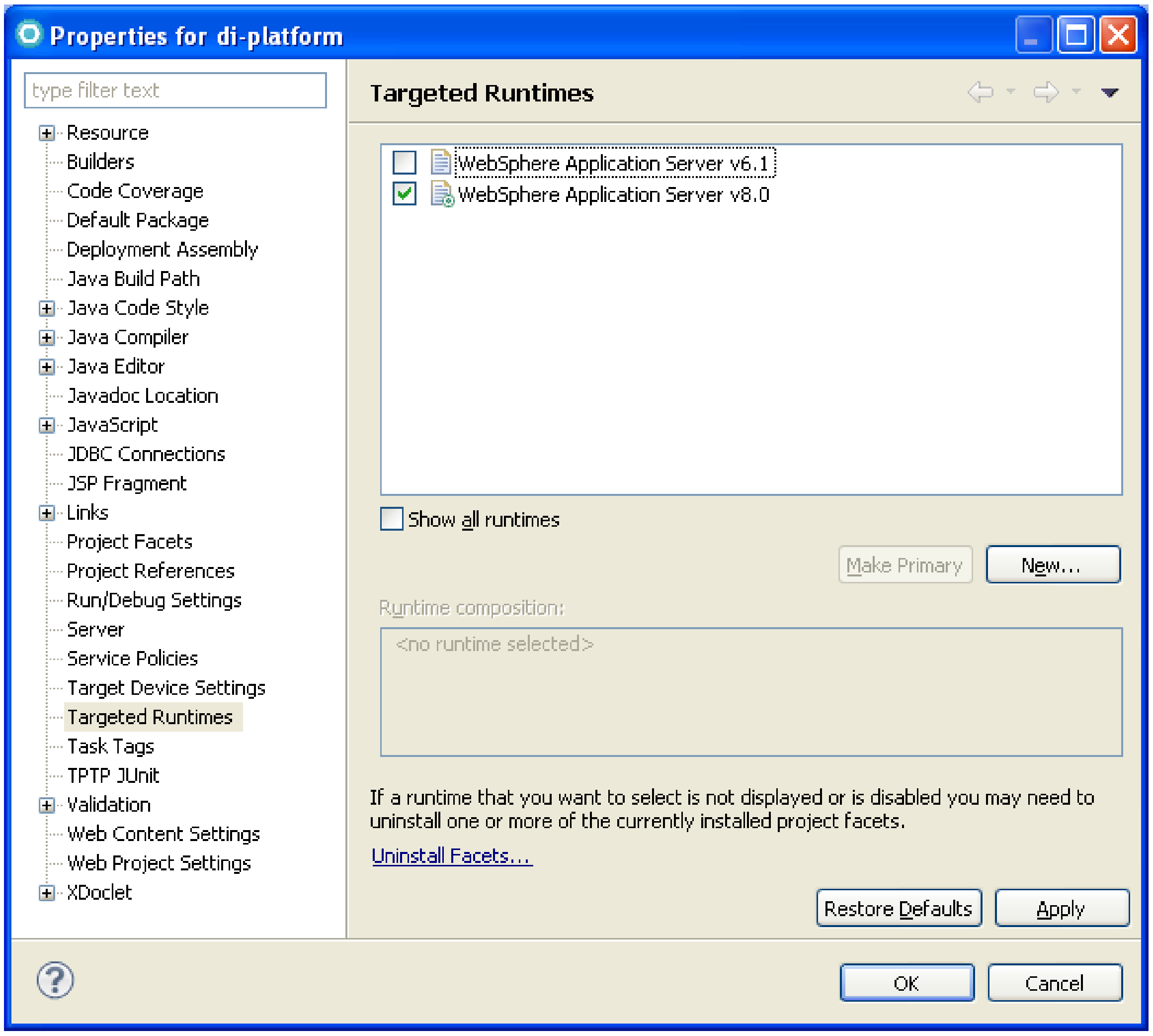
[ ] Show all runtimes

Runtime composition:

<no runtime selected>

If a runtime that you want to select is not displayed or is disabled you may need to uninstall one or more of the currently installed project facets.

3. Uncheck Websphere Application Server v6.1 and check Websphere Application Server v8.0



- [ ] WebSphere Application Server v6.1
- [✓] WebSphere Application Server v8.0

[ ] Show all runtimes

Runtime composition:

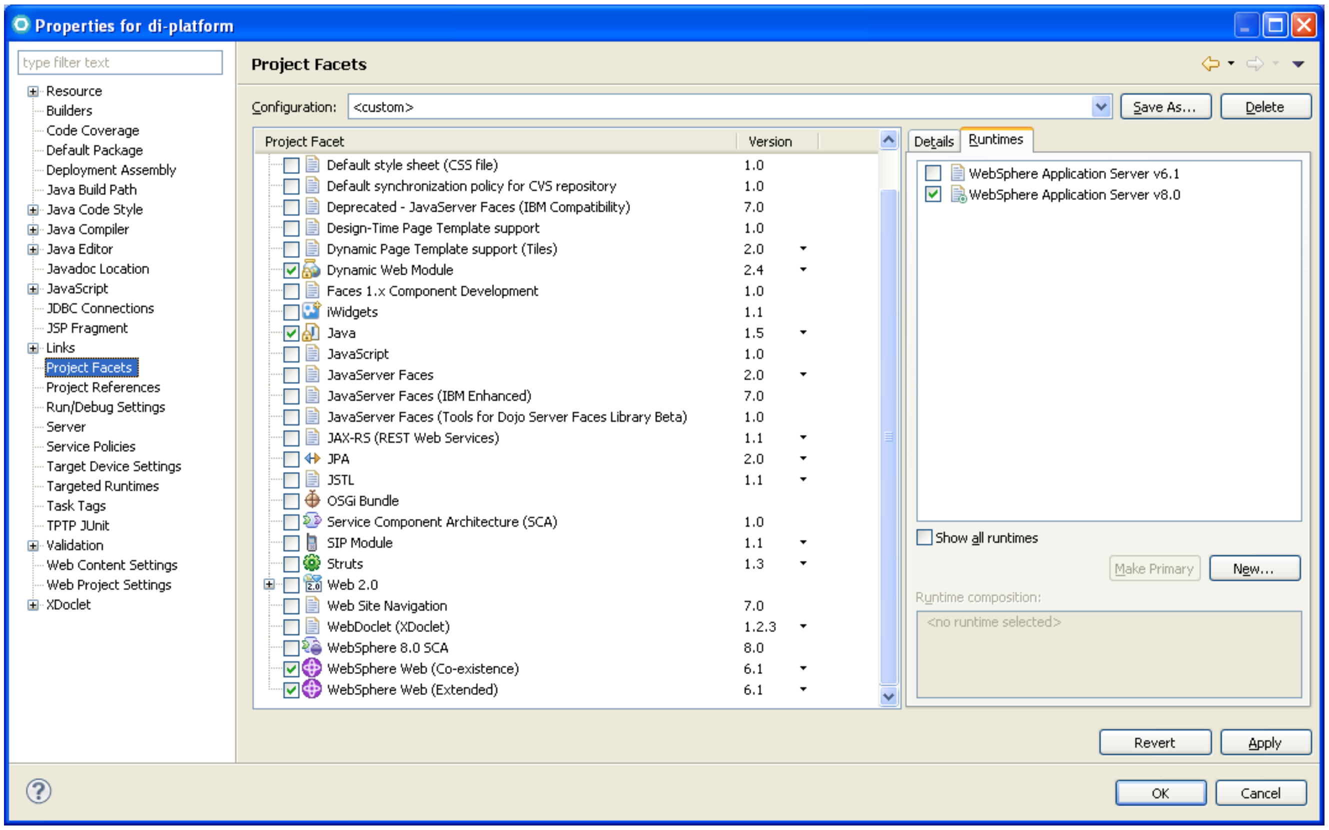
<no runtime selected>

If a runtime that you want to select is not displayed or is disabled you may need to uninstall one or more of the currently installed project facets.

Uninstall Facets...

4. Click **Apply**

5. Select **Project Facets** and click **Runtimes** tab



Project Facets

Configuration: <custom>

	Project Facet	Version
☐	Default style sheet (CSS file)	1.0
☐	Default synchronization policy for CVS repository	1.0
☐	Deprecated - JavaServer Faces (IBM Compatibility)	7.0
☐	Design-Time Page Template support	1.0
☐	Dynamic Page Template support (Tiles)	2.0
✓	Dynamic Web Module	2.4
☐	Faces 1.x Component Development	1.0
☐	iWidgets	1.1
✓	Java	1.5
☐	JavaScript	1.0
☐	JavaServer Faces	2.0
☐	JavaServer Faces (IBM Enhanced)	7.0
☐	JavaServer Faces (Tools for Dojo Server Faces Library Beta)	1.0
☐	JAX-RS (REST Web Services)	1.1
☐	JPA	2.0
☐	JSTL	1.1
☐	OSGi Bundle	
☐	Service Component Architecture (SCA)	1.0
☐	SIP Module	1.1
☐	Struts	1.3
☐	Web 2.0	
☐	Web Site Navigation	7.0
☐	WebDoclet (XDoclet)	1.2.3
☐	WebSphere 8.0 SCA	8.0
✓	WebSphere Web (Co-existence)	6.1
✓	WebSphere Web (Extended)	6.1

Runtimes:
- ☐ WebSphere Application Server v6.1
- ✓ WebSphere Application Server v8.0

Runtime composition:
<no runtime selected>

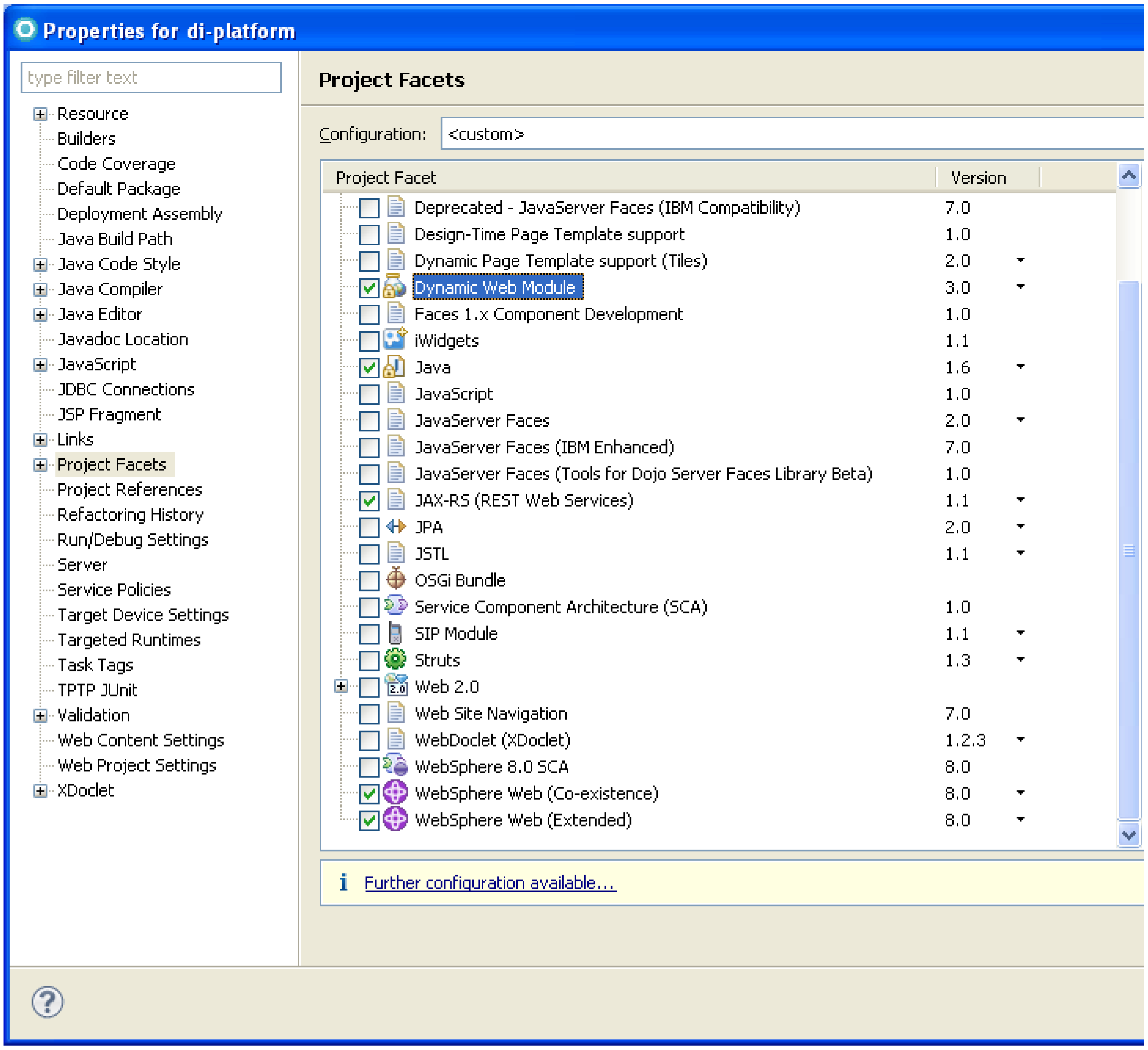
6. Click **Apply**

Now, the targeted runtimes is changed to Websphere Application Server v8.x

Appendix C: Set IBM JAX-RS Configuration

1. Go to project properties dialog and select **Project Facets**
2. Select **JAX-RS (REST Web Services)** and click **Apply**

*Note: Ignore warning **Further configuration required…***



Properties for di-platform

type filter text

- Resource
- Builders
- Code Coverage
- Default Package
- Deployment Assembly
- Java Build Path
- Java Code Style
- Java Compiler
- Java Editor
- Javadoc Location
- JavaScript
- JDBC Connections
- JSP Fragment
- Links
- Project Facets
- Project References
- Refactoring History
- Run/Debug Settings
- Server
- Service Policies
- Target Device Settings
- Targeted Runtimes
- Task Tags
- TPTP JUnit
- Validation
- Web Content Settings
- Web Project Settings
- XDoclet

Project Facets

Configuration: <custom>

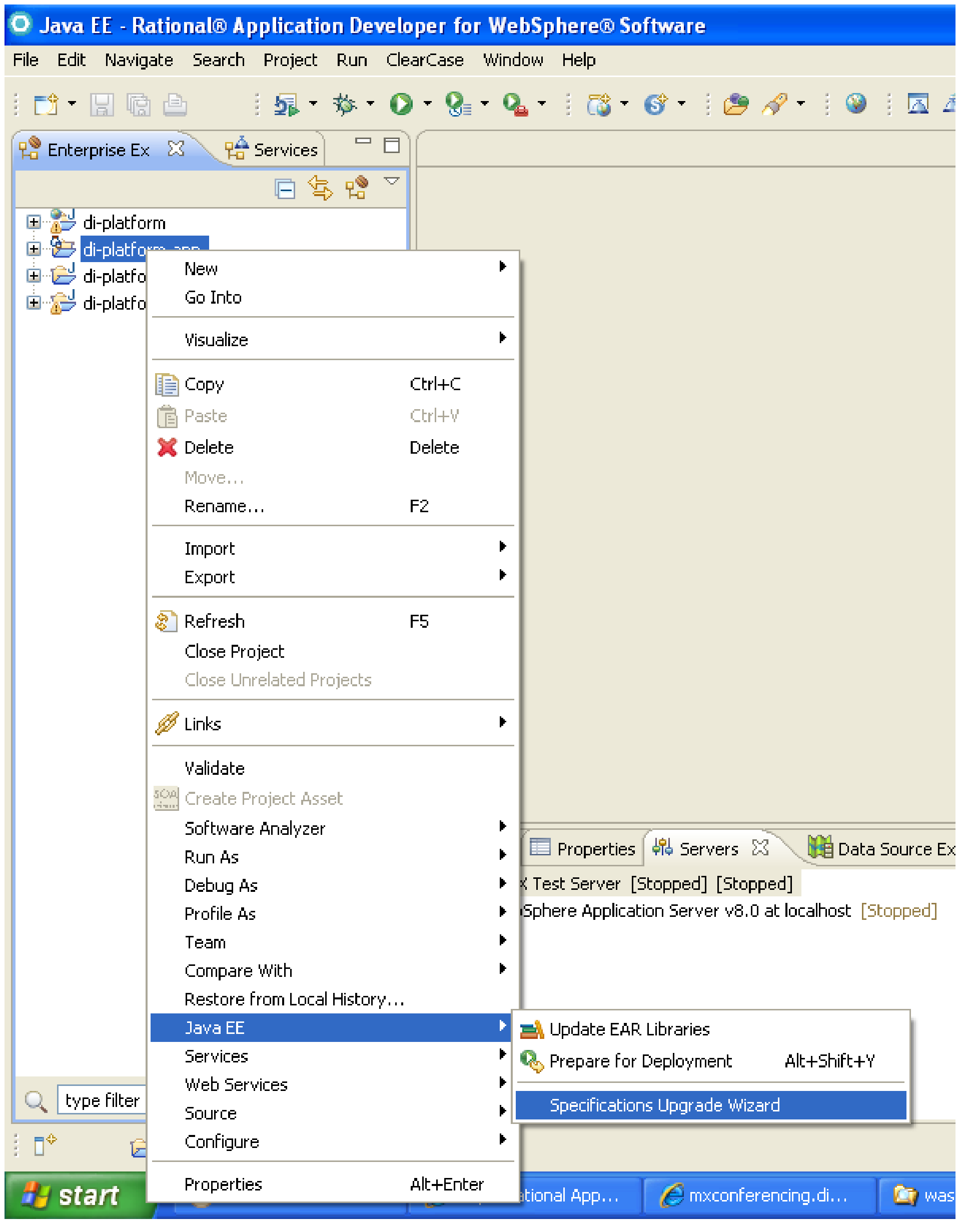
	Project Facet	Version
☐	Deprecated - JavaServer Faces (IBM Compatibility)	7.0
☐	Design-Time Page Template support	1.0
☐	Dynamic Page Template support (Tiles)	2.0
✓	Dynamic Web Module	3.0
☐	Faces 1.x Component Development	1.0
☐	iWidgets	1.1
✓	Java	1.6
☐	JavaScript	1.0
☐	JavaServer Faces	2.0
☐	JavaServer Faces (IBM Enhanced)	7.0
☐	JavaServer Faces (Tools for Dojo Server Faces Library Beta)	1.0
✓	JAX-RS (REST Web Services)	1.1
☐	JPA	2.0
☐	JSTL	1.1
☐	OSGi Bundle	
☐	Service Component Architecture (SCA)	1.0
☐	SIP Module	1.1
☐	Struts	1.3
☐	Web 2.0	
☐	Web Site Navigation	7.0
☐	WebDoclet (XDoclet)	1.2.3
☐	WebSphere 8.0 SCA	8.0
✓	WebSphere Web (Co-existence)	8.0
✓	WebSphere Web (Extended)	8.0

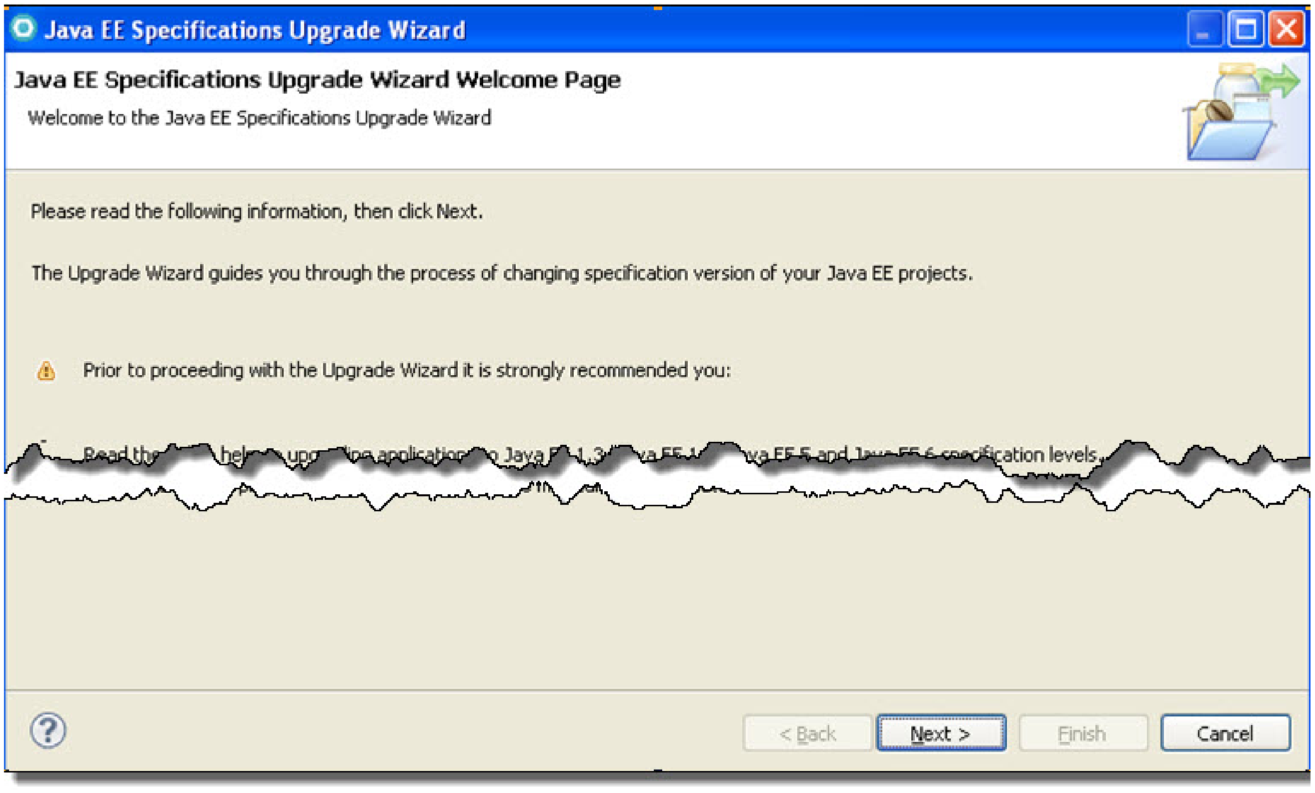
Further configuration available…

3. Click **OK**

Appendix D: Run Java EE Specification Upgrade Wizard

1. Right click on EAR project, select **Java EE**->**Specification Upgrade Wizard**





Java EE Specifications Upgrade Wizard Welcome Page

Welcome to the Java EE Specifications Upgrade Wizard

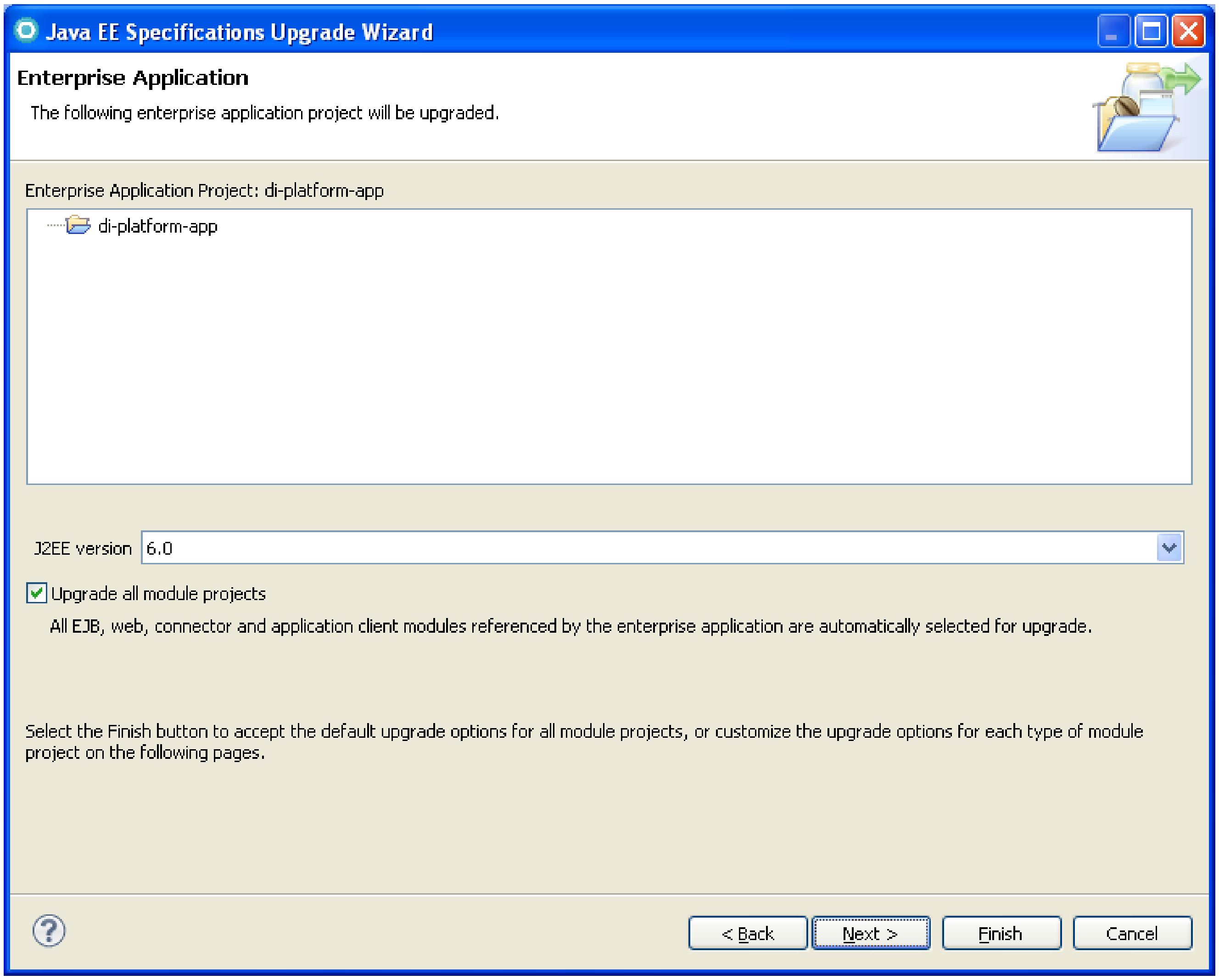
Please read the following information, then click Next.

The Upgrade Wizard guides you through the process of changing specification version of your Java EE projects.

Prior to proceeding with the Upgrade Wizard it is strongly recommended you:

Read the help on upgrading applications to Java EE 1.3, Java EE 1.4, Java EE 5 and Java EE 6 specification levels.

2. Click **Next**
3. In the following screen, select **6.0** from the drop-down selections in the **Java EE specification version** box.
4. Check **Upgrade all module projects** checkbox.



Enterprise Application

The following enterprise application project will be upgraded.

Enterprise Application Project: di-platform-app

di-platform-app

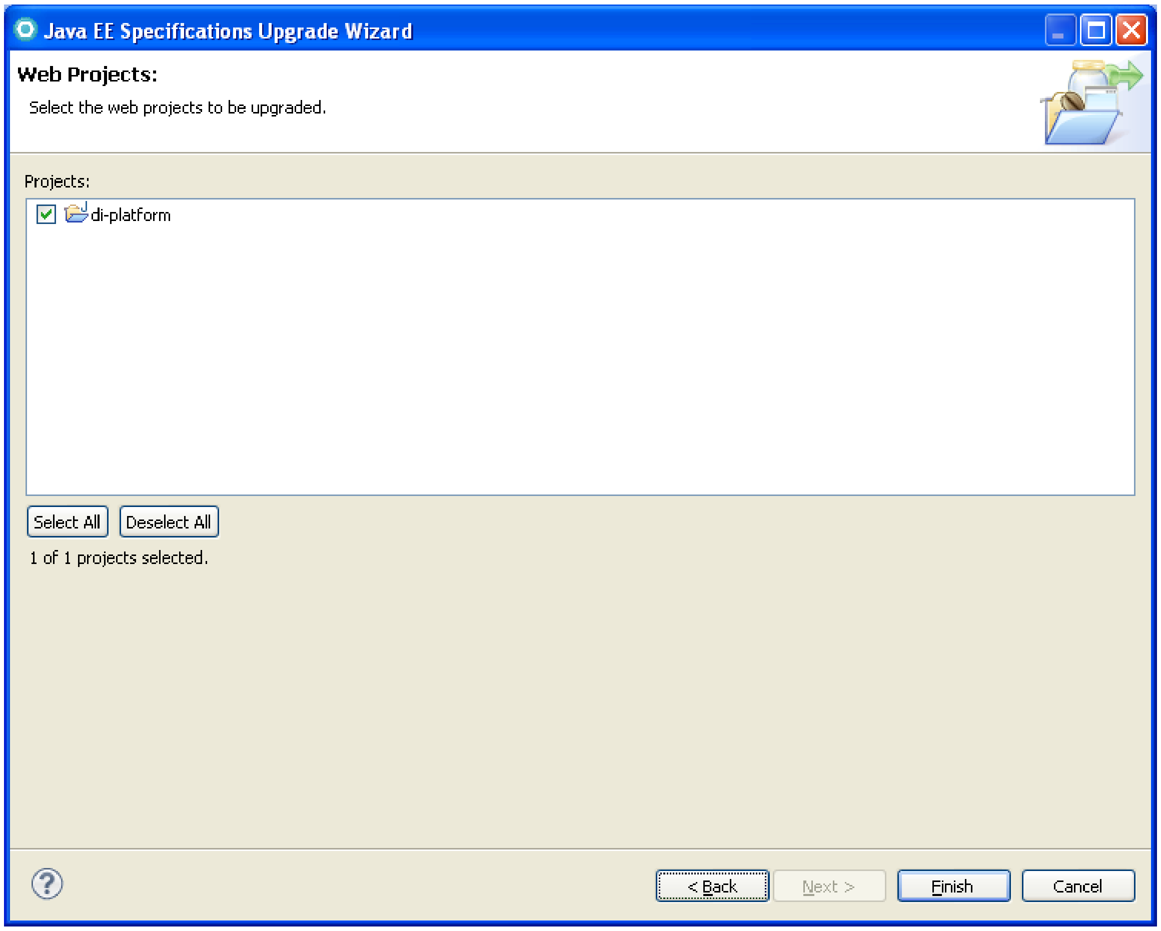
J2EE version 6.0

[✓] Upgrade all module projects

All EJB, web, connector and application client modules referenced by the enterprise application are automatically selected for upgrade.

Select the Finish button to accept the default upgrade options for all module projects, or customize the upgrade options for each type of module project on the following pages.

5. Click **Next**



Web Projects:

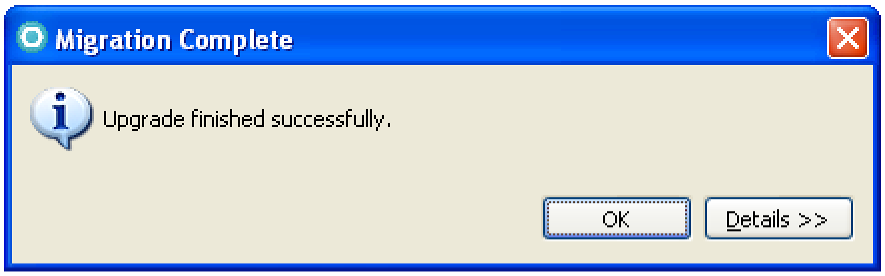
Select the web projects to be upgraded.

Projects:

[✓] di-platform

1 of 1 projects selected.

6. Click Finish



Upgrade finished successfully.

7. Click **OK**

Revision History and Contributors

Revision Date	Approved By	Author	Changes	Revision
01/31/2013		R Hauenstein	Initial Release	1.0

Reviewers/Contributors:

Name	Department/Contribution

Meta Tags

Note: Meta Tag in bold is the unique identifier

BT_restful_service_was_v8x_migration_guide

BT_restful_service

BT_migration

BT_standards

BT_guidelines

BT_best_practices