# Learn to Accelerate Your Web App Development with the Liberty Profile Lab Instructions

Authors: Alasdair Nottingham, Senior Software Engineer Tim deBoer, Senior Technical Staff Member Ross Pavitt, Software Engineer

© 2013 Copyright IBM Corporation

Page 1 of 118

# Table of Contents

Table of Contents	2
Objective	3
Prerequisite Knowledge	3
Setting up Eclipse	3
Instructions	3
The Hello World Application	6
Instructions	6
Summary	18
Registration Application	18
Defining the JPA entity	18
Create the Persistence Unit	29
Building the EJB	33
Adding Servlets	39
Adding CDI to the Application	49
Importing JSP resources	51
Create the Add Attendee page	52
Update the landing page	62
Configure the server	63
Running the application	72
Summary	79
Adding RESTful services using JAX-RS	79
Adding the client-side capability	79
Adding JAX-RS to the Web Project	82
Creating the JAX-RS application	86
Creating a JAX-RS GET request handling method	93
Creating JAX-RS POST requests	95
Summary	100
Adding another EJB	100
Securing the application	107
Add an SSL certificate	107
Configuring the application to use transport guarantees	109
Configure security on the Liberty profile	112
Testing security	114
Summary	114
Packaging the server for deployment	114
Summary	117

# Objective

In this lab, you learn:

- How to install the IBM WebSphere Application Server V8.5.5 Liberty Profile.
- How to create and deploy a simple web application using the IBM WebSphere Application Server Developer Tools for Eclipse V8.5.5.
- How to create and deploy a simple registration web application that uses Servlets, JPA, EJBs, Context and Dependency Injection, and JAX-RS.
- How to secure applications to use SSL.
- How to generate a customised Liberty profile image, and use that image to deploy your application outside of a development environment.

# Prerequisite Knowledge

To get the most out of this lab, knowledge of the following areas is useful

- Basic knowledge of Java EE
- Basic familiarity with the Eclipse IDE

# Setting up Eclipse

To run this lab you need to start Eclipse, configure a workspace and import the required resources.

### Instructions

- 1. Go to <u>https://www.ibmdw.net/wasdev/websphere-application-server-developer-tools-v8-5-5/</u>, and follow instructions to install WebSphere Application Server Developer Tool for Eclipse V8.5.5.
- 2. Launch the Eclipse IDE. If you are prompted to provide a workspace location, provide a path to an empty folder that can be used to store your work with Eclipse and click **OK**.

O Workspace	Launcher		<b>X</b>
Select a wor	kspace		
Eclipse store Choose a wo	is your projects in a folder called a workspace. orkspace folder to use for this session.		
<u>W</u> orkspace:	C:\Registration\workspace	<b>.</b>	<u>B</u> rowse
🔲 <u>U</u> se this a	s the default and do not ask again	ОК	Cancel

3. Select File > Import...



4. In the **Import** dialog, expand the **General** section, and select **Existing Projects into Workspace**, then click **Next**.

Import	
Select Create new projects from an archive file or directory.	Ľ
Select an import source:	
General     General     Archive File     Existing Projects into Workspace     File System     Preferences	Ē
▷ 🗁 CVS	-
(?) < <u>Back</u> <u>Next &gt; Finish</u>	h Cancel

5. Select the option for Select archive file, and use the Browse button to navigate to the copy of Lab-AppDevelopment.zip that came with this set of instructions. Once this is done, click Finish.

nport Projects Select a directory to sea	arch for existing Eclipse projects.	
Select root directory		Browse
Select <u>a</u> rchive file:	C:\Registration\Lab-AppDevelopment.zip	Browse
rojects:		
Resources (Reso	ources)	Select All
		Deselect All
		R <u>e</u> fresh
✓ Copy projects into v	workspace	
Working sets		
Add projec <u>t</u> to wo	rking sets	3 <u>1</u>
Working sets:	-	Select

### The Hello World Application

In this exercise you will install the Liberty profile, create a simple web project, write an application consisting of a single HTML page, and deploy that application to a Liberty profile server. This application forms the basis for later parts of the lab.

#### Instructions

1. Create a new Web Project by clicking **File > New > Web Project**.

🔘 Ja	wa EE - Eclipse							
File	Edit Navigate	Search	Project	Run	Wind	low	Help	_
	New		AI	t+Shift	+N ►	<b>₩</b>	OSGi Bundle Project	8 - 6 - 6
	Open File					₹\$	OSGi Application Project	
	Close			Ctrl-	+w d	<u></u>	Web Project	
	Close All		Ctr	l+Shift-	+w	<b>₽</b>	JPA Project	
						Ê	Enterprise Application Project	1
	Save			Ctr	I+S	<b>(</b>	Dynamic Web Project	1
9	Save As					<u></u>	EJB Project	1
R	Save All		Ct	rl+Shif	t+S		Connector Project	
	Revert					R	Application Client Project	1
	Move					<b>a</b>	Static Web Proiect	

 Enter the name of the project as RegistrationAppWeb. Select the Simple option from the Project Templates, and check the Programming model is set to Java EE. Click Finish. If prompted to open the Web perspective, click Yes.

New	Web Project	
Web Pr Create	<b>oject</b> a new web project.	
Na <u>m</u> e:	RegistrationAppWeb	
Project	Iemplates:         Image: Dojo Toolkit         Image: REST Services         Image: Simple         Image: jQuery	Simple This creates a basic web project. Programming Model: © <u>C</u> lient-side only (HTML, JavaScript,) @ <u>J</u> ava EE © OSGi
?		< <u>Back</u> <u>N</u> ext > <u>Finish</u> Cancel

 To create the web page right click the **RegistrationAppWeb** project and select New > Web Page.

😤 Enterprise Explorer 🛛		🖻 🔄 😫 🎽			
RegistrationAppWel Resources	-	New	•	C2	Project
		Go Into		¢	File
		Show In	Alt+Shift+W ▶	Ċ	Folder
	D	Сору	Ctrl+C	B	SQL File
	Þ	Copy Qualified Name	(	<b>S</b>	Web Page
	Ē	Paste	Ctrl+V	Ø\$	Annotation

4. Set the File Name to *index.html*, and set the **Template** to *HTML* from the **Basic Templates**, then click **Finish**.

🔵 New Web	Page	
Web Page Select a na	me, location, and template for a new web page.	<>
F <u>i</u> le Name:	index.html	
F <u>o</u> lder:	/RegistrationAppWeb/WebContent	Browse
T <u>e</u> mplate:	Preview:	
	<ul> <li>▲ Basic Templates</li> <li>♦ HTML</li> <li>♦ JSP</li> <li>♦ JSP Fragment</li> <li>♦ My Templates</li> <li>♦ HTML</li> </ul>	
	Template Details: Basic page template "HTML"	
	✓ Link page to template	
Options	,	
?	Finish	Cancel

 In the web page, add <h1>Hello World</h1> between the opening and closing tags of the body element.

index.html ⊠	
	6
Hello World	
<pre><!DOCTYPE HTML>     @ <html> @ <head> <title>index</title> <meta content="text/html; charset=utf-8" http-equiv="Content-Type"/> </head></html></pre>	
4	
Design Source Split	

- 6. Save the file using the **control-S** shortcut.
- 7. Now the basic application is complete. To deploy, right click on the *index.html* file and select **Run As > Run on Server**.

s 1	😤 Enterprise Explo	orer 8	X	] 🔄 😤 🦉 🔽 🗖 🗍	in 😰	dex.html 🔀
н	a 🛃 Registratio	nApp	Web			
	⊳ 🧟 JAX-W	S Weł	b Services			
	Kegistr	ation	AppWeb			
	Java Re	sour	ces		H	ello World
		npt ro	esources +			cho work
		TA-IN	l JF			
	> 🗁 WE	B-INF	:			
	📄 ind		New		×	
	P A Resources		Show In	Alt+Sh	ift+W ▶	
			Open		F3	(IDOCTYPE HTML>
			Open With		•	(html> (head>
		-				<pre>title&gt;index</pre>
			Сору	(	Ctrl+C	<pre>imeta http-equiv="Content-Type" content="text/html; cha</pre>
		•	Copy Qualified Name			(body>
		Ē	Paste	(	Ctrl+V	<pre>th1&gt;Hello World</pre>
		×	Delete		Delete	<pre>(/body&gt; (/html&gt;)</pre>
		<u>_</u>	Remove from Context	Ctrl+Alt+Shift+	Down	
		દ્રન	Mark as Landmark	Ctrl+Alt+Shi	ft+Up	
			Build Path		•	
			Move			
			Rename		F2	
		2	Import			
		4	Export			
		8	Refresh		F5	
		Ø	Links		×	
			Validate			Source Split
			Show in Remote Systems v	view		
			Run As		•	1 Run on Server Alt+Shift+X R
			Dehua As		•	2 DOH Rupper

8. Select Manually define a server. Expand IBM, and select WebSphere Application Server V8.5 Liberty Profile. Check Always use this server when running this project. Click Next.

💮 Run On Server		
Run On Server Select which server to use		
How do you want to select th Choose an existing serve Manually define a new se	e server? r erver	al cenver adapters
Select the <u>s</u> erver type:		
<ul> <li>▷ ➢ Basic</li> <li>▲ ➢ IBM</li> <li>☐ IBM WebSphere w</li> <li>☐ Web Preview Service</li> <li>☑ WebSphere Appli</li> <li>Lightweight profile for WebSp</li> </ul>	6.0 rer cation Server V8.5 Liberty Profile ohere Application Server.	A III
Server's <u>h</u> ost name:	localhost	
Server name:	WebSphere Application Server V8.5 Liberty Profile at loca	lhost
?	< Back	Cancel

9. Click the **download or install** link.

💽 Run On Server	
Liberty Profile Runtime Environment Specify the installation folder and JRE.	
Name: WebSphere Application Server V8.5 Liberty Profile Installation folder Click to download or install, or provide a path to an installed runtime environm Path:	nent.

10. Select the option to Install a new runtime from an archive, and use the Browse button to select the wlp-developers-runtime-8.5.5.0.jar and click Next.

Install a new runtime environment from an archive:	
C:\Registration\wlp-developers-runtime-8.5.5.0.jar	Browse
Download and install a new runtime environment from:	

11. Click Next on the Install Add-ons screen.

🔘 Install R	untime Environment		x					
Install Ac	<b>ld-ons</b> d-ons to install on the runtime environment.							
Select add	Select add-ons to include in this installation:							
	IBM WebSphere Application Server V8.5.5.0 Liberty Profile Ex Content	tended						
	Download and install the IBM WebSphere Application Server V8.5.5.0 Liberty Profile Extended Content runtime directly from ibm.com	23.6 MB Add	Ξ					
	Java Architecture for XML Bindings							
	This feature enables support for the Java Architecture for XML Binding 2.2 specification, which provides easy mapping of Java classes to XML documents.	5.1 MB Add						
	MongoDB Java Driver							
	This feature enables the usage of the MongoDB Java Driver and allows DB instances to be configured in the server configuration, injected into managed components such as EJBs, and accessed through JNDI. Applications interact with these DB instances via the MongoDB APIs	342.8 KB Add						
	Web Cache Monitor							
	This feature provides monitoring of the distributed map cache and provides simple cache statistics, cache entries, and cache policy information for Servlet cache instances.	807.3 KB Add						
	Enterprise Java Beans (EJB) Sample							
	The following sample demonstrates injection of an EJB into a servlet and provides an application consisting of a servlet and stateless session	123.2 KB	-					
Add <u>A</u> rcł	nive Ad	dd-ons to insta	II: O					
?	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish	Cancel						

12. Select the radio button for *I accept the terms of the license agreement* and click **Next**.

💮 Install Runtime Environment
License Acceptance The license must be accepted before proceeding with the installation.
International License Agreement for Non-Warranted Programs
Part 1 - General Terms
BY DOWNLOADING, INSTALLING, COPYING, ACCESSING, CLICKING ON AN "ACCEPT" BUTTON, OR OTHERWISE USING THE PROGRAM, LICENSEE AGREES TO THE TERMS OF THIS AGREEMENT. IF YOU ARE ACCEPTING THESE TERMS ON BEHALF OF LICENSEE, YOU REPRESENT AND WARRANT THAT YOU HAVE FULL AUTHORITY TO BIND LICENSEE TO THESE TERMS. IF YOU DO NOT AGREE TO THESE TERMS,
* DO NOT DOWNLOAD, INSTALL, COPY, ACCESS, CLICK ON AN "ACCEPT" BUTTON, OR USE THE PROGRAM; AND
* PROMPTLY RETURN THE UNUSED MEDIA AND DOCUMENTATION TO THE PARTY FROM WHOM IT WAS OBTAINED FOR A REFUND OF THE AMOUNT PAID. IF THE PROGRAM WAS DOWNLOADED, DESTROY ALL COPIES OF THE PROGRAM.
1. Definitions
<ul> <li>I accept the terms of all the license agreements</li> <li>I do not accept the terms of all the license agreements</li> </ul>
Image: Mext >     Einish     Cancel

13. Set the Target installation folder to C:\wlp, and then click Finish.

Install Runtime Environment	
Installation Folder Select the folder to install the runtime environment.	
Target installation folder C:\wlp	▼ B <u>r</u> owse
Total size: 50.3 MB Install content: wlp-developers-runtime-8.5.5.0.jar	
(?) < <u>Back</u> <u>Next</u> > <u>Finish</u>	Cancel

14. Click **OK** when presented with a message box stating the installation was successful.

Liberty Profile		
Installatio wlp-de	n was successful: relopers-runtime-8.5.5.0.jar	
	ОК	

15. The next step is to define a new server. To do this click **Next**.

💽 Run On Server					
Liberty Profile Runtime En Specify the installation folde	and JRE.				
Na <u>m</u> e: WebSphere Application Server V8.5 Liberty Profile <u>Installation folder</u> Click to <u>download or install</u> , or provide a path to an installed runtime environment.					
JRE O U <u>s</u> e a specific JRE:	Java70	PTOM25			
Ose def <u>a</u> ult JRE (current)	/ 'Java70')	<u>Configure JREs</u> <u>Advanced options</u>			
?	< <u>B</u> ack <u>N</u> ext >	<u>F</u> inish Cancel			

16. Leave the Server name as *defaultServer* and click Finish.

🔆 Run On Server				
New Liberty Profile Server Specify the name of the new server.				
User <u>d</u> irectory:	WebSphere Application Server V8.5 Liberty Profile			
Server name:	defaultServer			
<u>T</u> emplate:	defaultServer	*		
Server configur	ation:			
🔊 Feature 🗷 HTTP I	e Manager jsp-2.2 Endpoint host=localhost httpPort=9080 httpsPort=9443			
?	< <u>B</u> ack <u>N</u> ext > <u>Finish</u>	Cancel		

17. The server is started and *index.html* file is opened in a browser.



### Summary

In this exercise you have learned how to

- Create a simple web application
- Install the Liberty profile
- Deploy an application to a server

This example illustrates how easy it is to get up and running with the Liberty profile and the WebSphere Application Server Developer Tools for Eclipse.

# **Registration Application**

In this next exercise you create a more complex application. You will learn how to create an application that uses Enterprise Java Beans (EJBs), Context and Dependency Injection (CDI), Java Server Pages (JSP), Java Persistence API (JPA), and Servlets to implement a simple registration application. Enterprise Java Beans provide a mechanism to separate business logic from presentation logic. The first task is to create the JPA entity that will be required by the EJB, and then to create the business logic of the EJB.

### Defining the JPA entity

As we are using WebSphere Application Server Developer Tools for Eclipse we benefit from tools to aid the creation of JPA entities. To do this we will include the JPA project Facet in Eclipse, and then define a new JPA entity called Person.

1. Right click on the RegistrationAppWeb project in Eclipse and select Properties.

😫 Enterprise Explorer 🛛			index.html	🥥 index 🔀
	<b>€</b> 5 ¶	<u>●</u> ▽	(+ -) <b>=</b>	http://localhost:9080/Registrati
RegistrationAppWeb     Age JAX-WS Web Service     Bag RegistrationAppWe		New		۲.
		Go Into		
<ul> <li>Java Resources</li> <li>JavaScript Resources</li> </ul>		Show In		Alt+Shift+W ▶
WebContent		Сору Сору Qu	ualified Name	Ctrl+C
index.html	Ē	Paste		Ctrl+V
Resources	×	Delete		Delete
WebSphere Applicatio	<u>_</u>	Remove	from Context	Ctrl+Alt+Shift+Down
		Build Pa Refactor	th r	► Alt+Shift+T ►
		Import		•
		Export		×
	\$	Refresh Close Pr	oject	F5
		Close Ur	nrelated Projects	
		Validate		
		Show in	Remote Systems vie	w
		Run As		•
		Debug A	4s	
		Team	45	
		Compar	e With	•
< III		Restore	from Local History	
type filter text		Java EE 1	Tools	•
ڬ RegistrationAppWeb		JPA Too	ls	+
		Configu	re	•
		Source		+
	<	Properti	es	Alt+Enter

2. Navigate to the **Targeted Runtimes** section, and ensure the checkbox is selected next to **WebSphere Application Server V8.5 Liberty Profile**.

Properties for RegistrationApp	Council and an o		
type filter text	Targeted Runtimes		<b>⇔ • • •</b>
Resource			
Builders	Web Preview Server Puntime		
Default Package	🕼 WebSphere Application Server V8.5 Liberty Profile		
Deployment Assembly			
Java Build Path			
Java Code Style			
Java Compiler			
Java Editor			
Javadoc Location			
JavaScript			
JSP Fragment			
b Liberty Profile			
Project Facets			
Project References			
Refactoring History			
Run/Debug Settings			
Server			
Service Policies			
Target Device Settings	Snow <u>all</u> runtimes		
Targeted Runtimes		Make Primary	N <u>e</u> w
Task Tags	Runtime composition:		
Validation	< no runtime selected>		
Web Content Settings			
Web Page Editor			
Web Project Settings			8
WikiText			
> XDoclet	If a runtime that you want to select is not displayed or is disabled you may need to uninstall one or more of the curren	tly installed project facets.	
	Uninstall Facets		
		Restore Defaults	Apply
?		ОК	Cancel

3. Next, navigate to the **Project Facets** section. Check the checkbox for **JPA**, and click **OK**. A prompt will appear asking to add the jpa-2.0 feature to the Liberty profile, select **Yes**.



4. Right click on the **RegistrationAppWeb** project and select **New > JPA Entity**.



5. Set the Java package to *com.ibm.websphere.sample.registration.data*, and the Class name to *Person*, then click Next.

New JPA Ent	ity				
Entity class Create a new J	Entity class Create a new JPA entity. Only JPA enabled projects may be selected.				
<u>P</u> roject:	RegistrationAppWeb	•			
Source fol <u>d</u> er:	\RegistrationAppWeb\src	Br <u>o</u> wse			
Java pac <u>k</u> age:	com.ibm.websphere.sample.registration.data	Browse			
Class name:	Person				
<u>S</u> uperclass:		Brows <u>e</u>			
Inheritance					
XML entity m Add to ent Mapping file	appings tity mappings in <u>X</u> ML	B <u>r</u> owse			
?	< <u>B</u> ack <u>Next</u> <u>Finish</u>	Cancel			

6. Click the **Add** button to add a new field to the class.

New JP	A Entity							
Entity Pro	Entity Properties Set entity name, table name, fields, and access type.							
Entity na Table na	Entity name: Person							
<mark> </mark>	lefault ame: Person							
Entity fi	eld <u>s</u>							
Кеу	Name	Туре	<u>A</u> dd <u>E</u> dit <u>R</u> emove					
Access t Fiel <u>d</u> <u>P</u> rope	ype erty							
?	< <u>B</u> ack	Next >	inish Cancel					

7. Click the Browse button to select a Java class type.

8.			
🔘 Enti	ty Fields		X
<u>T</u> ype:			B <u>r</u> owse
Na <u>m</u> e:			
		ОК	Cancel

9. In the filter box enter *String*, and select the String from the java.lang package, then click **OK**.

Choose Type		
Choose the type of the entity field.		•
String		
Matching items:		
<b>⊙</b> <sup>S</sup> STRING		*
G <sup>F</sup> String - java.lang - [Java70]		
G String - org.apache.xpath.operations - [Java70]	10	+
<		Þ.
🖶 java.lang - [Java70]		
0	OK	Cancel
	UK	Cancer

10. Set the Name to Email and click OK.

🔘 Entit	y Fields	×			
<u>T</u> ype:	java.lang.String	B <u>r</u> owse			
Na <u>m</u> e:	Na <u>m</u> e: Email				
	ОК	Cancel			

11. To set this field to a Primary Key (ensuring this field must be unique across all entities), check the checkbox under the **Key** header. Then click the **Add...** button to add a new field.

New JPA	Entity		
Entity Pro	<b>perties</b> name, table name, fields, and	l access type.	<b>G</b>
Entity na <u>r</u> Table na <u>V</u> use da Table na	me efault Person		
Entity fie	lds	Tune	Add
<b>N</b> ey	Email	java.lang.String	<u>E</u> dit <u>R</u> emove
Access ty Fiel <u>d</u> <u>P</u> rope	/pe rty		
?	< <u>B</u> ack	Next > Einish	Cancel

12. Click the Browse button again, and select the String class from the java.lang package again, which should be stored in your history of Matching Items.

Choose Type		x
Choose the type of the entity field.		•
Matching items: OF       String - java.lang		
🖶 java.lang - [Java70]		
? Ок	Cancel	

13. Enter a Name of Name, then click OK.

🔘 Entit	ty Fields	×
<u>Т</u> уре:	java.lang.String	B <u>r</u> owse
Na <u>m</u> e:	Name	]
	ОК	Cancel

14. Click the Add... button again to add the last field to this entity.

New JP	A Entity		
Entity Pr Set entity	<b>operties</b> / name, table name, fields,	and access type.	
Entity na	me: Person		
<mark>▼ U</mark> se o <u>T</u> able r	default name: Person		
Entity fi	eld <u>s</u>		
Key	Name	Туре	<u>A</u> dd
	Email Name	java.lang.String java.lang.String	<u>E</u> dit <u>R</u> emove
Access f Fiel <u>d</u>	type erty		
?	< <u>B</u> ack	Next > <u>Finish</u>	Cancel

15. Select the **Browse** button again.

Entity F	ields	×
<u>Т</u> уре:		Browse
Na <u>m</u> e:		
	ОК	Cancel

16. Type Boolean into the filter box, and select the Boolean class from the java.lang package. Then click **OK**.

Choose Type			x
Choose the type of the entity field.			•
Boolean			
<u>M</u> atching items:			
⊖ <sup>F</sup> Boolean - java.lang			*
Ges Boolean Action			
<b>Boolean</b> ArrayPack			
G <sup>F</sup> BooleanArravSerializer			<b>T</b>
🖶 java.lang - [Java70]			
3	ОК	Cancel	

17. Set a Name of Arrived, then click OK.

🔘 Entit	y Fields	×
<u>T</u> ype:	java.lang.Boolean	B <u>r</u> owse
Na <u>m</u> e:	Arrived	]
	ОК	Cancel

18. The set up of the entity should look as below. Click **Finish** to create the entity.

New JP	A Entity		
Set entit	<b>operties</b> y name, table name, fields,	and access type.	<b>e</b>
Entity na	a <u>m</u> e: Person		
Table n	ame		
Use (	default		
Table	Derron		
Table I	lame: Person		
Entity fi	eld <u>s</u>		
Key	Name	Туре	<u>A</u> dd
	Email	java.lang.String	Edit
	Name	java.lang.String	
	Arrived	java.lang.Boolean	Remove
Access	type		
• Field	ĺ.		
O Prop	erty		
1	K Back	Next > Finish	Cancel
0			Cancer

### **Create the Persistence Unit**

JPA uses Persistence Units to define the types of data required to be stored by your application. We will configure this in an XML file called persistence.xml, which was created for us when the JPA project Facet was added.

1. Expand the JPA Content section of the RegistrationAppWeb project, and double click on the persistence.xml file.



2. Switch to the **Connection** tab, and set the **JTA data source** to *java:comp/env/jdbc/DerbyDataSource*.

index.html 🛛 🎯	) index	🕽 Person.java	🔒 *persistence.xml 🛿	3
Connection				?
Persistence Unit Co	onnection			
Configure the data	source or	JDBC connection prop	erties.	
Transaction type	: De	fault (JTA)	•	
Database				
JTA data source:	ja	va:comp/env/jdbc/Dei	byDataSource	
Non-JTA data so	urce:			
-JDBC connection	n propertie	s		
Populate from c	onnection.	<u></u>		
Driver:			Br	rowse
URL:				
User:				
Password:				
General Connection	Options	Properties Source		

3. Switch to the Properties tab, and click the Add... button.

index.html	🎯 index	🚺 Person.java	🛛 🔒 *persistence.xml 🛛	- 8
Properties				?
This table lists a	ll properties tha	at are defined for this	persistence unit.	
Name	Value		A	\dd
			Re	move

4. Set the **Name** to *openjpa.jdbc.SynchronizeMappings*, and the **Value** to *buildSchema*.

index.html	🎱 index	J Persor	.java	🔒 *persistence.xn	n 🖾 🗖 🗆
Properties					?
This table lists al	l properties tha	at are defined	for this p	ersistence unit.	
Name			Value		Add
openjpa.jdł	oc.Synchronize	Mappings	buildSch	iema	Remove

5. Switch to the **General** tab. Expand the **Managed Classes** section. Check the checkbox for **Exclude unlisted classes**, then click the **Add...** button.

registration	persistence.xml 🔀 🚺 AddAttendee	» <sub>6</sub> □
General		(?
General		
Name:	RegistrationAppWeb	
Persistence provider:		
Description:		
	2 I I I I I I I I I I I I I I I I I I I	
		Add Open Remove
Exclude unlisted cl	asses	Add Open Remove
Exclude unlisted classical statements of the second statement of the secon	asses	Add Open Remove

 In the filter box, type Person. Select the Person class from the package com.ibm.websphere.sample.registration.data, and then click OK.

Class Selection	x
Enter type name prefix or pattern (*, ?, or camel case):	•
Person	
Matching items:	
Person - com.ibm.websphere.sample.registration.data	-
Ge <sup>s</sup> PersonalCertImportFileNamePanel	
PersonalCertsButtonPanel	-
🖶 com.ibm.websphere.sample.registration.data - RegistrationAppWeb/src	
OK   Cancel	

7. Use the **control-S** shortcut to save the file.

### Building the EJB

Enterprise Java Beans (EJBs) are a useful programming model to assist separating Business logic from presentation logic. This application uses an EJB to separate to process of managing an attendee from the presenting of information about attendees. Creating this EJB is the next task.

1. Right click on the **RegistrationAppWeb** and select **New > Other...**.



2. In the New Wizard dialog, expand the section for EJB and select the Session Bean (EJB 3.x). Then click Next.

New	
Select a wizard Create a new EJB 3.x Session Bean	
<u>W</u> izards:	
type filter text	-
(?) < <u>B</u> ack <u>N</u> ext > <u>Finish</u>	Cancel

3. Set the Java package to *com.ibm.websphere.sample.registration.ejb*, and set the Class name to *RegistrationBean*. Click Finish.

Create EJB 3	x Session Bean	
Create EJB 3.: Specify class f	x Session Bean ïle destination.	
<u>P</u> roject:	RegistrationAppWeb -	]
Source fol <u>d</u> er:	\RegistrationAppWeb\src	Br <u>o</u> wse
Java pac <u>k</u> age:	com.ibm.websphere.sample.registration.ejb	Bro <u>w</u> se
Class na <u>m</u> e:	RegistrationBean	
<u>Superclass</u> :		Brows <u>e</u>
State <u>t</u> ype:	Stateless -	]
Create busin	ess interface	
Remote	com.ibm.websphere.sample.registration.ejb.view.RegistrationBeanRemote	
🔲 Local	com.ibm.websphere.sample.registration.ejb.view.RegistrationBeanLocal	
<b></b> No- <u>i</u> nterfa	ace View	
?	< <u>B</u> ack <u>N</u> ext > <u>Finish</u>	Cancel

4. This will open the RegistrationBean. This class requires access to an EntityManager to provide JPA function, so add the following lines just below the class definition:

@PersistenceContext(unitName = "RegistrationAppWeb")
EntityManager em;

5. This will cause errors to appear, as the classes have not been imported. To fix this, begin by clicking on the red X to the left of the <code>@PersistenceContext</code> annotation, and double clicking **Import 'PersistenceContext' (javax.persitence)** from the menu.



 Next hover over the EntityManager and select Import 'EntityManager' (javax.persistence) from the pop-up menu.



7. Now we will add the business methods. First, add the method to register users by placing the following code below the RegistrationBean constructor:

```
public void register(String name, String email) {
   Person p = new Person();
   p.setEmail(email);
   p.setName(name);
   em.persist(p);
}
```

8. Next, add a method to remove users by adding the code below just below the previous one:

```
public void unregister(String email) {
   Person p = em.find(Person.class, email);
   em.remove(p);
}
```

© 2013 Copyright IBM Corporation
9. Next, add the functions to mark whether a person has attended or not by adding the following code below the previous set:

```
public void markAttended(String email) {
   Person p = em.find(Person.class, email);
   p.setArrived(true);
}
public void markUnattended(String email) {
   Person p = em.find(Person.class, email);
   p.setArrived(false);
}
```

10. Add the following code below the previous set, in order to allow searches and retrieval of specific attendees.

```
public List<Person> getPeople() {
  return searchForPeople("*");
}
public Person getPerson(String email) {
  return em.find(Person.class, email);
}
public List<Person> searchForPeople(String searchTerm)
{
  String queryString = "SELECT p FROM Person p WHERE
p.Name LIKE :searchTerm OR p.Email LIKE :searchTerm";
  Query query = em.createQuery(queryString);
  String cleanedSearchTerm = "%";
  if (searchTerm != null) {
    cleanedSearchTerm = "%" +
searchTerm.replaceAll("\\*", "%").toLowerCase() + "%";
  }
  query.setParameter("searchTerm", cleanedSearchTerm);
  @SuppressWarnings("unchecked")
  List<Person> people = query.getResultList();
 return people;
}
```

11. Use the **control-shift-o** shortcut to automatically set up imports. This will present a dialog asking for clarification of which packages to use for specific classes. To begin, select **javax.persistence.Query**, then click **Next**.

Organize Imports	
<u>C</u> hoose type to import:	Page 1 of 2
<ul> <li>javax.management.Query</li> <li>javax.persistence.Query</li> <li>org.apache.openjpa.kernel.Query</li> </ul>	
(?) < <u>Back</u> <u>Mext &gt;</u> <u>Finish</u>	Cancel

12. On the next dialog, select java.util.List, then click Finish.

Organize Imports	
<u>C</u> hoose type to import:	Page 2 of 2
<ul> <li>com.sun.xml.internal.bind.v2.schemagen.xmlschema.List</li> <li>java.awt.List</li> <li>java.util.List</li> </ul>	
( <u>Back</u> ) <u>Next</u> > <u>Finish</u>	Cancel

13. Use the **control-S** shortcut to save the file. The file should resemble the screenshot below.

```
index.html
                                                             🕽 RegistrationBean.java 🔀
             🌚 index
                          J Person.java
                                          persistence.xml
    public class RegistrationBean {
        @PersistenceContext(unitName = "RegistrationAppWeb")
  0
        EntityManager em;
        /**
  \Theta
          * Default constructor.
        public RegistrationBean() {
  Θ
2
            // TODO Auto-generated constructor stub
        3
  Θ
        public void register(String name, String email) {
            Person p = new Person();
            p.setEmail(email);
            p.setName(name);
            em.persist(p);
        }
  Θ
        public void unregister(String email) {
            Person p = em.find(Person.class, email);
            em.remove(p);
        }
  Θ
        public void markAttended(String email) {
            Person p = em.find(Person.class, email);
            p.setArrived(true);
        3
        public void markUnattended(String email) {
  \Theta
            Person p = em.find(Person.class, email);
            p.setArrived(false);
        3
        public List<Person> getPeople() {
           return searchForPeople("*");
        }
  Θ
        public Person getPerson(String email) {
            return em.find(Person.class, email);
        3
  \Theta
        public List<Person> searchForPeople(String searchTerm) {
            String queryString = "SELECT p FROM Person p WHERE p.Name LIKE :searchTerm OR p.Email LIKE :searchTerm";
            Query query = em.createQuery(queryString);
            String cleanedSearchTerm = "%";
            if (searchTerm != null) {
                 cleanedSearchTerm = "%" + searchTerm.replaceAll("\\*", "%").toLowerCase() + "%";
            }
            query.setParameter("searchTerm", cleanedSearchTerm);
            @SuppressWarnings("unchecked")
            List<Person> people = query.getResultList();
            return people;
        3
```

# **Adding Servlets**

To register and edit users, you will use servlets to drive methods of the RegistrationBean. Note that this could also be achieved via JAX-RS, and we will use JAX-RS later in the lab.

#### AddAttendee servlet

1. Right click on the RegistrationAppWeb project, and select New >Servlet.© 2013 Copyright IBM CorporationPage 39 of 118



2. Provide a Java package of *com.ibm.websphere.sample.registration.servlets*, and a Class name of *AddAttendee*. Click Finish.

Create Servi	et 📼	
Create Servle Specify class f	<b>t</b> ile destination.	S
Project:	RegistrationAppWeb	·
Source fol <u>d</u> er:	\RegistrationAppWeb\src	Br <u>o</u> wse
Java pac <u>k</u> age:	com.ibm.websphere.sample.registration.servlets	Bro <u>w</u> se
Class na <u>m</u> e:	AddAttendee	
<u>Superclass</u> :	javax.servlet.http.HttpServlet	Brows <u>e</u>
🔲 <u>U</u> se an exist	ing Servlet class or JSP	
Class na <u>m</u> e:	AddAttendee	Browse
?	< <u>B</u> ack <u>N</u> ext > <u>Finish</u>	Cancel

3. We need to define the data resources used by JPA. These can be defined in multiple locations, in our case we are using an annotation. Add this just below the WebServlet annotation in the AddAttendee class:

<pre>@Resources({</pre>	
<pre>@Resource(name = "jdbc/DerbyDataSource", type</pre>	=
javax.sql.DataSource.class),	
<pre>@Resource(name = "jdbc/NonTxDerbyDataSource",</pre>	type
<pre>= javax.sql.DataSource.class) })</pre>	

- 4. Use the **control-shift-o** shortcut to import the required packages and classes.
- 5. This servlet will be used to register the attendees, so you need to inject the RegistrationBean. To do this, add the following line to the class, just above the constructor:

```
@Inject RegistrationBean rb;
```

6. Hover over the @Inject annotation and select Import 'Inject' (javax.inject) from the dialog.



 Next, hover over the RegistrationBean class and select Import 'RegistrationBean' (com.ibm.websphere.sample.registration.ejb) from the popup menu.



8. Next, add the following lines to the doPost() method of the AddAttendee class:

9. Hover over RequestDispatcher and select Import 'RequestDispatcher' (javax.servlet) from the dialog.



10. Save the file using the **control-S** shortcut. The completed file should match the image below.



#### EditAttendee servlet

As well as adding an attendee, we also need to be able to edit attendees. To do this, we will create the EditAttendee servlet.

 Expand the Java Resources section of the RegistrationAppWeb project, then expand the src directory inside. Next, expand the com.ibm.websphere.sample.registration.servlets package. Right click on the AddAttendee servlet and select New > Servlet.

<ul> <li>Enterprise Explorer &amp;</li> <li>RegistrationAppWeb</li> <li>A JAX-WS Web Services</li> <li>RegistrationAppWeb</li> <li>HPA Content</li> <li>Java Resources</li> <li>A src</li> </ul>		Registration AddAtter package com.ibm.websph @ import java.io.IOExcep @ /** * Servlet implementat */	ndee. ere. tion	☆ ¾ = sample.registration.s ; class AddAttendee
b      com.ibm.websphere.s	New	•		Project
<ul> <li>com.ibm.websphere.s</li> <li>com.ibm.websphere.s</li> </ul>	Open Type Hierarchy	F4	@	Annotation
AddAttendee.java	Show In	Alt+Shift+W	G	Enum
	Open With	•	G	Interface
Javascript Resources     WebContent	🗎 Сору	Ctrl+C	<b>₿</b>	Package
D 🗁 META-INF	Copy Qualified Name		5	HTML File
D 🗁 WEB-INF	🎦 Paste	Ctrl+V	1	JSP File
Index.html b 1 → Resources	🗙 Delete	Delete		Filter
WebSphere Application Server V8	Remove from Context	Ctrl+Alt+Shift+Down	48	Listener
	Build Path	+	6	Servlet
	Source	Alt+Shift+S ►	F\$	Frample

2. Clear the checkbox for **Use an existing Servlet class or JSP**, and provide a **Class name** of *EditAttendee*. Click **Finish**.

Create Servie	et 📃	
Create Servler Specify class f	<b>t</b> ile destination.	S
<u>P</u> roject:	RegistrationAppWeb 🗸	
Source fol <u>d</u> er:	/RegistrationAppWeb/src	Br <u>o</u> wse
Java pac <u>k</u> age:	com.ibm.websphere.sample.registration.servlets	Bro <u>w</u> se
Class na <u>m</u> e:	EditAttendee	
<u>S</u> uperclass:	javax.servlet.http.HttpServlet	Brows <u>e</u>
🔲 <u>U</u> se an exist	ing Servlet class or JSP	
Class na <u>m</u> e:	EditAttendee	Browse
?	< <u>B</u> ack <u>N</u> ext > Finish	Cancel

3. Add the following line just above the constructor of EditAttendee.java:

@Inject RegistrationBean rb;

4. Hover over the @Inject annotation and select Import 'Inject' (javax.inject) from the dialog.



 Hover over the RegistrationBean class and select Import 'RegistrationBean' (com.ibm.websphere.sample.registration.ejb) from the dialog.



- 6. In the doPost () method, add the following code:
- © 2013 Copyright IBM Corporation

```
String name = request.getParameter("fullname");
String email = request.getParameter("email");
String oldemail = request.getParameter("oldemail");
boolean arrived = false;
if(request.getParameter("arrived") != null
   && request.getParameter("arrived").equals("on")){
  arrived = true;
}
rb.unregister(oldemail);
rb.register(name, email);
if (arrived) {
  rb.markAttended(email);
}
RequestDispatcher dispatcher =
request.getRequestDispatcher(
               "ListAttendee.jsp?email="+email);
request.setAttribute("RETURN_MESSAGE", "Updated");
dispatcher.forward(request, response);
```

7. Hover over RequestDispatcher and select Import 'RequestDispatcher' (javax.servlet) from the dialog.



8. Save the file using the **control-S** shortcut. The completed file should match the image below.

```
🚺 AddAttendee.... 🚺 EditAttendee... 💥 🖃
🎯 index.html 🛛 🎯 index 🚺 Person.java 🚽
                                          🔒 persistence.xml 🛛 🚺 Registration...
    package com.ibm.websphere.sample.registration.servlets;
  import java.io.IOException;
  ⊖ /**
     * Servlet implementation class EditAttendee
      *)
    @WebServlet("/EditAttendee")
    public class EditAttendee extends HttpServlet {
        private static final long serialVersionUID = 1L;
        @Inject RegistrationBean rb;
  \Theta
          * @see HttpServlet#HttpServlet()
        public EditAttendee() {
  Θ
            super();
// TODO Auto-generated constructor stub
        3
  Θ
          * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
         */
Θ
        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOExc
            // TODO Auto-generated method stub
         }
  \Theta
          * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
▲ ⊖
        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOEx
2
             // TODO Auto-generated method stub
             String name = request.getParameter("fullname");
             String email = request.getParameter("email");
             String oldemail = request.getParameter("oldemail");
             boolean arrived = false;
            if(request.getParameter("arrived") != null
                && request.getParameter("arrived").equals("on")){
              arrived = true;
             }
             rb.unregister(oldemail);
             rb.register(name, email);
             if (arrived) {
              rb.markAttended(email);
             RequestDispatcher dispatcher = request.getRequestDispatcher(
             "ListAttendee.jsp?email="+email);
request.setAttribute("RETURN_MESSAGE", "Updated");
             dispatcher.forward(request, response);
        }
    }
```

# Adding CDI to the Application

To use CDI we need ensure the beans.xml file is added. One way to add the beans.xml is to add the CDI project facet to the Eclipse project. This ensures the beans.xml file is added and the tooling is configured correctly.

1. Right click on the **RegistrationAppWeb** project, and select **Properties**.

RegistrationAppWeb			
JAX-WS Web Servi		New	+
RegistrationAppW		Go Into	
JPA Content			
🔺 🈕 Java Resources		Show In	Alt+Shift+W ►
⊿ / ع src ⊳ 🖶 com.ibm.w		Сору	Ctrl+C
b 🖶 com.ibm.w		Copy Qualified Name	
a 🌐 com.ibm.w	Ē	Paste	Ctrl+V
AddAtt	×	Delete	Delete
EditAtte	<u>_</u>	Remove from Context	Ctrl+Alt+Shift+Down
D D META-INF		Build Path	
<ul> <li>A Libraries</li> <li>A JavaScript Resource</li> </ul>		Refactor	Alt+Shift+T ►
a 🗁 WebContent		Import	۱.
D > > META-INF D > > WEB-INF		Export	•
index.html	8	Refresh	F5
Kesources WebSphere Application		Close Project	
V Websphere Application		Close Unrelated Projects	
		Validate	
		Show in Remote Systems view	
		Run As	۲.
		Debug As	•
		Profile As	•
		Team	•
		Compare With	•
		Restore from Local History	
type filter text		Java EE Tools	•
RegistrationAppWeb		IPA Tools	
2 11		Configure	
		Source	•
		Properties	Alt+Enter

2. Navigate to the **Project Facets**, and check the checkbox for **Context and dependency injection (CDI)**, then click **OK**.



3. If a prompt appears to install the cdi-1.0 feature, click Yes.

Liberty Profile	×
User Input Required Issues were found while publishing	
One or more applications are missing required features. Do you v feature(s) to the server configuration? Always perform this action Application RegistrationAppWeb requires feature(s): cdi-1.0	vant to add these
Yes No	Cancel

### **Importing JSP resources**

All required JSP files are provided in the **Resources** project.

1. Copy ListAttendee.jsp and ListAttendees.jsp files by dragging and dropping the files from the Web/jsp directory in the Resources project to the WebContent directory of the RegistrationAppWeb project.



Create the Add Attendee page

To drive the AddAttendee servlet, we need to create a new HTML page that contains a form. To create this page, we will use the Web Page editor built into Eclipse.

1. Right click on the **RegistrationAppWeb** project, and select **New > HTML File**.



2. Give the file a name of AddAttendee.html. Click Next.

New HTML File	
HTML Create a new HTML file.	<>
Enter or select the parent folder:	
RegistrationAppWeb/WebContent	
<ul> <li>Image: Image: Section AppWeb</li> <li>Image: Section Section</li></ul>	
File name: AddAttendee.html	
<u>A</u> dvanced >>	
(?) < <u>Back</u> <u>Next</u> > Finish	Cancel

3. Ensure the HTML 5 template is selected. Click Finish.

<mark> </mark>		
Name	Description	*
New HTML File (4.01 frameset)	html 4.01 frameset	
New HTML File (4.01 strict)	html 4.01 strict	
New HTML File (4.01 transitional)	html 4.01 transitional	
New HTML File (5)	html 5	=
New XHTML File (1.0 frameset)	xhtml 1.0 frameset	
New XHTML File (1.0 strict)	xhtml 1.0 strict	-

 $\ensuremath{\textcircled{}^\circ}$  2013 Copyright IBM Corporation

Page 54 of 118

4. In the HTML editor, click the *Palette* icon at the top to open the **Palette** view.

🍥 index	🚺 Person.java	🔒 persistence.xml	Registration	🕽 AddAttendee	J EditAttende	📄 AddAttendee 🔀	<b>»</b> 1
				😳 🗖 🗄	🖻   🗔 🖻 👰 📃	🔲 🕹 🔟 Devi	ce:

5. From the **Palette**, select the **Heading 1** from the **HTML Tags** section, and drag and drop it into the **Design** section.



6. Double click the box it creates, and type in "Add Attendee".



7. Drag the *form* element into the main area just below the heading. Right click on the form tag in the **Source** section, and then select **Properties**.

 $\ominus$ <body> <h1>Add Atter <form></form></h1></body>		Open With Show In	► Alt+Shift+W
	ob	Cut	Ctrl+X
 	Ð	Сору	Ctrl+C
	Ē	Paste	Ctrl+V
		Quick Fix	Ctrl+1
		Source	+
•		Refactor	+
Design Source Split	K	Add to Snippets	
		Properties	
影 Problems 鍋 Servers [ HTML Tags		Open Selection	F3
		M. P. L.	

8. Set the **Action** to *AddAttendee*, and set the **Method** to *Post*.

🔝 Problems 🛛 🔀 Servers 🔲 Properties 🔀 📃 Console 🔮 Palette	
html 🕨 body 🕨 form	
🐼 Tag 🎆 Styles 🛗 Layout 🔲 All	
form	
Id:	
Action: AddAttendee	
Method: 🔘 Get 💿 Post	

9. Switch back to the **Palette** view, and drag a *Text field* element into the **Source** view, between the *form* tags.

	html	
	⊖ <html></html>	
	⊖ <head></head>	
	<meta charset="utf-8"/>	
	<title>Insert title here</title>	
	<h1>Add Attendee</h1>	
	<pre><form action="AddAttendee" methode"post"="">k/1</form></pre>	
		-
	4 11	
	Design Source Split	
	🥷 Probl 👭 Serv 🔲 Prop 📃 Cons 😲 Palette 🔀 📃	
	Description	
	🔁 HTML Tags	$\Leftrightarrow$
	💋 Link	
	The second secon	
	image	
	P Paragraph	
	H1 Heading 1	
	📙 HTML Form Tags	$\Leftrightarrow$
		_
	Check Box	
	Text Area	
$\boldsymbol{\wedge}$	Taxt Field	
<b>۱</b>	1 I EXCITED	
	File Selection Field	

10. Right click on the input tag you just inserted in the **Source** view and select **Properties**. Set the *name* to "name"

🛐 Problen	ns 🕴 Serv	ers 🔲 Prop	erties 🛛	E Console	😳 Palette
html ▶	body ▶ for	m 🕨 input			
🐼 Tag	🌄 Styles 🖣	🗄 Layout 🔳	AII		
input					
Id:					
Name:	name				
Input typ	e: 🖲 Text	Password			

© 2013 Copyright IBM Corporation

11. Under the **All** tab set the **placeholder** property to "Enter name here".

💦 Problems 🛛 👭 Se	ervers 🔲 Properties 🔀 📮 Console  😵 Palette						
html ▶ body ▶ form ▶ input							
💿 Tag 🎆 Styles 📅 Layout 🥅 All							
place							
Property Value							
placeholder Enter name here							

12. Drag another *Text field* into the *form*. Right click on the new input tag and select **Properties**.



13. In the Properties view, set the **name** to "email" in the **Tag** tab, and the **placeholder** to "Enter email here".

💦 Problems 🛛 👭 Servers 🔲 Properties 🔀 📮 C	🛐 Problems 🛛 👭 S	ervers 🔲 Properties 🛿 📃 Co
html ▶ body ▶ form ▶ input	html ⊧ body ⊧	form 🕨 input
🐼 Tag 🎆 Styles 🛗 Layout 🔲 All	💿 Tag 🎆 Styles	ti 🕂 Layout 🔲 All
input	place	
Id:	Property	Value
	placeholder	Enter email here
Name: email		
Input type: 💿 Text 🔘 Password		
Columns:		
Maximum length:		

14. Finally, drag a Submit button into the form.

Add Attendee <pre> Mode Attendee</pre> Add Attendee mation="AddAttendee" method="post"> input type="text" name="name" placeholder="Enter name here"> input type="text" name="email" placeholder="Enter email here"> input type="text" placeholder="email here"> input type="text" placeholder="emai	
💦 Problems 🕴 Servers 🔲 Properties 💻 Console 🚱 Palette 🔀	- 8
🔁 HTML Tags	∞
🖉 Link	
🔣 Image	
P Paragraph	
H1 Heading 1	
📙 HTML Form Tags	$^{\odot}$
Form	

15. Use the **control-S** shortcut to save the page. The completed for should match the image below.

index.html	🎯 index	🕽 Person.java	🔒 persistence.xml	J	Regist	ration	<u></u>	Add
						ē 🖷	:::	
Add A	ttend	ee						
			Submit Query	/				
+								
<pre><!--DOCTYP<br-->@ <html> @ <head> <meta cha<br=""/><title>In </title></head> @ <body> <h1> @ </h1></body> </html></pre>	E html> arset="ISO- nsert title Add Attende m action="A <input type<br=""/> <input type<br=""/> type="s rm>	8859-1"> here e ddAttendee" meth ="text" name="na ="text" name="en ubmit">	nod=" <i>post"&gt;</i> me" placeholder="En mail" placeholder="N	nter Ente	name r emai	here"> il here	"> <in< td=""><td>put</td></in<>	put
Design Source	Split							

# Update the landing page

In a previous step we created the landing page index.html. We now need to update this so that the pages we need to return to are easily available.

- 1. Expand the WebContent folder in the RegistrationAppWeb project, and double click the index.html file to open it.
- 2. Update the contents of the <body> tag to be

```
<h1>Hello World</h1>
<a href="AddAttendee.html">Add Attendee</a><br>
<a href="ListAttendees.jsp">List Attendees</a>
```

3. Use the **control-S** shortcut to save the file.

## **Configure the server**

Now all the resources are in place to add and edit attendees. We will now configure the server. This involves adding the derby.jar to allow JDBC function, and configuring the server's configuration file, server.xml.

 Open the WebSphere Application Server V8.5 Liberty Profile project, and expand the shared folder. Right click on the resources directory, and select New > Folder.



2. Call the folder derby. Click Finish.

<ul> <li>Resources</li> <li>WebSphere Application Server V8.5 Liberty Profile</li> <li>.settings</li> <li>servers</li> <li>shared</li> <li>apps</li> </ul>
i config
resources
Folder <u>n</u> ame: derby
<u>A</u> dvanced >>
Finish Cancel

3. Open the **Resources** project and expand the Jars folder. Drag and drop the derby.jar from this folder into the shared/resources/derby folder you just created in the **WebSphere Application Server V8.5 Liberty Profile** project.

a 🗁 Resources
Business
4 🗁 Jars
derby.jar
🔺 🗁 Web
> 🗁 dojo
html
🔺 🗁 jsp
Audit.jsp
a 🗁 WebSphere Application Server V8.5 Liberty Profile
E servers
🔺 🗁 shared
🗁 apps
🗁 config
resources
🕞 derby

- 4. Open the **Servers** view and expand the **WebSphere Application Server V8.5 Liberty Profile at localhost** server definition.
- © 2013 Copyright IBM Corporation



5. Double click on the Server Configuration. This opens the server.xml configuration tool.

index.html	🎯 index	🕽 Person.java	🔒 persistence.xml	Registration	server.xml 🔀	<b>»</b> 3	- 8
🐁 Server C	onfigurati	on: defaultSer	ver (server.xml)				
Configuration	Structure		↓ <sup>a</sup> ⊞ ⊡	Configuration Details			æ
Define the ma	ain contents of	the configuration in	this section.	Description: new server	r		*
type filter te	ext						-
⊿ 👸 Serv ≝ F ⊠ F	er Configuratio eature Manage HTTP Endpoint	er	Remove				
	Application Mo Web Applicatio	n n	Up Down				
Design Source							

6. Click on the Server Configuration element, then click Add.



7. Using the filter box, type *JDBC*, then select the **JDBC Driver** element and click **OK**.

Add Element	
Select the element to add Select the element to add from the list below.	
Context: Server Configuration	
jdbc	
JDBC Driver	
Description: Identifies a JDBC driver.	
ОК	Cancel

8. Set the **Id** to be *DerbyDriver*.

Configuration Structure		JDBC Driver	
Define the main contents of the confi this section.	guration in	ID:	DerbyDriver
type filter text  type filter text  Server Configuration  Feature Manager  HTTP Endpoint  Application Monitoring  Web Application  JDBC Driver	Add Remove Up Down	Shared library reference: XADataSource class: ConnectionPoolDataSource implementation class: DataSource implementation class:	▼ Add ▼

9. Right click on the JDBC Driver and select Add > Shared Library.

Configuration Structure			↓ <mark>a</mark> ⊞ ⊟		JDBC Driver	
Define the main contents of this section.	fthe	configuration in			ID:	
type filter text	on Jer t onitoring		Add Remove Up		Shared library reference: XADataSource class: ConnectionPoolDataSource im DataSource implementation cl	
JDBC Driver	4	Add	Down	▶ 📎	Shared Library	
	×	Remove		Г		
	<b>앞</b>	Mo Mo	<b>ve Up</b> ve Down			
	_					

10. Right click on the **Shared Library** and select **Add > File**.

Configuration Structure			↓ <mark>a</mark> 🕀 🕞	5	Shared Library
Define the main contents of t	the co	onfi	guration in	I	Name:
					Description:
type filter text					Fileset reference:
🔺 🐘 Server Configuration	n		Add		
Feature Manage Image: Manage Image	r		Remove		
Application Mor	nitorii	ng	d		
Web Application	n				
JDBC Driver			Down		
🛞 Shared Library				Allowed API type	
	<ul> <li>♣ A</li> <li>★ R</li> <li>★ M</li> </ul>		dd 🕨 🕨	8	File
			emove	۲	Fileset
			love Up	۲	Folder
	Ŧ	Μ	love Down		
	_			-	

11. Click on the File element, and select the Browse button under File.

File	
Name*:	Browse

12. Select the *shared.resource.dir* from the left hand navigation pane. Expand the derby folder, click on the derby.jar and then click **OK**.

Browse for File Location Specify the file location Search for a file location by search a relative resource on the	electing a path variable on the left
type filter text Relative Path server.config.dir server.output.dir shared ann dir	Path: C:\wlp\usr\shared\resources
<u>S</u> shared.config.dir	Enter the path to the file or select from above:
Wip.user.dir	derby/derby.jar
Selected location:	\${shared.resource.dir}/derby/derby.jar
?	OK Cancel

13. Click on the **Server Configuration** and click the **Add** button.



© 2013 Copyright IBM Corporation

14. Using the filter, type *data* and then select the **Data Source** element and click **OK**.

Add Element
Select the element to add
Select the element to add from the list below.
Context: Server Configuration
data
Data Source     Matching elements     Authentication Data     HTTP Session Database
Description: Defines a data source configuration.
OK Cancel

15. Set the **Id** to *jdbc/DerbyDataSource*, and the **JNDI Name** to *jdbc/DerbyDataSource*. Set the **JDBC driver** to *DerbyDriver* using the dropdown.

Data Source	
ID:	jdbc/DerbyDataSource
JNDI name*:	jdbc/DerbyDataSource
JDBC driver reference:	DerbyDriver
Connection manager reference:	▼ Add ▼
Туре:	<b>.</b>
Connection matching:	MatchOriginalRequest -
Default container managed authentication data:	▼ Add ▼
Transaction isolation level:	
Cached statements per connection:	10
Participate in transactions	
Enlist scrolling APIs	
Enlist vendor APIs	
Commit or roll back on cleanup:	
Query timeout:	
Recovery authentication data reference:	▼ Add ▼
Synchronize query and transaction timeouts	
Supplemental JDBC trace	

16. Right click on the data source you have just created and select Add > Derby Embedded Properties.

type filter text					JNDI name*:	jdł
a 🐻 Server Configuration			Add	•	JDBC driver reference:	Der
😕 Feature Manager 📨 HTTP Endpoint			Remov	/e	Connection manager reference:	
<ul> <li>Application Monitoring</li> <li>Web Application</li> <li>JDBC Driver</li> <li>Shared Library</li> </ul>			Up Dowr	1	Type: Connection matching: Default container managed authentication data:	Ma
Data Source: idbc/Derby	-+-C	ourco			I ransaction isolation level:	
	÷	Add	+	-	Connection Manager	
3	×	Remove		0	DataDirect Connect for JDBC (sqlserver) Properties	
	ŵ	Move Un		0	DB2 for i Native JDBC Driver Properties	
	Move Do		wn	0	DB2 for i Toolbox JDBC Driver Properties	
	~	more bo		0	DB2 JCC Properties	
					Default Container Managed Authentication Data	
				0	Derby Embedded Properties	
				0	Derby Network Client Properties	
				0	Generic JDBC Driver Properties	
				0	Informix JCC Properties	
				0	Informix JDBC Driver Properties	
				0	JDBC Driver	
				0	Microsoft SQL Server JDBC Driver Properties	
				0	Oracle Properties	
				6	Recovery Authentication Data	
				0	Sybase Properties	
				_		_

17. Set the **Database name** to *RegistrationDB*, and the **Create database** dropdown to *create*.

Derby Embedded Properties						
Create database:	create		-			
Database name:	RegistrationDB					
Connection attributes:						
Login timeout:						
Password:			Set			
Shutdown database:			-			
User:						
Additional properties:	Кеу	Value	Add			
			Edit			
			Remove			

18. Save the file using the **control-S** shortcut. The configuration should resemble the image below.



### **Running the application**

Now the application is ready, we can run it in the server and test it.
8. Go to **Window > Web browser**, and select *Firefox* from the list of options. This redirects any web pages opened in Eclipse to the Firefox web browser instead, which is required for some of the technologies being used in this lab.

er V8.5 Liberty Profile,	/servers/defaultServer/server.xn	nl - Ecl	ipse		
h Project Run 🛛	/indow Help		_		
\$ <b>• () • (</b>	New Window New Editor Hide Toolbar		*		ndex.htm ∋≺servo
	Open Perspective Show View	+			⊖ <
re.sample.registratio re.sample.registratio re.sample.registratio iva va	Customize Perspective Save Perspective As Reset Perspective Close Perspective Close All Perspectives Navigation	•			<.
	Web Browser	•	$\checkmark$	0 Internal Web Browser	
	Preferences			1 Default system web browser	
-				2 Firefox	
				3 Internet Explorer	

9. Go to the **Console** view. The application will be started, and you should have a line like this:

[AUDIT ] CWWKT0016I: Web application available (default\_host): http://localhost:9080/RegistrationAppWeb/

10. Click the hyperlink in the **Console** view to open the landing page of the application.

Firefox 🔻	
index	+
÷ >	Socalhost:9080/RegistrationAppWeb/
Hell Add Atter List Atter	<b>World</b> ndee ndees

11. To add an attendee, click the **AddAttendee** link on the landing page.

Firefox 🔻					
Insert title he	ere +				
♦ Iocalhost:9080/RegistrationAppWeb/AddAttendee.html					
Add A	<b>ttendee</b> ere Enter email here <b>Submit Query</b>				

Note that the placeholder text may not appear, depending on the browser used.

12. Provide an email address and name for the user, then click **submit**. This will take you to the ListAttendee.jsp page, but display an error message.



- 13. This is because the EJBLite feature is not enabled, meaning your EJB is not being run correctly. To fix this we need to enable the missing feature in server.xml. Open server.xml.
- 14. Click on the Feature Manager section under the server Configuration, then click the Add... button under the detailed Feature Manager panel.



15. From the dialog, type *ejb* into the filter box. Then, select the **ejbLite-3.1** feature, and then click **OK**.

Add Features			
Select the features to ena Select one or more features	able to enable.		
ejb			
Feature	Name		
🔊 appSecurity-1.0	Application Security		
🛞 appSecurity-2.0	Application Security		
💌 ejbLite-3.1	Enterprise JavaBeans Lite		
Description: This feature ena subset of the EJB 3.1 specific Enables: jndi-1.0	ables support for Enterprise JavaBeans written to the EJB Lite ation.		
Enabled by: webProfile-6.0			
?	OK Cancel		

- 16. Save the file using the **control-S** shortcut.
- 17. Reopen the homepage by clicking on the link in the console.
- 18. Try adding a new Attendee again. This time it works, but the page you are taken to shows empty fields.

Firefox <b>T</b>	
Attendee	+
♦ ♦ localhost:9080/F	RegistrationAppWeb/AddAttendee
Attendee	
E-mail	
Name	
Arrived? 🔲 Submit	
Back to main page	

- 19. This is because the RegistrationBean is not marked as @Named, and so is not being managed correctly. Open the RegistrationBean class.
- 20. Between the *@*LocalBean annotation and the class definition line, add the @Named annotation. To import this annotation, hover over it and select Import 'Named' (javax.inject) from the dialog.



- 21. Save the file using the **control-S** shortcut.
- 22. Return to the landing page, and click the **Add Attendee** link again.
- 23. Fill out the details of a **different** Attendee, and click **Submit**. This will take you to the **List Attendee** page, where you can update details.

Firefox 💌
Attendee Ross +
♦ Iocalhost:9080/RegistrationAppWeb/AddAttendee
Attendee Ross
E-mail pavittr@uk.ibm.com
Name Ross
Arrived? 🔲 Submit
Back to main page

24. Mark the attendee as arrived, and click **Submit**. You will get a message that says "Updated", and the checkbox will remained checked.

Firefox 🔻
Attendee Ross +
♦ Iocalhost:9080/RegistrationAppWeb/EditAttendee
Attendee Ross
E-mail pavittr@uk.ibm.com
Name Ross
Arrived? 🗹 Submit
Updated
Back to main page

25. Return to the landing page and click the **List Attendees** link. This shows a list of users you have added and their current state of attendance.



#### Summary

In this section you learned:

- How to create and deploy an EJB
- How to enable CDI
- How to configure the Liberty profile server for JPA

# Adding RESTful services using JAX-RS

JAX-RS allows you to write a RESTful interface that uses POJOs to communicate to clients. Web requests are made to classes with annotations that identify them as handling different HTTP requests. We will use JAX-RS here to enable a search facility.

## Adding the client-side capability

To enable searching we are going to use client-side technologies such as Dojo. The **Resources** project contains Dojo modules, plus some modules in the registration folder that are used to send requests to a web server and receive responses. These resources will be added to the Web project so they form part of our application.

 Copy the SearchForAttendees.html file from the Web/html folder of the Resources project into the WebContent folder of the RegistrationAppWeb project.

- A 2 RegistrationAppWeb
  - JAX-WS Web Services
  - RegistrationAppWeb
  - JPA Content
  - Java Resources
    - 🔺 进 src
      - a 🌐 com.ibm.websphere.sample.registration.data
        - a 🚺 Person.java
          - Person
      - a 
         com.ibm.websphere.sample.registration.ejb
        - RegistrationBean.java
      - a 
         com.ibm.websphere.sample.registration.servlets
        - AddAttendee.java
        - EditAttendee.java
      - META-INF
        - persistence.xml

#### Libraries



- AddAttendee.html
- index.html
- ListAttendee.jsp
- ListAttendees.jsp
- Resources
  - Business
    - 🕞 Jars
  - 🔺 🗁 Web
- - 👂 🗁 dojo

🔺 🕞 html

SearchForAttendees.html

- 15D Audit.jsp
- 2. Copy the dojo folder from the Web folder of the Resources project by dragging it into the WebContent directory of the RegistrationAppWeb project.

- a 🞥 RegistrationAppWeb
  - JAX-WS Web Services
  - RegistrationAppWeb
  - JPA Content
  - a 🈕 Java Resources
    - 🔺 进 src
      - a 🌐 com.ibm.websphere.sample.registration.data
        - a 🚺 Person.java
          - b G Person
      - a 🌐 com.ibm.websphere.sample.registration.ejb
        - RegistrationBean.java
      - a 🖶 com.ibm.websphere.sample.registration.servlets
        - AddAttendee.java
        - EditAttendee.java
      - a 🗁 Meta-Inf
        - 🔒 persistence.xml
    - 👂 🛋 Libraries
  - JavaScript Resources
- WebContent
   META-INF
   MANIFEST.MF
   WEB-INF
   AddAttendee.html
   index.html
   ListAttendee.jsp
   ListAttendee.jsp
   SearchForAttendees.html
   SearchForAttendees.html
   Methods
  - → Jars → <del>→ Web</del> → <del>→</del> dojo
- 3. Open the index.html file in the **RegistrationAppWeb** project, and update the <body> tag to be:

```
<hl>Hello World</hl>
<a href="AddAttendee.html">Add Attendee</a><br />
<a href="ListAttendees.jsp">List Attendees</a><br />
<a href="SearchForAttendees.html">Search For
Attendees</a><br />
```

4. Save the file using the **control-S** shortcut.

# Adding JAX-RS to the Web Project

The SearchForAttendees.html page is now available. However, when you attempt to search, you will receive an error like:

Firefox T
http://localhost:90chForAttendees.html +
Search for: *
Error: RequestError: Unable to load rest/search?filter=* status: 404
Back to main page

We are going to solve this by providing a JAX-RS class that will answer these requests.

1. Right click on the **RegistrationAppWeb** project and select **Properties**.

Project Explorer 🙁 🛛	= <b>4</b>	5 S T T	
<ul> <li>RegistrationAppWeb</li> <li>Deployment Descrip</li> <li>JAX-WS Web Servic</li> </ul>		New Go Into	۶.
<ul> <li>IPA Content</li> <li>Ipava Resources</li> </ul>		Show In	Alt+Shift+W ►
<ul> <li>JavaScript Resource</li> <li>Geometric</li> <li>Geometric</li> <li>Geometric</li> <li>Metric</li> </ul>		Copy Copy Qualified Name Paste	Ctrl+C Ctrl+V
<ul> <li>WEB-INF</li> <li>AddAttendee.ht</li> <li>index.html</li> <li>ListAttendee.jsp</li> </ul>	2	Delete Remove from Context Build Path Refactor	Delete Ctrl+Alt+Shift+Down Alt+Shift+T •
<ul> <li>ListAttendees.js</li> <li>SearchForAttendes.js</li> <li>Resources</li> <li>Employee Business</li> </ul>	8	Import Export	۱ ۱
<ul> <li>Jars</li> <li> &gt; &gt; Web</li> <li> WebSphere Application</li> </ul>		Refresh Close Project Close Unrelated Projects	F5
		Validate Show in Remote Systems view Run As Debug As Profile As Team Compare With Restore from Local History Java EE Tools JPA Tools Configure Source	
		Properties	Alt+Enter

2. On the **Project Facets**, check the checkbox for **JAX-RS (REST Web Services)**.

Properties for RegistrationApp	Web		
type filter text	Project Facets		
<ul> <li>Resource         Builders         Default Package         Deployment Assembly         Java Build Path         Java Code Style         Java Editor         Java2 Compiler         Java2 Compiler</li></ul>	Configuration: <custom>         Project Facet          ✓       Context and dependency injection (CDI)         C CXF.2x. Web Services          Default style sheet (CSS file)          Default style sheet (CSS file)          Dynamic Web Module          ✓       Java         ✓       JavaScript         ✓       JavaScript         ✓       JAx/RS (REST Web Services)         ✓       JAX         ✓       OSGi Bundle         ✓       WebDoclet (XDoclet)</custom>	Version           1.0           1.0           1.0           1.0           1.0           1.0           1.0           1.0           2.0           1.2.3	Save As Delete  Details <u>Runtimes</u> Axis2 Web Services  Enables Web services generation through the Axis2 Web services engine.
⊳ XDoclet			Revert
?			OK Cancel

- 3. Click the link at the bottom labelled **Further configuration required...**.
- 4. Click Add.

Modify Faceted Project					
JAX-RS Capabilities					
Add JAX-RS capabilities to t	his Web Project				
JAX-RS Implementation Lib	prary				
Type: IBM WebSphere App	plication Server Liberty JAX-RS Library	•			
Update Deployment Desc	riptor				
Servlet Information					
JAX-RS servlet name:	JAX-RS Servlet				
JAX-RS servlet class name:	com.ibm.websphere.jaxrs.server.IBMRestServlet				
<u>U</u> RL mapping patterns:	/jaxrs/*	Add <u>R</u> emove			
<u> </u>					

5. Specify the **URL Pattern** as /rest/\* and click **OK**.

Specify URL Pattern	×	
URL Pattern		
/rest/*		
	OK Cancel	

- 6. Click **OK** to exit the **Modify Faceted Project** dialog.
- 7. Click **Apply**, you are prompted to add the jaxrs-1.1 feature. Click **Yes**.

Liberty Profile	×
User Input Required Issues were found while publishing	
One or more applications are missing required features. Do feature(s) to the server configuration?	you want to add these :-1.1
? Yes No	D Cancel

8. Click OK.

### Creating the JAX-RS application

The JAX-RS application consists of a class that handles making the application available, a class that performs the request handling, and some configuration in the web.xml file. Adding the JAX-RS project facet provides most of the web.xml configuration, however we still need to complete that and also add the other classes.

1. Right click on the **RegistrationAppWeb** project and select **New > Class**.



2. Set the package name to *com.ibm.websphere.sample.registration.rest*, and the **Class name** to *SearchAttendees*. Click *Finish*.

New Java Class		
Java Class Create a new Java	class.	C
Source fol <u>d</u> er:	RegistrationAppWeb/src	Br <u>o</u> wse
Pac <u>k</u> age:	com.ibm.websphere.sample.registration.rest	Bro <u>w</u> se
Enclosing type:		Browse
Na <u>m</u> e: Modifiers:	SearchAttendees	
<u>S</u> uperclass:	java.lang.Object	Brows <u>e</u>
Interfaces:		<u>A</u> dd
		Remove
Which method stub	os would you like to create?	
	public static void main(String[] args)	
	Constructors from superclass	
Do you want to add	✓ In <u>h</u> erited abstract methods I comments? (Configure templates and default value <u>here</u> ) ○ Generate comments	
?	Finish	Cancel

3. Right click on the com.*ibm.webshere.sample.registration.rest* package and select **New > Class**.

<ul> <li>         ■ RegistrationAppWeb         ■ Deployment Descriptor: RegistrationAppWeb         ■ AX-WS Web Services         ■ AAX-WS Web Services         ■ JAX-WS Web Services         ■ JAX-WS Web Services         ■ JAX-WS Web Services         ■ Java Resources         ■ Mark Services         ■ Com.ibm.websphere.sample.registration.data         ■ ∰ com.ibm.websphere.sample.registration.data         ■ ∰ com.ibm.websphere.sample.registration.data     </li> </ul>	<pre>package com.ibm.websphere.sample.registra public class SearchAttendees { }</pre>	tio /	<ul> <li>Com.ibm.websph</li> <li>GearchAttendees</li> </ul>
∠ Group Communication Comm	New +	1	Project
com.ibm.websphere.sample.registration.servle	Go Into	@	Annotation
META-INF Libraries	Open Type Hierarchy F4	G	Class
JavaScript Resources	Show In Alt+Shift+W 🕨	G	Enum
A 🗁 WebContent	Conv. Ctrl+C	Ø	Interface

#### 4. Set the name to *SearchAttendeesApplication*.

New Java Class	- O I Rectification and 1	
Java Class Create a new Java	class.	C
Source fol <u>d</u> er:	RegistrationAppWeb/src	Br <u>o</u> wse
Pac <u>k</u> age:	com.ibm.websphere.sample.registration.rest	Bro <u>w</u> se
Enclosing type:		Bro <u>w</u> se
Na <u>m</u> e: Modifiers:	SearchAttendeesApplication	
<u>Superclass</u> :	java.lang.Object	Brows <u>e</u>
Interfaces:		<u>A</u> dd
		<u>R</u> emove
Which method stub	os would you like to create?	
	public static void main(String[] args)	
	Constructors from superclass           Inherited abstract methods	
Do you want to add	comments? (Configure templates and default value <u>here</u> )	
?	<u> </u>	Cancel

5. Click the **Browse** button next to the **Superclass** field. Using the filter box, type *Application*, and select the Application class from the *javax.ws.rs.core* package. Click **OK**.

Lhoose a type:	
Application	
<u>A</u> atching items:	
G <sup>A</sup> Application - javax.faces.application - C:\wlp\dev\api	\spec\com.ibm.ws.javaee.jsf.2.0_1.0.2.jar
Application - javax.ws.rs.core - C:\wlp\dev\api\spec\c	com.ibm.ws.javaee.jaxrs.1.1_1.0.1.jar
Application organachowink common model wad	<u>Ciluda/deu}</u> ani/third_narth/com/ihm/unhr
javax.ws.rs.core - C:\wlp\dev\api\spec\com.ibm.ws.jav	vaee.jaxrs.1.1_1.0.1.jar

- 6. Click Finish.
- 7. Add the following method to the SearchAttendeesApplication class:

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

8. Hover over the Set class and select Import 'Set' (java.util) from the dialog.



9. Hover over HashSet and select Import 'HashSet' (java.util) from the dialog.



10. Save the file using the **control-S** shortcut. The completed class should match the following image.



11. Open the web.xml file in the WebContent/WEB-INF folder of the RegistrationAppWeb project.



- 12. Right click on the **Servlet** element on the left pane and select **Add > Initialization Parameter**.
- © 2013 Copyright IBM Corporation

#### 🔞 Web Application 3.0 Deployment Descriptor Editor

)verview			E	Details	
type filter text				Display Name:	
Web Application (Registratio)	nApp	Web) Add		Servlet Name*:	JA
Servlet (JAX-RS Servlet)	÷	Add 🕨	<b>[</b> +	Initialization Parameter	
🧐 Serviet Mapping (JAX-KS	×	Remove	۹,	Run As	5
	全	Move Up	2 D	Security Role Reference	1
	7	Move Down		Description:	JA
	_		-		

13. In the **Details** panel on the right, set the **Name** to *javax.ws.rs.Application*. In the **Value**, enter

com.ibm.websphere.sample.registration.rest.SearchAttendeesApplication.

Details		Ţ.
Name*:	javax.ws.rs.Application	
Value*:	com. ibm. we bs phere. sample. registration. rest. Search Attendees Application	
Description:		*
		-

- 14. Save the file using the **control-S** shortcut.
- 15. Open the SearchAttendees.java class.
- 16. Add the following annotation to the SearchAttendees class declaration:

@Path("search")



17. Save the file using the **control-S** shortcut. Note that the file contains an import error. We will resolve this during the next few steps.

#### Creating a JAX-RS GET request handling method

The JAX-RS application is now in place. However, it does not contain any methods that are configured to handle requests. The first one we need to add is to handle GET requests. These will come from searches.

1. Add the following method into the SearchAttendees class to enable the search request to be processed:

2. To receive requests via HTTP GET requests, and return a collection of Person objects as JSON, add the following annotations to the method:

3. Inject the RegistrationBean EJB so that we can look up a list of users based on the filter by adding this line to the SearchAttendees class:

```
@Inject RegistrationBean rb;
```

4. Change the return statement of the searchForAttendees() method to call the RegistrationBean.

return rb.searchForPeople(searchTerm);

5. Lastly, update the filter parameter with the @QueryParam annotation:

© 2013 Copyright IBM Corporation

6. The file at this stage should resemble the image below.



- 7. To resolve the import errors, use the **control-shift-o** keyboard shortcut. This will cause a prompt for ambiguous imports.
- 8. For the Produces annotation, select javax.ws.rs.Produces. Click Next.

🔘 Organi	ze Imports	
<u>C</u> hoose t	ype to import:	Page 1 of 2
@ jav	ax.ws.rs.Produces	
?	< <u>B</u> ack <u>N</u> ext > <u>Finish</u>	Cancel

9. For MediaType class, select javax.ws.rs.core.MediaType. Click Finish.

Organize Imports	
<u>C</u> hoose type to import:	Page 2 of 2
Java.awt.PageAttributes.MediaType	
(?) < <u>Back</u> <u>Next</u> > <u>Finish</u>	Cancel

10. Save the file using the **control-S** shortcut. The file should resemble the image below. Ensure the import statements match.



#### **Creating JAX-RS POST requests**

Now that the search handler is in place, the search function should work (you can try this out by opening the Search For Attendees page in Firefox and setting the search term to "\*").

Firefox •
http://localhost:90chForAttendees.html +
<ul> <li>♦ ♦ localhost:9080/RegistrationA ☆ マ C</li> <li>♦ Google</li> <li>A I </li> <li>♥ </li> </ul>
Search for: *
Error: RequestError: Unable to load rest/search?filter=* status: 404
• Remove Attendee <u>Ross (pavittr@uk.ibm.com) - Not Arrived</u>
• Remove Attendee <u>Steve (steve@uk.ibm.com) - Arrived</u>
Back to main page

Each search result displayed on the page includes a checkbox to mark whether the attendee is attending or not, and a button to allow the attendee to be unregistered. However clicking these currently returns a 404 error. This is because no handlers have been defined for these requests.

1. Open the SearchAttendees java class and copy the method below into the class to allow processing of POST requests with a URL of */search/update*. We still return a string of JSON text, but this time the return type is Person, not Collection<Person>, so the returned JSON string only contains the details of one person object (the one we have updated).

```
@POST
     @Produces (MediaType.APPLICATION JSON)
     @Path("/update")
     public Person changeAttendance(
                @QueryParam("email") String emailAddress,
                @QueryParam("state") boolean attending)
                                            throws Exception {
       String email =
               emailAddress.substring("checkbox".length());
       if (attending)
         rb.markAttended(email);
       else
         rb.markUnattended(email);
       return rb.getPerson(email);
© 2013 Copyright IBM Corporation
                                                    Page 96 of 118
```

- }
- 2. Add the following method to the SearchAttendees class to allow the delete request to be processed, and to return a representation of the person who was unregistered:

- 3. Resolve any import issues by using the **control-shift-o** shortcut again.
- 4. Save the file using the **control-S** shortcut. The class should resemble the image below.

```
🚺 SearchAttendees.java 🔀  🛞 server.xml
                                      SearchAttendeesApplication.java
                                                                       🥹 v
    package com.ibm.websphere.sample.registration.rest;
  import java.util.Collection;
    import javax.inject.Inject;
    import javax.ws.rs.GET;
    import javax.ws.rs.POST;
    import javax.ws.rs.Path;
    import javax.ws.rs.Produces;
    import javax.ws.rs.QueryParam;
    import javax.ws.rs.core.MediaType;
    import com.ibm.websphere.sample.registration.data.Person;
    import com.ibm.websphere.sample.registration.ejb.RegistrationBean;
    @Path("search")
    public class SearchAttendees {
        @Inject RegistrationBean rb;
  Θ
        @GET
        @Produces(MediaType.APPLICATION_JSON)
        public Collection<Person> searchForAttendees
           (@QueryParam("filter") String searchTerm) {
            return rb.searchForPeople(searchTerm);
        }
        @POST
  @Produces(MediaType.APPLICATION_JSON)
        @Path("/update")
        public Person changeAttendance(
                   @QueryParam("email") String emailAddress,
                   @QueryParam("state") boolean attending)
                                              throws Exception {
          String email =
                  emailAddress.substring("checkbox".length());
          if (attending)
            rb.markAttended(email);
          else
            rb.markUnattended(email);
          return rb.getPerson(email);
        }
  Θ
        @POST
        @Produces(MediaType.APPLICATION_JSON)
        @Path("/delete")
        public Person changeAttendance(
                   @QueryParam("email") String emailAddress)
                                              throws Exception {
          String email =
                    emailAddress.substring("button".length());
          Person p = rb.getPerson(email);
          rb.unregister(email);
          return p;
        }
    }
```

© 2013 Copyright IBM Corporation

5. Return to the landing page in your browser, and select the **Search Attendees** link. Perform the search again. Now, attempt to toggle the checkbox for the attendee. This causes the attendee to be updated as shown below.

Firefox 🔻	
http://loca	Ihost:90chForAttendees.html +
<b>←</b> → [	Iocalhost:9080/RegistrationAppWeb/SearchForAttendees.html
Search for:	*
• 🛛 🗌	Remove Attendee (Updated) Alasdair (not@uk.ibm.com) - Arrived
• 🗹 🗌	Remove Attendee Ross (pavittr@uk.ibm.com) - Arrived
Back to mai	n page

6. Remove one of the attendees by clicking the **Remove Attendee** button. This removes the button and marks the attendee as removed.

Firefox T
http://localhost:90chForAttendees.html +
♦ ♦ localhost:9080/RegistrationAppWeb/SearchForAttendees.html
Search for: *
<ul> <li>Remove Attendee (Updated) Alasdair (not@uk.ibm.com) - Arrived</li> <li>(Removed) Ross (parittr@uk.ibm.com) - Arrived</li> </ul>
Back to main page

7. Performing a new search shows the attendee has gone.

Firefox T
http://localhost:90chForAttendees.html +
♦ Iocalhost:9080/RegistrationAppWeb/SearchForAttendees.html
Search for: *
• Remove Attendee <u>Alasdair (not@uk.ibm.com) - Arrived</u>
Back to main page

#### Summary

In this section you learned:

- How to enable JAX-RS support for an application
- How to interact with client side technologies using AJAX techniques

# Adding another EJB

Currently our application uses one EJB to handle the business logic of the application.

In this section, we will add another function, which will log use of certain business logic methods on the RegistrationBean. As we want to keep our logging separate, we will use a separate EJB to manage that data access.

1. Open the Business/view folder of the Resources project, right click on the AuditEntry class and click on Move.



#### 2 . Expand the RegistrationAppWeb project down to

src/com/ibm/websphere/sample/registration/data.



Page 101 of 118

- 3. Click on the data folder and click **OK**. This is the same folder that contains the Person.java class.
- 4. Open the Business folder of the Resources project, click on the AuditBean class. Next, right click and select Move.

-			
J SearchAtte		Move	
D SearchAtte Com.ibm.web		Rename F2	2
META-INF	2	Import	
Libraries	പ	Export	
Services			_
JavaScript Resources	8	Refresh FS	;
▲ B WebContent		Validate	
META-INF		Show in Remote Systems view	
🖌 🗁 WEB-INF		Run As	+
🔁 lib		Debug As	+
🗴 beans.xml		Profile As	+
AddAttendee.htm		Java EE Tools	+
index.html		Team	+
ListAttendee.jsp		Compare With	
ListAttendees.jsp		Replace With	
SearchForAttende			
a 🗁 Resources		Web Services	1
▲ Business		Properties Alt+Ente	r
J AuditBean.java	_	alco	_
🗁 Jars		<	
🖻 🗁 Web			
WebSphere Application S	erver	V8.5 Liberty Profile 🛛 🕅 Markers 🔲 Properti	ᆥ

5. Expand the **RegistrationAppWeb** project down to

src/com/ibm/websphere/sample/registration/ejb.

Move Resources		x
Choose destination for 'AuditBean.java':		
▲		-
🗁 .apt_generated		
> 🗁 .settings		
a 🧁 src		=
🔺 🗁 com		
🔺 🗁 ibm		
🔺 🧁 websphere		
a 🥭 sample		
a 🥭 registration		
data		
ejp		-
Previe <u>w</u> > OK	Canc	el

- 6. Click on the ejb folder and click **OK**.
- 7. Open the RegistrationBean class and add the injection for the AuditBean:

```
@Inject AuditBean ab;
```

public class RegistrationBean {

8. Hover over the @Inject annotation and select Import 'Inject' (javax.inject) from the dialog.

9. Next, add audit lines to the register and unregister methods:

```
public void register(String name, String email) {
     ab.logUpdate("Registering attendee "
                         + name + " with email " + email);
  public void unregister(String email) {
     ab.logUpdate("Unregistering attendee with email "
                                                         + email);
 public void register(String name, String email) {
    ab.logUpdate("Registering attendee
           + name + " with email " + email);
    Person p = new Person();
    p.setEmail(email);
    p.setName(name);
    em.persist(p);
}
 public void unregister(String email) {
    ab.logUpdate("Unregistering attendee with email "
           + email);
    Person p = em.find(Person.class, email);
    em.remove(p);
 }
10. Save the file using the control-S shortcut.
```

- 11. Open the persistence.xml file.
- 12. Switch to the **General** tab, and click the **Add...** button.

🎯 web.xml 💿 🚺 Regi	istration	🔒 persistence.xml 🔀	» <sub>3</sub>	- 8
General				?
General				
Name:				
Persistence provider:				
Description:				
<ul> <li>Managed Classes</li> </ul>				
Specify the list of class	es to be manag	ged in this persistence un	it.	
Gcom.ibm.websphe	re.sample.regi	stration.data.Person		Add
				Open
				Remove
Exclude unlisted clas	sses			
XML Mapping Files				
JAR Files				

#### 13. Type Audit in the filter box, and select AuditEntry from the

com.ibm.websphere.sample.registration.data package.

Class Selection	
Enter type name prefix or pattern (*, ?, or camel case):	•
Audit	
Matching items:	
<ul> <li>G AuditBean</li> <li>G AuditCallback</li> <li>G <sup>F</sup> Audited</li> </ul>	
G AuditEntry - com.ibm.websphere.sample.registration.dat	ta
AuditLogger     AuditManager	
eom.ibm.websphere.sample.registration.data - Registratio	onAppWeb/src
?	OK Cancel

14. Save the file using the  ${\bf control-S}$  shortcut. @ 2013 Copyright IBM Corporation

- 15. Open the landing page, click on the **Add Attendees**, and add a couple of attendees.
- 16. Copy the Audit.jsp file into the application by dragging it from the Web/jsp folder of the Resources project, and dropping it into the WebContent folder of the RegistrationAppWeb project.



- 17. Navigate to the page using the URL

<u>http://localhost:9080/RegistrationAppWeb/Audit.jsp</u>. This contains a line for the register of an attendee that you just performed. It will also show any unregisters you perform, and any edits to an attendee which will show the attendee being removed and then re-registered.

Fire	efox 🔻		index ×	AuditLog	×	+
÷	>	•	localhost:9080/RegistrationAppWeb/Audit.jsp			

# Audit Log

ID	Date	Message
1	Sun Jun 16 06:37:13 BST 2013	Registering attendee Adam with email adamg@us.ibm.com

2 Sun Jun 16 06:37:50 BST 2013 Registering attendee Tim with email deboer@ca.ibm.com Back to main page

# Securing the application

Applications may handle sensitive data such as personal information, and may require the use of SSL to ensure secure transportation of data over a network. The following section will add SSL to the application, and ensure it is enforced in all communication between the application and the client.

# Add an SSL certificate

1. Open the Servers view, right click on the server and select Utilities > Create SSL Certificate.



2. Provide a password for your keystore, then click Finish.

Create SSL Certific	ate			
Create SSL Certificate Create an SSL certificate for this server				
Create a default sec	ure socket laye	er (SSL) certificate to use with this server.		
Keystore <u>p</u> assword:	•••••		Show	
E <u>n</u> coding:	xor (encode using XOR) 👻			
Key:				
Specify validity period (days):		365		
Specify subject (I	DN):	CN=R9VHY9H,OU=defaultServer,O=ibr	n,C=us	
?		Finish	Cancel	

 A Console window will appear, containing the configuration for SSL. Highlight the configuration section in the console, and copy it into memory using the control-C shortcut.



- 4. In the **Servers** view, expand the server definition to show the **Server Configuration** element.
- 5. Right click on the Server Configuration element and select Open.
6. Switch to the **Source** tab, and paste the contents of the clipboard using the **control-V** shortcut.



7. Save the file using the **control-S** shortcut.

### Configuring the application to use transport guarantees

The configuration above will enable the use of SSL, but it does not force the application to only work over SSL, meaning it can be accessed via HTTP. However, we can configure this in the application to insist on using the CONFIDENTIAL transport guarantee. Be aware that there is one more configuration step required after this to ensure the Liberty profile server also enforces SSL only.

- Open the web.xml file of the WebContent/WEB-INF folder of the RegistrationAppWeb project.
- 2. Click on Web Application (RegistrationAppWeb), and click the Add button.

🔞 Web Application 3.0 Deployment Desc	riptor Editor	
Overview		General
type filter text		Display
<ul> <li>Web Application (RegistrationAppWeb)</li> <li>Servlet (JAX-RS Servlet)</li> <li>Initialization Parameter (javax.ws.rs.Appl</li> <li>Servlet Class (com.ibm.websphere.jaxrs.:</li> <li>Servlet Mapping (JAX-RS Servlet)</li> </ul>	Add Remove Up Down	Version <sup>3</sup> Distribur Metadal Module Descript

3. Using the filter, type *Security*, and select the **Security Constraint** element. Click **OK**.

Add Item	
Select item to add to Web Application (Registr	rationAppWeb):
security	
Security Constraint	
Contraction Contractica Contra	
ОК	Cancel

4. Click the newly created **Web Resource Collection** element, and click **Add** next to the **URL Pattern.** 

) SearchAttendees.java 🛛 🔯 server.xml 🛛 🚺 Se	archAttendeesApplicat	ion.j 🔇 🕲 *web.xml	🛿 🚺 RegistrationBean.java	🔒 persistence.xml 🛛 🖻 🖻
Web Application 3.0 Deployment D	escriptor Editor	1 error detected		?
Overview		Details		
type filter text		Web Resource Name*:		$\frown$
Web Application (RegistrationAppWeb)	Add	URL Pattern*:		Add
Web Resource Collection ()	Remove			Remove
Servlet (JAX-RS Servlet)	Up			Up
<ul> <li>Gervlet Class (com.ibm.websphere.ja</li> </ul>	xrs.: Down		•	Down
Servlet Mapping (JAX-RS Servlet)		Description:		

5. Type the pattern as "/\*". Provide a **Web Resource Name** of *Secure Connection*.

n.j 🔇 📎 *web.xml	🛛 🚺 RegistrationBean.java	🔒 persistence.xml	
			?
Details			Ē
Web Resource Name*:	Secure Connection		
URL Pattern*:	/*		Add
			Remove
			Up
	<	4	Down
Description:			

6. Right click on the Web Resource Collection in the left pane, and add a HTTP Method. In the Details section set the HTTP Method to GET.

Overview			Det	ails	
type filter text			Wel	o Resource Name*:	Secure Connec
eb Application (RegistrationAppWeb)	וך	Add	URL	. Pattern*:	/*
Security Constraint ()	- 11	Remove			
Web Resource Collection (Secure Connection)			0		
Servlet (JAX-RS Servlet)	۰.	Add 🕨	C	HITP Method	
Initialization Parameter (javax.ws.rs.Application)	) 🗙 F	Remove	🚱 HTTP Method (		Omission
<ul> <li>Gervlet Class (com.ibm.websphere.jaxrs.server.IE</li> <li>Fervlet Mapping (JAX-RS Servlet)</li> </ul>	Â	Move Up	Γ		•
	-	Move Down	es	cription:	

7. Right click on the **Web Resource Collection** in the left pane again, add another **HTTP Method**. In the **Details** section set the **HTTP Method** to *POST*.

Overview				Details	
type filter text				Web Resource Name*:	S
Application (RegistrationAppWeb)		Add		URL Pattern*:	1
curity Constraint ()		Remove			
Web Resource Collection (Secure Connection)	_	Kentove			4
HTTP Method (GET)	÷	Add	►	HTTP Method	
rvlet (JAX-RS Servlet)	×	Remove			T"
<ul> <li>Initialization Parameter (javax.ws.rs.Application) Servlet Class (com.ibm.websphere.jaxrs.server.If rylet Mapping (JAX-RS Servlet)</li> </ul>	Ŷ	Move Up Move Down		Description:	•
	~		_	1	

8. Click on the **Security Constraint**. Scroll down to the **User Data Constraint** (optional) section, and change the **Transport Guarantee** dropdown to *CONFIDENTIAL*.

SearchAttendees.java 🔊 server.xml 🛛	SearchAttendeesApplication.j	🛛 🎱 *web.xml 🔀	RegistrationBean.java	🔒 persistence.xml	- e
Web Application 3.0 Deployme	nt Descriptor Editor				?
Overview	Detai	ls			Ē
type filter text	Displa	ay Name:			
<ul> <li>Web Application (RegistrationAppWeb</li> <li>Security Constraint ()</li> </ul>	) Add	uthorization Constr	aint (optional)		R
Web Resource Collection (Security HTTP Method (GET)	re Conne Role	Name:			Add
HTTP Method (POST)	Ор			F	Remove
Servlet (JAX-RS Servlet)	Down				Un
Initialization Parameter (Javax.w	/s.rs.Appl				
Servlet Mapping (JAX-RS Servlet)	Elejaxis.	•		<b>)</b>	Down
	Des	cription:			*
					-
	<b>•</b> U	ser Data Constraint	(optional)		
	Trar	sport Guarantee*:	CONFIDENTIAL		-

9. Save the file using the **control-S** shortcut.

### Configure security on the Liberty profile

The final stage to enable security is to ensure the security feature is enabled on the Liberty profile.

1. In the **Servers** view, expand the server definition to show the **Server Configuration** element.

© 2013 Copyright IBM Corporation

- 2. Right click on the Server Configuration element and select Open.
- 3. Switch to the **Design** tab.
- 4. Expand **Server Configuration** and click on any **Feature Manager** element (there may be more than one).
- 5. In the right pane, click the **Add** button.

SearchAttendees.java 🛛 🛞 server.xml 🙁 🚺	SearchAttendeesApplication.ja	🛞 web.xml	RegistrationBean.java	🔒 persistence.xml	- 6
Server Configuration: defaultServ	ver (server.xml)				
Configuration Structure	↓ª 🕀 🕞 Feature I	Manager		_	
Define the main contents of the configuration in	this section. Feature:	Feature	Name		Add
type filter text	Add Remove Up Down	<ul> <li>★ cdi-1.0</li> <li>★ ejbLite-3.1</li> <li>★ jaxrs-1.1</li> <li>★ jpa-2.0</li> <li>★ jsp-2.2</li> <li>★ localConnector-1.0</li> </ul>	Contexts and Dependency Enterprise JavaBeans Lite Java API for XML RESTful Java Persistence API JavaServer Pages Local JMX Connector	y Injection Web Services	Remove
JDBC Driver		Description:			
Data Source: jdbc/DerbyDataSource		Enables:			

6. From the dialog, select appSecurity-2.0. Click OK.



Page 113 of 118

7. Save the file using the **control-S** shortcut.

#### **Testing security**

The final stage is to attempt to connect to the server via an unsecured port, and ensure your traffic is redirect to the SSL port.

- 1. In Firefox, open a connection to <a href="http://localhost:9080/RegistrationAppWeb">http://localhost:9080/RegistrationAppWeb</a>.
- 2. If this succeeds, you will be redirected to the SSL port (port 9443).
- 3. Accept the security exception caused by the unknown certificate. This is we are using the certificate you generated in a previous step.
- 4. Ensure the page displays as expected, the protocol is set to HTTPS, and the port is the secured 9443 HTTPS port.

Firefox 🔻	
index	× 🖸 AuditLog
+ + https://localhost:9	443/RigistrationAppWeb/
Hello World	
Add Attendee List Attendees Search For Attendees	

### Summary

In this section you learned:

- How to secure the Liberty profile server for secure communication using SSL
- How to enforce SSL only traffic on applications

# Packaging the server for deployment

The Liberty profile contains the *Minify* function. This allows you to build a customised Liberty package that contains only the features you require to run the applications you have installed.

1. In the **Servers** view, shutdown the server by right clicking on the server and selecting **Stop**.



2. Right click on the server in the Server view, and select Utilities > Package Server.

🖲 Markers 🔲 Properties 🖓 Servers 🛠 🙀 Data Source Explorer 🕓 Spinnets 📃 Console	Utilities	•	Create SSL Certificate
Web Preview Server [Stopped]	Properties	Alt+Enter	Generate Web Server Plug-in
WebSphere Application Server V8.5 Liberty Profile at localhost [defaultServer] [Stopped, Synchronized]			Package Server
RegistrationAppWeb [Stopped, Synchronized]			Join Collective
B Server Configuration [server.xml] new server			
			Generate Dump for Support

3. Click Browse next to the Archive name. Choose a folder, and set the name to package.zip. Under the Include section, change the dropdown to Minimal Runtime (minify), and click Finish.

Packag	ge Server	x
Package Package	e a server into a distributable archive	
Package and/or a <u>A</u> rchive C:\Re	e a server to create an archive file containing all of the binaries, server configura applications required to distribute the server. /e egistration\package.zip	ition,
Ove Include:	erwrite existing files	]•
	Server configuration and applications are always included	
?	Finish Cance	

© 2013 Copyright IBM Corporation

Page 115 of 118

4. Once the packaging process is complete and the package.zip file is created, use Windows Explorer to navigate to the folder package.zip was created. Right click on the file and select Extract All.

(C:) ► Registration ►		•	Search Registratio	n
New folder				
Name		Date modified	Туре	Size
퉬 eclipse		16/06/2013 05:47	File folder	
퉬 workspace		16/06/2013 01:27	File folder	
🚹 package.zip		16/06/2012 06:58	Compressed (zipp	46,515 KB
🚹 Resources.zip	Open		Compressed (zipp	14,590 KB
📓 wlp-developers-runti	Open in new win	dow	Executable Jar File	51,492 KB
	Extract All			
	Extract to "packa	ge\"		
	Add to "package	zin"		

- 5. Follow the instructions to extract the zip file.
- 6. Open a command prompt, from the start menu and navigate to the location where the package.zip file was extracted to.



7. Navigate to the bin directory inside the extract location, and then run the command server start defaultServer.



8. Navigate to <u>http://localhost:9080/RegistrationAppWeb</u>. This will redirect to the secure port. You are now accessing your application from the packaged minified server.



## Summary

In this lab you learned:

- How to install the IBM WebSphere Application Server V8.5.5 Liberty profile
- How to create and deploy a simple web application using the IBM WebSphere Application Server Developer Tools for Eclipse V8.5.5.
- How to create and deploy a simple registration web application that uses Servlets, JPA, EJBs, Context and Dependency Injection, and JAX-RS.
- How to secure applications to use SSL.

© 2013 Copyright IBM Corporation

• How to generate a customised Liberty profile image, and use that image to deploy your application outside of a development environment.

If you are interested in learning more please visit <u>http://wasdev.net</u>. WASdev is the developer focussed community for WebSphere Application Server developers, providing:

- Useful articles on getting started
- Samples and tutorials of specific features
- Configuration snippets
- The latest releases of available Early Access Programs for Liberty and related products.
- Forums for finding further information from other developers, and getting answers to questions.