



Red Hat JBoss Developer Studio 8.1

Start Developing

Tutorial for first time users

Red Hat JBoss Developer Studio 8.1 Start Developing

Tutorial for first time users

[Red Hat Customer Content Services](#)

Legal Notice

Copyright © 2015 Red Hat, Inc.

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This document details how to start developing applications with JBoss Developer Studio.

Table of Contents

CHAPTER 1. START DEVELOPING WITH JBOSS DEVELOPER STUDIO	3
1.1. ABOUT START DEVELOPING	3
1.2. ABOUT THE TICKETMONSTER APPLICATION	3
1.3. TUTORIAL ASSUMPTIONS	3
1.4. TUTORIAL OUTLINE	4
CHAPTER 2. CREATING THE TICKETMONSTER APPLICATION	5
2.1. OPEN THE JBOSS PERSPECTIVE	5
2.2. CONFIGURE MAVEN TO USE THE JBOSS EAP MAVEN REPOSITORY	5
2.3. SET UP A SERVER USING RUNTIME DETECTION	6
2.4. CREATE A JAVA EE WEB PROJECT USING JBOSS CENTRAL	7
2.5. ADD AN ENTITY USING FORGE TOOLS	9
2.6. ADD AN ENTITY USING HIBERNATE TOOLS	11
2.7. PREPARE EVENT RECORDS TO POPULATE THE DATABASE	14
2.8. DEPLOY THE APPLICATION USING JBOSS SERVER TOOLS	15
2.9. ACCESS AND QUERY THE DATABASE	17
2.10. ADD A RESTFUL WEB SERVICE	19
2.11. ADD A USER INTERFACE OPTIMIZED FOR MOBILE DEVICES	22
2.12. TEST THE MOBILE USER INTERFACE USING BROWSERSIM	26
APPENDIX A. REVISION HISTORY	27

CHAPTER 1. START DEVELOPING WITH JBOSS DEVELOPER STUDIO

1.1. ABOUT START DEVELOPING

Start Developing is a step-by-step tutorial that introduces JBoss Developer Studio to new users. It aims to give you a taste of developing applications with JBoss Developer Studio. The tutorial here demonstrates how to create a web application, TicketMonster, that includes typical enterprise functionality by using JBoss Developer Studio.

In guiding you through the tutorial steps, a number of assumptions are made about your system. You must ensure your system complies with the assumptions before proceeding to complete the tutorial. For information, see [Section 1.3, “Tutorial Assumptions”](#).

The information in this tutorial is minimal. It describes one way of completing each step in the tutorial; in fact, there may be several alternatives for each step in the tutorial. For more information about alternative ways to complete tasks, see the [Red Hat JBoss Developer Studio 8.1 documentation](#) available on the Red Hat Customer Portal.

[Report a bug](#)

1.2. ABOUT THE TICKETMONSTER APPLICATION

TicketMonster is an example web application demonstrating ticket purchasing. The application provides an online environment in which users can purchase tickets for events and administrators can manage the data relating to events and ticket sales. You can see a running example of the TicketMonster application at <https://ticketmonster-jdf.rhcloud.com/>.

The application consists of a user interface and web services. The user interface allows users to purchase tickets for listed events and it enables administrators to access and modify event and ticket information. The user interface is optimized for desktop and mobile clients. The web services enable access to information about members, events and venues stored in a database.

TicketMonster demonstrates the combining of Red Hat and JBoss technologies and frameworks to build, test and deploy applications. For example, the running instance of TicketMonster at <https://ticketmonster-jdf.rhcloud.com/> is hosted in the cloud on the OpenShift platform.

[Report a bug](#)

1.3. TUTORIAL ASSUMPTIONS

The TicketMonster application is built on a Maven-based Java EE Web project. To ensure successful completion of the tutorial, a number of assumptions are made about your system:

- JBoss Developer Studio 8.1 stand-alone or Bring-Your-Own-Eclipse (BYOE) is installed; for instructions, see [Install Red Hat JBoss Developer Studio](#).
- JBoss Enterprise Application Platform 6.x is installed
- Corresponding JBoss Enterprise Application Platform 6.x Maven repository is installed but Maven is not necessarily configured to use it

[Report a bug](#)

1.4. TUTORIAL OUTLINE

The diversity and complexity of the TicketMonster application makes it ideal for demonstrating some of the capabilities of JBoss Developer Studio. This tutorial sets the stage for building TicketMonster. It shows you how to develop a number of the TicketMonster application features using JBoss Developer Studio:

- Open the JBoss perspective
- Configure Maven for use with JBoss Central projects
- Set up a server using JBoss Server Tools runtime detection
- Create a Java EE web project to which the TicketMonster functionality is added using JBoss Central
- Create **Event** and **Venue** entity classes for the database entities using Forge Tools and Hibernate Tools
- Prepare event records to populate the database at runtime
- Deploy the application on a runtime server using JBoss Server Tools
- Access and query the database holding the member, event and venue records
- Add a RESTful web service so that the application can retrieve the information in the database
- Add a user interface to the application that is optimized for mobile devices using Mobile Web Tools
- Test the mobile user interface using BrowserSim

You must work through these tasks in the order they are presented because the earlier tasks are prerequisites for the later tutorial tasks.



NOTE

For information about building the complete TicketMonster application using JBoss Developer Studio see <http://www.jboss.org/ticket-monster/> on the JBoss Developer website.

[Report a bug](#)

CHAPTER 2. CREATING THE TICKETMONSTER APPLICATION

2.1. OPEN THE JBOSS PERSPECTIVE

When following this tutorial, you must use the **JBoss** perspective. Specific views, toolbars, and menu options most used by JBoss developers are available when this perspective is open. Instructions in the tutorial are given based on the **JBoss** perspective and if you use an alternative perspective you might not be able to find some of the items as detailed in the tutorial procedures. The procedure below details how to open the **JBoss** perspective.

Procedure 2.1. Open the JBoss Perspective

1. Click **Window** → **Open Perspective** → **Other**.
2. From the list of perspectives, double-click **JBoss**.

[Report a bug](#)

2.2. CONFIGURE MAVEN TO USE THE JBOSS EAP MAVEN REPOSITORY

The Java EE Web project, on which the tutorial TicketMonster application is built, is constructed from an enterprise Maven archetype. Maven is distributed with JBoss Developer Studio so it is not necessary to install Maven but, to be able to use the Java EE Web project enterprise archetype, Maven must be correctly configured to use the JBoss EAP 6.x Maven repository. Instructions are given here for editing your Maven configuration file from within JBoss Developer Studio to achieve this.

Procedure 2.2. Configure Maven to use the JBoss EAP Maven Repository

1. Click **Window** → **Preferences**, expand **JBoss Tools** and select **JBoss Maven Integration**.
2. Click **Configure Maven Repositories**.
3. Click **Add Repository**.
4. Click **Recognize JBoss Maven Enterprise Repositories**.
5. Navigate to **path/to/jboss-eap-6.x.y-maven-repository** and click **OK**. JBoss Maven Tools recursively scans the path searching for the Maven repository.
6. Modify the information in the **ID** and **Name** fields as desired, ensure the **Active by default** check box is selected and click **OK**.

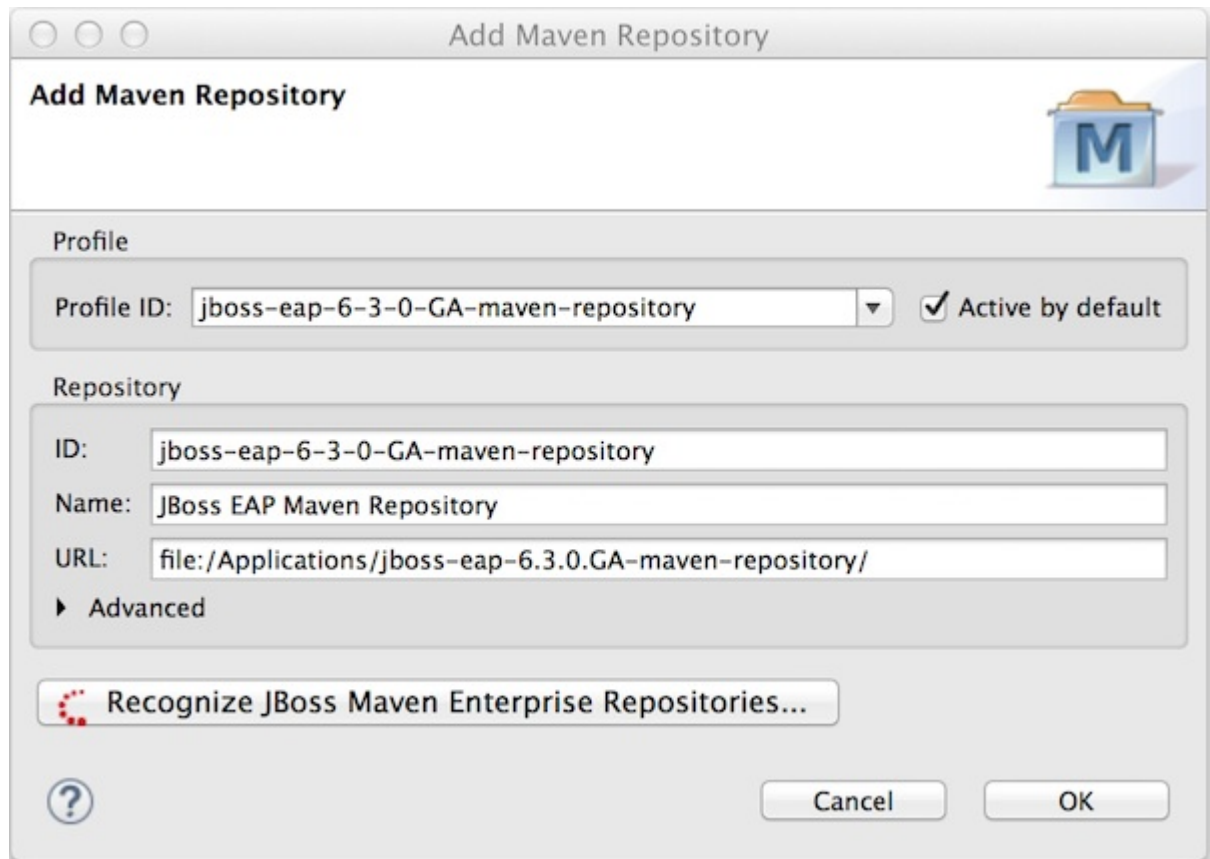
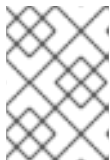


Figure 2.1. JBoss EAP Maven Repository Listed for Adding to the Maven Configuration File

- Click **Finish** and at the prompt asking if you are sure you want to update the Maven configuration file click **Yes**. If the specified configuration file does not exist, JBoss Maven Tools creates it.



NOTE

Maven settings, such as the configuration file, are specified in **Preferences** under **Maven** → **User Settings**. These settings can be customized.

- Click **Apply** and click **OK** to close the **Preferences** window.

[Report a bug](#)

2.3. SET UP A SERVER USING RUNTIME DETECTION

Servers can be defined from within the **Java EE Web Project** wizard. But for the purposes of exploring JBoss Developer Studio the procedure below defines the server first using JBoss Server Tools runtime detection.

Procedure 2.3. Define a Server Using Runtime Detection

- Click **Window** → **Preferences**, expand **JBoss Tools** and select **JBoss Runtime Detection**.
- Click **Add**.

3. Select the directory containing the **JBoss EAP 6.x** installation and click **OK**. The directory is now scanned for application servers and JBoss EAP 6.x found.

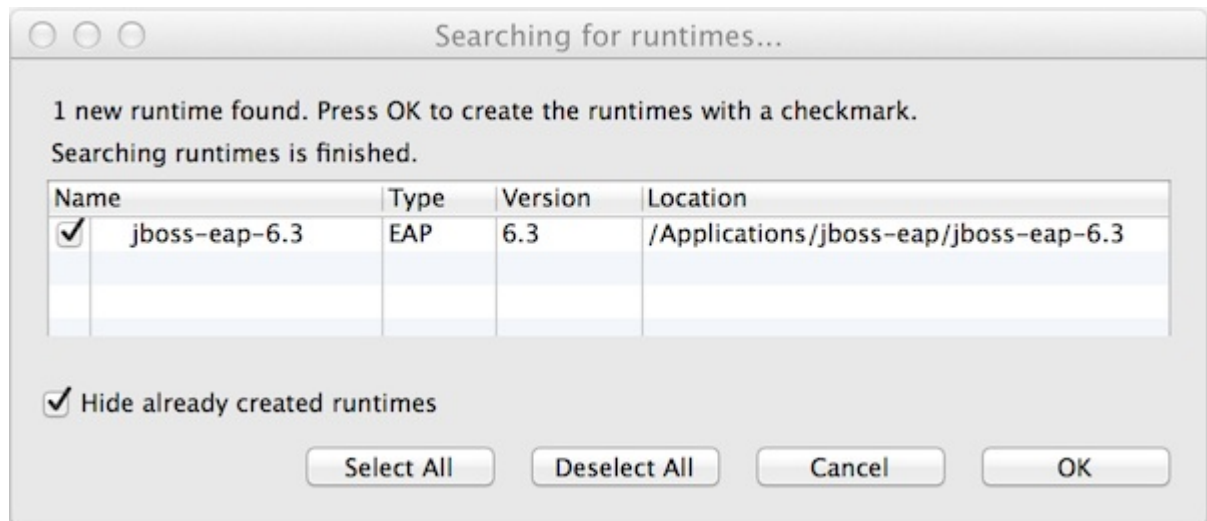


Figure 2.2. JBoss EAP 6.x Found by Runtime Detection

4. To create a server for JBoss EAP 6.x, ensure the **jboss-eap-6.x** check box is selected and click **OK**.
5. Click **Apply** and click **OK** to close the **Preferences** window. The server is listed in the **Servers** view.



Figure 2.3. JBoss EAP 6.x Server Listed in Servers View

6. The server is initially shown in stopped mode. Start the server by right-clicking **jboss-eap-6.x** and clicking **Start**. After a short pause, the view switches to the **Console** view and shows the startup output of the server.

[Report a bug](#)

2.4. CREATE A JAVA EE WEB PROJECT USING JBOSS CENTRAL

The TicketMonster application is based on a Java EE web application. A wizard is provided for creating this type of project in JBoss Central and the procedure below guides you through using the wizard.

Procedure 2.4. Create a Web Application Using the Java EE Web Project Wizard

1. Click the **JBoss Central** tab.
2. In JBoss Central under **Start from scratch**, click **Java EE Web Project** to open the **Java EE Web Project** wizard. If JBoss Central is not visible, click **Help** → **JBoss Central**.



Figure 2.4. Java EE Web Project in JBoss Central

- From the **Target Runtime** list, select **jboss-eap-6.x Runtime** and click **Next**.

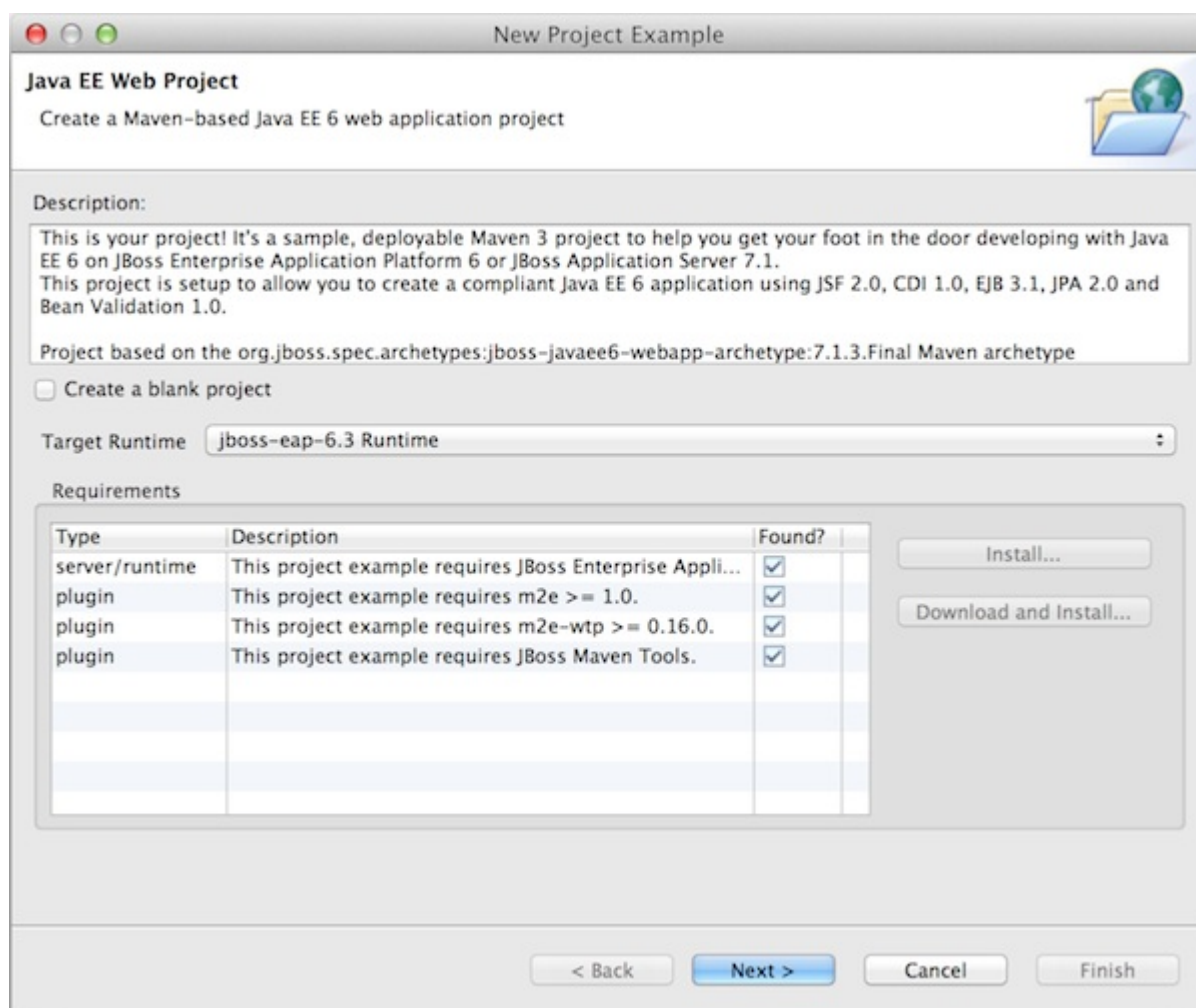


Figure 2.5. JBoss EAP 6.x Selected as Target Runtime in Java EE Web Project Wizard

- Complete the fields about the project as follows:
 - In the **Project name** field, type **ticket-monster**.
 - In the **Package** field, type **org.jboss.jdf.example.ticketmonster**.

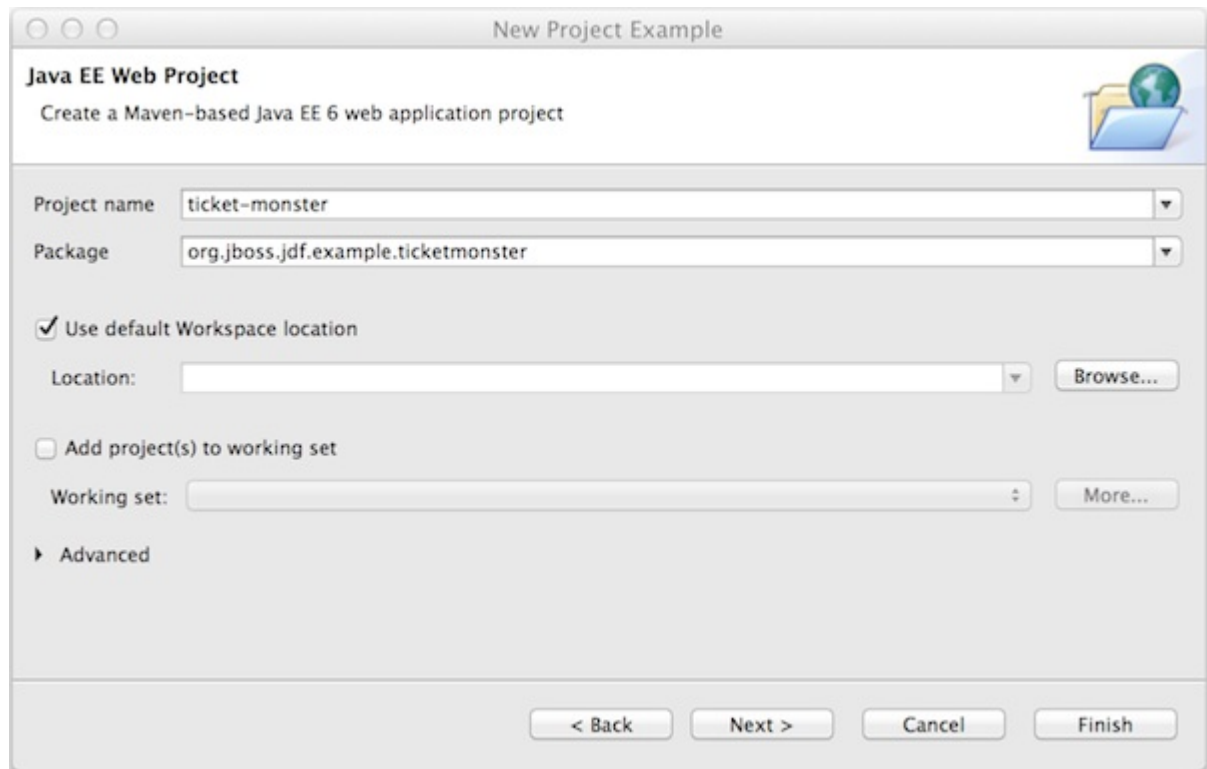


Figure 2.6. Completed Project Name and Package Fields in Java EE Web Project Wizard

5. Click **Finish** to create the project.

During project creation, the wizard imports project dependencies. When the **Java EE Web Project** wizard displays '**Java EE Web Project**' **Project is now ready**, click **Finish** to close the wizard. A **README.md** file for the project automatically opens for viewing and the project is listed in the **Project Explorer** view.

[Report a bug](#)

2.5. ADD AN ENTITY USING FORGE TOOLS

A new entity class can be added to the TicketMonster project using Forge Tools. In the procedure below an **Event** entity class is created to hold information about the events for which users can buy tickets. This **Event** entity has id, name, description, major and picture fields.

Procedure 2.5. Add an Entity Using Forge Tools

1. In the **Project Explorer** view, expand **ticket-monster** → **Java Resources** → **src/main/java**.
2. Click **org.jboss.jdf.example.ticketmonster.model** and press **Ctrl+4** (**Cmd+4**).
3. From the Forge wizards list, click **JPA: New Entity**.
4. In the **Entity name** field, type **Event** and click **Finish**.

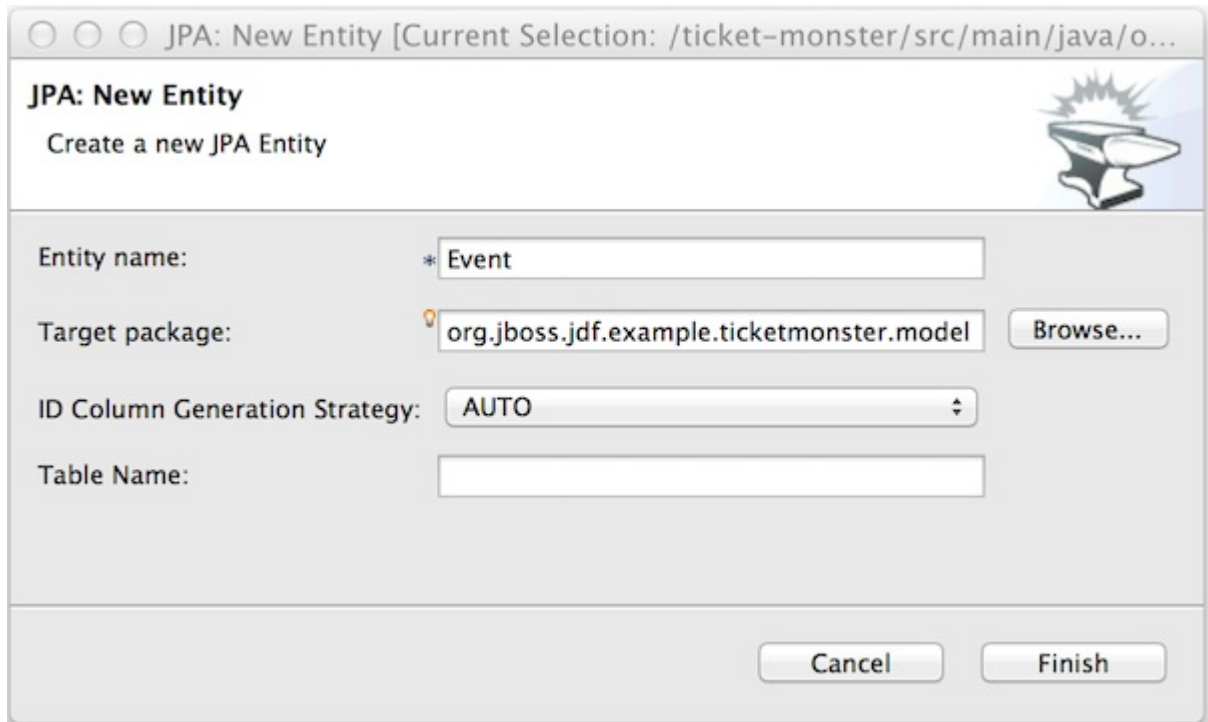


Figure 2.7. New Event Entity using Forge 2 JPA: New Entity Wizard

5. In the **Project Explorer** view, ensure **org.jboss.jdf.example.ticketmonster** → **Event.java** is selected and press **Ctrl+4**.
6. From the Forge wizards list, click **JPA: New Field**.
7. Complete the fields about the field as follows:
 - o In the **Field name** field, type **name**.
 - o In the **Length** field, type **50**.

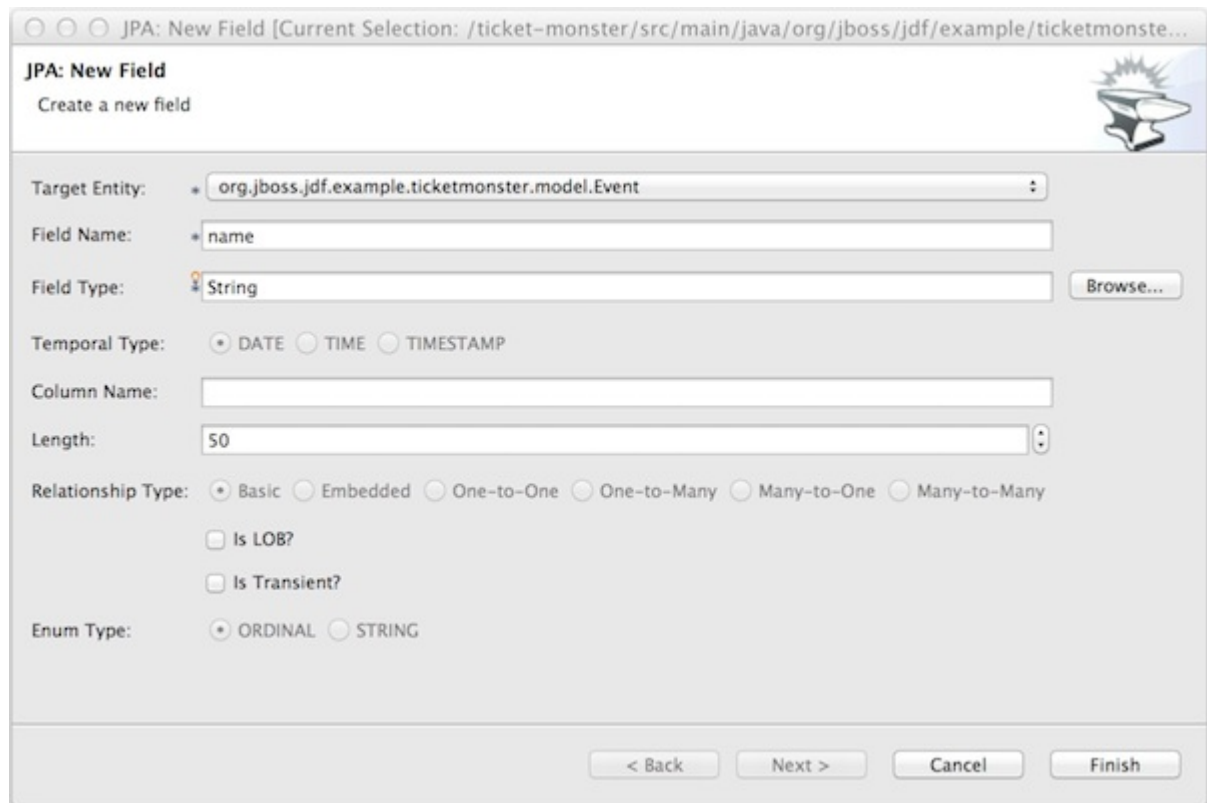


Figure 2.8. New name Field using Forge 2 JPA: New Field Wizard

8. Click **Finish** to create the field.
9. Repeat steps 5 and 6 to create three more fields for the **Event** entity as follows:
 - **description**, with **Field Type String** and **Length 1000**
 - **major**, with **Field Type Boolean**
 - **picture**, with **Field Type String**
10. In the **Project Explorer** view, right-click **Event.java** and click **Show In → Forge Console**.
11. In the **Forge Console** view, on the Forge 2 command line enter

```
constraint-add --onProperty name
```

and from the list of options enter the number corresponding to **NotNull**.

[Report a bug](#)

2.6. ADD AN ENTITY USING HIBERNATE TOOLS

A new entity class can be added to the TicketMonster project using Hibernate Tools. In the procedure below a **Venue** entity class is created to hold information about the venues where events will be held. This **Venue** entity has id, name, description and capacity fields.

Procedure 2.6. Add an Entity Using Hibernate Tools

1. In the **Project Explorer** view, expand **ticket-monster** → **Java Resources** → **src/main/java**.
2. Right-click **org.jboss.jdf.example.ticketmonster.model** and click **New** → **Class**.
3. In the **Name** field, type **Venue** and click **Finish**. This creates a **Venue.java** file that is automatically opened in a Java editor.

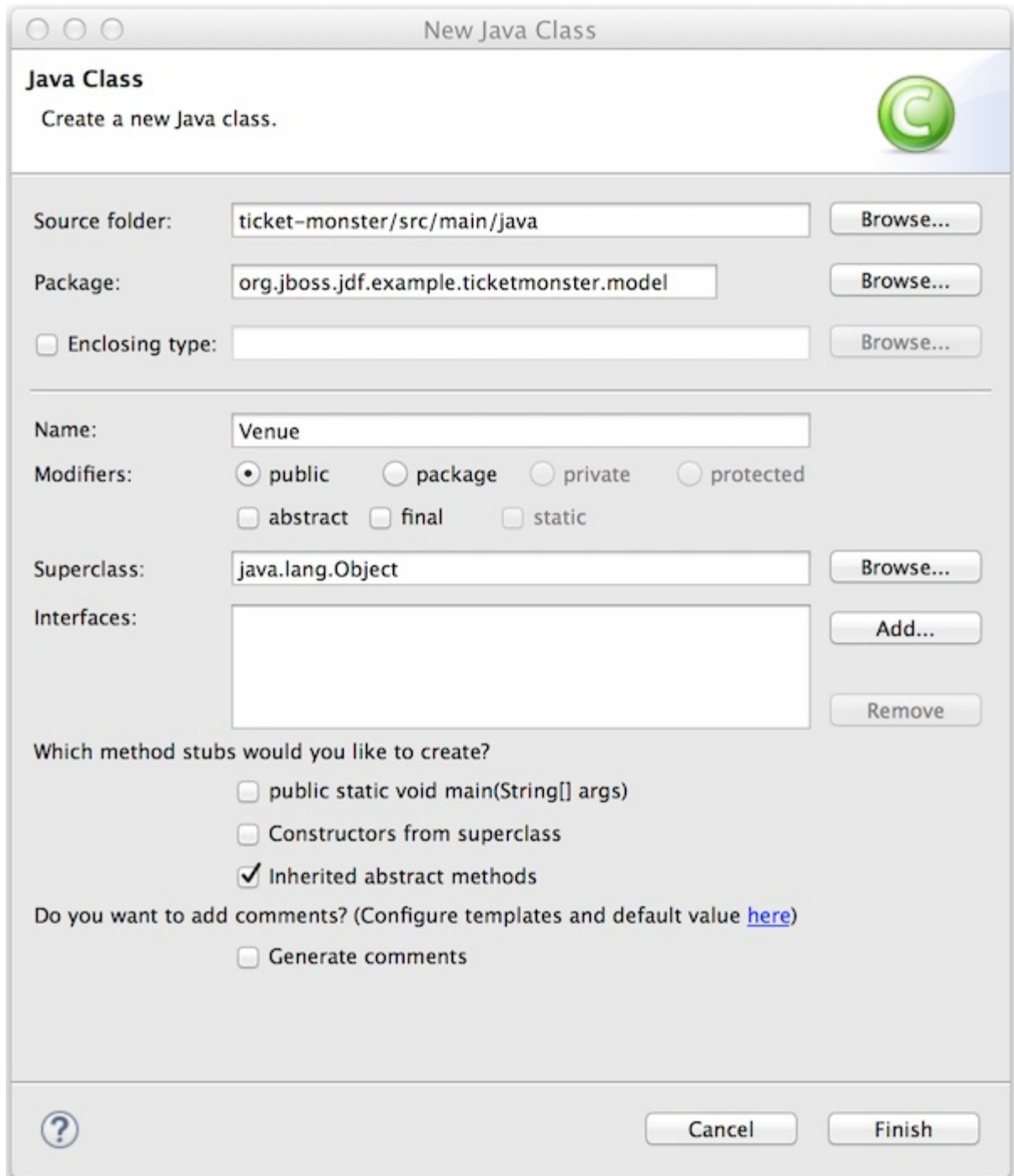


Figure 2.9. New Venue Class using Java Class Wizard

4. Add fields to the entity by adding the following lines in between the braces of the **Venue** class

```
private Long id;
private String name;
private String description;
private int capacity;
```


5. Save the `Venue.java` file by pressing **Ctrl+S** (**Cmd+S**).
6. Add **get** and **set** methods by right-clicking on a new line after `private int capacity;` and clicking **Source** → **Generate Getters and Setters**.
7. Click **Select All** and click **OK**.

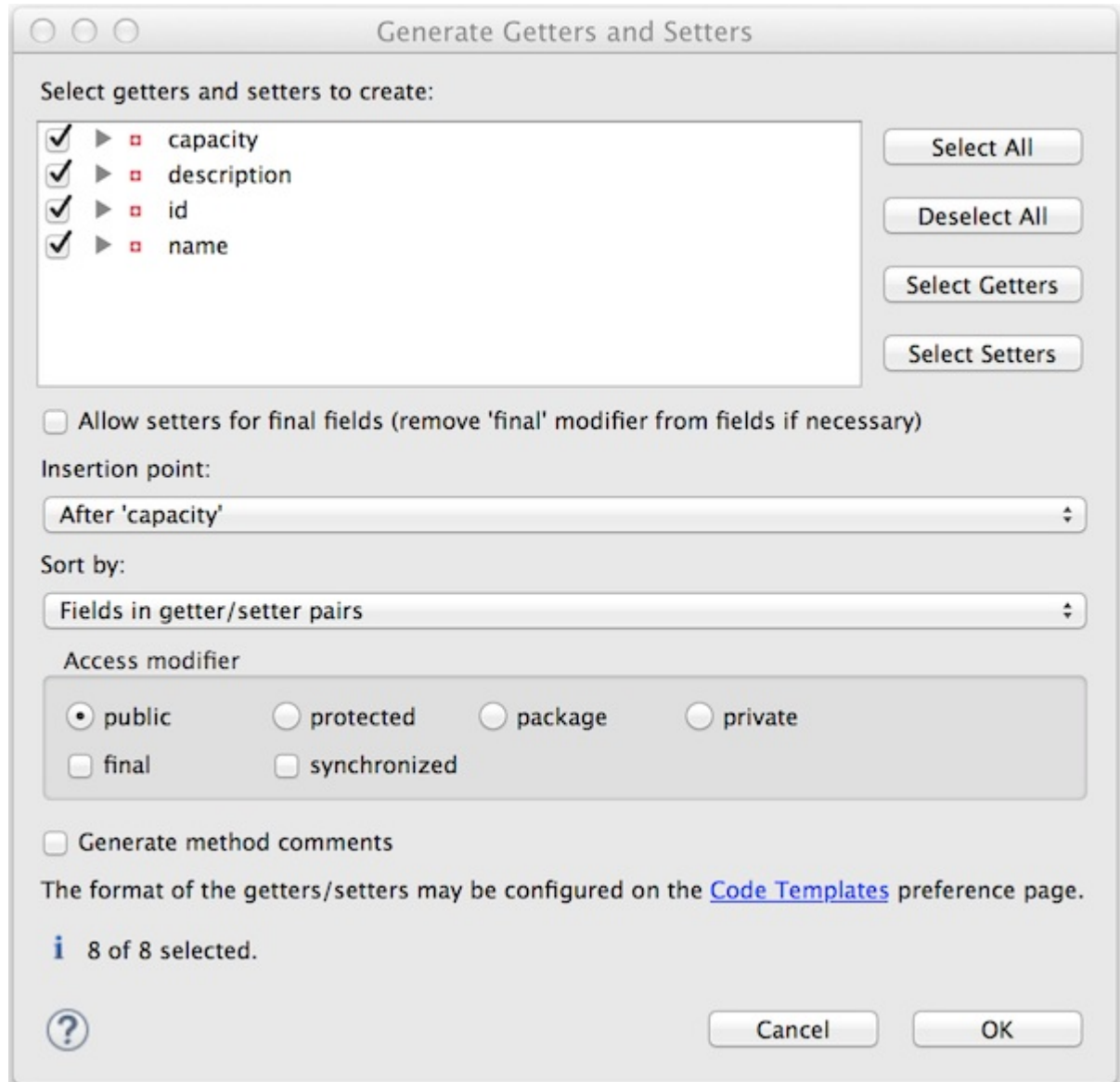


Figure 2.10. List of Attributes Selected for Creating Get and Set Methods

8. Save the `Venue.java` file.
9. Make the class an entity by right-clicking anywhere in the `Venue` class and clicking **Source** → **Generate Hibernate/JPA annotations**.
10. Ensure `org.jboss.jdf.example.ticketmonster.model.Venue` is selected and click **Next**.

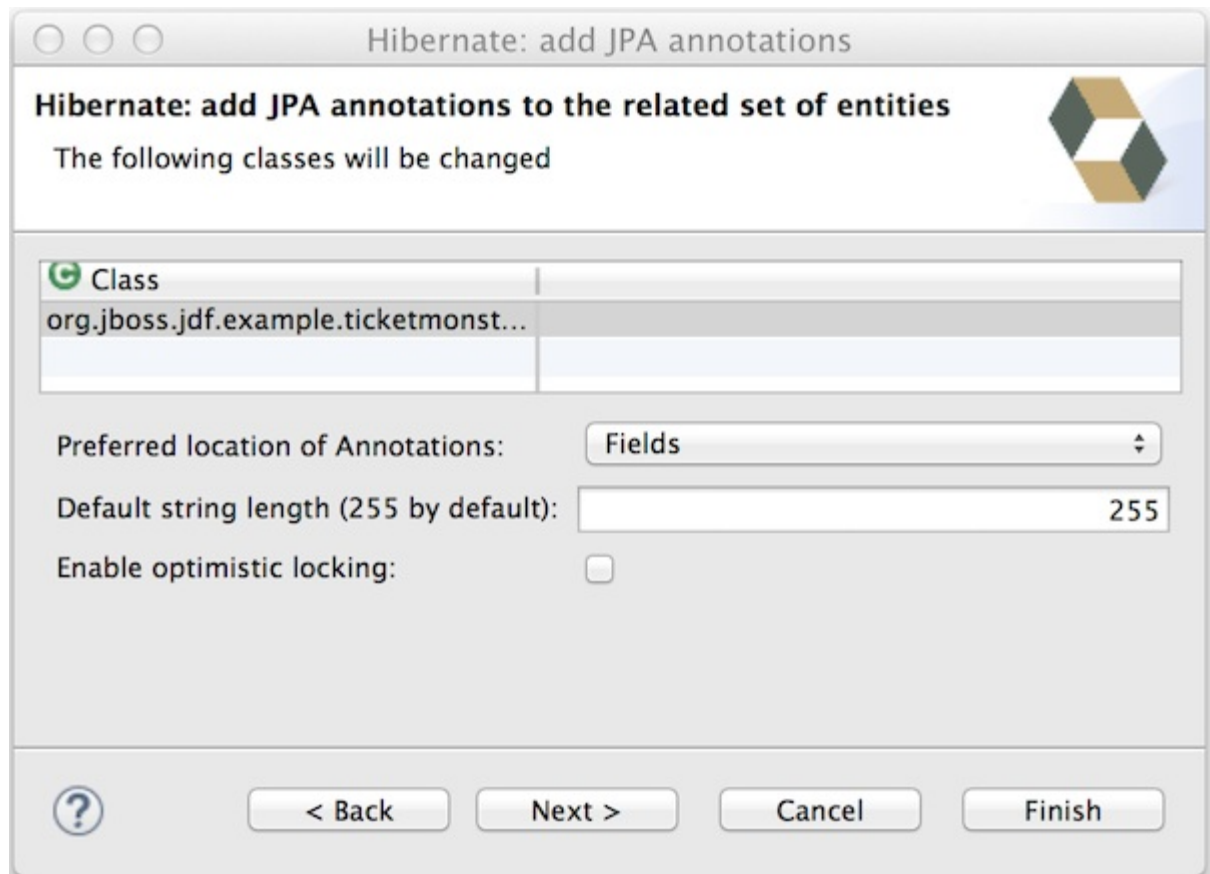


Figure 2.11. Venue Class Selected for Adding JPA Annotations

11. Click **Finish** and save the **Venue.java** file.

[Report a bug](#)

2.7. PREPARE EVENT RECORDS TO POPULATE THE DATABASE

This web project uses H2 for data storage by default. H2 is a Java database integrated with the application servers of JBoss EAP 6.x. It is an in-memory database which means that the data does not persist once the application using the database stops running. For this reason, seed data must be stored in a file within the project and added to the database each time the application starts. While it might not seem practical to have to construct the database each time the application starts, this type of data storage is useful during development for testing purposes.



WARNING

The use of H2 for data storage is for testing purposes only and must not be used during production.

The procedure below demonstrates preparing two sample Event records for addition to the H2 database once the TicketMonster application starts.

Procedure 2.7. Prepare Event Records

1. In the **Project Explorer** view, expand **ticket-monster** → **Java Resources** → **src/main/resources**.
2. Double-click the **import.sql** file to open it for editing.
3. Double-click the **import.sql** view label to make the editor fill the window.
4. Create a new Event record by copying the following on one line after the existing Member record

```
insert into Event (id, name, description, major, picture, version)
values (1, 'Shane''s Sock Puppets', 'This critically acclaimed
masterpiece...', true, 'http://dl.dropbox.com/u/65660684/640px-
Carnival_Puppets.jpg', 1);
```

5. Create a second new Event record by copying the following on one line after the first new Event record

```
insert into Event (id, name, description, major, picture, version)
values (2, 'Rock concert of the decade', 'Get ready to rock...',
true, 'http://dl.dropbox.com/u/65660684/640px-
Weir%2C_Bob_(2007)_2.jpg', 1);
```

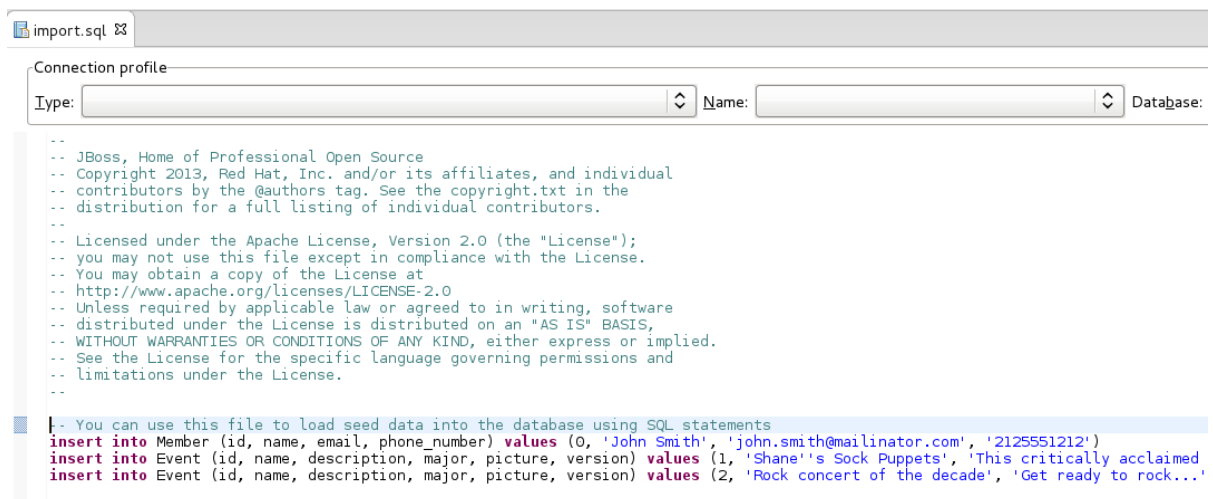


Figure 2.12. Content of the `import.sql` File for this TicketMonster Workflow

6. Save the **import.sql** file.
7. Double-click the **import.sql** view label to make the editor return to its previous size.

[Report a bug](#)

2.8. DEPLOY THE APPLICATION USING JBOSS SERVER TOOLS

The TicketMonster application can be deployed on the JBoss EAP 6.x server created earlier in the tutorial. The procedure below guides you through how to do this.

Procedure 2.8. Deploy the Application Using JBoss Server Tools

1. In the **Project Explorer** view, right-click the **ticket-monster** project name and click **Run As** → **Run on Server**.

2. In the **Run On Server** wizard, ensure **Choose an existing server** is selected.
3. From the **Select the server that you want to use** table, select **jboss-eap-6.x** and click **Finish**.

If not already in view, the **Console** view automatically opens and shows the runtime server output. Also, a **Web Browser** automatically opens and displays the web page of the running TicketMonster application.

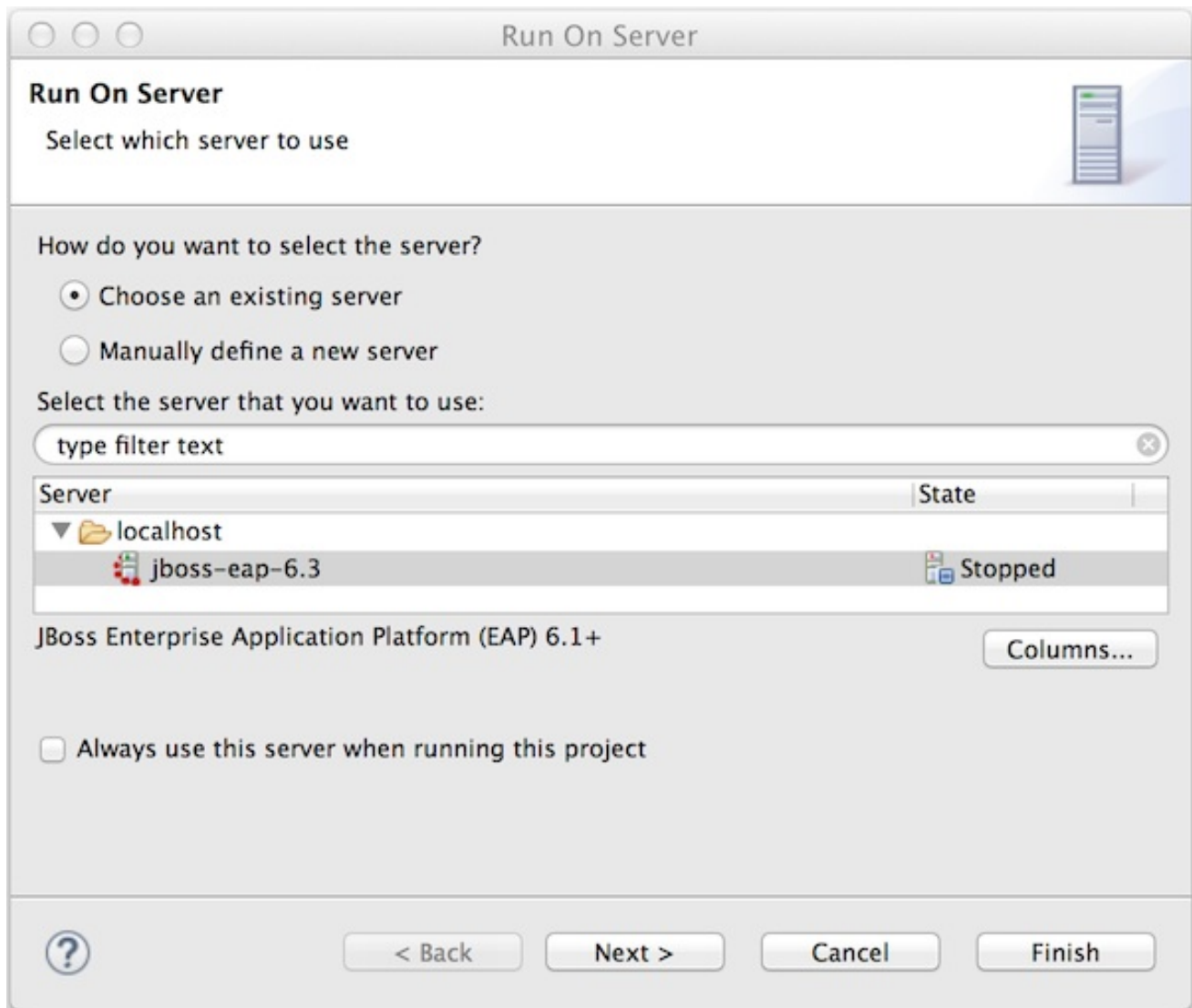


Figure 2.13. JBoss EAP 6.x Selected in Run on Server Wizard

In the **Web Browser**, enter sample Member data in the Member Registration fields and click **Register**. The submitted data is displayed in the **Members** table, which is visible at the bottom of the web page.

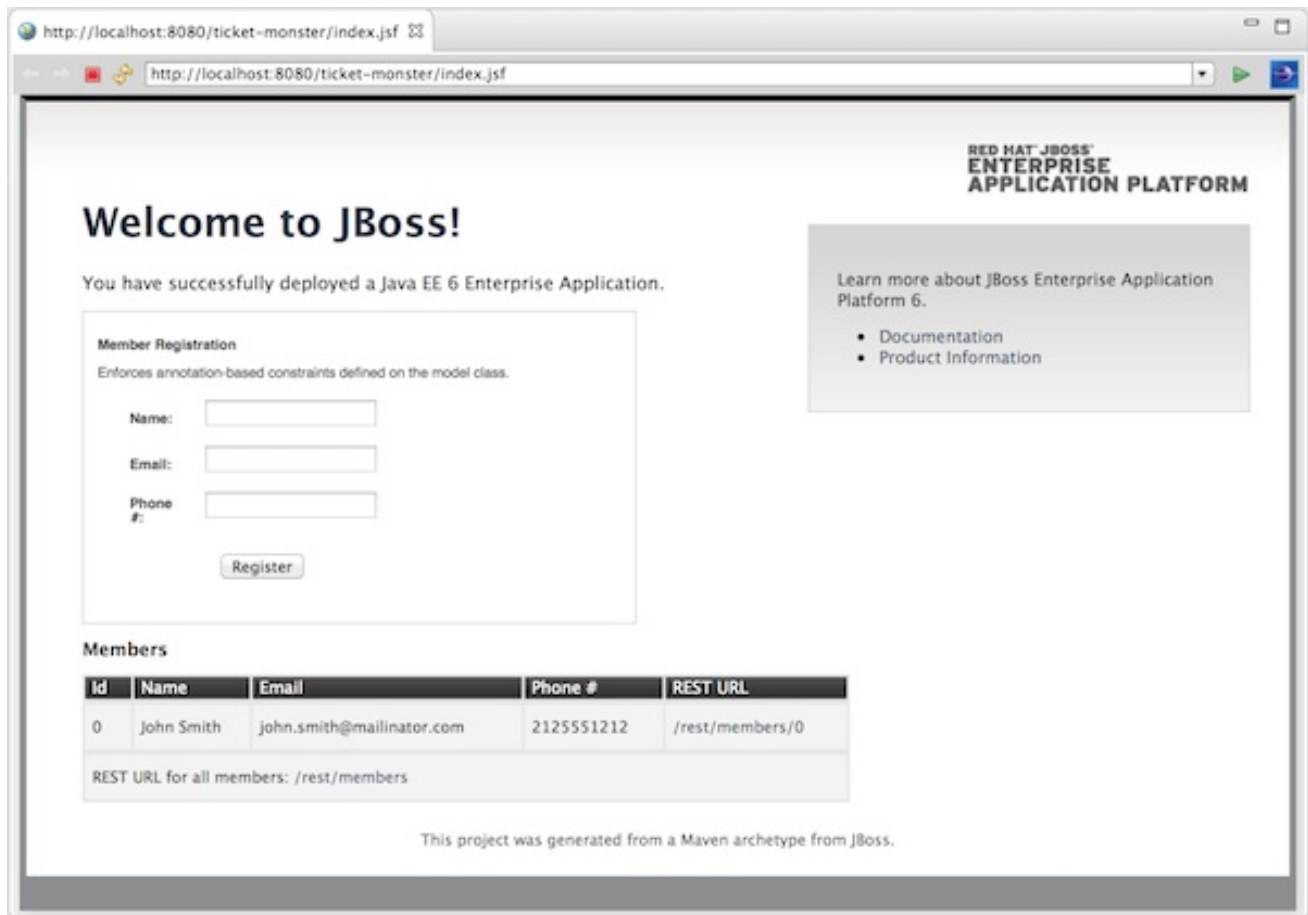


Figure 2.14. Sample Data in the Member Registration Fields

[Report a bug](#)

2.9. ACCESS AND QUERY THE DATABASE

This web project uses the in-memory database, H2, of the application servers in JBoss EAP 6.x. The H2 console web application can be used to access the H2 database but this is not packaged with JBoss EAP 6.x. As the procedure below explains, the H2 console web application must be downloaded and deployed on the application server from outside JBoss Developer Studio before the H2 database can be accessed.

Procedure 2.9. Access and Query the Database

1. Go to <http://www.jboss.org/quickstarts/eap/h2-console/>.
2. Click **Download**.
3. Extract the contents of the archive

```
$ unzip jboss-eap-quickstarts-<version>.zip
```

4. Copy the **h2console.war** file to the JBoss EAP 6.x **deployments** directory

```
$ cp jboss-eap-quickstarts-<version>/h2-console/h2console.war
/path/to/jboss-eap/standalone/deployments/
```

- Confirm the H2 console web application is running by looking at the server output in the **Console** view in JBoss Developer Studio.

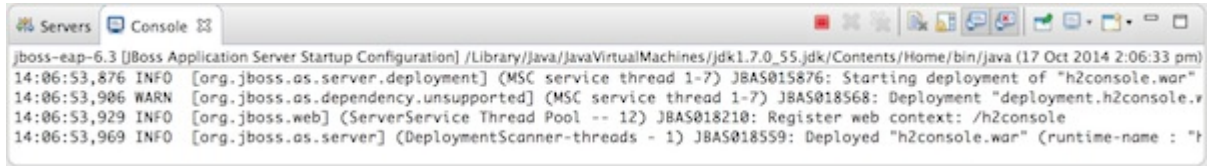


Figure 2.15. Console Output showing h2console.war Deployed

- In the IDE open a new **Web Browser** by clicking **Window** → **Show View** → **Other**, expand **General** and double-click **Internal Web Browser**.
- In the address bar of the **Web Browser**, enter <http://localhost:8080/h2console>.
- In the **Login** area, in the **JDBC URL** field type `jdbc:h2:mem:ticket-monster` and in both the **User Name** and **Password** fields type `sa`. These settings are defined in the `ticket-monster-ds.xml` file in the `ticket-monster/src/main/webapp/WEB-INF` directory.

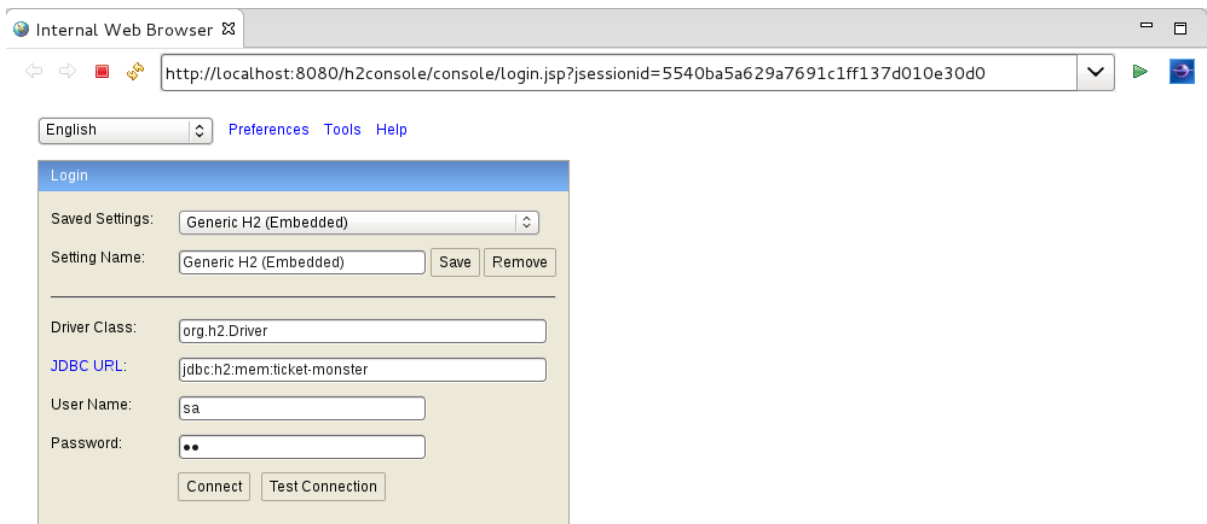


Figure 2.16. Completed H2 Login Web Page

- Click **Connect** to connect to the database.

The **Event**, **Member** and **Venue** entities created in the TicketMonster project are displayed under `jdbc:h2:mem:ticket-monster`. Expanding the entities shows the fields associated with each. The records stored in the database are viewed using SQL search statements. For example, to display all the Event records stored in the database, click **Event**, which creates an SQL search statement based on Event in the **SQL statement** field, and click **Run**. The details of the Event sample records you added to `import.sql` earlier in the tutorial are visible in the table below the **SQL statement** field.

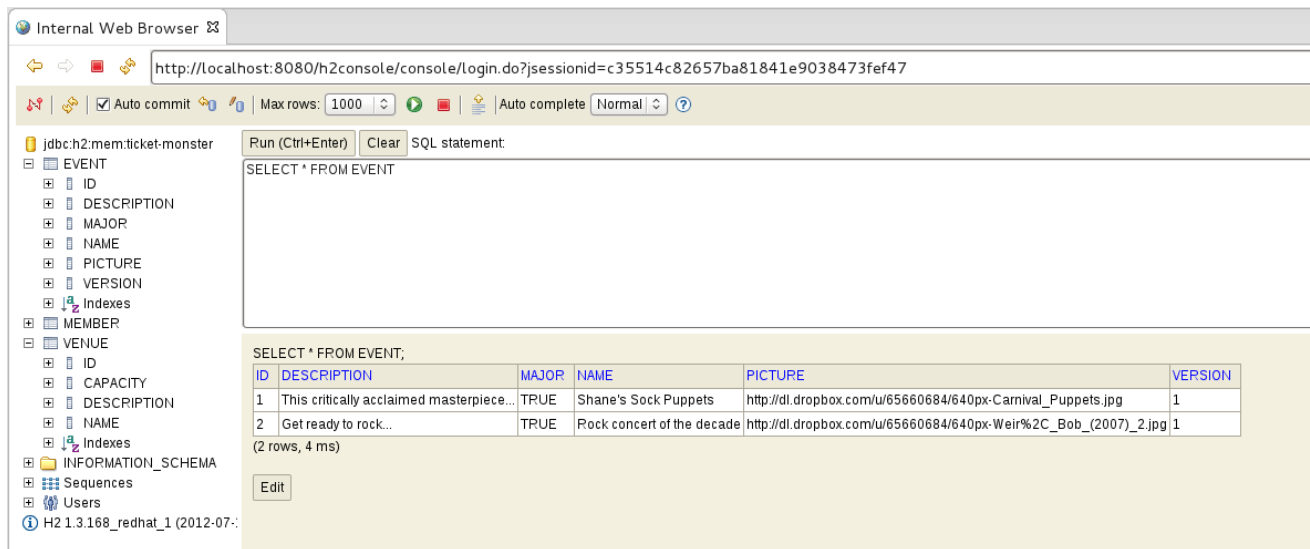


Figure 2.17. Event Records Stored in the H2 Database

[Report a bug](#)

2.10. ADD A RESTFUL WEB SERVICE

For the TicketMonster application to be able to access the information stored in the database, web services are needed. The procedure below creates a RESTful web service that returns all the events in the database. It generates a POJO (plain old Java object) and adds JAX-RS annotations to create an endpoint.

Procedure 2.10. Add a RESTful Web Service

1. In the **Project Explorer** view, expand **ticket-monster** → **Java Resources** → **src/main/java**.
2. Right-click **org.jboss.jdf.example.ticketmonster.rest** and click **New** → **Class**.
3. In the **Name** field, type **EventService** and click **Finish**. This creates a **EventService.java** file that is automatically opened in a Java editor.

New Java Class

Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs 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 [here](#))

☐ Generate comments

Figure 2.18. Completed Name Field in Java Class Wizard

4. Add the following lines immediately above **public class EventService** {

```
@Path("/events")
@RequestScoped
```

and add the following lines in between the braces of the **EventService** class

```
@Inject
private EntityManager em;

@GET
@Produces(MediaType.APPLICATION_JSON)
public List<Event> getAllEvents() {
```



```

final List<Event> results =
    em.createQuery("select e from Event e order by
e.name").getResultList();
return results;
}

```

5. Save the **EventService.java** file.
6. Resolve the errors relating to missing imports by right-clicking anywhere in the **EventService** class and clicking **Source** → **Organize Imports**.
7. For each class name that is not unique, select the class to use as follows and click **Next**:
 - For **MediaType** select **javax.ws.rs.core.MediaType**
 - For **Event** select **org.jboss.jdf.example.ticketmonster.model.Event**
 - For **Produces** select **javax.ws.rs.Produces**
 - For **List** select **java.util.List**
 - For **Inject** select **java.inject.Inject**
 - For **RequestScoped** select **java.enterprise.context.RequestScoped**

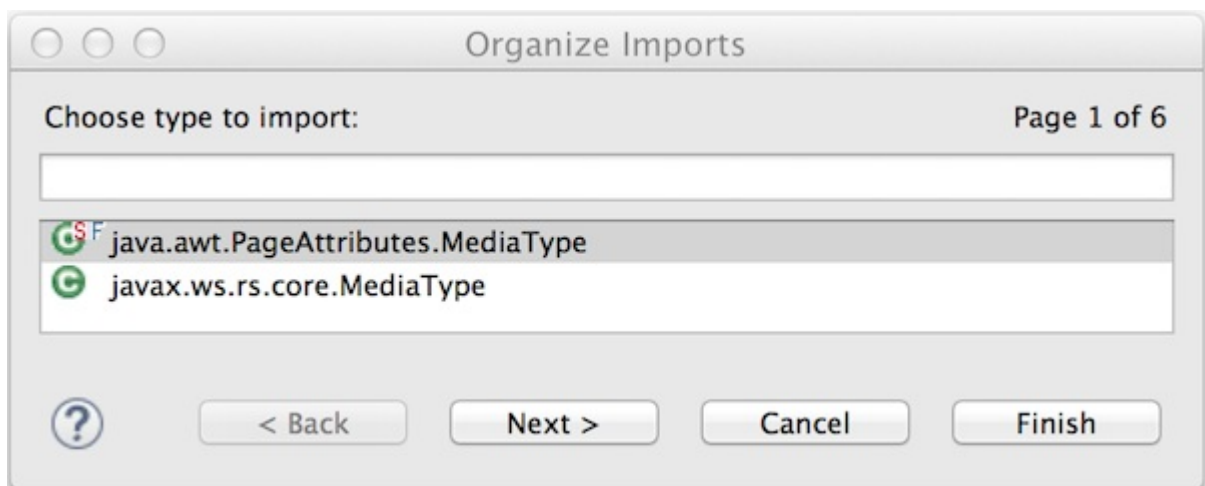


Figure 2.19. javax.ws.rs.core.MediaType Selected for MediaType Class

8. When all the classes have been chosen, click **Finish**. The import statements corresponding to the class names selected in the previous step are added to the **EventService.java** file.
9. Save the **EventService.java** file.
10. In the **Servers** view expand **jboss-eap-6.x**, right-click **ticket-monster** and click **Full Publish** to update the deployed version of the application.
11. Open a new **Web Browser** and in the address bar of the **Web Browser** enter <http://localhost:8080/ticket-monster/rest/events>. Depending on your operating system, the output of the new RESTful endpoint is displayed or you are prompted to save the JSON output to a file that can be viewed with a text editor. In both cases, details of the two event records created earlier in the tutorial are listed as below.

```
[
  {"id":2,"version":1,"name":"Rock concert of the
  decade","description":"Get ready to
  rock...","major":true,"picture":null}
,
  {"id":1,"version":1,"name":"Shane's Sock
  Puppets","description":"This critically acclaimed
  masterpiece...","major":true,"picture":null}
]
```

[Report a bug](#)

2.11. ADD A USER INTERFACE OPTIMIZED FOR MOBILE DEVICES

The TicketMonster user interface displays on desktop and mobile clients but it is not optimized for the latter. The procedure below uses the jQuery Mobile API to create a mobile optimized user interface. The mobile user interface created varies from the existing user interface by displaying the names of the events returned by the previously created web service rather than the Member Registration web page.

Procedure 2.11. Add a User Interface Optimized for Mobile Devices

1. In the **Project Explorer** view, expand **ticket-monster** → **src** → **main**.
2. Right-click **webapp** and click **New** → **HTML file**.
3. Complete the information about the file as follows:
 - In the **Enter or select the parent folder** field, type **ticket-monster/src/main/webapp**.
 - In the **File name** field, type **mobile.html**.

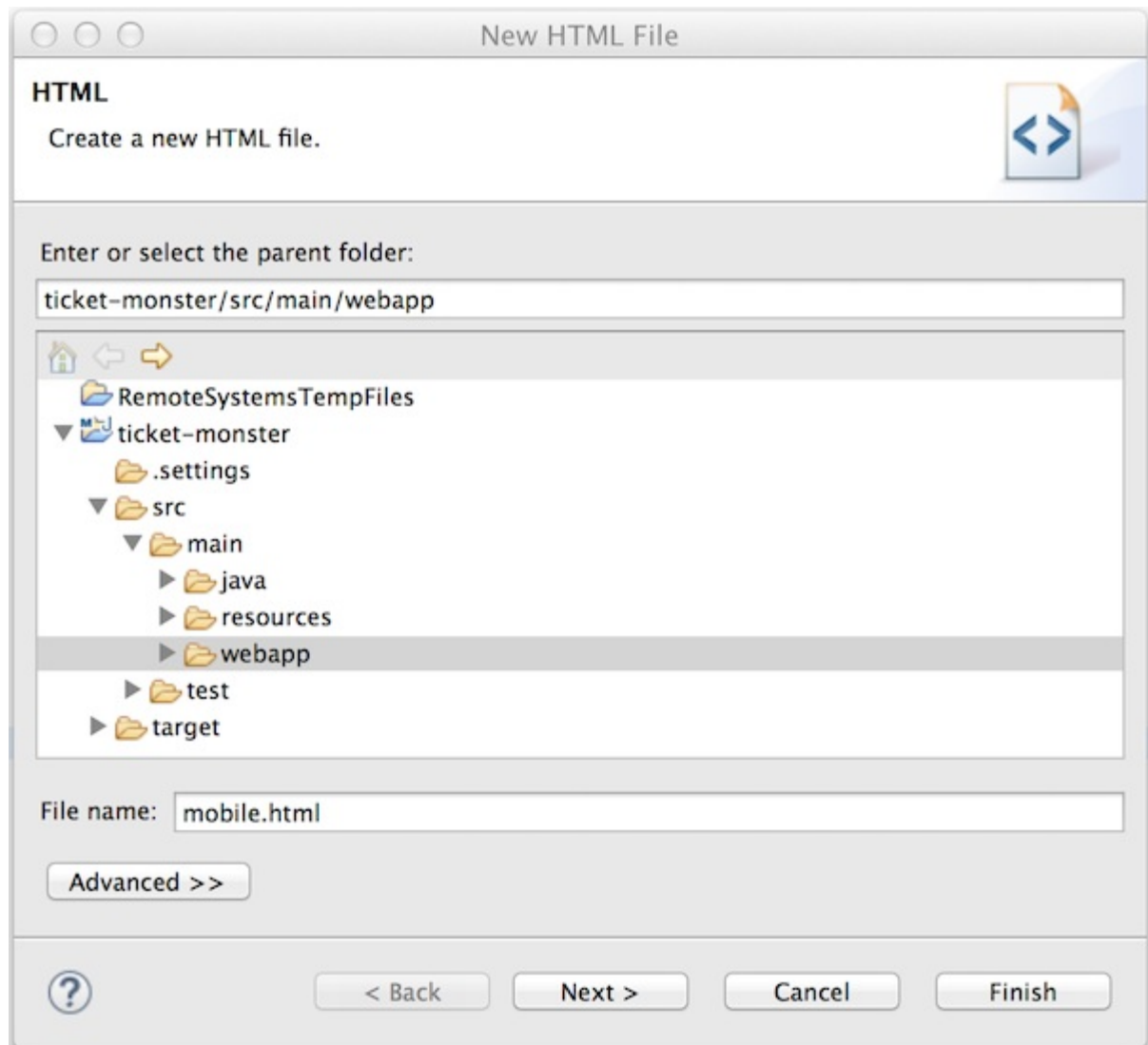


Figure 2.20. New HTML File using the New HTML File Wizard

4. Click **Next**.
5. From the **Templates** list, select **HTML5 jQuery Mobile Page (1.4)** and click **Finish**. This creates a **mobile.html** file that is automatically opened in the **Visual/Source** tab of the Visual Page Editor.

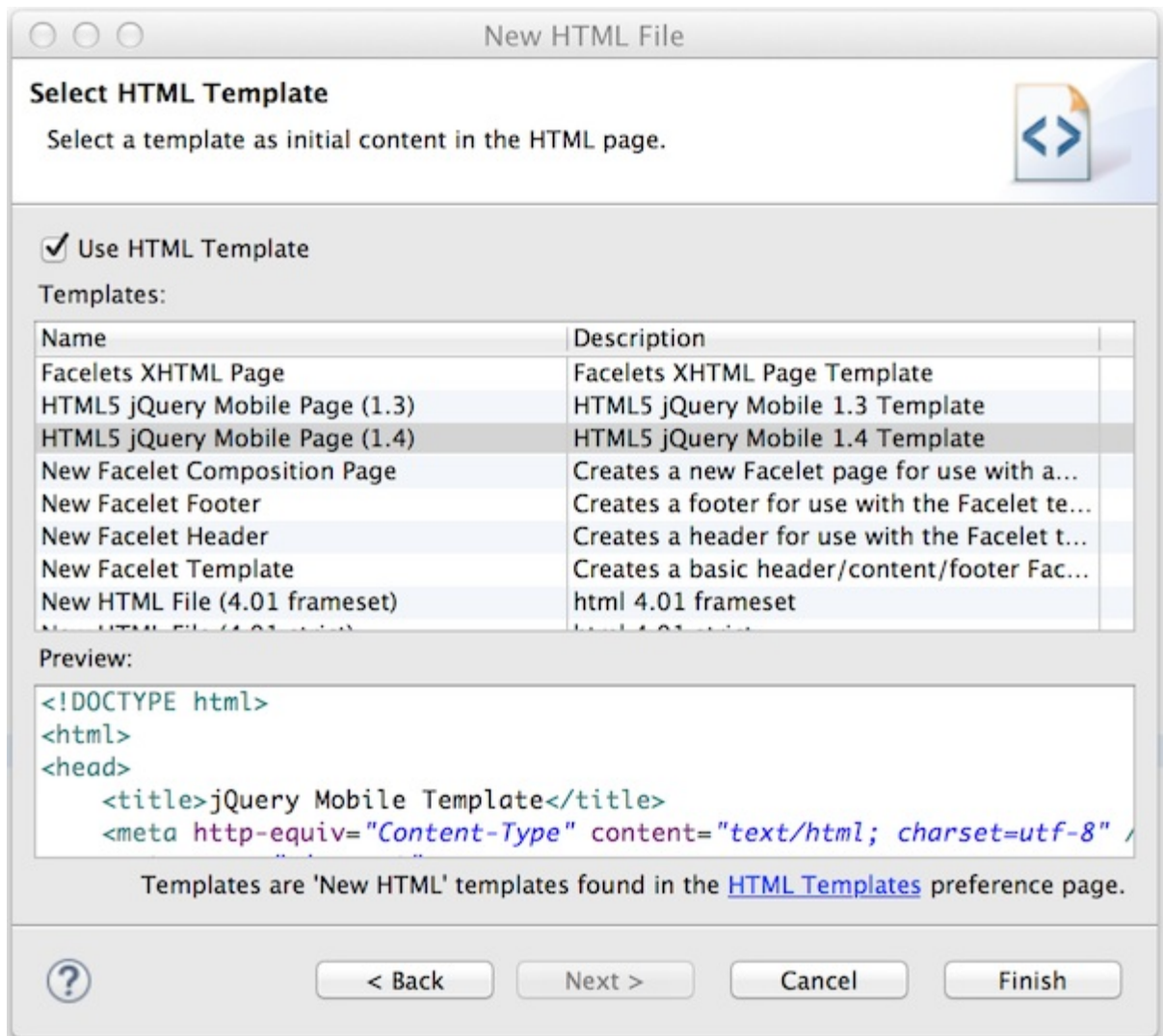


Figure 2.21. Selected HTML5 jQuery Mobile Page (1.4) Template in New HTML File Wizard

- List the events returned by the web service in the mobile user interface by adding the following lines immediately after the `alert("Ready To Go");` line

```
$.getJSON("rest/events", function(events) {
  // console.log("returned are " + events);
  var listOfEvents = $("#listOfItems");
  listOfEvents.empty();
  $.each(events, function(index, event) {
    // console.log(event.name);
    listOfEvents.append("<li><a href='#'>" + event.name + "</a>");
  });
  listOfEvents.listview("refresh");
});
```



```

<!DOCTYPE html>
<html>
<head>
  <title>jQuery Mobile Template</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta name="viewport"
    content="width=device-width, initial-scale=1" />
  <link rel="stylesheet"
    href="http://code.jquery.com/mobile/1.4.4/jquery.mobile-1.4.4.min.css" />
  <script type="text/javascript"
    src="http://code.jquery.com/jquery-1.10.2.min.js"></script>
  <script type="text/javascript"
    src="http://code.jquery.com/mobile/1.4.4/jquery.mobile-1.4.4.min.js"></script>
  <script type="text/javascript">
    $(document).on("pageinit", "#page1", function(event){
      alert("Ready To Go");

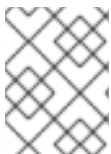
      $.getJSON("rest/events", function(events) {
        // console.log("returned are " + events);
        var listOfEvents = $("#listOfItems");
        listOfEvents.empty();
        $.each(events, function(index, event) {
          // console.log(event.name);
          listOfEvents.append("<li><a href='#'>" + event.name + "</a>");
        });
        listOfEvents.listview("refresh");
      });
    });
  </script>
</head>

```

Figure 2.22. Completed `mobile.html` File

7. Save the `mobile.html` file.

View the mobile interface, by opening an external web browser and in the address bar entering <http://localhost:8080/ticket-monster/mobile.html>. At the **Ready To Go** prompt, click **OK**. The names of the events in the database are displayed.



NOTE

HTML pages are deployed immediately so there is no need to do a full publish of the TicketMonster application to see changes.




Figure 2.23. Events Displayed via Mobile Interface in External Web Browser

[Report a bug](#)

2.12. TEST THE MOBILE USER INTERFACE USING BROWSERSIM

Mobile user interfaces are best tested on mobile devices and JBoss Developer Studio provides BrowserSim for this purpose. The procedure below guides you through viewing the TicketMonster application on simulated mobile devices using BrowserSim.

Procedure 2.12. Test the Mobile User Interface Using BrowserSim

1. On the toolbar click the **BrowserSim** icon . A simulated device is displayed.
2. At the **Ready To Go** prompt, click **Close**. The names of the events in the database are displayed.



NOTE

If the names of the events in the database are not displayed, check the address bar of the simulated device. The address bar must state <http://localhost:8080/ticket-monster/mobile.html>.

3. To rotate the mobile device, click in any corner of the simulated device.



Figure 2.24. Rotated Simulated Device

4. Change the type of simulated device displayed by right-clicking anywhere on the simulated device, clicking **Skin** and selecting from the different skins listed.
5. Close BrowserSim by right-clicking anywhere on the simulated device and clicking **Close**.

[Report a bug](#)

APPENDIX A. REVISION HISTORY

Revision 8.1.0-2

Test build for GA

Mon Mar 23 2015**Misha Husnain Ali****Revision 8.1.0-1**

Generated for Beta release

Wed Feb 18 2015**Supriya Bharadwaj**