What is TicketMonster?

Preamble

TicketMonster is an example application that focuses on Java EE6 - JPA 2, CDI, EJB 3.1 and JAX-RS along with HTML5 and jQuery Mobile. It is a moderately complex application that demonstrates how to build modern web applications optimized for mobile & desktop. TicketMonster is representative of an online ticketing broker - providing access to events (e.g. concerts, shows, etc) with an online booking application.

Apart from being a demo, TicketMonster provides an already existing application structure that you can use as a starting point for your app. You could try out your use cases, test your own ideas, or, contribute improvements back to the community.

Fork us on GitHub!

The accompanying tutorials walk you through the various tools & technologies needed to build TicketMonster on your own. Alternatively you can download TicketMonster as a completed application and import it into your favorite IDE.

Before we dive into the code, let's discuss the requirements for the application.

Use cases

We have grouped the current use cases in two major categories: end user oriented, and administrative.

What can end users do?

The end users of the application want to attend some cool events. They will try to find shows, create bookings, or cancel bookings. The use cases are:

- look for current events;
- look for venues;
- select shows (events taking place at specific venues) and choose a performance time;
- book tickets;
- view current bookings;
- cancel bookings;

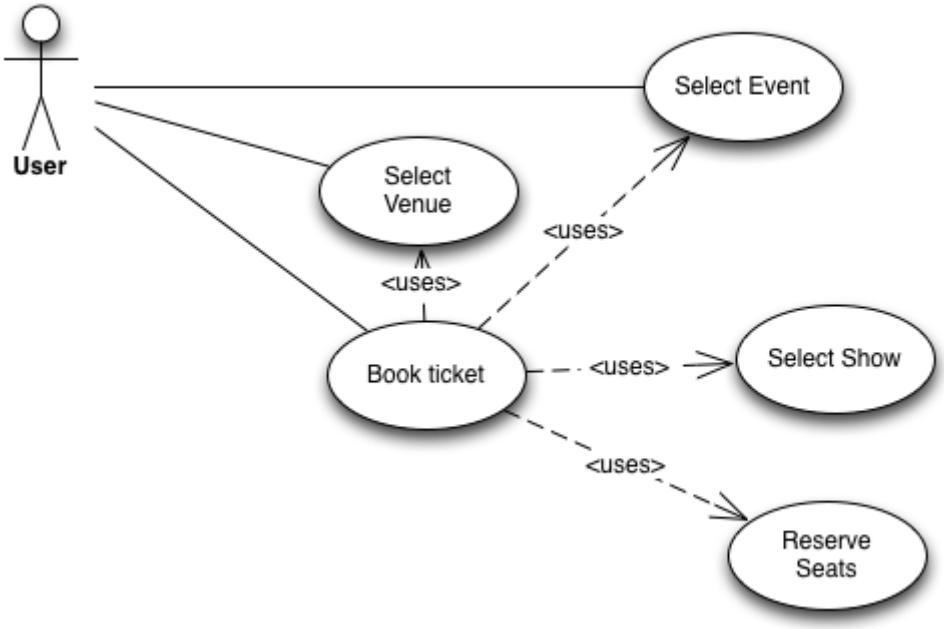

Figure 1. End user use cases

What can administrators do?

Administrators are more concerned the operation of the business. They will manage the *master data*: information about venues, events and shows, and will want to see how many tickets have been sold. The use cases are:

- add, remove and update events;
- add, remove and update venues (including venue layouts);
- add, remove and update shows and performances;
- monitor ticket sales for current shows;

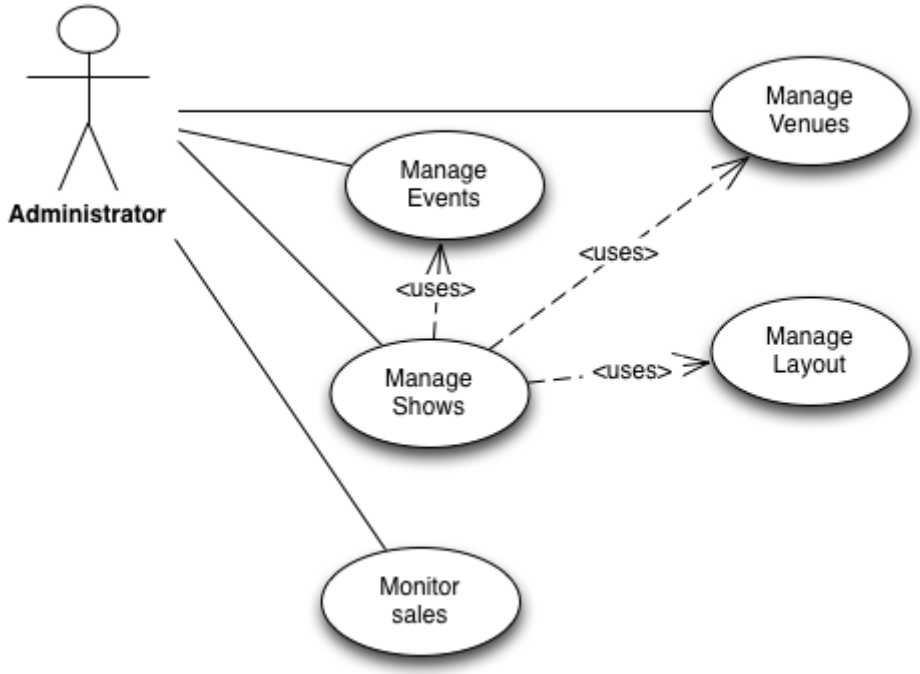

Figure 2. Administration use cases

Architecture

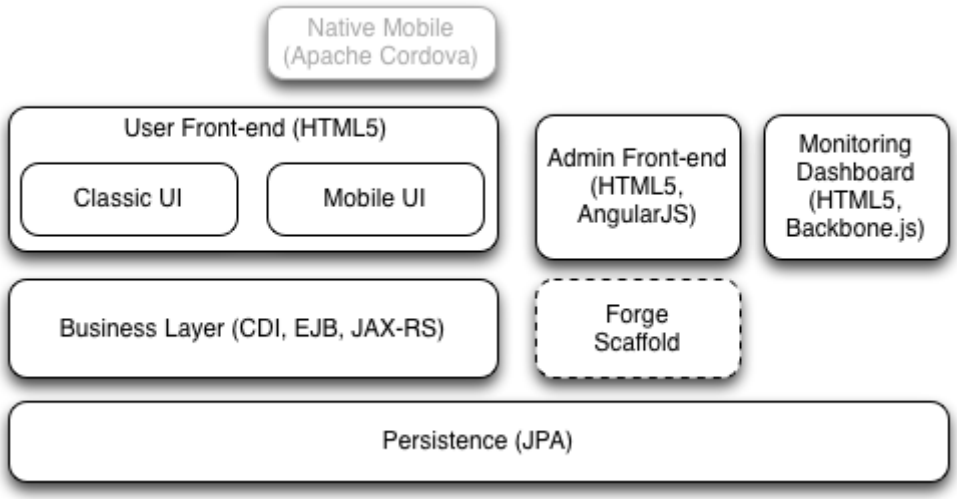

Figure 3. TicketMonster architecture

The application uses Java EE 6 services to provide business logic and persistence, utilizing technologies such as CDI, EJB 3.1 and JAX-RS, JPA 2. These services back the user-facing booking process, which is implemented using HTML5 and JavaScript, with support for mobile devices through jQuery Mobile.

The administration site is centered around CRUD use cases, so instead of writing everything manually, the business layer and UI are generated by Forge, using EJB 3.1, CDI and JAX-RS. For a better user experience, Twitter Bootstrap is used.

Monitoring sales requires staying in touch with the latest changes on the server side, so this part of

the application will be developed in HTML5 and JavaScript using a polling solution.

How can you run it?

Building TicketMonster

CAUTION	In order to build the application, you will need you to configure Maven to use the JBoss Enterprise Maven repositories. For instructions on configure the Maven repositories, visit the JBoss Enterprise Application Platform 6.3 documentation.

TicketMonster can be built from Maven, by runnning the following Maven command:

```
mvn clean package
```

This prepares a WAR file that you can deploy right away in a JBoss Enterprise Application Platform instance. It would use the in-built H2 database.

If you want to run the Arquillian tests as part of the build, you can enable one of the two available Arquillian profiles.

For running the tests in an *already running* application server instance, use the `arq-jbossas-remote` profile.

```
mvn clean package -Parq-jbossas-remote
```

If you want the test runner to *start* an application server instance, use the `arq-jbossas-managed` profile. You must set up the `JBOSS_HOME` property to point to the server location, or update the `src/main/test/resources/arquillian.xml` file.

```
mvn clean package -Parq-jbossas-managed
```

If you intend to deploy into OpenShift with the PostgreSQL cartridge, you can use the `postgresql-openshift` profile:

```
mvn clean package -Ppostgresql-openshift
```

If you intend to deploy into OpenShift with the MySQL cartridge, you can use the `mysql-openshift` profile:

```
mvn clean package -Pmysql-openshift
```

Running TicketMonster

You can run TicketMonster into a local JBoss EAP 6.3 instance or on OpenShift.

Running TicketMonster locally

Start JBoss Enterprise Application Platform 6.3.

1. Open a command line and navigate to the root of the JBoss server directory.
2. The following shows the command line to start the server with the web profile:

```
For Linux:   JBOSS_HOME/bin/standalone.sh
For Windows: JBOSS_HOME\bin\standalone.bat
```

Then, *deploy TicketMonster*.

1. Make sure you have started the JBoss Server as described above.
2. Type this command to build and deploy the archive into a running server instance.

```
mvn clean package jboss-as:deploy
```

   (You can use the `arq-jbossas-remote` profile for running tests as well)

3. This will deploy `target/ticket-monster.war` to the running instance of the server.
4. Now you can see the application running at http://localhost:8080/ticket-monster.

Running TicketMonster in OpenShift

First, *create an OpenShift project*.

1. Make sure that you have an OpenShift domain and you have created an application using the `jbosseap-6` cartridge (for more details, get started here). If you want to use PostgreSQL, add the `postgresql-9.2` cartridge too. Or for MySQL, add the `mysql-5.5` cartridge.
2. Ensure that the Git repository of the project is checked out.

Then, *build and deploy it*.

1. Build TicketMonster using either:
   - the default profile (with H2 database support)

```
mvn clean package
```

   - the `postgresql-openshift` profile (with PostgreSQL support) if the PostgreSQL cartrdige is enabled in OpenShift.

```
mvn clean package -Ppostgresql-openshift
```

- the `mysql-openshift` profile (with MySQL support) if the MySQL cartrdige is enabled in OpenShift.

```
mvn clean package -Pmysql-openshift
```

2. Copy the `target/ticket-monster.war` file in the OpenShift Git repository (located at `<root-of-openshift-application-git-repository>`).

```
cp target/ticket-monster.war <root-of-openshift-application-git-
repository>/deployments/ROOT.war
```

3. Navigate to `<root-of-openshift-application-git-repository>` folder.
4. Remove the existing `src` folder and `pom.xml` file.

```
git rm -r src
git rm pom.xml
```

5. Add the copied file to the repository, commit and push to Openshift

```
git add deployments/ROOT.war
git commit -m "Deploy TicketMonster"
git push
```

6. Now you can see the application running at at `http://<app-name>-<domain-name>.rhcloud.com`

Learn more

The example is accompanied by a series of tutorials that will walk you through the process of creating the TicketMonster application from end to end.

After reading this series you will understand how to:

- set up your project;
- define the persistence layer of the application;
- design and implement the business layer and expose it to the front-end via RESTful endpoints;
- implement a mobile-ready front-end using HTML 5, JSON, JavaScript and jQuery Mobile;
- develop a HTML5-based administration interface rapidly using JBoss Forge;
- thoroughly test your project using JUnit and Arquillian;

Throughout the series, you will be shown how to achieve these goals using JBoss Developer Studio.