Creating RESTful Web Services in NetBeans 7 : Part 1

Purpose

This tutorial demonstrates building RESTful Web Services in NetBeans 7.

Time to Complete

Approximately 30 minutes.

Overview

Representational State Transfer (REST), is a Web Service model is a simpler alternative to SOAP and Web Services Description Language (WSDL) based Web Services. Building Web Services using the RESTful approach is emerging as a popular Web Service model due to its lightweight nature.

RESTful Web Services enable message exchange over HTTP using formats like -XML, JSON etc. RESTful Web Services are really just a collection of web resources identifiable by URIs, which can be manipulated by a small number of operations – GET, PUT, POST and DELETE. Exposing a system's resources through a RESTful API is a flexible way to integrate applications.

REST is just an architectural style, not a technology. Java specification (JSR 311) describes how REST should be implemented in Java.

There have been several implementations of this standard. Jersey is its official reference implementation and the one that is most widely used in development and production.

NetBeans 7.1 supports the rapid development of RESTful Web Services using JSR-311 (Java API for RESTful Web Services – JAX-RS) and Jersey, the reference implementation for JAX-RS. The IDE supports building and testing services, as well as creating client applications that access these services. The following are the RESTful features provided in NetBeans :

- Creating RESTful Web Services
- Supports testing of RESTful Web Services
- Building client applications that access RESTful Web Services

In this tutorial, you will

- Create a database connection
- Create RESTful Web Services
- Test RESTful Web Services

Software and Hardware Requirements

The following software is required to complete this tutorial in **Windows** platform. You must install the software in the given order.

- Download and install Java 7 (JDK) from this link.
- Download and install NetBeans IDE 7.1 Java EE Version which includes GlassFish 3.1.1 (Java EE download bundle) from this link.

Prerequisites

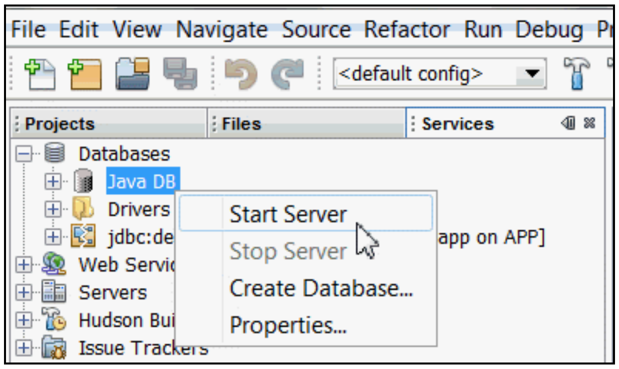
- Before starting this tutorial, you should have the software installed as listed under Software Requirements.
- NetBeans is running.
- Download and unzip the files.zip file that contains the file you need to perform this tutorial.

Create a database connection

Java DB database server is part of NetBeans. We will use Java DB as the database server. The following steps demonstrate creating the database **playerDB**.

1 . To start the Java DB Database from NetBeans, perform the following steps.

a. Click **Services** tab.
b. Expand **Databases** node.
c. Right-click **Java DB** icon.
d. Select **Start Server**.



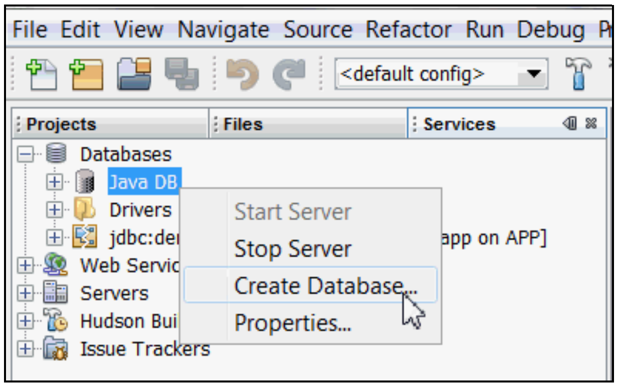
Note the following output in the Output window, indicating that the DB server has started:

Screenshot for Step

Note that the DBserver version could vary from the version shown in the screenshot depending on the JDK build updates.

2. To create `playerDB` database, perform the following steps:

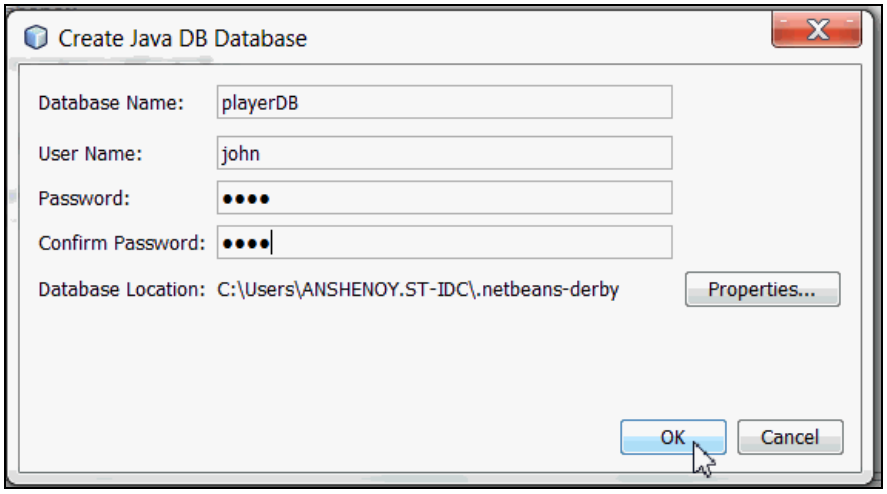
a. Right-click **Java DB** icon, select **Create Database**.



b. Enter the following information for the database:

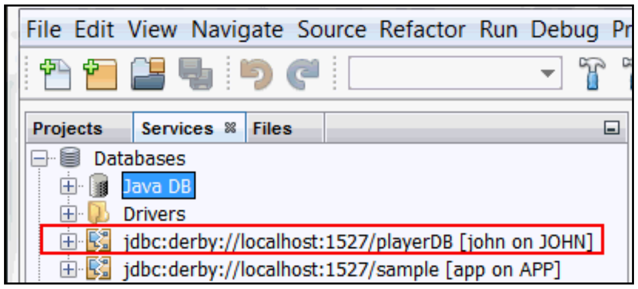
Database Name: **playerDB**
User Name: **john**
Password: **john**
Confirm Password:**john**

c. Click OK.



Database Name:	playerDB
User Name:	john
Password:	••••
Confirm Password:	••••
Database Location:	C:\Users\ANSHENOY.ST-IDC\.netbeans-derby

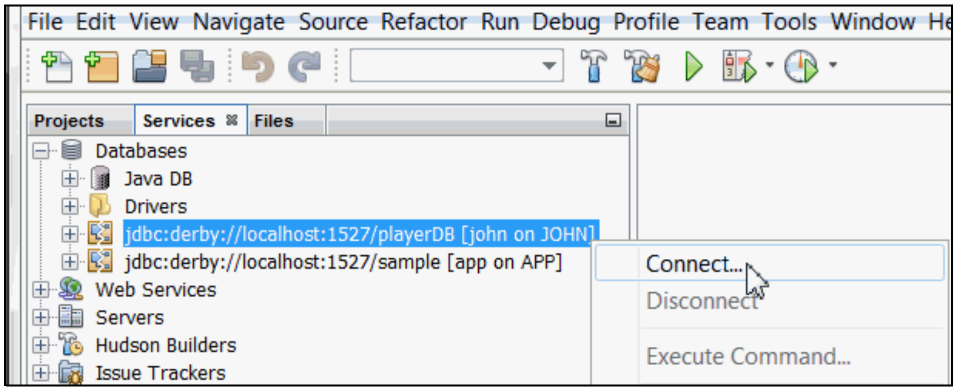
This creates the database and adds a connection for the database under the **Databases** icon.


File Edit View Navigate Source Refactor Run Debug
Projects
Services
Files
Databases
Java DB
Drivers
jdbc:derby://localhost:1527/playerDB [john on JOHN]
jdbc:derby://localhost:1527/sample [app on APP]

3 . To connect to the newly created database `playerDB`, perform the following steps :

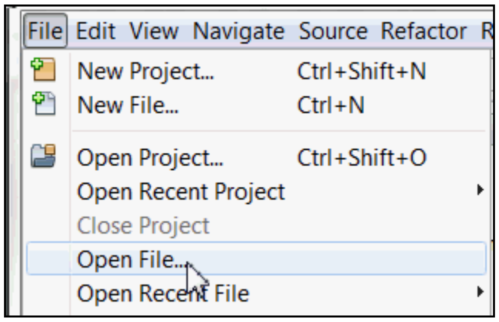
a. Right-click `jdbc:derby://localhost:1527/playerDB` connection.

b. Select **Connect**.


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window
Projects
Services
Files
Databases
Java DB
Drivers
jdbc:derby://localhost:1527/playerDB [john on JOHN]
jdbc:derby://localhost:1527/sample [app on APP]
Web Services
Servers
Hudson Builders
Issue Trackers
Connect...
Disconnect
Execute Command...

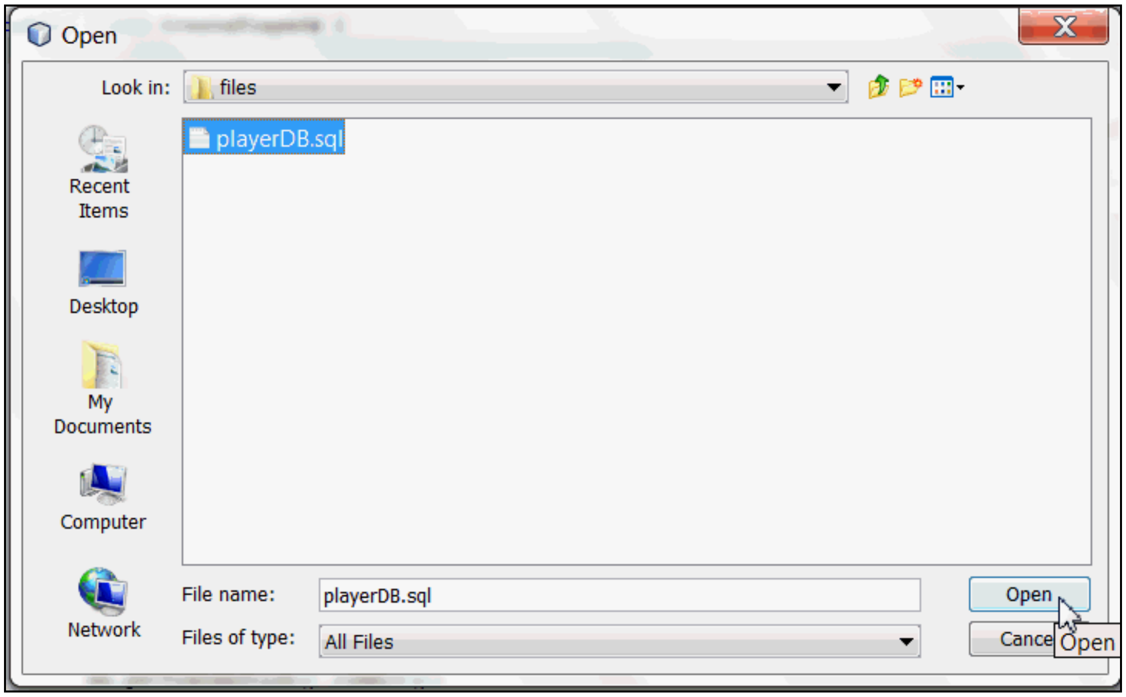
4. Create tables and populate them with data in **`playerDB`** database.

a. In NetBeans select File > Open File.


File Edit View Navigate Source Refactor
New Project... Ctrl+Shift+N
New File... Ctrl+N
Open Project... Ctrl+Shift+O
Open Recent Project
Close Project
Open File...
Open Recent File

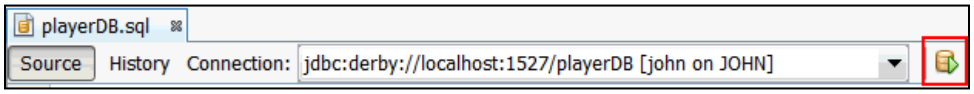
b. In the file browser navigate to the directory, where you unzipped the files from the Prerequisites section and select `playersDB.sql`.

c. Click Open. The script automatically opens in the SQL Editor.



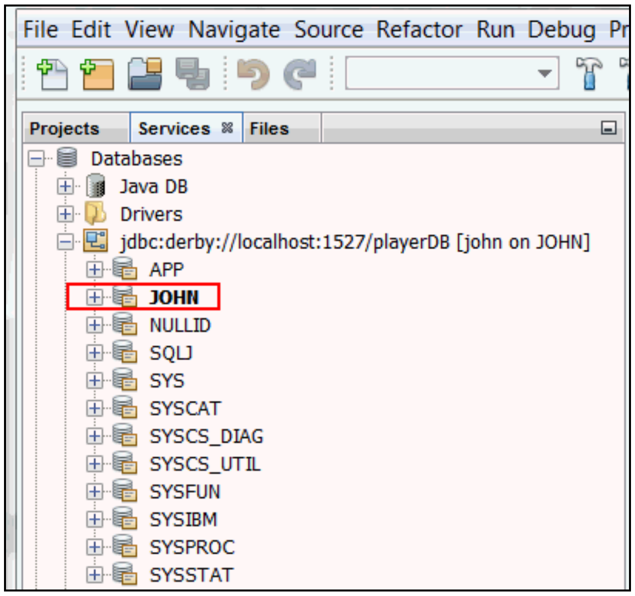
d. Select `jdbc:derby://localhost:1527/playerDB` in Connection drop-down box in the SQL Editor toolbar.

e. Click the **`Run SQL`** icon to execute the SQL statement.

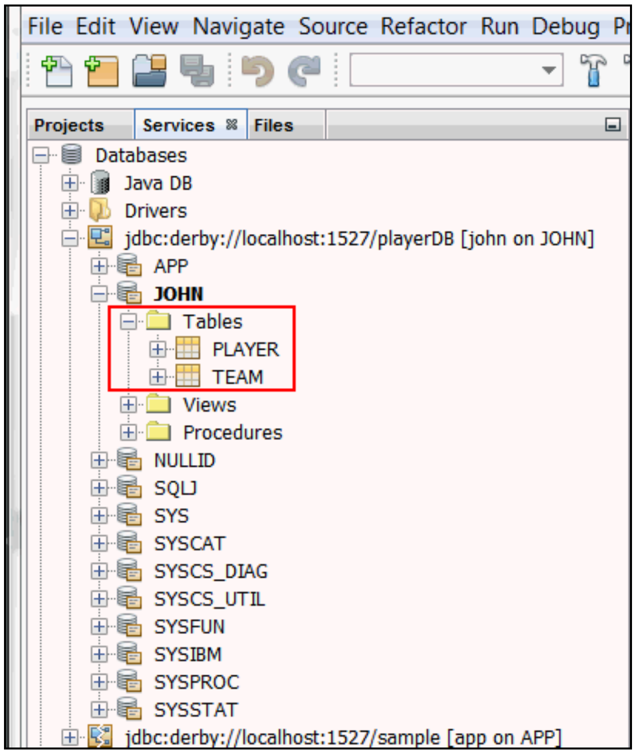

playerDB.sql
Source
History
Connection: jdbc:derby://localhost:1527/playerDB [john on JOHN]

5. Examine the contents of the database.

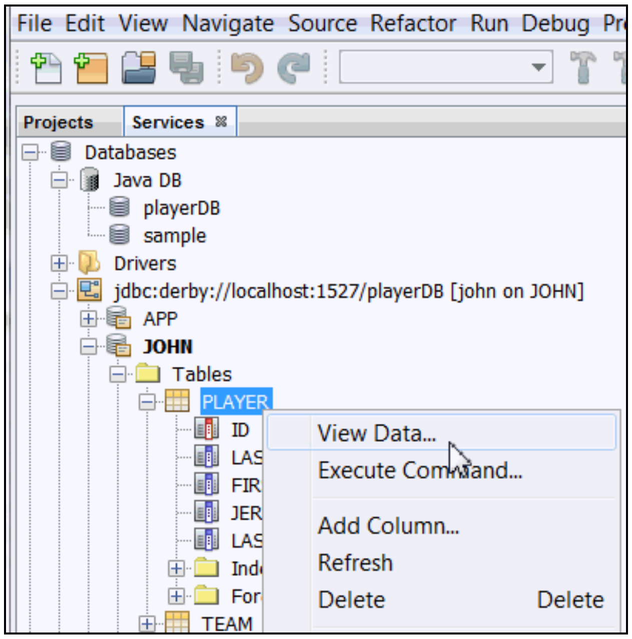
a. In the **Services** window, expand the `jdbc:derby://localhost:1527/playerDB` connection under the Databases node.
b. Right-click the connection and select **Refresh**.
c. Expand the **john** schema. You see the nodes for Tables, Views, and Procedures.



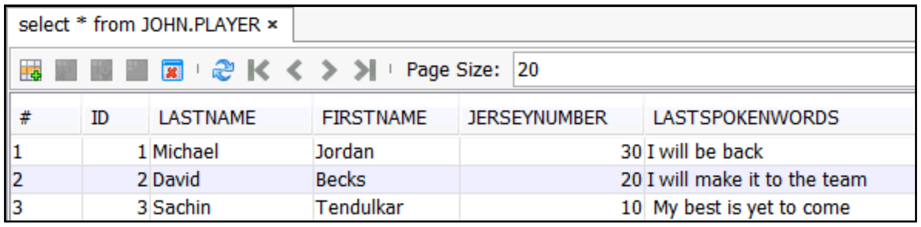
d. Expand the Tables node to see the PLAYER and TEAM tables.


File Edit View Navigate Source Refactor Run Debug
Projects
Services
Files
Databases
Java DB
Drivers
jdbc:derby://localhost:1527/playerDB [john on JOHN]
APP
JOHN
Tables
PLAYER
TEAM
Views
Procedures
NULLID
SQLJ
SYS
SYSCAT
SYSCS_DIAG
SYSCS_UTIL
SYSFUN
SYSIBM
SYSPROC
SYSSTAT
jdbc:derby://localhost:1527/sample [app on APP]

e. Right-click the PLAYER table node and select **View Data**.



An SQL command window opens and executes an SQL command to display the data in the table.



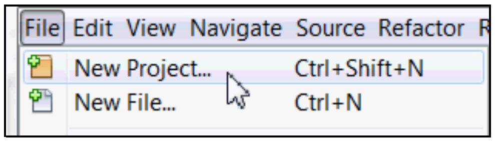
#	ID	LASTNAME	FIRSTNAME	JERSEYNUMBER	LASTSPOKENWORDS
1	1	Michael	Jordan	30	I will be back
2	2	David	Becks	20	I will make it to the team
3	3	Sachin	Tendulkar	10	My best is yet to come

f. Repeat the previous step for the TEAM table.

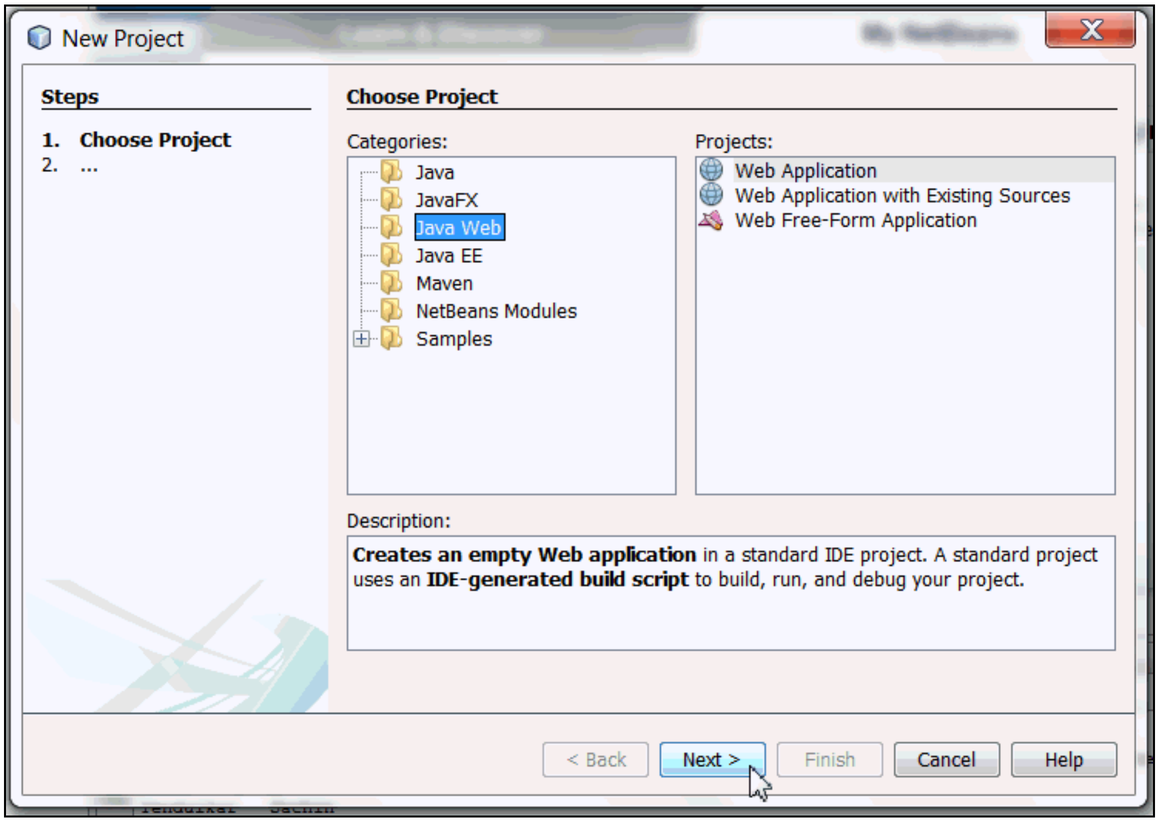
Building a Sample Web Application

To create RESTful Web Services, you need a Java Web application project. In the below section you will create a demo Java web project, PlayerServer.

1. To create new Java Web Project, select File > New Project.



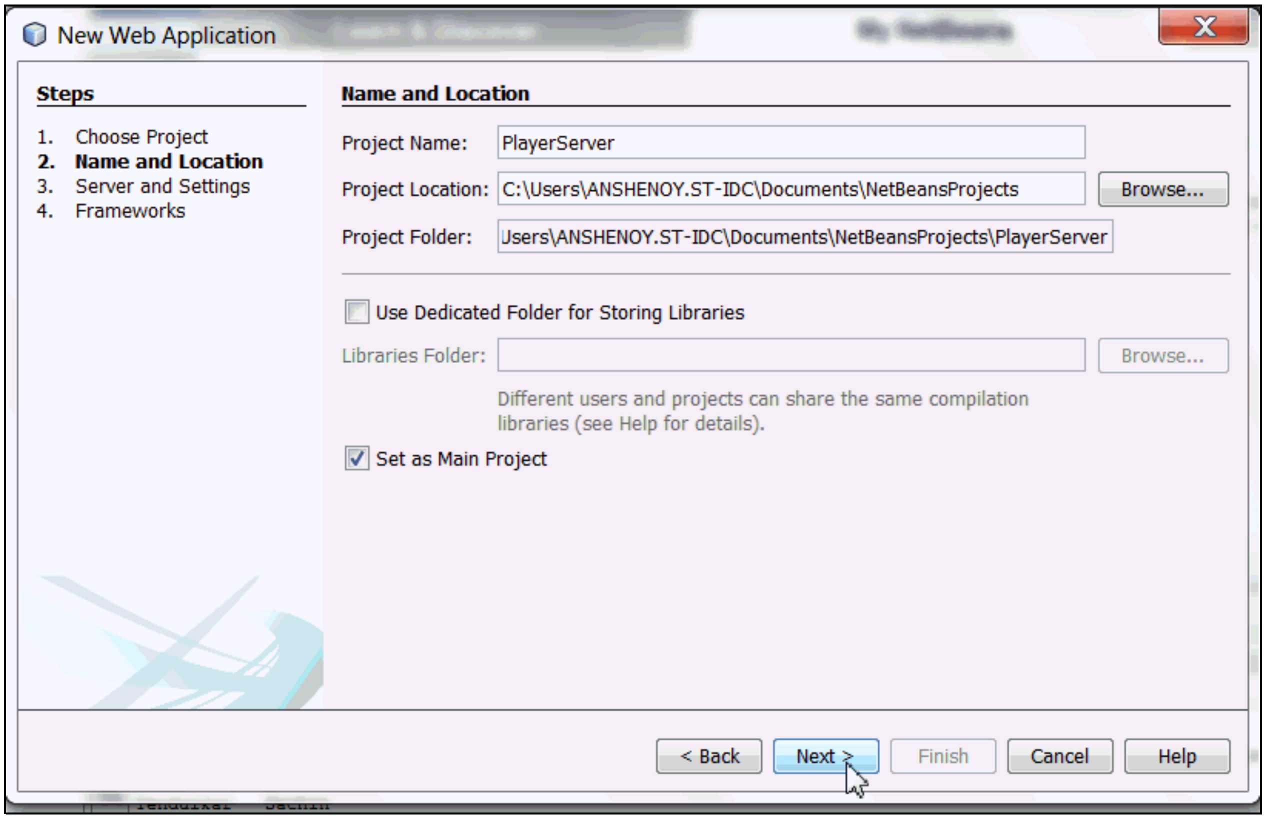
2. Select **Java Web** from the **Categories** column and **Web Application** from the **Projects** column and click **Next**.



Creates an empty Web application in a standard IDE project. A standard project uses an **IDE-generated build script** to build, run, and debug your project.

3. Perform the following steps:

   a. Name the project **PlayerServer**.
   b. Click Next.



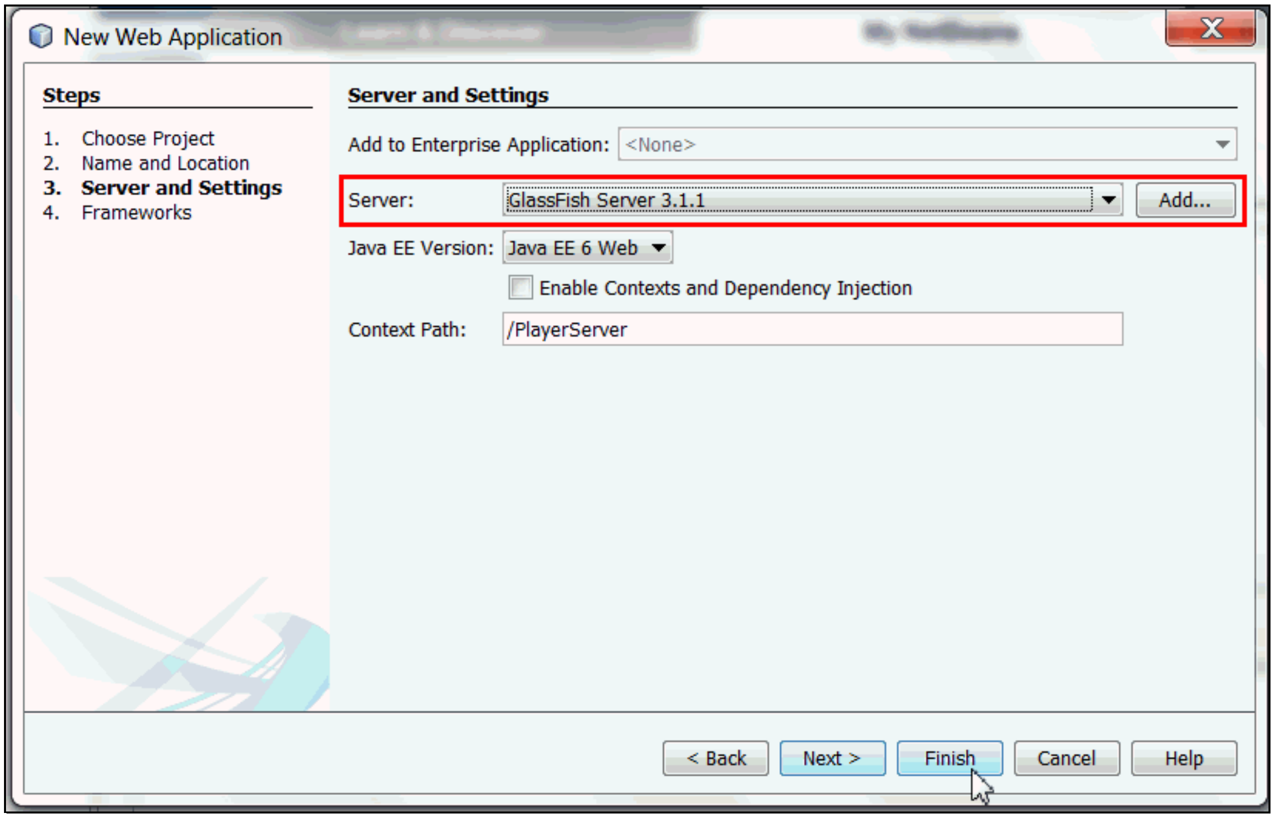
New Web Application

Steps
1. Choose Project
2. **Name and Location**
3. Server and Settings
4. Frameworks

Name and Location

Project Name: PlayerServer
Project Location: C:\Users\ANSHENOY.ST-IDC\Documents\NetBeansProjects
Project Folder: Jsers\ANSHENOY.ST-IDC\Documents\NetBeansProjects\PlayerServer
Use Dedicated Folder for Storing Libraries
Libraries Folder:
Different users and projects can share the same compilation libraries (see Help for details).
Set as Main Project

4. In the Server and Settings window, verify GlassFish Server is selected as Server and click Finish.



New Web Application

Steps
1. Choose Project
2. Name and Location
3. **Server and Settings**
4. Frameworks

Server and Settings

Add to Enterprise Application: <None>
Server: GlassFish Server 3.1.1
Java EE Version: Java EE 6 Web
Enable Contexts and Dependency Injection
Context Path: /PlayerServer

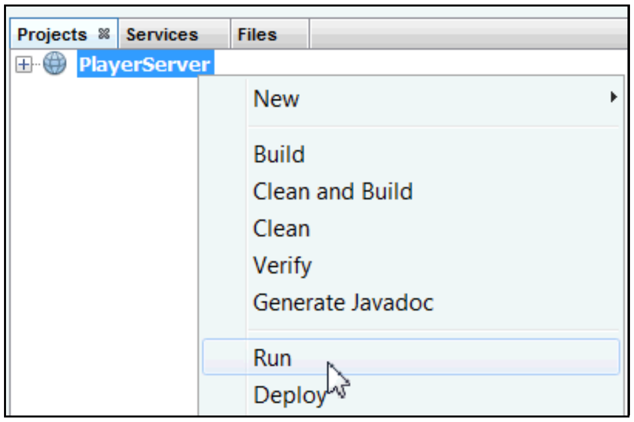
5. To start the Server, perform the following steps :

a. Open the Projects tab.

b. Right-click PlayerServer project.

c. Select Run.

This action starts the GlassFish server and deploys the application.



On successful deployment of the application - a default jsp page with url, `http://localhost:8080/PlayerServer/` is opened in the browser displaying "Hello World".

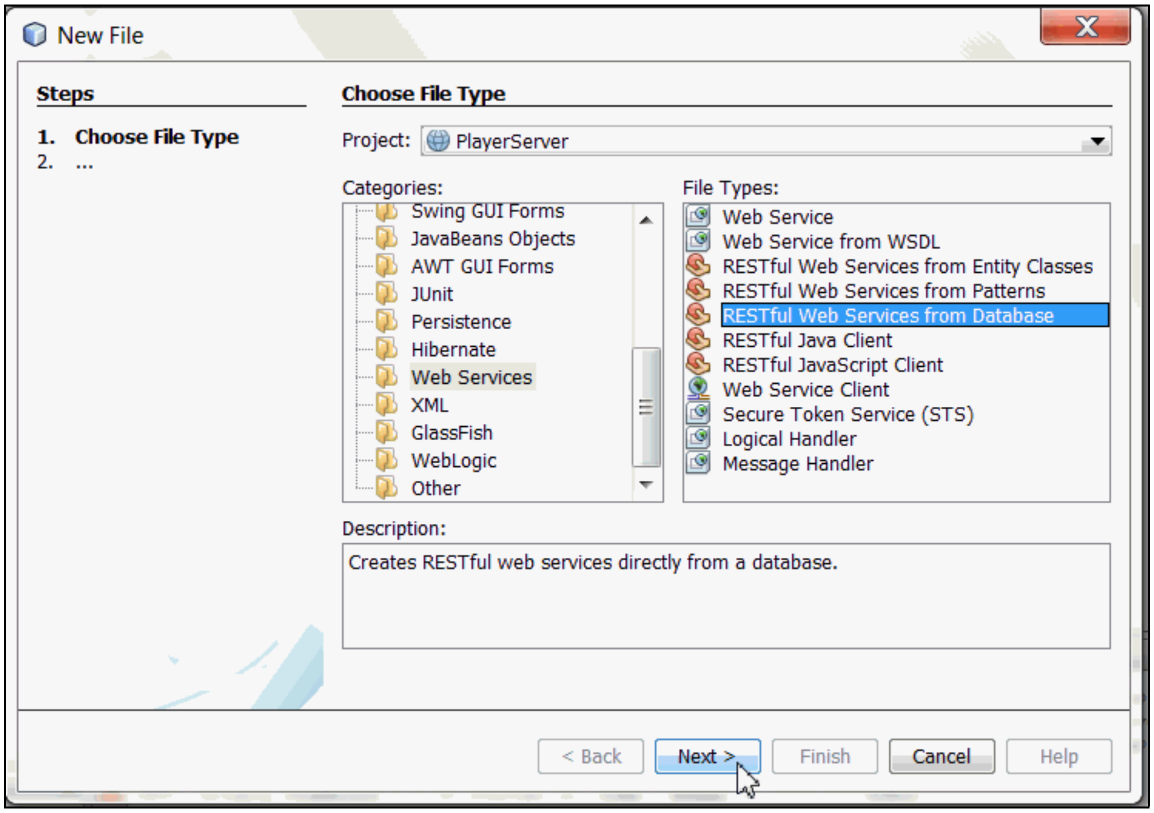
Generate RESTful Web Services

Creation of RESTful Web Services in Java depends on the Java Persistence API to communicate with a database. You can use NetBeans IDE either to create entity classes and RESTful Web Services in the same process, or you can use the IDE to create RESTful Web Services from existing entity classes.

The below section demonstrates how to use the **RESTful Services from Database wizard** to generate entity classes and RESTful Web Services in the same process.

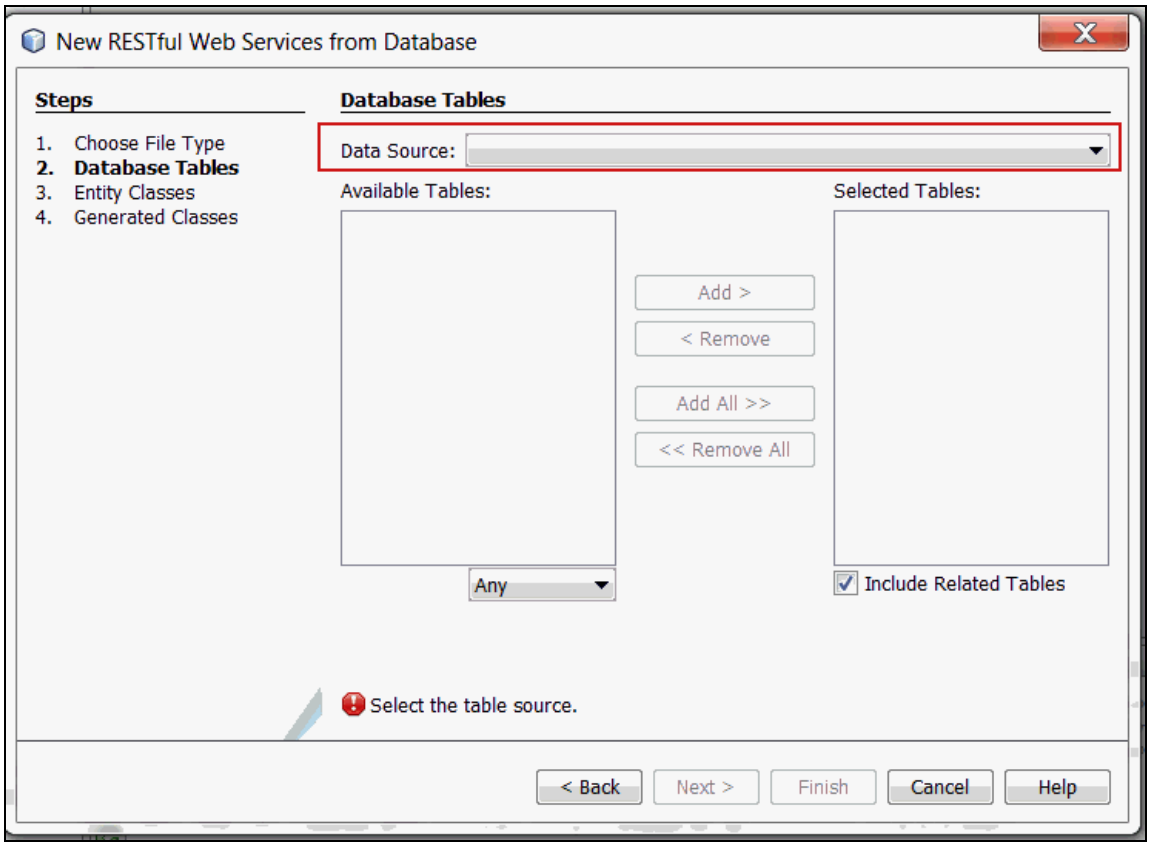
1 To generate RESTful Web Services do the following:

Right-click the PlayerServer and choose **`New > Other > Web Services > RESTful Web Services from Database`**. The New RESTful Web Service wizard opens on the Database Tables panel.

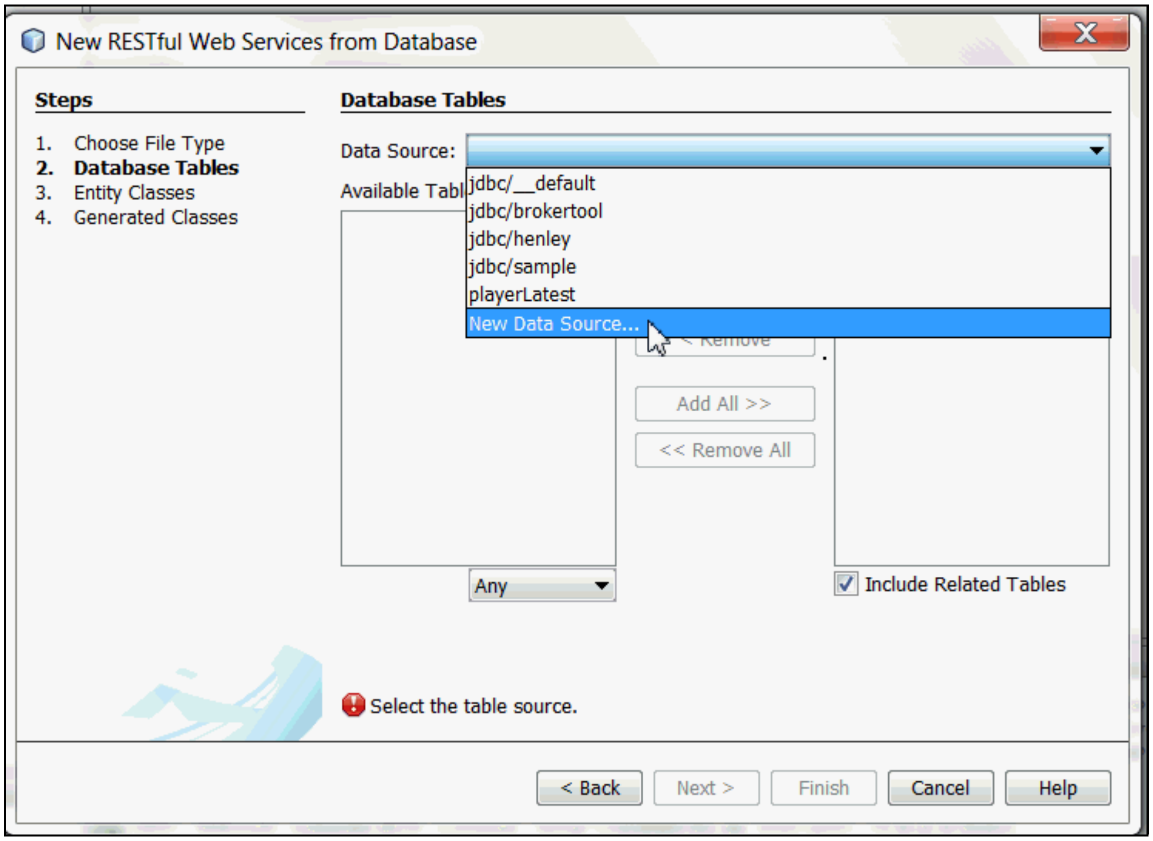


Creates RESTful web services directly from a database.

2. In the Database Tables window, select `Data Source`.



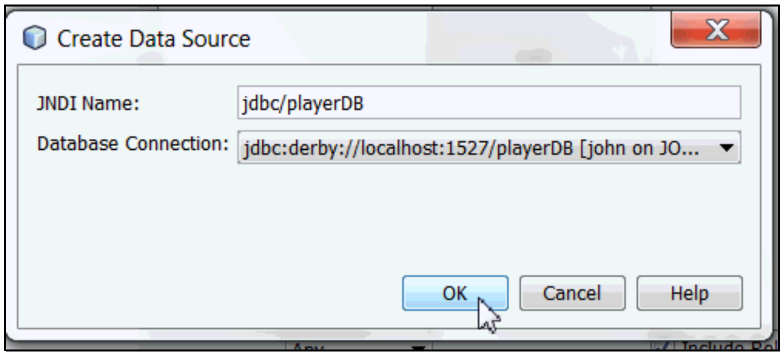
3. Next select "New Data Source" from the Drop-down list.



a. In the Create Data Source Window, enter the following information:

JNDI name: `jdbc/playerDB`
Database connection : select `jdbc:derby://localhost:1527/playerDB[john on JOHN]`

b. Click OK.

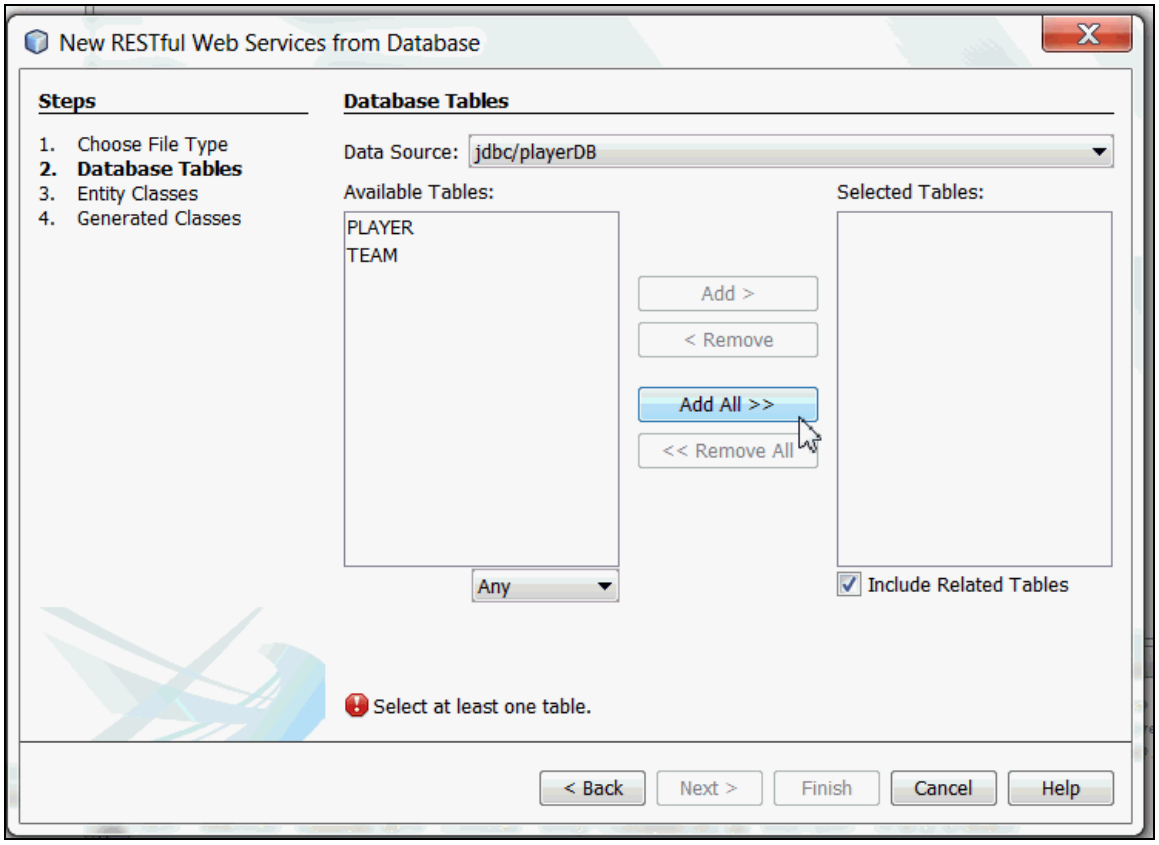

Create Data Source

JNDI Name: jdbc/playerDB

Database Connection: jdbc:derby://localhost:1527/playerDB [john on JO...

The PLAYER and TEAM tables are displayed under the Available Tables column.

c. Click **Add All**. The PLAYER and TEAM tables are displayed under the Selected Tables column.


New RESTful Web Services from Database

Steps

1. Choose File Type
2. **Database Tables**
3. Entity Classes
4. Generated Classes

Database Tables

Data Source: jdbc/playerDB

Available Tables:

PLAYER
TEAM

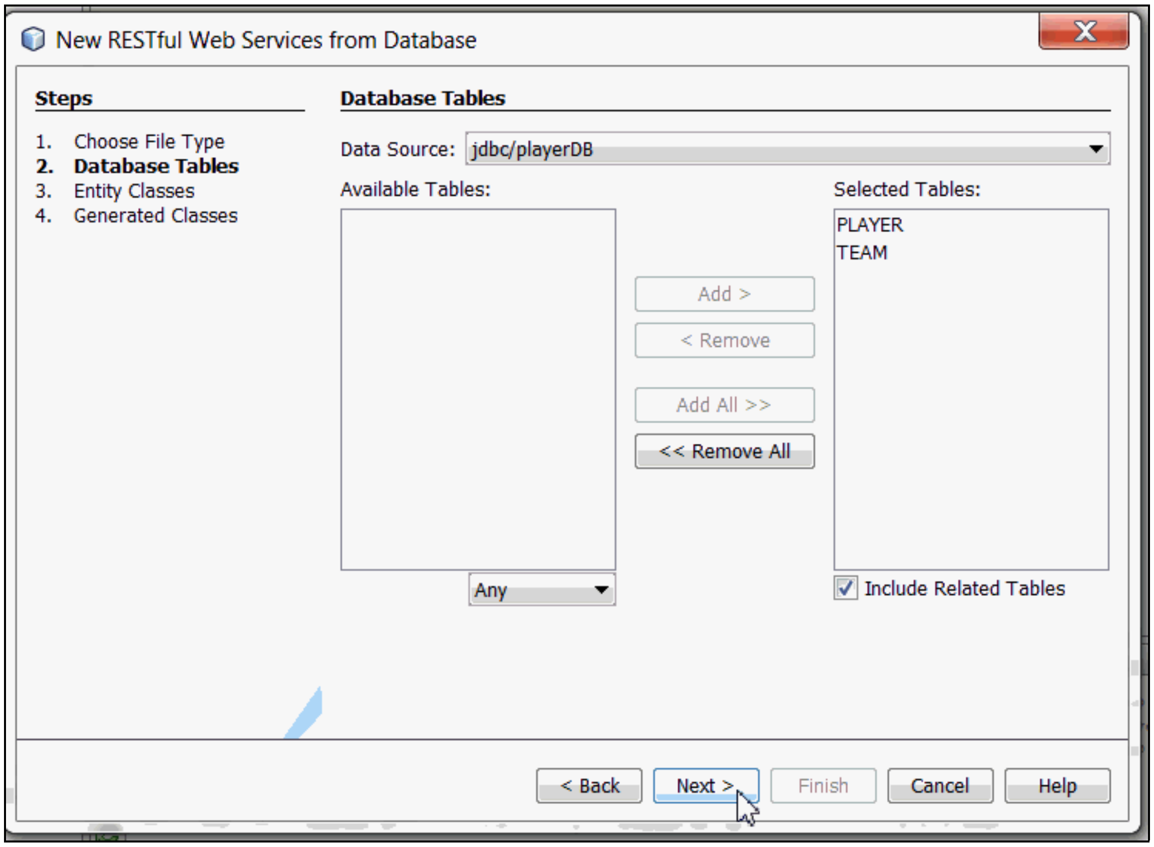
Selected Tables:

Any

☑ Include Related Tables

Select at least one table.

d. Click Next.


New RESTful Web Services from Database

Steps

1. Choose File Type
2. **Database Tables**
3. Entity Classes
4. Generated Classes

Database Tables

Data Source: jdbc/playerDB

Available Tables:

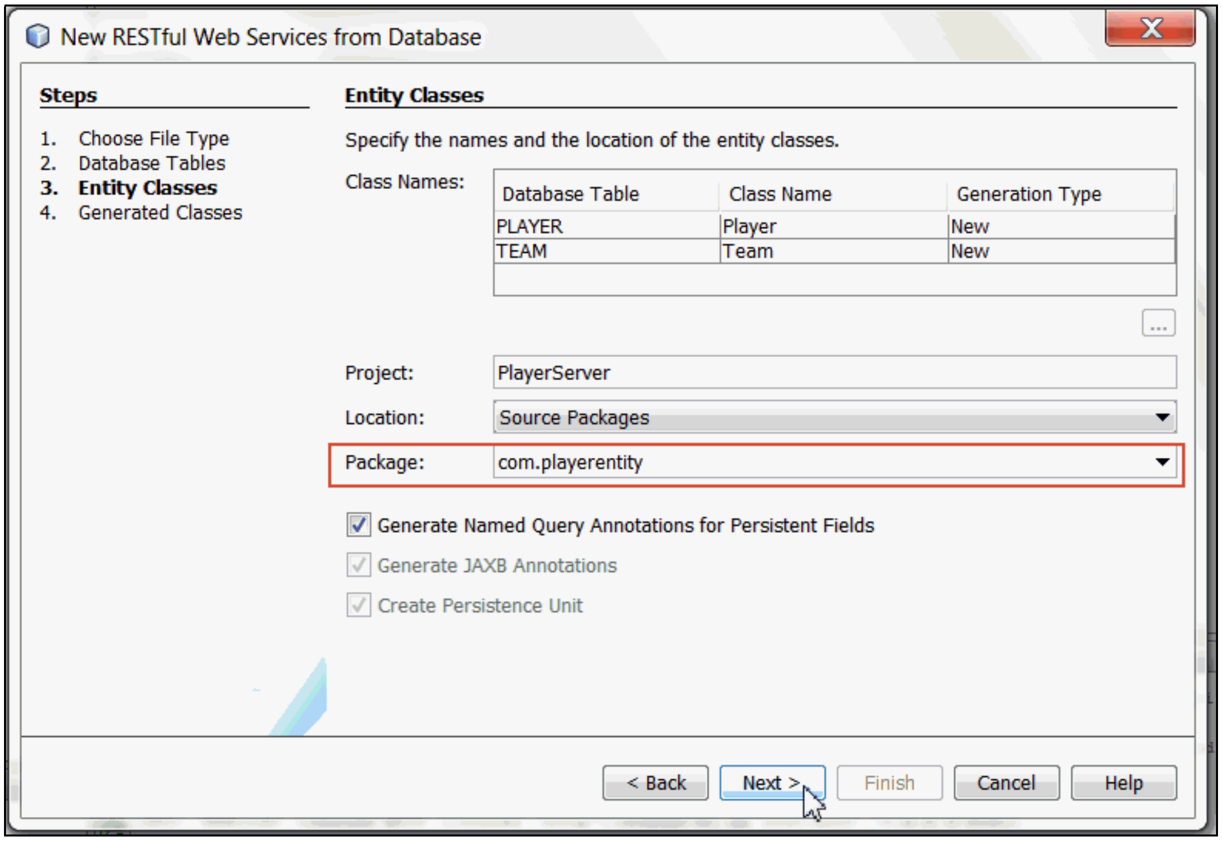
Selected Tables:
PLAYER
TEAM

Any

☑ Include Related Tables

4. In the **Entity Classes** window, complete the following steps:

a. Enter the package name as `com.playerentity`.


New RESTful Web Services from Database

Steps

1. Choose File Type
2. Database Tables
3. **Entity Classes**
4. Generated Classes

Entity Classes

Specify the names and the location of the entity classes.

Class Names:

Database Table	Class Name	Generation Type
PLAYER	Player	New
TEAM	Team	New

Project: PlayerServer

Location: Source Packages

Package: com.playerentity

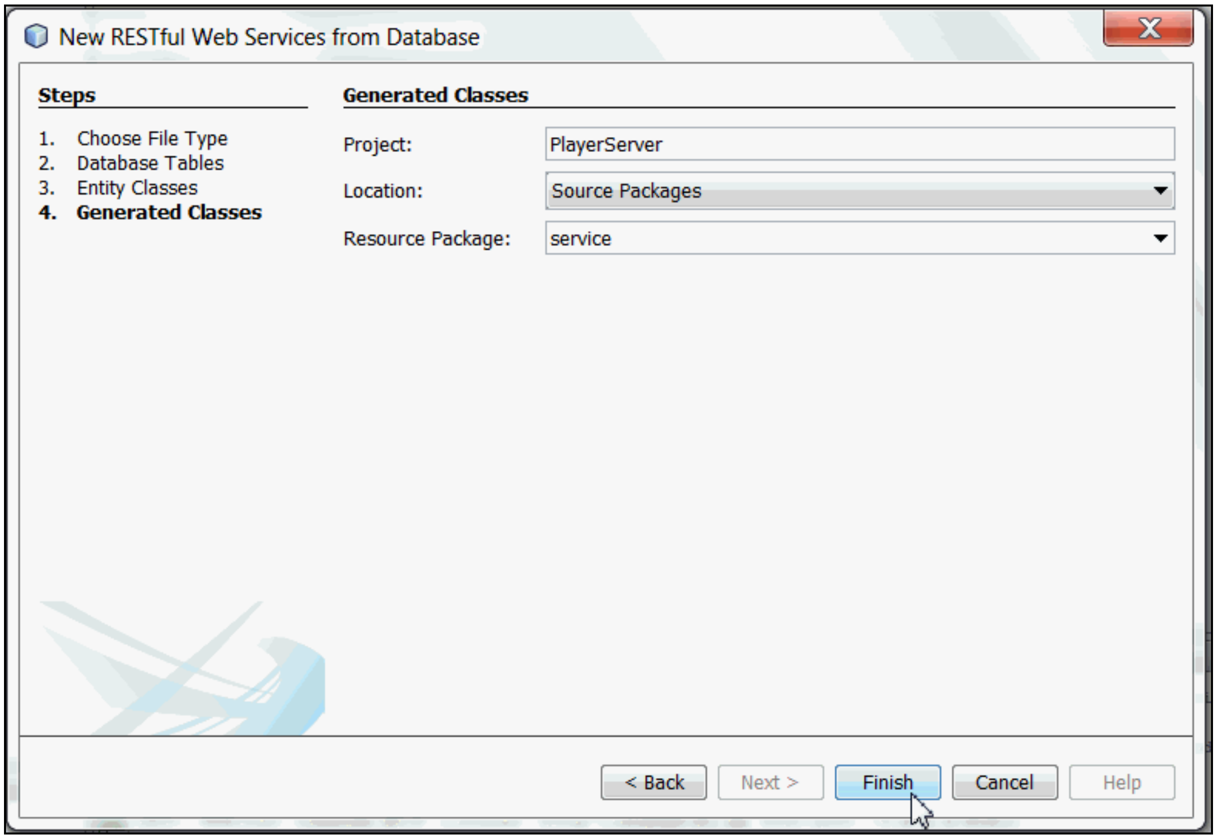
☑ Generate Named Query Annotations for Persistent Fields

☑ Generate JAXB Annotations

☑ Create Persistence Unit

b. Click Next.

5. In the **Generated Classes** Window, click Finish with default values.


New RESTful Web Services from Database

Steps
1. Choose File Type
2. Database Tables
3. Entity Classes
4. **Generated Classes**

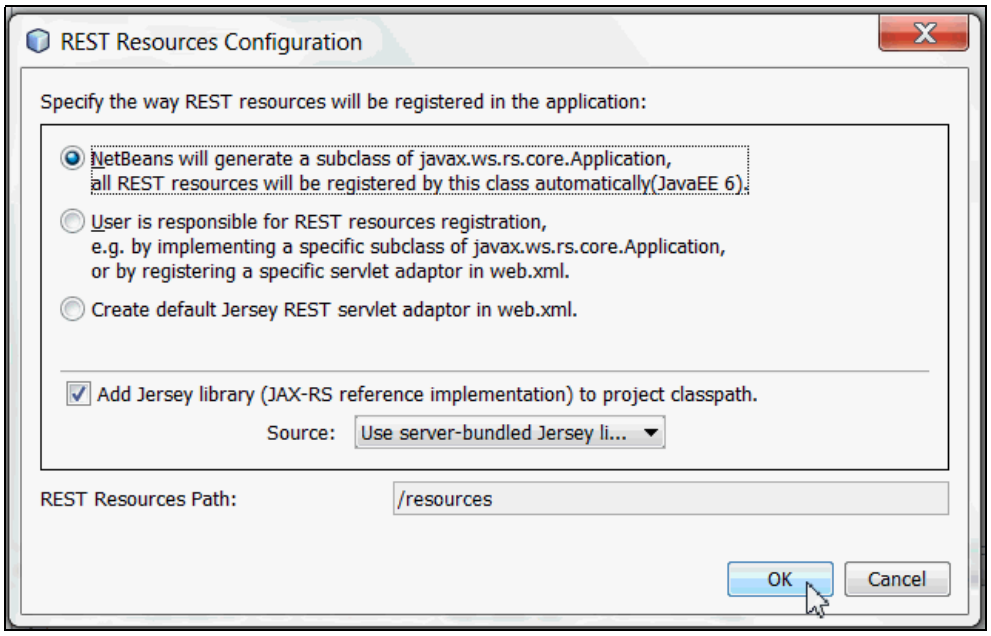
Generated Classes

Project: PlayerServer

Location: Source Packages

Resource Package: service

6. In the **Rest Resources Configuration** Window, click OK.


REST Resources Configuration

Specify the way REST resources will be registered in the application:

● NetBeans will generate a subclass of javax.ws.rs.core.Application, all REST resources will be registered by this class automatically(JavaEE 6).

○ User is responsible for REST resources registration, e.g. by implementing a specific subclass of javax.ws.rs.core.Application, or by registering a specific servlet adaptor in web.xml.

○ Create default Jersey REST servlet adaptor in web.xml.

✓ Add Jersey library (JAX-RS reference implementation) to project classpath.

Source: Use server-bundled Jersey li...

REST Resources Path: /resources

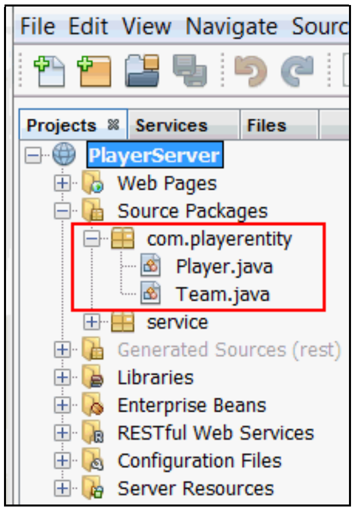
7. In the Projects pane, perform the below steps :

a. Select and expand the **PlayerServer** project.

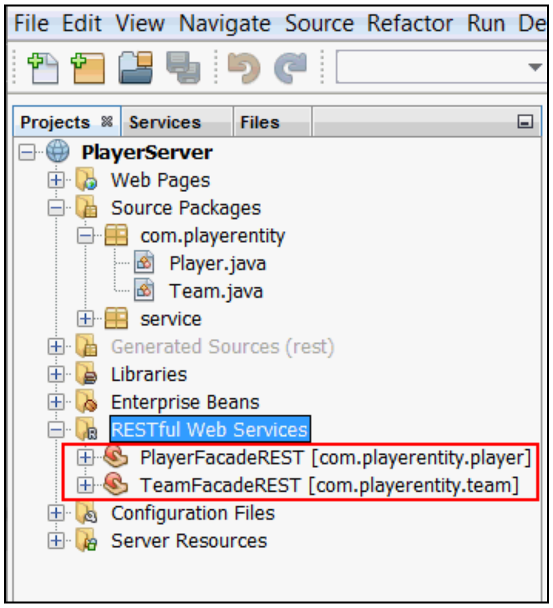
b. Expand the **source packages** of the project.

c. Expand the **`com.playerentity`** package.

d. Verify the creation of Entity Classes **`Player.java`** and **`Team.java`**.



e. Expand the **RESTful Web Services folder** and and verify the creation of two RESTful Web Services `PlayerFacadeREST[com.playerentity.player]` and `TeamFacadeREST[com.playerentity.team]`.

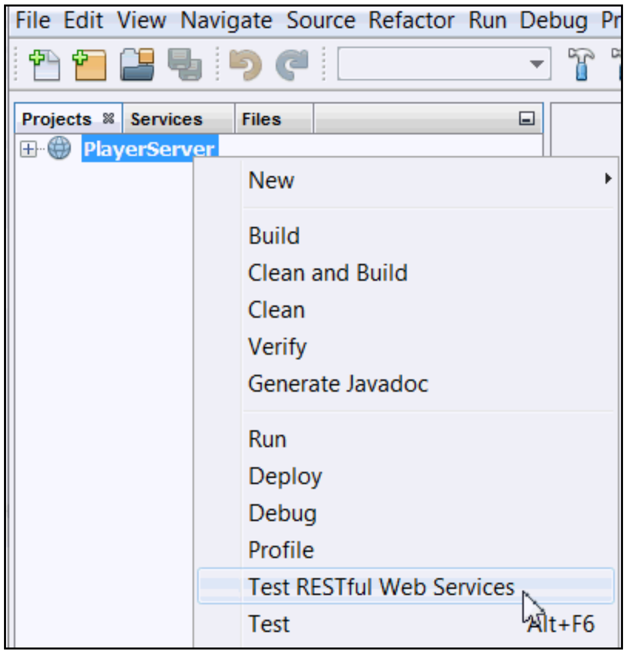


The above generated two Web Services are the REST front end of our application. For each entity, there is a resource that lists all the entity's instances.

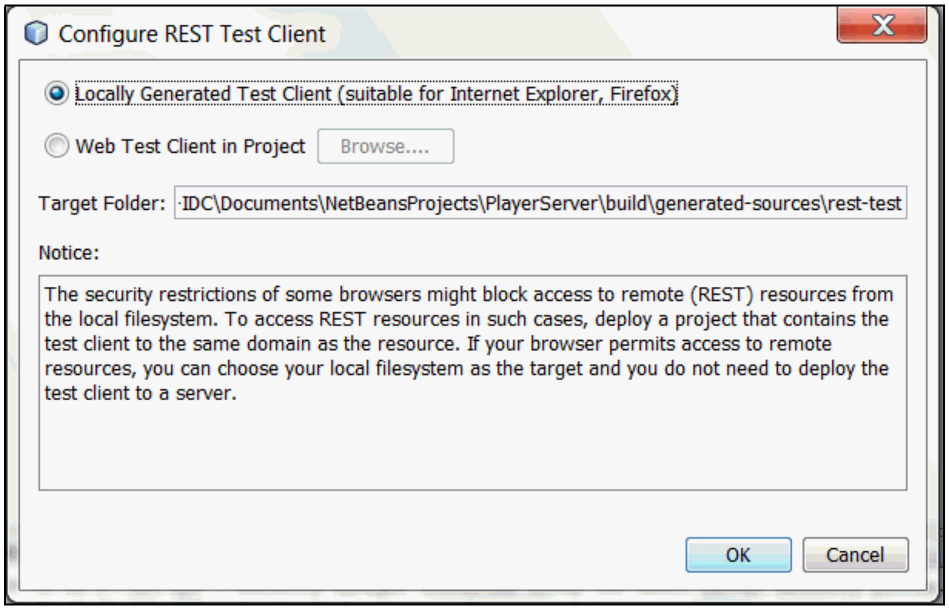
Test RESTful Web Services

The following section demonstrates how to test RESTful Web Services with tests that are generated using the test framework provided by Jersey.

1 . To Generate Web Services Test client, complete the following steps.
a. Select **PlayerServer** project.
b. Right-click and select **Test RESTful Web Services.**



Configure REST test Client window opens as shown in the below screenshot.



(●) Locally Generated Test Client (suitable for Internet Explorer, Firefox)

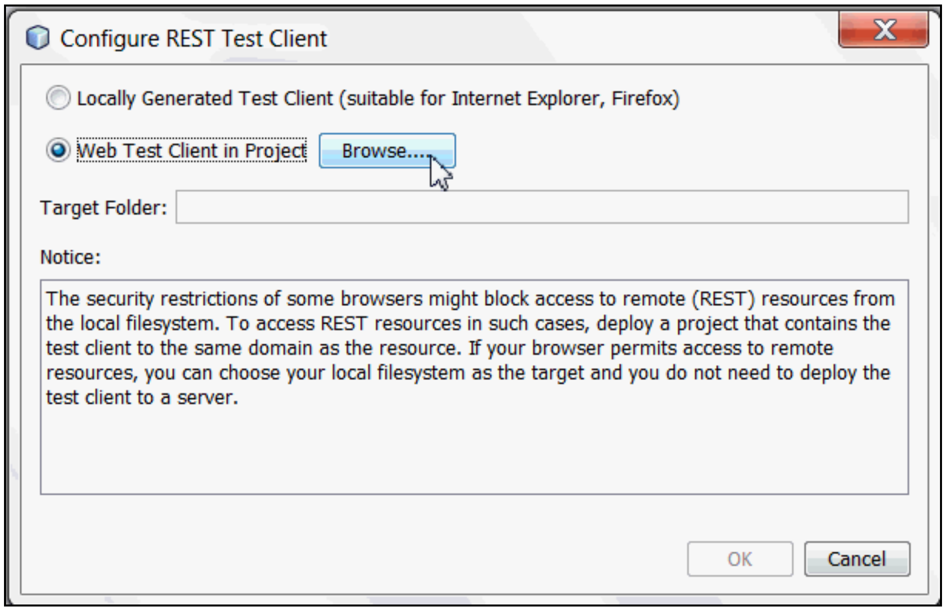
( ) Web Test Client in Project

Target Folder: ·IDC\Documents\NetBeansProjects\PlayerServer\build\generated-sources\rest-test

Notice:

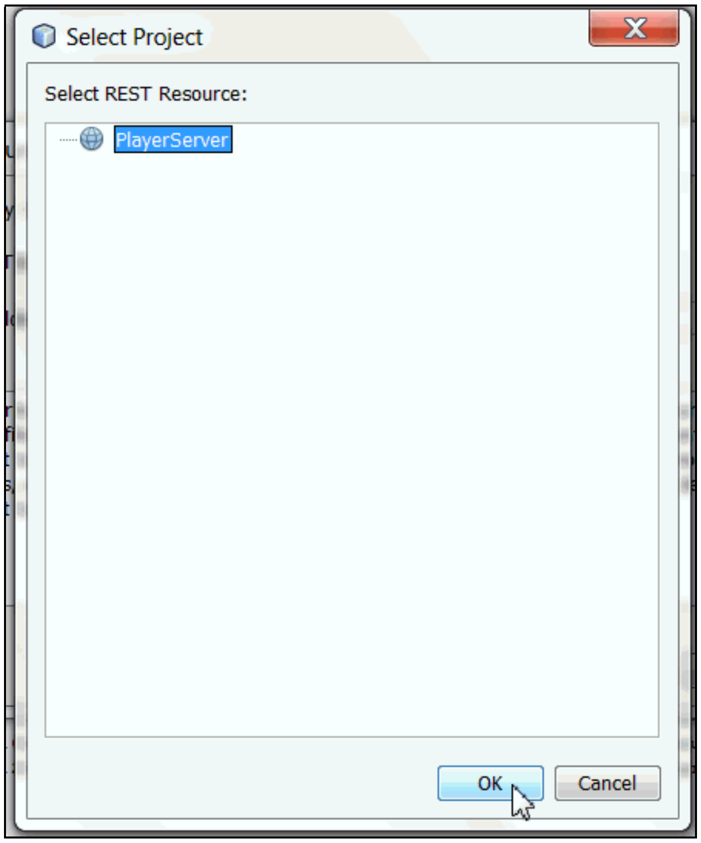
The security restrictions of some browsers might block access to remote (REST) resources from the local filesystem. To access REST resources in such cases, deploy a project that contains the test client to the same domain as the resource. If your browser permits access to remote resources, you can choose your local filesystem as the target and you do not need to deploy the test client to a server.

2 . Select "Web Test Client in project" and click Browse.

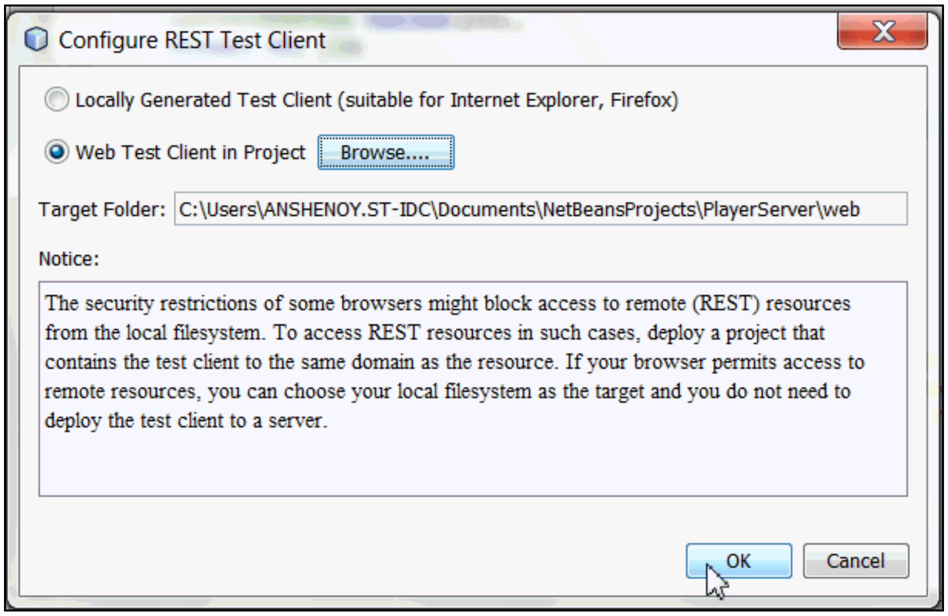

Configure REST Test Client
Locally Generated Test Client (suitable for Internet Explorer, Firefox)
Web Test Client in Project
Browse....
Target Folder:
Notice:
OK
Cancel

The security restrictions of some browsers might block access to remote (REST) resources from the local filesystem. To access REST resources in such cases, deploy a project that contains the test client to the same domain as the resource. If your browser permits access to remote resources, you can choose your local filesystem as the target and you do not need to deploy the test client to a server.

3. a. In the Select Project dialog box, select **PlayerServer** and click OK.



b. **Configure Rest Test Client window** is displayed, click OK.


Configure REST Test Client

Locally Generated Test Client (suitable for Internet Explorer, Firefox)

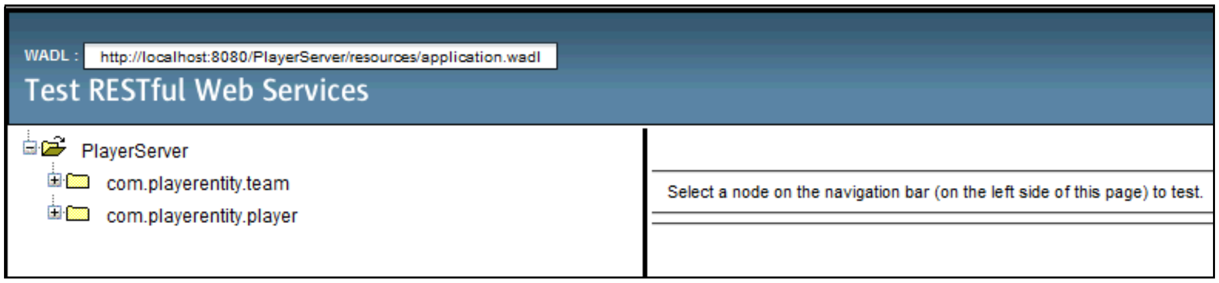
Web Test Client in Project [✓] Browse....

Target Folder: C:\Users\ANSHENOY.ST-IDC\Documents\NetBeansProjects\PlayerServer\web

Notice:

The security restrictions of some browsers might block access to remote (REST) resources from the local filesystem. To access REST resources in such cases, deploy a project that contains the test client to the same domain as the resource. If your browser permits access to remote resources, you can choose your local filesystem as the target and you do not need to deploy the test client to a server.

The server starts and the application is deployed. When deployment is complete, the browser displays your application, with a link for each of the Web Services.


WADL : http://localhost:8080/PlayerServer/resources/application.wadl

Test RESTful Web Services

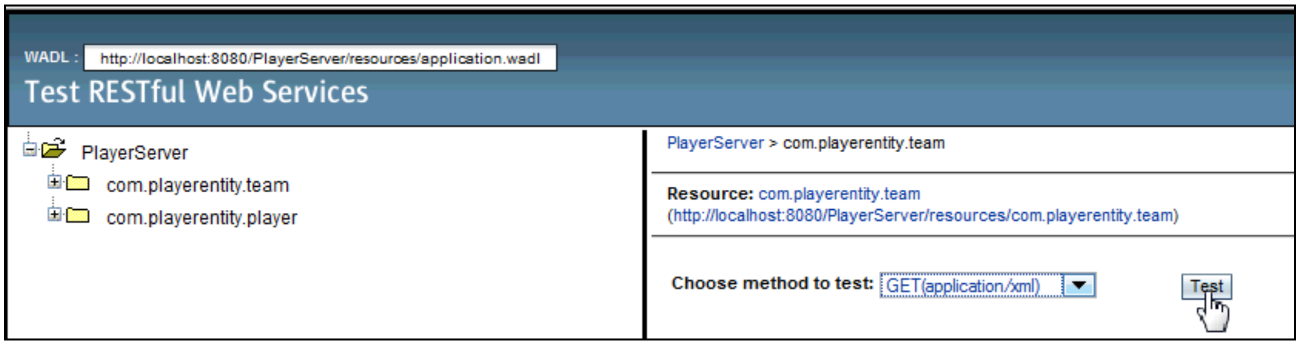
PlayerServer
com.playerentity.team
com.playerentity.player

Select a node on the navigation bar (on the left side of this page) to test.

On the left-hand side is the set of root resources named **`com.playerentity.team`** and **`com.playerentity.player`**.

4 . To test the Web Services Client, complete the following steps.

a. Select one resource node such as `com.playerentity.team`.
b. In the "Choose method to test" field, select either GET (application/json) or GET (application/xml).
c. Click Test.


WADL : http://localhost:8080/PlayerServer/resources/application.wadl

Test RESTful Web Services

PlayerServer
com.playerentity.team
com.playerentity.player

PlayerServer > com.playerentity.team

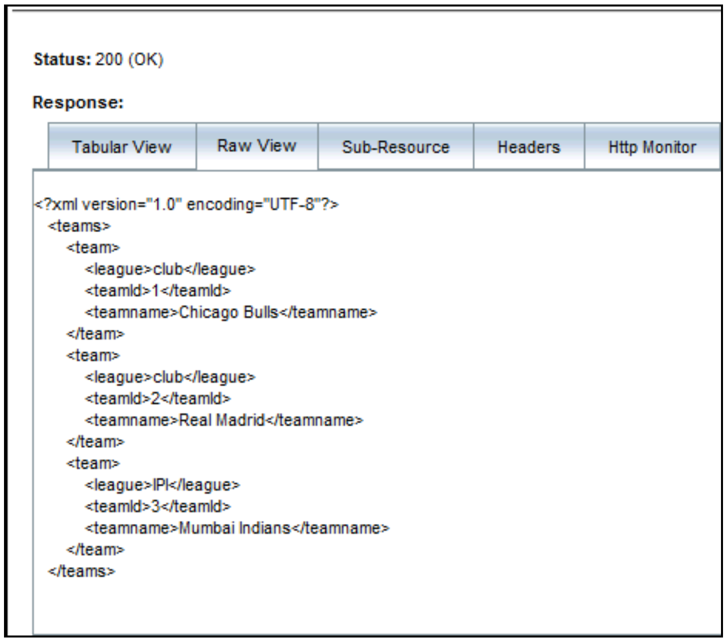
Resource: com.playerentity.team (http://localhost:8080/PlayerServer/resources/com.playerentity.team)

Choose method to test: GET(application/xml)

The test client sends a request and displays the result in the Test Output section. The test client displays the Raw View by default.

d. Examine the output:

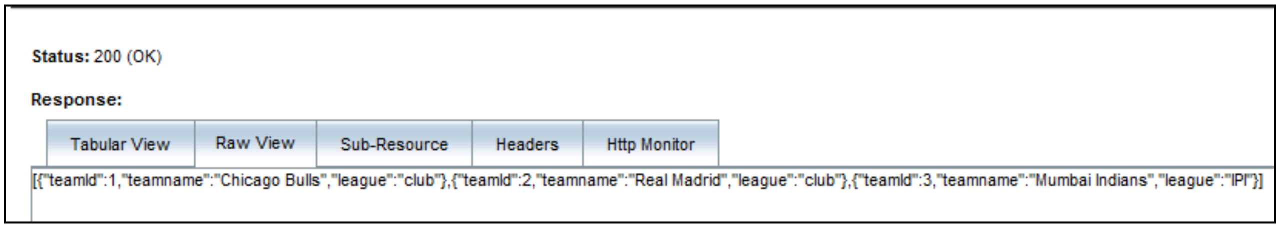
Response to an `application/xml` request.


Status: 200 (OK)

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <teams>
    <team>
      <league>club</league>
      <teamId>1</teamId>
      <teamname>Chicago Bulls</teamname>
    </team>
    <team>
      <league>club</league>
      <teamId>2</teamId>
      <teamname>Real Madrid</teamname>
    </team>
    <team>
      <league>IPl</league>
      <teamId>3</teamId>
      <teamname>Mumbai Indians</teamname>
    </team>
  </teams>
```

Response to an `application/json` request.


Status: 200 (OK)

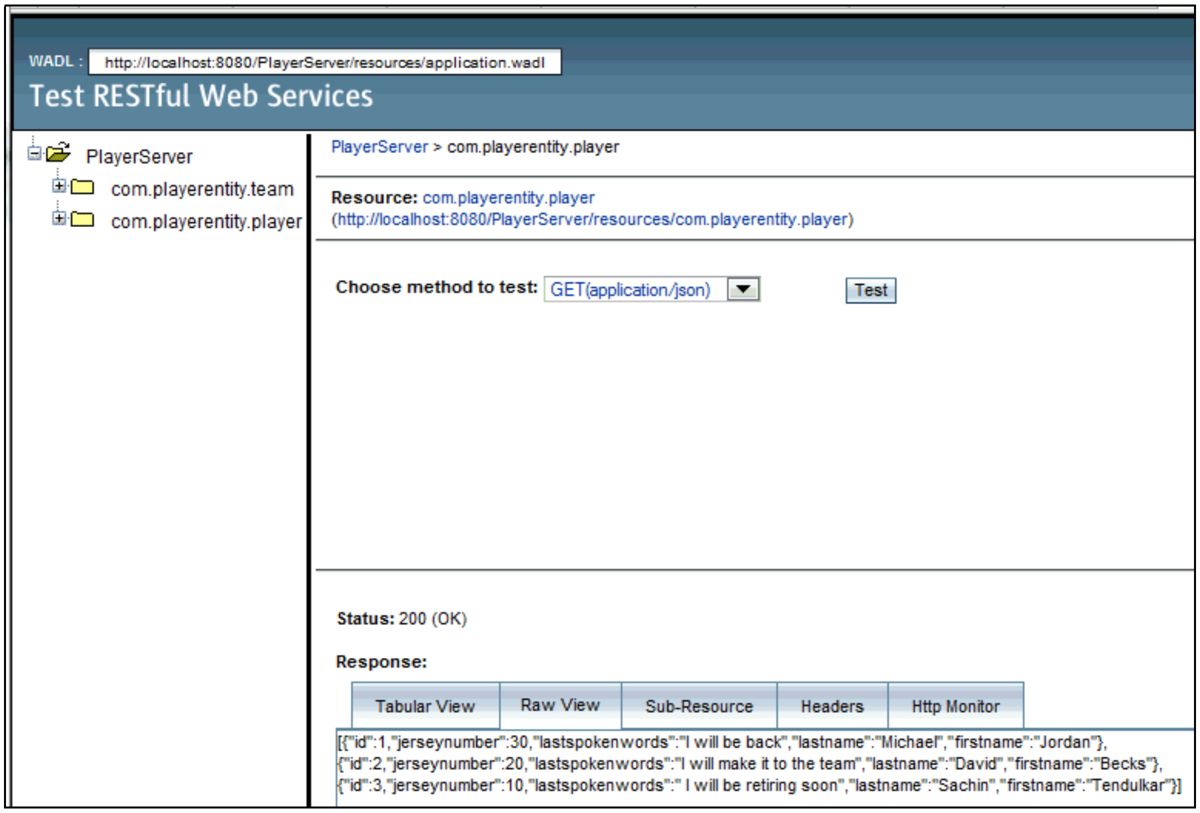
Response:

```
[{"teamId":1,"teamname":"Chicago Bulls","league":"club"},{"teamId":2,"teamname":"Real Madrid","league":"club"},{"teamId":3,"teamname":"Mumbai Indians","league":"IPl"}]
```

JSON is widely used in REST-based applications, because this format is more compact than XML.

Also it can easily be used with most common programming languages, including JavaScript.

e. Likewise select the `com.playerentity.player` node and test the Web Service.


WADL : http://localhost:8080/PlayerServer/resources/application.wadl

Test RESTful Web Services

PlayerServer > com.playerentity.player

Resource: com.playerentity.player (http://localhost:8080/PlayerServer/resources/com.playerentity.player)

Choose method to test: GET(application/json) Test

Status: 200 (OK)

Response:

```
[{"id":1,"jerseynumber":30,"lastspokenwords":"I will be back","lastname":"Michael","firstname":"Jordan"},
{"id":2,"jerseynumber":20,"lastspokenwords":"I will make it to the team","lastname":"David","firstname":"Becks"},
{"id":3,"jerseynumber":10,"lastspokenwords":" I will be retiring soon","lastname":"Sachin","firstname":"Tendulkar"}]
```

Summary

This tutorial provides an introduction to RESTful Web Services. You learned how NetBeans allows you to qiuckly develop and test REST-based applications.

Resources

- Using Java Persistence API for Java SE 7 Desktop applications in NetBeans 7
- JSR 311: JAX-RS: The Java API for RESTful Web Services
- The Jersey project home page

Credits

Curriculum Developer: Anjana Shenoy