



[Documentation Home](#) > [The Java EE 6 Tutorial, Volume I](#) > [Part III Web Services](#) > [Chapter 13 Building RESTful Web Services with JAX-RS and Jersey](#) > [Example Applications for JAX-RS and Jersey](#) > [Creating a RESTful Web Service](#) > [Creating a RESTful Web Service Using NetBeans IDE](#)

The Java EE 6 Tutorial, Volume I

« **Previous:** [Creating a RESTful Root Resource Class](#)

Next: [Creating a RESTful Web Service From Examples](#) »

▼ Creating a RESTful Web Service Using NetBeans IDE

This section describes, using a very simple example, how to create a Jersey-annotated web application from NetBeans IDE.

1. In NetBeans IDE, create a simple web application. This example creates a very simple “Hello, World” web application.

- a. Open NetBeans IDE.
- b. Select File->New Project.
- c. From Categories, select Java Web. From Projects, select Web Application. Click Next.

Note –

For this step, you could also create a RESTful web service in a Maven web project by selecting Maven as the category and Maven Web Project as the project. The remaining steps would be the same.

- d. Enter a project name, **HelloWorldApplication** , click Next.
 - e. Make sure the Server is Sun GlassFish v3 (or similar wording.)
 - f. Click Finish. You may be prompted for your server Administrator User Name and Password. If so, enter this information.
2. The project will be created. The file `index.jsp` will display in the Source pane.
3. Right-click the project and select New, then select RESTful Web Services from Patterns.

- a. Select Simple Root Resource. Click Next.
- b. Enter a Resource Package name, like **helloWorld** .
- c. Enter **helloworld** in the Path field. Enter **HelloWorld** in the Class Name field. For MIME Type select `text/html` .
- d. Click Finish.
- e. The REST Resources Configuration page displays. Select OK.

A new resource, `HelloWorld.java` , is added to the project and displays in the Source pane. This file provides a template for

creating a RESTful web service.

4. In `HelloWorld.java`, find the `getHtml()` method. Replace the `//TODO` comment and the exception with the following text, so that the finished product resembles the following method.

Note –

Because the MIME type that is produces is HTML, you can use HTML tags in your return statement.

```
/**
 * Retrieves representation of an instance of helloWorld.HelloWorld
 * @return an instance of java.lang.String
 */
@GET
@Produces("text/html")
public String getHtml() {
    return "<html><body><h1>Hello, World!!</body></h1></html>";
}
```

5. Test the web service. To do this, right-click the project node and click Test RESTful Web Services.

This step will deploy the application and bring up a test client in the browser.

6. When the test client displays, select the `helloworld` resource in the left pane, and click the Test button in the right pane.

The words Hello, World!! will display in the Response window below.

7. Deploy and Run the application.

- a. Set the Run Properties. To do this, right-click the project node, select Properties, and then select the Run category. Set the Relative URL to the location of the RESTful web service relative to the Context Path, which for this example is `resources/helloworld`.

Tip –

You can find the value for the Relative URL in the Test RESTful Web Services browser window. In the top of the right pane, after Resource, is the URL for the RESTful web service being tested. The part following the Context Path (`http://localhost:8080/HelloWorldApp`) is the Relative URL that needs to be entered here.

If you don't set this property, by default the file `index.jsp` will display when the application is run. As this file also contains `Hello World` as its default value, you might not notice that your RESTful web service isn't running, so just be aware of this default and the need to set this property, or update `index.jsp` to provide a link to the RESTful web service.

- b. Right-click the project and select Deploy.

- c. Right-click the project and select Run.

A browser window opens and displays the return value of `Hello, World!!`

See Also

For other sample applications that demonstrate deploying and running Jersey applications using NetBeans, read [Example: Creating a Simple Hello World Application Using JAX-RS and Jersey](#) and/or look at the tutorials on the NetBeans tutorial site, such as the one titled [Getting Started with RESTful Web Services](#). This tutorial includes a section on creating a CRUD application from a database. Create, read, update and delete (CRUD) are the four basic functions of persistent storage and relational databases.

« Previous: Creating a RESTful Root Resource Class

Next: Creating a RESTful Web Service From Examples »