Preparing for REST Development

At the IntelliJ IDEA level, REST development is supported through the *WebServices* plugin. The plugin is bundled with IntelliJ IDEA and enabled by default. If not, enable the plugin in the Plugin Configuration Wizard or in the Plugins page of the Settings dialog box.

REST development in a module does not require any specific module structure or configuration files. Actually, it is a plain Java module with Jersey reference implementation `.jar` files configured as libraries.

To enable REST development, create a project with a module for REST from scratch, or add a REST module to a project, or attach a REST Web Service facet to an existing module.

On this page:

Creating a project with a module for REST from scratch

1. Start creating a project from scratch. Do one of the following:
   - If no project is currently opened in IntelliJ IDEA, click **Create New Project** on the **Welcome** screen.
   - If you already have an opened project, choose **File | New Project** on the main menu.

   The New Project Wizard starts.
2. On the **Main Settings** page, choose **Java Module** in the **Java** area.
   1. In the right-hand pane, specify the project name, location, and the location of the JDK to use in the project.
   2. In the **More Settings** area, specify the module name, the location of the content root folder and of the module .iml file.

   Click **Next**.
3. On the Technologies page that opens, select the **RESTful WebServices** check box.
4. In the **Libraries** area, appoint the `.jar` files to use:
   - To use previously downloaded `.jar` files configured as a library, choose the **Use Library** option and select the relevant library from the drop-down list.
   - To use previously downloaded `.jar` files that have not been arranged in a library yet, choose the **Use Library** option, click the **Create** button, and then select the relevant `.jar` files.

     By default, IntelliJ IDEA assigns the library the name `JAX-RS` and configures it at the *project level*, whereupon the library can be used in the current module and in all other modules of the current project. To be re-used in any IntelliJ IDEA project, a library should be configured as *global*. A *module* library will be available in the current module only.

     To edit the default settings and possibly attach other `.jar` files, click the **Configure** button and re-configure the library in the **Create Library** dialog box that opens. For more details about library levels and configuring libraries, see the topics Configuring Module Dependencies and Libraries and Configuring Project and Global Libraries .
   - To have IntelliJ IDEA download the relevant files, choose the **Download** option.

     By default, when you click **Finish**, IntelliJ IDEA will download the entire Jersey reference implementation version 1.12, configure it as a project library named *JAX-RS-jersey-1.12*, and store it in the `lib` folder. To have another version downloaded or change these default settings, click the **Configure** button and set up the library as you need in the **Downloading Options** dialog box that opens:
     - Choose the relevant implementation from the **Version** drop-down list.
     - Specify the library name and level.
     - Specify the `.jar` files to download by selecting the check boxes next to the relevant files.
     - Specify the folder to the store the new library in.
   - To have the module created without the REST implementation classes, possibly to use a previously configured *global* library, choose the **Set up library later** option.
5. Specify whether you want IntelliJ IDEA to generate the code for a sample application.
   - To have the server side application code, select the **Generate server code sample** check box.
   - To have the client side application code, select the **Generate client code sample** check box.

   For more details, see the topic Coding Assistance for REST Development.

Adding a module to an existing project

1. Choose **File | New Module** on the main menu or **New | Module** on the context menu of the **Project** tool window. The New Project Wizard starts.
2. On the **Main Settings** page, choose **Java Module** in the **Java** area.
   1. In the right-hand pane, specify the module name, the location of the content root folder, of the module .iml file, and the location of the JDK to use in the module.
   2. To have a separate folder for source files created, expand the **More Settings** area, select the **Create source root** check box, and specify the location of the folder in the text box.

   Click **Next**.
3. On the Technologies page that opens, select the **RESTful WebServices** check box.
4. In the **Libraries** area, appoint the `.jar` files to use:
   - To use previously downloaded `.jar` files configured as a library, choose the **Use Library** option and select the relevant library from the drop-down list.
   - To use previously downloaded `.jar` files that have not been arranged in a library yet, choose the **Use Library** option, click the **Create** button, and then select the relevant `.jar` files.

     By default, IntelliJ IDEA assigns the library the name `JAX-RS` and configures it at the *project level*, whereupon the library can be used in the current module and in all other modules of the current project. To be re-used in any IntelliJ IDEA project, a library should be configured as *global*. A *module* library will be available in the current module only.

     To edit the default settings and possibly attach other `.jar` files, click the **Configure** button and re-configure the library in the **Create Library** dialog box that opens. For more details about library levels and configuring libraries, see the topics Configuring Module Dependencies and Libraries and Configuring Project and Global Libraries .
   - To have IntelliJ IDEA download the relevant files, choose the **Download** option.

     By default, when you click **Finish**, IntelliJ IDEA will download the entire Jersey reference implementation version 1.12, configure it as a project library named *JAX-RS-jersey-1.12*, and store it in the `lib` folder. To have another version downloaded or change these default settings, click the **Configure** button and set up the library as you need in the **Downloading Options** dialog box that opens:
     - Choose the relevant implementation from the **Version** drop-down list.
     - Specify the library name and level.
     - Specify the `.jar` files to download by selecting the check boxes next to the relevant files.
     - Specify the folder to the store the new library in.
   - To have the module created without the REST implementation classes, possibly to use a previously configured *global* library, choose the **Set up library later** option.
5. Specify whether you want IntelliJ IDEA to generate the code for a sample application.
   - To have the server side application code, select the **Generate server code sample** check box.
   - To have the client side application code, select the **Generate client code sample** check box.

   For more details, see the topic Coding Assistance for REST Development.

Attaching a RESTful WebServices facet to an existing module

1. Open the Project Structure dialog box.
2. Under **Project Settings**, select **Modules**.
3. Select the module you want to add an Android facet to, click +, and choose **RESTful WebServices**. The **Add RESTful WebServices Support** dialog box opens.
4. In the **Libraries** area, appoint the `.jar` files to use:
   - To use previously downloaded `.jar` files configured as a library, choose the **Use Library** option and select the relevant library from the drop-down list.
   - To use previously downloaded `.jar` files that have not been arranged in a library yet, choose the **Use Library** option, click the **Create** button, and then select the relevant `.jar` files.

     By default, IntelliJ IDEA assigns the library the name `JAX-RS` and configures it at the *project level*, whereupon the library can be used in the current module and in all other modules of the current project. To be re-used in any IntelliJ IDEA project, a library should be configured as *global*. A *module* library will be available in the current module only.

     To edit the default settings and possibly attach other `.jar` files, click the **Configure** button and re-configure the library in the **Create Library** dialog box that opens. For more details about library levels and configuring libraries, see the topics Configuring Module Dependencies and Libraries and Configuring Project and Global Libraries .
   - To have IntelliJ IDEA download the relevant files, choose the **Download** option.

     By default, when you click **Finish**, IntelliJ IDEA will download the entire Jersey reference implementation version 1.12, configure it as a project library named *JAX-RS-jersey-1.12*, and store it in the `lib` folder. To have another version downloaded or change these default settings, click the **Configure** button and set up the library as you need in the **Downloading Options** dialog box that opens:
     - Choose the relevant implementation from the **Version** drop-down list.
     - Specify the library name and level.
     - Specify the `.jar` files to download by selecting the check boxes next to the relevant files.
     - Specify the folder to the store the new library in.
   - To have the module created without the REST implementation classes, possibly to use a previously configured *global* library, choose the **Set up library later** option.
5. Specify whether you want IntelliJ IDEA to generate the code for a sample application.
   - To have the server side application code, select the **Generate server code sample** check box.
   - To have the client side application code, select the **Generate client code sample** check box.

   For more details, see the topic Coding Assistance for REST Development.

See Also

Concepts:

- RESTful WebServices
- Artifact

Procedures:

- Enabling Web Service Development Support

- Java EE

Reference:

- Web Services Reference
- REST Client Tool Window
- Web Services Facet Page
- Artifacts

Web Resources:

- Developer Community