Hello World RESTful web service tutorial

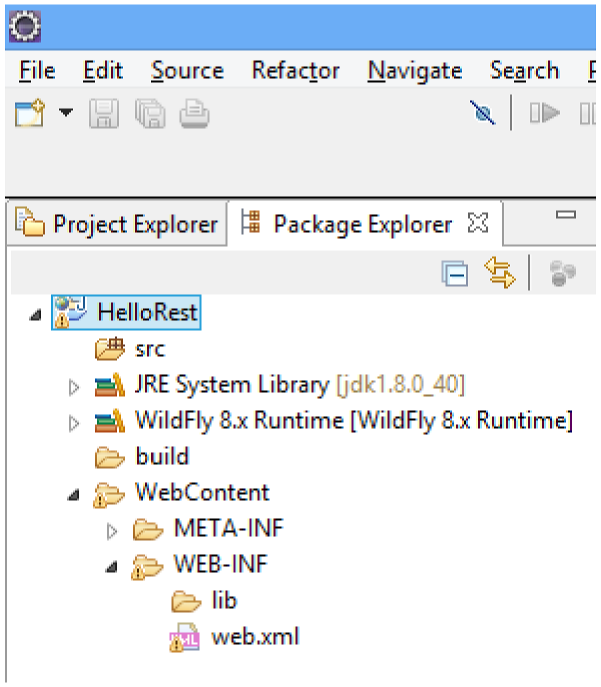
Balázs Simon (sbalazs@iit.bme.hu), BME IIT, 2015

1 Introduction

This document describes how to create a Hello World RESTful web service in Eclipse using JAX-RS and WildFly.

2 RESTful service

Create a new **Dynamic Web Project** in Eclipse called **HelloRest**. Make sure to select **Generate web.xml deployment descriptor** at the end of the wizard. The web project should look like this:

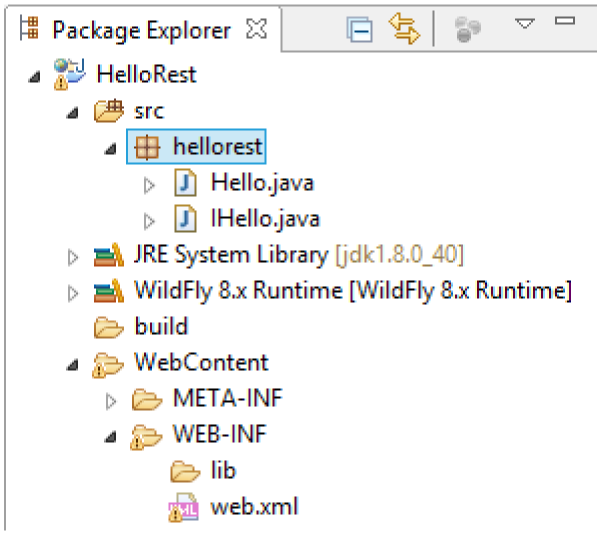


```
HelloRest
    src
    JRE System Library [jdk1.8.0_40]
    WildFly 8.x Runtime [WildFly 8.x Runtime]
    build
    WebContent
        META-INF
        WEB-INF
            lib
            web.xml
```

The JAX-RS annotated classes will be handled by a special servlet. This servlet must be declared in the **web.xml** file. Change the **web.xml** file so that it contains the following lines shown with yellow background:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="3.1"
         xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
  <display-name>HelloRest</display-name>
  <servlet-mapping>
    <servlet-name>javax.ws.rs.core.Application</servlet-name>
    <url-pattern>/rest/*</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

Create a new interface called **IHello** and a class called **Hello** in the **src** folder under the package **hellorest**:



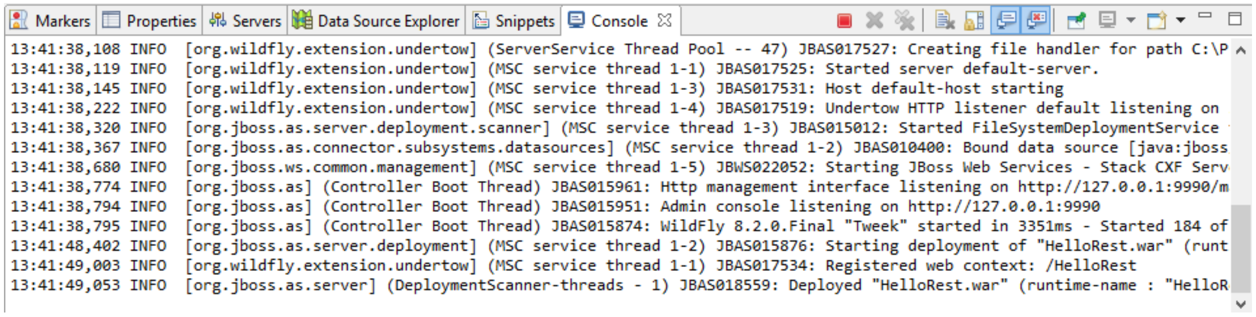
The **IHello.java** should contain the following code:

```java
package hellorest;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;

@Path("hello")
public interface IHello {
    @GET
    @Path("sayHello")
    public String sayHello(@QueryParam("name") String name);
}
```

The **Hello.java** should contain the following code:

```java
package hellorest;

public class Hello implements IHello {

    @Override
    public String sayHello(String name) {
        return "Hello: "+name;
    }

}
```

Deploy the application to the server. It should be deployed without errors:

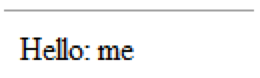


```
13:41:38,108 INFO  [org.wildfly.extension.undertow] (ServerService Thread Pool -- 47) JBAS017527: Creating file handler for path C:\P
13:41:38,119 INFO  [org.wildfly.extension.undertow] (MSC service thread 1-1) JBAS017525: Started server default-server.
13:41:38,145 INFO  [org.wildfly.extension.undertow] (MSC service thread 1-3) JBAS017531: Host default-host starting
13:41:38,222 INFO  [org.wildfly.extension.undertow] (MSC service thread 1-4) JBAS017519: Undertow HTTP listener default listening on
13:41:38,320 INFO  [org.jboss.as.server.deployment.scanner] (MSC service thread 1-3) JBAS015012: Started FileSystemDeploymentService
13:41:38,367 INFO  [org.jboss.as.connector.subsystems.datasources] (MSC service thread 1-2) JBAS010400: Bound data source [java:jboss
13:41:38,680 INFO  [org.jboss.ws.common.management] (MSC service thread 1-5) JBWS022052: Starting JBoss Web Services - Stack CXF Serv
13:41:38,774 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015961: Http management interface listening on http://127.0.0.1:9990/m
13:41:38,794 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015951: Admin console listening on http://127.0.0.1:9990
13:41:38,795 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015874: WildFly 8.2.0.Final "Tweek" started in 3351ms - Started 184 of
13:41:48,402 INFO  [org.jboss.as.server.deployment] (MSC service thread 1-2) JBAS015876: Starting deployment of "HelloRest.war" (runt
13:41:49,003 INFO  [org.wildfly.extension.undertow] (MSC service thread 1-1) JBAS017534: Registered web context: /HelloRest
13:41:49,053 INFO  [org.jboss.as.server] (DeploymentScanner-threads - 1) JBAS018559: Deployed "HelloRest.war" (runtime-name : "HelloR
```

To test the service, type the following URL into a browser (FireFox, Chrome, IE, etc.):

`http://localhost:8080/HelloRest/rest/hello/sayHello?name=me`

The response should be:



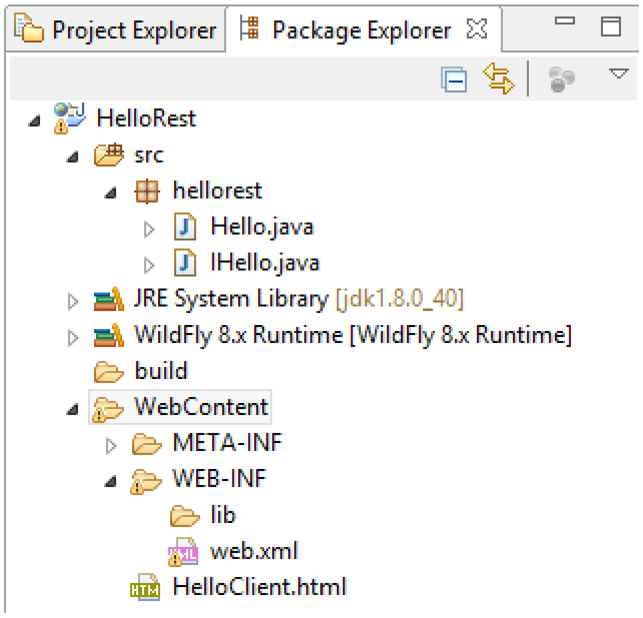
Hello: me

The URL has the following structure:

- **http://localhost:8080** – the protocol, domain name and port number of the server
- **/HelloRest** – the name of the web application
- **/rest** – the mapping of the REST servlet in the **web.xml** file
- **/hello** – the **@Path** mapping of the **IHello** interface
- **/sayHello** – the **@Path** mapping of the **sayHello** operation
- **?name=me** – the name and value passed as a **@QueryParam**

3 HTML client

Create an HTML file called **HelloClient.html** under the **WebContent** folder:



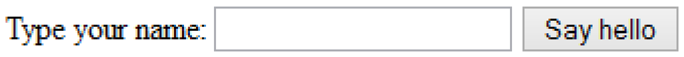
The contents of the file should be the following:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="ISO-8859-1">
    <title>Hello REST client</title>
</head>
<body>
    <form method="get" action="rest/hello/sayHello">
        Type your name: <input type="text" name="name"/>
        <input type="submit" value="Say hello"/>
    </form>
</body>
</html>
```

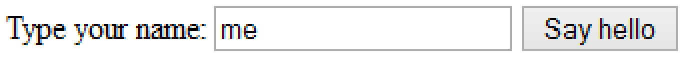
Redeploy the application to the server, and open the following URL in a browser:

```
http://localhost:8080/HelloRest/HelloClient.html
```

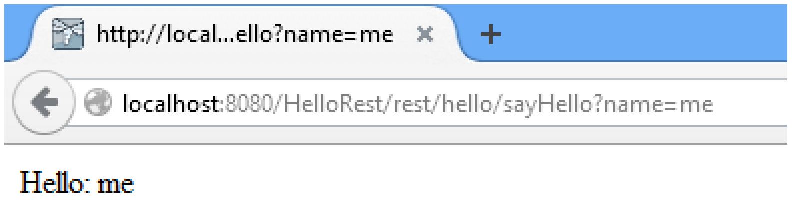
This is where the HTML is accessible. The following page should be loaded in the browser:



Type something in the text field and click the **Say hello** button:



The result should be the following:


http://local...ello?name=me
localhost:8080/HelloRest/rest/hello/sayHello?name=me

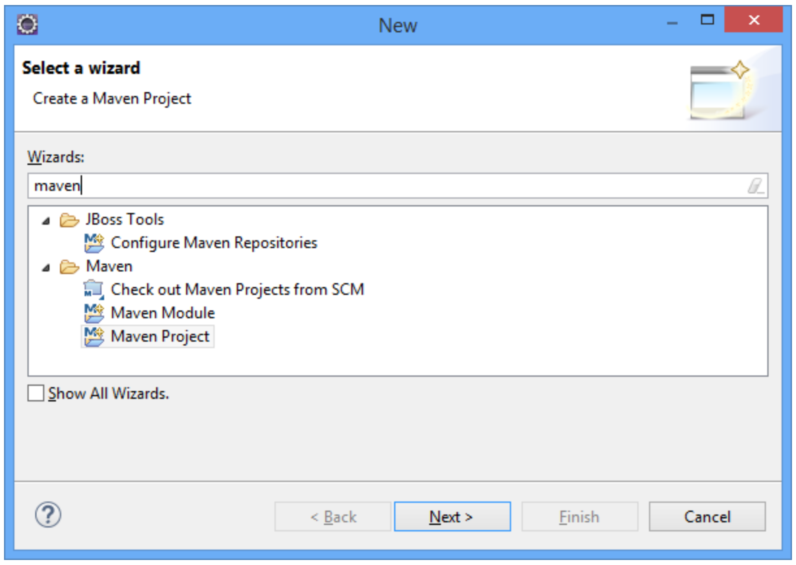
Hello: me

4 Java client using RESTeasy

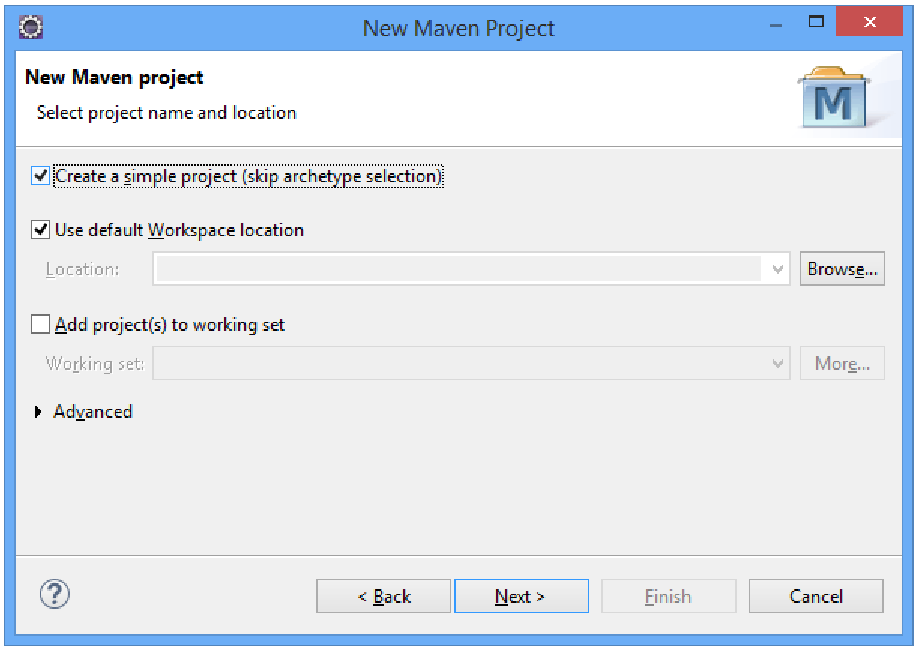
RESTeasy is the JAX-RS implementation of the WildFly server. It also has a client library, which can be used in console applications. This example will show how.

RESTeasy requires some additional dependencies, so the easiest way to create a client is to use Maven.

Create a new **Maven Project** in Eclipse called **HelloRestClient**:



Click **Next**, and select the **Create a simple project** check-box:


New Maven Project

New Maven project

Select project name and location

☑ Create a simple project (skip archetype selection)

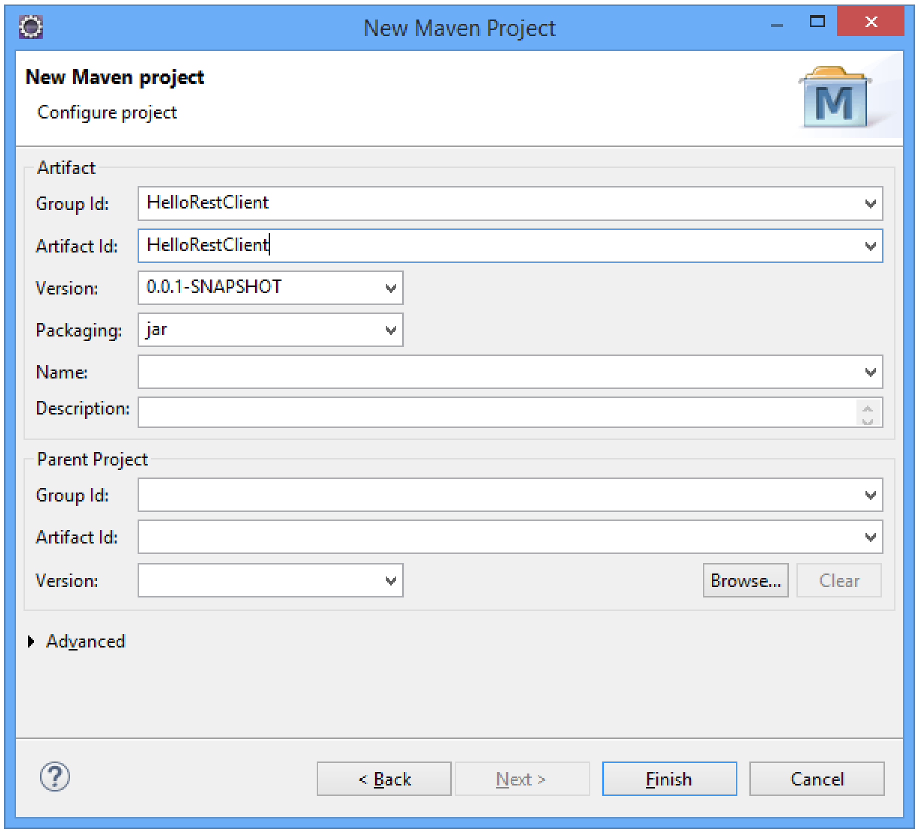
☑ Use default Workspace location

Location:

☐ Add project(s) to working set

Working set:

Click **Next** and type **HelloRestClient** for both **Group Id** and **Artifact Id**:


New Maven Project

New Maven project

Configure project

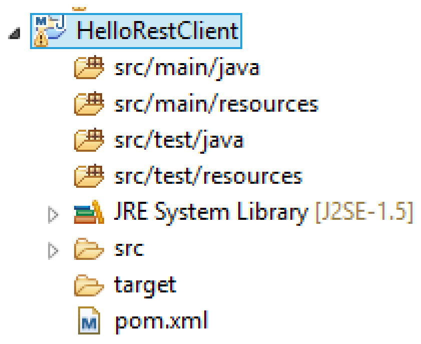
Artifact

Group Id:	HelloRestClient
Artifact Id:	HelloRestClient
Version:	0.0.1-SNAPSHOT
Packaging:	jar
Name:	
Description:	

Parent Project

Group Id:	
Artifact Id:	
Version:	

Click **Finish**, and the project should look like this:


HelloRestClient
src/main/java
src/main/resources
src/test/java
src/test/resources
JRE System Library [J2SE-1.5]
src
target
pom.xml

Edit the **pom.xml** and add the following lines shown with yellow background:

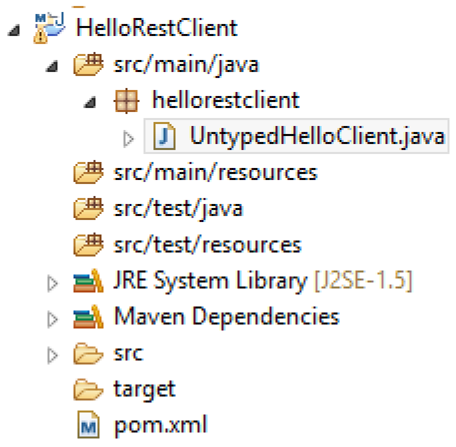
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>HelloRestClient</groupId>
  <artifactId>HelloRestClient</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
    <dependency>
      <groupId>org.jboss.resteasy</groupId>
      <artifactId>resteasy-client</artifactId>
      <version>3.0.10.Final</version>
    </dependency>
  </dependencies>
</project>
```

4.1 Untyped client

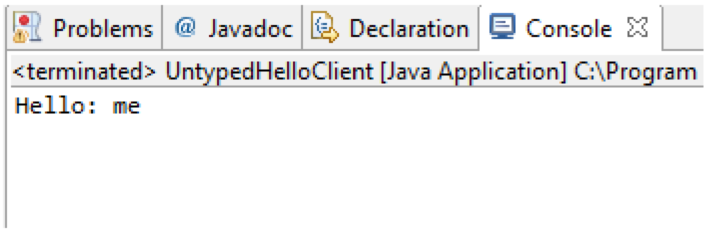
Create a new class called **UntypedHelloClient** under the package **hellorestclient** with the following content:

```
package hellorestclient;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.Response;

public class UntypedHelloClient {

    public static void main(String[] args) {
        try{
            // Create a new RESTeasy client through the JAX-RS API:
            Client client = ClientBuilder.newClient();
            // The base URL of the service:
            WebTarget target = client.target("http://localhost:8080/HelloRest/rest");
            // Building the relative URL manually for the sayHello method:
            WebTarget hello =
                target.path("hello").path("sayHello").queryParam("name", "me");
            // Get the response from the target URL:
            Response response = hello.request().get();
            // Read the result as a String:
            String result = response.readEntity(String.class);
            // Print the result to the standard output:
            System.out.println(result);
            // Close the connection:
            response.close();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

}
```

The project should look like this:



Right click on the **UntypedHelloClient.java** and select **Run As > Java Application**. It should print the following:



```
<terminated> UntypedHelloClient [Java Application] C:\Program
Hello: me
```

4.2 Typed client

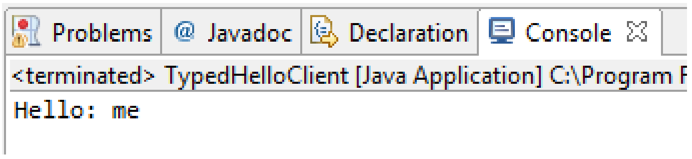
Create an interface called **IHello** under the package **hellorestclient** with the following content:

```
package hellorestclient;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;

@Path("hello")
public interface IHello {
    @GET
    @Path("sayHello")
    public String sayHello(@QueryParam("name") String name);
}
```

Create a class called **TypedHelloClient** under the package **hellorestclient** with the following content:

```
package hellorestclient;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.WebTarget;
import org.jboss.resteasy.client.jaxrs.ResteasyWebTarget;

public class TypedHelloClient {
    public static void main(String[] args) {
        try {
            // Create a new RESTeasy client through the JAX-RS API:
            Client client = ClientBuilder.newClient();
            // The base URL of the service:
            WebTarget target = client.target("http://localhost:8080/HelloRest/rest");
            // Cast it to ResteasyWebTarget:
            ResteasyWebTarget rtarget = (ResteasyWebTarget)target;
            // Get a typed interface:
            IHello hello = rtarget.proxy(IHello.class);
            // Call the service as a normal Java object:
            String result = hello.sayHello("me");
            // Print the result:
            System.out.println(result);
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

Right click on the **TypedHelloClient.java** and select **Run As > Java Application**. It should print the following:



```
<terminated> TypedHelloClient [Java Application] C:\Program F
Hello: me
```