Exercise 3

Create an embedded Java REST Service using JAX-RS and Spring Boot

Prior Knowledge
Basic understanding HTTP verbs, REST architecture
Some Java coding skill

Objectives
Understand what it takes to create REST services. Interact with a REST service using simple web clients in Chrome, on the command line.
See how Gradle can be used.

Software Requirements
(see separate document for installation of these)

- Java Development Kit 8
- Gradle build system
- Spring Boot and Jersey
- Visual Studio Code, including extensions:
    - Java
    - Spring Boot
    - SpringInitializr
    - Gradle
- curl
- Google Chrome/Chromium plus Chrome Advanced REST extension

Overview
There are many technologies for creating RESTful Web Services in Java. In order to create a simple approach, we are going to use the Java standard for creating REST services, which is called JAX-RS. The “official” Oracle implementation of JAX-RS is Jersey, although there are other implementations such as CXF and RESTEasy.

Spring Boot is a framework for running microservices.

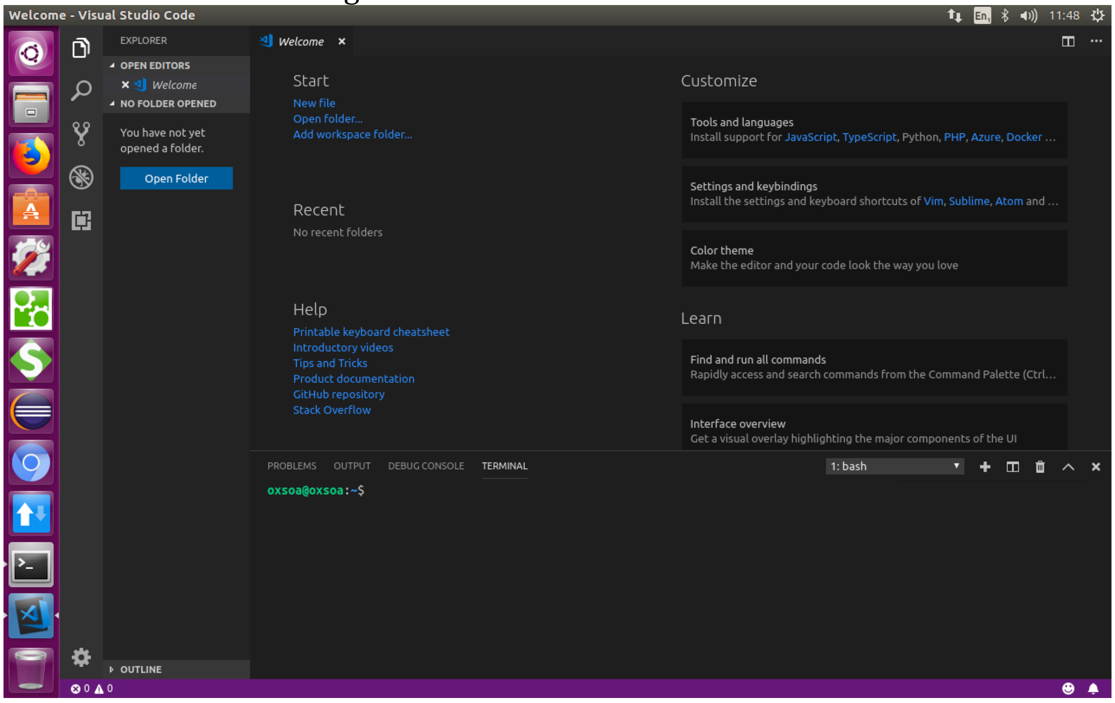
© Paul Fremantle 2016. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

Steps

1. Create a new directory:
```
mkdir ~/ex3
cd ~/ex3
```

2. Start Visual Studio Code:
```
code
```

You should see something like:



You have not yet opened a folder.

No recent folders

Install support for JavaScript, TypeScript, Python, PHP, Azure, Docker ...

Install the settings and keyboard shortcuts of Vim, Sublime, Atom and ...

Make the editor and your code look the way you love

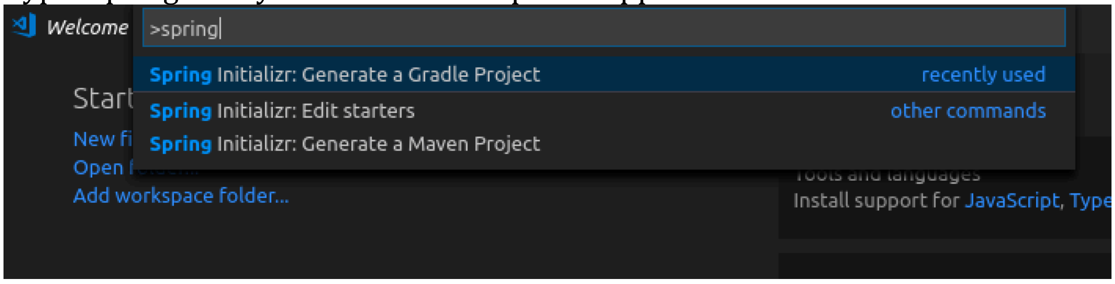
Rapidly access and search commands from the Command Palette (Ctrl...

Get a visual overlay highlighting the major components of the UI

```
oxsoa@oxsoa:~$
```

3. Show the vscode "Command Palette" (Ctrl-Shift-P)

4. Type "spring" and you should some options appear:



```
>spring
Spring Initializr: Generate a Gradle Project        recently used
Spring Initializr: Edit starters                    other commands
Spring Initializr: Generate a Maven Project
```

5. Select **Spring Initalizr: Generate a Gradle Project**

© Paul Fremantle 2016. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

6. Select **Java**

7. Type **me.freo** for the Group

8. Type **hello** for the artifact id

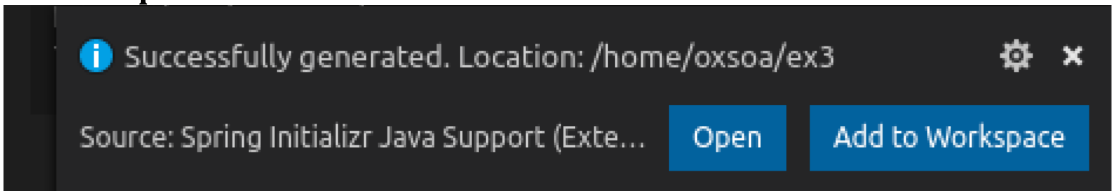
9. Choose **2.1.1** for the Spring Boot version

10. Type **Jersey** and then select Jersey (JAX-RS) for the dependency.

11. Then hit **Enter** to continue.

12. The window should be pointing to your ex3 directory and then click **Generate into this Folder.** If clicking doesn't work, try hitting **Enter**

13. You should see a notification that the code has been generated. Click **Add to Workspace:**



Successfully generated. Location: /home/oxsoa/ex3

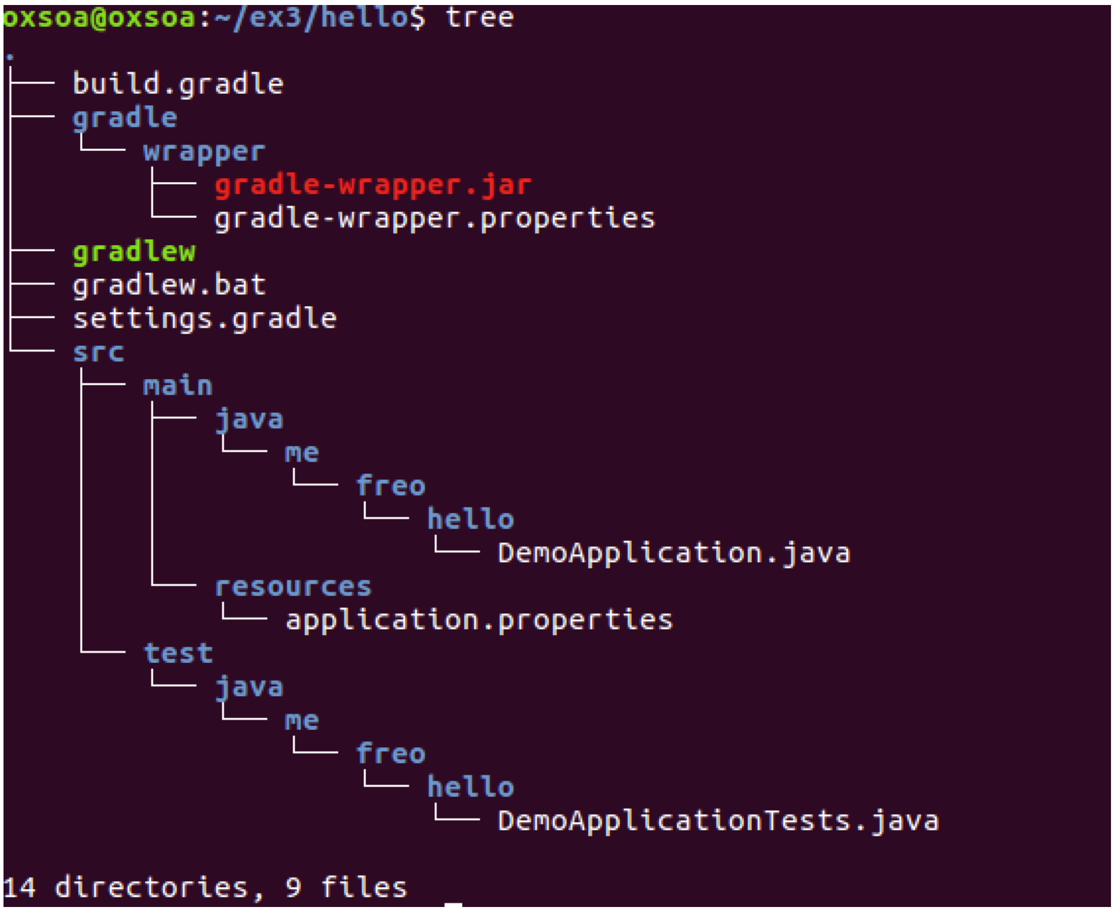
Source: Spring Initializr Java Support (Exte...

14. On a command line:

```
cd ~/ex3/hello
tree
```

© Paul Fremantle 2016. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

You should see:

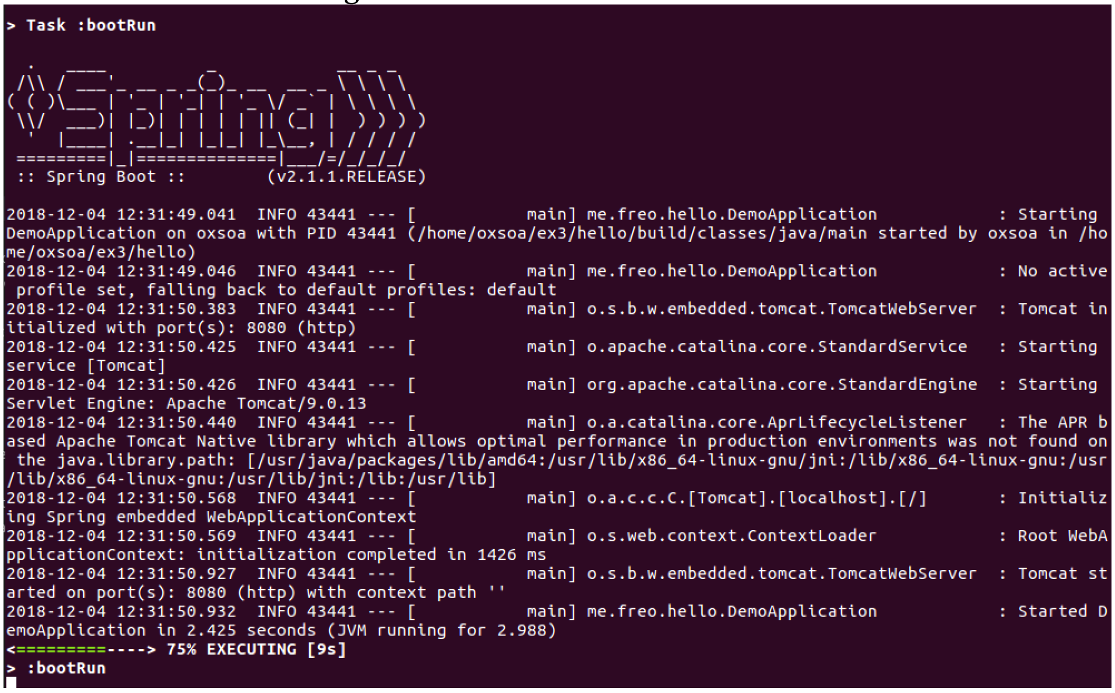


```
oxsoa@oxsoa:~/ex3/hello$ tree
.
├── build.gradle
├── gradle
│   └── wrapper
│       ├── gradle-wrapper.jar
│       └── gradle-wrapper.properties
├── gradlew
├── gradlew.bat
├── settings.gradle
└── src
    ├── main
    │   ├── java
    │   │   └── me
    │   │       └── freo
    │   │           └── hello
    │   │               └── DemoApplication.java
    │   └── resources
    │       └── application.properties
    └── test
        └── java
            └── me
                └── freo
                    └── hello
                        └── DemoApplicationTests.java

14 directories, 9 files
```

15. In the same place, type:
```
gradle build
```

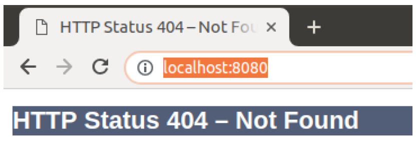
© Paul Fremantle 2016. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

16. This will build your Spring Boot application. However, so far there is no logic, so it will just start a web server that doesn't do anything useful. Try it:
    `gradle bootRun`
    You should see something like:



```
> Task :bootRun

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.1.1.RELEASE)

2018-12-04 12:31:49.041  INFO 43441 --- [           main] me.freo.hello.DemoApplication            : Starting
DemoApplication on oxsoa with PID 43441 (/home/oxsoa/ex3/hello/build/classes/java/main started by oxsoa in /ho
me/oxsoa/ex3/hello)
2018-12-04 12:31:49.046  INFO 43441 --- [           main] me.freo.hello.DemoApplication            : No active
 profile set, falling back to default profiles: default
2018-12-04 12:31:50.383  INFO 43441 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat in
itialized with port(s): 8080 (http)
2018-12-04 12:31:50.425  INFO 43441 --- [           main] o.apache.catalina.core.StandardService   : Starting
service [Tomcat]
2018-12-04 12:31:50.426  INFO 43441 --- [           main] org.apache.catalina.core.StandardEngine  : Starting
Servlet Engine: Apache Tomcat/9.0.13
2018-12-04 12:31:50.440  INFO 43441 --- [           main] o.a.catalina.core.AprLifecycleListener   : The APR b
ased Apache Tomcat Native library which allows optimal performance in production environments was not found on
 the java.library.path: [/usr/java/packages/lib/amd64:/usr/lib/x86_64-linux-gnu/jni:/lib/x86_64-linux-gnu:/usr
/lib/x86_64-linux-gnu:/usr/lib/jni:/lib:/usr/lib]
2018-12-04 12:31:50.568  INFO 43441 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializ
ing Spring embedded WebApplicationContext
2018-12-04 12:31:50.569  INFO 43441 --- [           main] o.s.web.context.ContextLoader            : Root WebA
pplicationContext: initialization completed in 1426 ms
2018-12-04 12:31:50.927  INFO 43441 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat st
arted on port(s): 8080 (http) with context path ''
2018-12-04 12:31:50.932  INFO 43441 --- [           main] me.freo.hello.DemoApplication            : Started D
emoApplication in 2.425 seconds (JVM running for 2.988)
<=========----> 75% EXECUTING [9s]
> :bootRun
```

17. You can see a server is running on port 8080, but there are no resources deployed:



```
HTTP Status 404 – Not Fo  ×  +
localhost:8080

HTTP Status 404 – Not Found
```

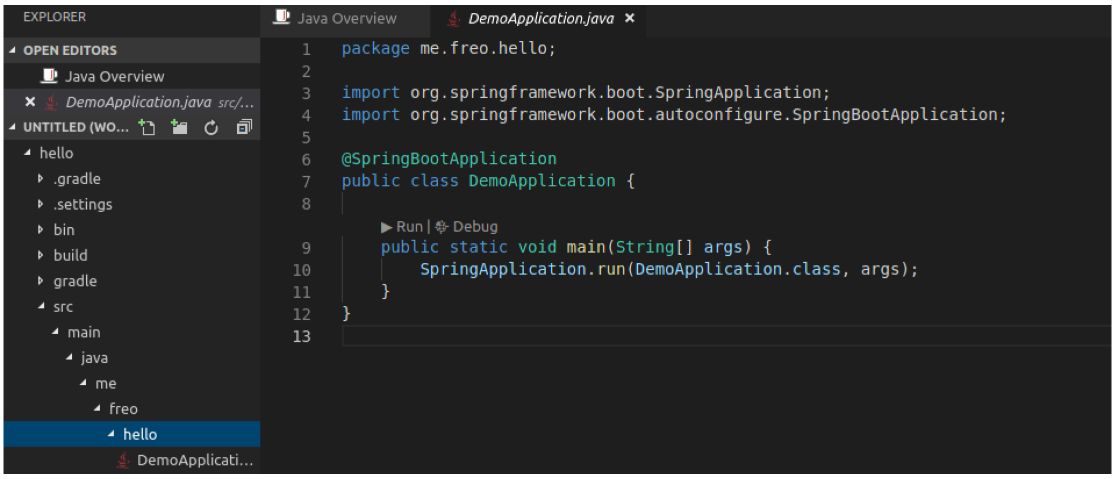
18. Stop the server / gradle run by hitting Ctrl-C

19. In vscode, Open the Folder (Ctrl-K Ctrl-O, or dropdown menu)
    `~/ex3/hello`

© Paul Fremantle 2016. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

20. You can open the src/main/java/me/freo/hello/DemoApplication.java file.

    You should see:



```
package me.freo.hello;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```

21. In the same Package (me.freo.hello), create a new Java file called **Resource.java**

22. Type the following Java. Hint if you leave out the imports, vscode will add them automatically as you do the rest.

```
package me.freo.hello;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import org.springframework.stereotype.Component;

@Component
@Path("/hello")
public class Resource {
    @GET
    @Path("/")
    public String sayHello() {
        return "hello world";
    }
}
```

    This file is our actual "Restful Service". It contains the logic to respond to HTTP requests based on the verb/path/content-type, etc

23. We need one more Java file, called **HelloConfiguration.java** (also in me/freo/hello.

© Paul Fremantle 2016. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

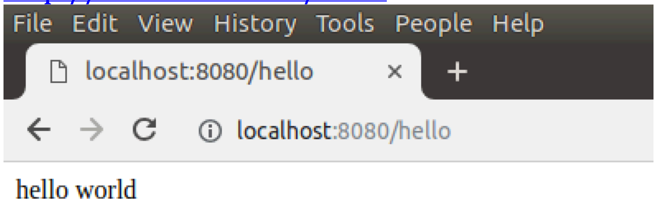
24. The contents are here. Once again, vscode can help you do all this. This is basically telling Spring Boot about your Resource class.

```
package me.freo.hello;

import javax.annotation.PostConstruct;
import javax.ws.rs.ApplicationPath;
import org.glassfish.jersey.server.ResourceConfig;
import org.springframework.context.annotation.Configuration;

@Configuration
@ApplicationPath("/")
public class HelloConfiguration extends ResourceConfig {
	public HelloConfiguration() {
	}

	@PostConstruct
	public void setUp() {
		register(Resource.class);
	}
}
```

25. Rebuild your gradle project and re-run:

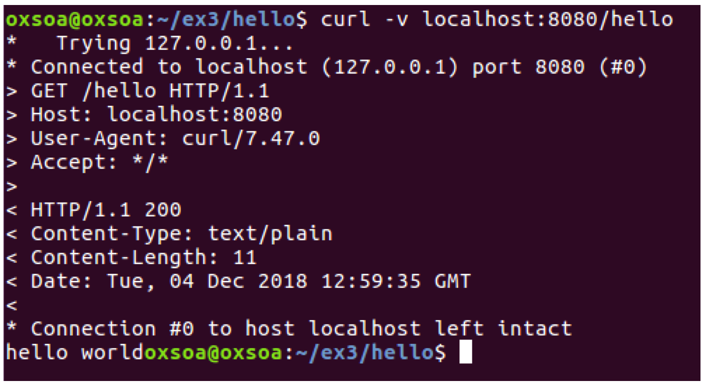
```
gradle build
gradle bootRun
```

26. Now we can see that this has created a better server. Browse to http://localhost:8080/hello



hello world

27. Try "curl -v":



```
oxsoa@oxsoa:~/ex3/hello$ curl -v localhost:8080/hello
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /hello HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.47.0
> Accept: */*
>
< HTTP/1.1 200
< Content-Type: text/plain
< Content-Length: 11
< Date: Tue, 04 Dec 2018 12:59:35 GMT
<
* Connection #0 to host localhost left intact
hello worldoxsoa@oxsoa:~/ex3/hello$ 
```

© Paul Fremantle 2016. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

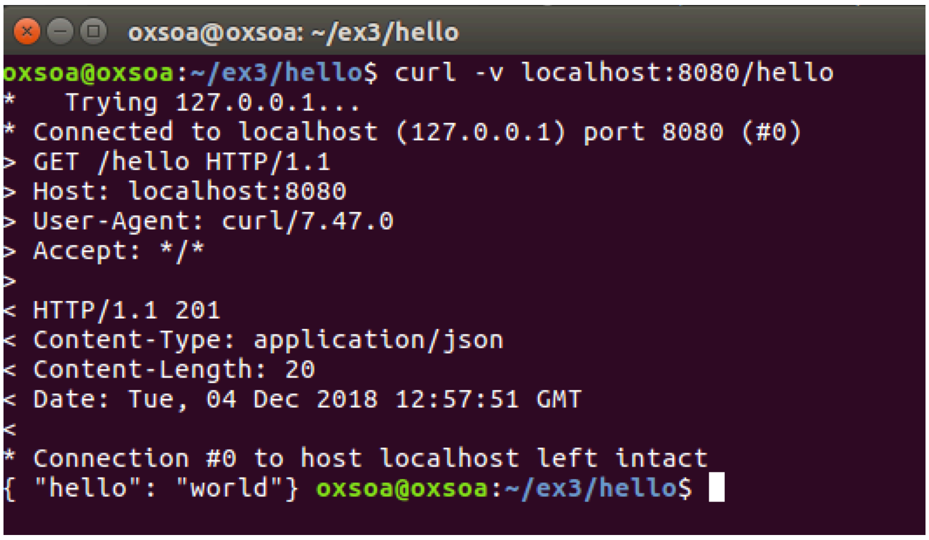
28. We are nearly done, but let's do one final improvement to the Resource. Change Resource.java so that it looks like this:

```
package me.freo.hello;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

import org.springframework.stereotype.Component;

@Component
@Path("/hello")
@Produces(MediaType.APPLICATION_JSON)
public class Resource {
    @GET
    @Path("/")
    public Response sayHello() {
        return Response.status(201).entity("{ \"hello\": \"world\"} ").build();
    }
}
```

What we have done is to take full control of the HTTP response. Instead of just returning a string, we are 'hand-crafting' the Response object. We have also explicitly told JAX-RS to return an application/json media type.

29. Rebuild and rerun.

30. Now "curl -v" this resource. You should see:



```
oxsoa@oxsoa:~/ex3/hello$ curl -v localhost:8080/hello
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /hello HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.47.0
> Accept: */*
>
< HTTP/1.1 201
< Content-Type: application/json
< Content-Length: 20
< Date: Tue, 04 Dec 2018 12:57:51 GMT
<
* Connection #0 to host localhost left intact
{ "hello": "world"} oxsoa@oxsoa:~/ex3/hello$
```

31. Notice the different return code, and the Content-Type. And of course the fact it is now JSON.

© Paul Fremantle 2016. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

32. One last thing. So far we have only run this using gradle. That is really a "dev" time thing. What Spring Boot has done is to package this all up into a single executable JAR file. From the ~/ex3/hello directory type:

    ```
    java –jar build/libs/hello-0.0.1-SNAPSHOT.jar
    ```

    Once again you should see the Spring logo etc and your service has started.

33. That's all for now. We have created a simple Spring Boot project, including JAX-RS, and used gradle to build into a standalone JAR file.

© Paul Fremantle 2016. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/