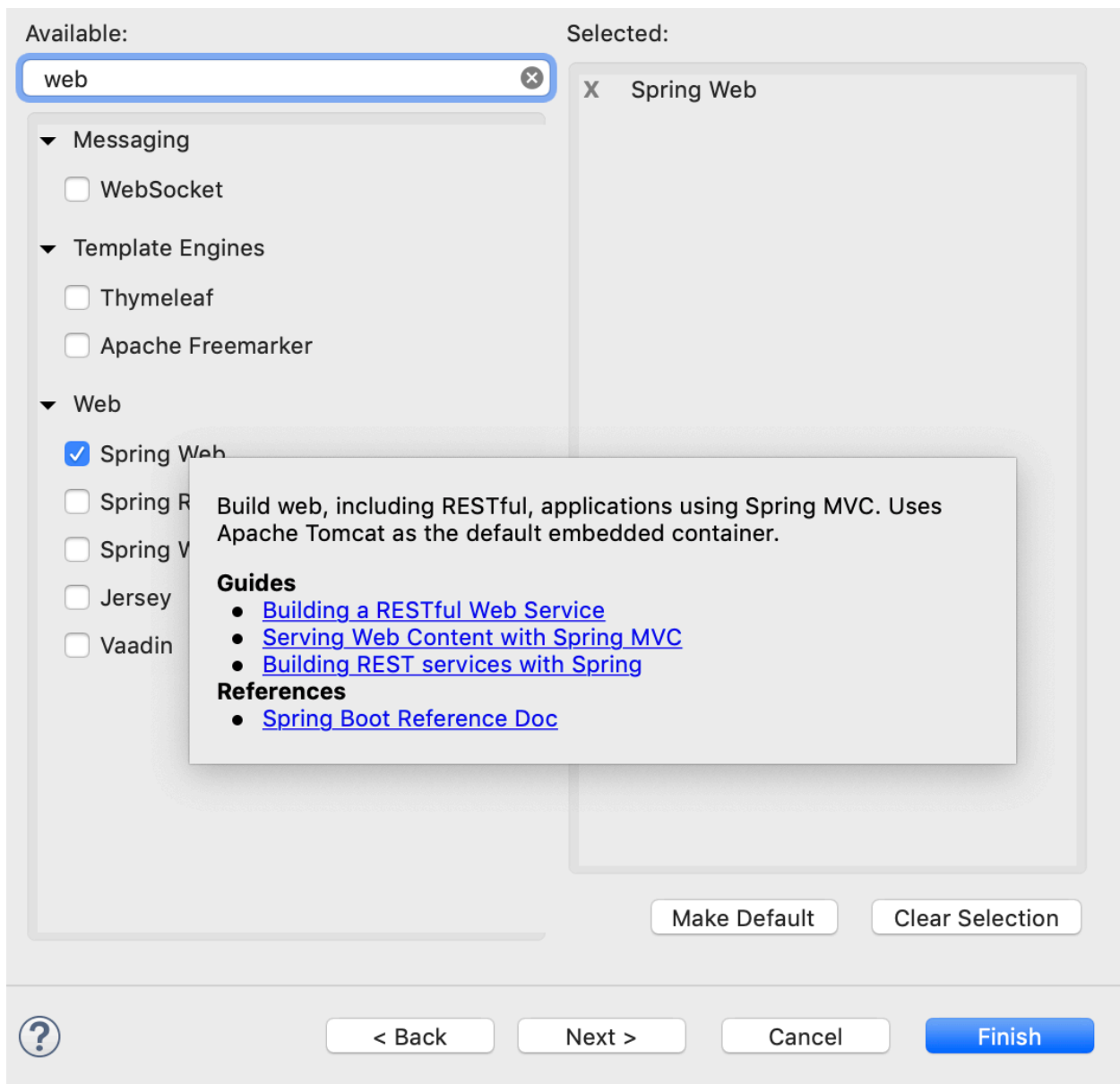


Create a REST API using Spring Boot for Hello World

1. File -> New -> Spring Starter Project
2. Name = helloworld-springboot
3. Group = com.example (default value)
4. Package = com.example.demo (default value)
5. Click on Next
6. Select "Spring Web" module



7. Click on Finish
8. Open pom.xml and add these dependencies for Dev Tools, Actuator,

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <optional>true</optional>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-rest-hal-explorer</artifactId>
</dependency>
```

HAL Explorer

9. Create a REST Controller class
- 9.1 Right click on src/main/java
- 9.2 New → Class
- 9.3 Class name = BasicController
- 9.4 Package = com.example.demo.controller

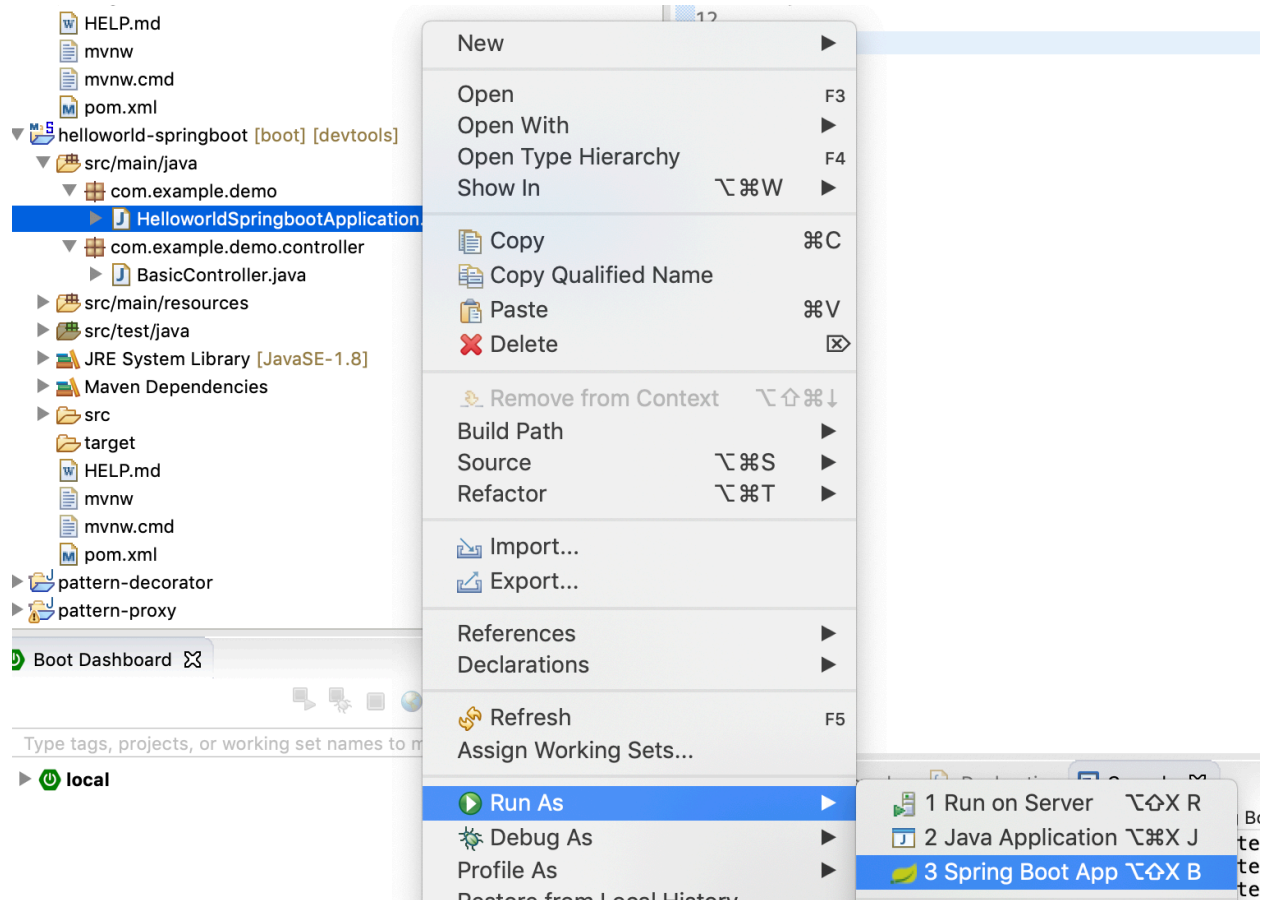
```
@RestController
public class BasicController {

    @GetMapping("/welcome")
    public String welcome() {
        return "Hello World!!!";
    }

}
```

9.5 Click on Finish

10. RUN the App



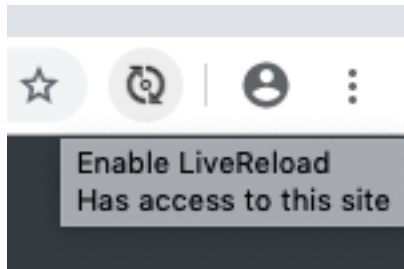
11. Observe logs on Startup

```

  ____  _
 / ___|| | | |
| |___| |_| |
 \___|____|_|_|_|
:: Spring Boot :: (v2.2.0.RELEASE)

2019-10-19 19:01:43.105 INFO 711 --- [ restartedMain] c.e.d.HelloworldSpringbootApplication : Starting HelloworldSpringbootApplication on Nandakumars-MBP, hc
2019-10-19 19:01:43.106 INFO 711 --- [ restartedMain] c.e.d.HelloworldSpringbootApplication : No active profile set, falling back to default profiles: default
2019-10-19 19:01:43.137 INFO 711 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-pr
2019-10-19 19:01:43.887 INFO 711 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2019-10-19 19:01:43.891 INFO 711 --- [ restartedMain] org.apache.catalina.core.StandardService : Starting service [Tomcat]
2019-10-19 19:01:43.891 INFO 711 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.27]
2019-10-19 19:01:43.921 INFO 711 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2019-10-19 19:01:43.922 INFO 711 --- [ restartedMain] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 785 ms
2019-10-19 19:01:44.405 INFO 711 --- [ restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2019-10-19 19:01:44.708 INFO 711 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2019-10-19 19:01:44.712 INFO 711 --- [ restartedMain] o.s.b.a.e.web.EndpointLinksResolver : Exposing 2 endpoint(s) beneath base path '/actuator'
2019-10-19 19:01:44.760 INFO 711 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2019-10-19 19:01:44.763 INFO 711 --- [ restartedMain] c.e.d.HelloworldSpringbootApplication : Started HelloworldSpringbootApplication in 1.835 seconds (JVM
2019-10-19 19:02:01.115 INFO 711 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2019-10-19 19:02:01.115 INFO 711 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2019-10-19 19:02:01.125 INFO 711 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 10 ms
```

12. Enable Live Reload



12.1 <http://localhost:8080/welcome>

13. Make changes to Controller class & Save

14. App restarts automatically as we added devtools

15. Browser automatically refreshes as “Live Reload” is enabled

16. Visit <http://localhost:8080> to navigate through HAL Browser

A screenshot of the HAL Explorer web application running in a browser. The browser's address bar shows the URL "localhost:8080/explorer/index.html#uri=/". The application has a dark theme and a navigation bar with "Theme", "Layout", and "About" options. Below the navigation bar, there is a "Go!" button and a "Response Status" section showing "200 (OK)". The "Response Headers" section lists headers: "date" (Sat, 19 Oct 2019 23:10:34 GMT), "vary" (Origin, Access-Control-Request-Method, Access-Control-Request-Headers), "transfer-encoding" (chunked), and "content-type" (application/hal+json). The "Response Body" section displays a JSON object:

```
{  "_links": {    "profile": {      "href": "http://localhost:8080/profile"    }  }}
```

 On the left side, there is a "Links" section with a table containing one row:

Relation	Name	Title	HTTP	Doc
profile			← → ↶ ↷ ✖	

17. Visit Actuator API to observe metrics

← → ↻ localhost:8080/explorer/index.html#uri=/actuator

HAL Explorer Theme Layout About

Edit Headers /actuator Go!

Links

Relation	Name	Title	HTTP	Doc
self			< + > > ✖	
health			< + > > ✖	
health-path			< + > > ✖	
info			< + > > ✖	

Response Status

200 (OK)

Response Headers

date	Sat, 19 Oct 2019 23:36:05 GMT
transfer-encoding	chunked
content-type	application/json

Response Body

```
{
  "_links": {
    "self": {
      "href": "http://localhost:8080/actuator",
      "templated": false
    },
    "health": {
      "href": "http://localhost:8080/actuator/health",
      "templated": false
    },
    "health-path": {
      "href": "http://localhost:8080/actuator/health/{*path}",
      "templated": true
    },
    "info": {
      "href": "http://localhost:8080/actuator/info",
      "templated": false
    }
  }
}
```

Extend Hello World REST API with JSON Response

18. Create a POJO to hold the response structure

We will create a **WelcomeBean.java** with a property to hold the message along with argument constructor & a getter method

```
package com.example.demo.model;

public class WelcomeBean {

    private String message;

    public WelcomeBean(String message) {
        super();
        this.message = message;
    }

    public String getMessage() {
        return message;
    }

}
```

19. Create an API Resource method in the BasicController.java to create and return an object of WelcomeBean

```
@GetMapping("/welcome-with-object")
public WelcomeBean welcomeWithObject() {

    return new WelcomeBean("Hello World");

}
```


20. If the application was already running, then it should restart automatically.

Otherwise,

HelloWorldSpringBootApplication ->

Right click -> Run As -> Spring Boot App

21. Visit: **<http://localhost:8080/welcome-with-object>**



← → ↻ ⓘ localhost:8080/welcome-with-object

```
{"message": "Hello World"}
```

Extend JSON Response with a name path variable

22. Let's consider an example

/welcome-with-parameter/name/Nanda

In this URI, **Nanda** is a value. I want to be able to map **Nanda** to a variable in the request method.

```
@GetMapping("/welcome-with-parameter/name/{name}")
public WelcomeBean welcomeWithParameter(@PathVariable String name) {
    return new WelcomeBean(String.format("Hello World, %s!", name));
}
```

23. If the application was already running, then it should restart automatically.

Otherwise,

HelloWorldSpringBootApplication ->
Right click -> Run As -> Spring Boot App

24. Visit: */welcome-with-parameter/name/Nanda*

← → ↻ ⓘ localhost:8080/welcome-with-parameter/name/Nanda

```
{"message": "Hello World, Nanda!"}
```