BridgeLabz

Employability Delivered

Spring App Development

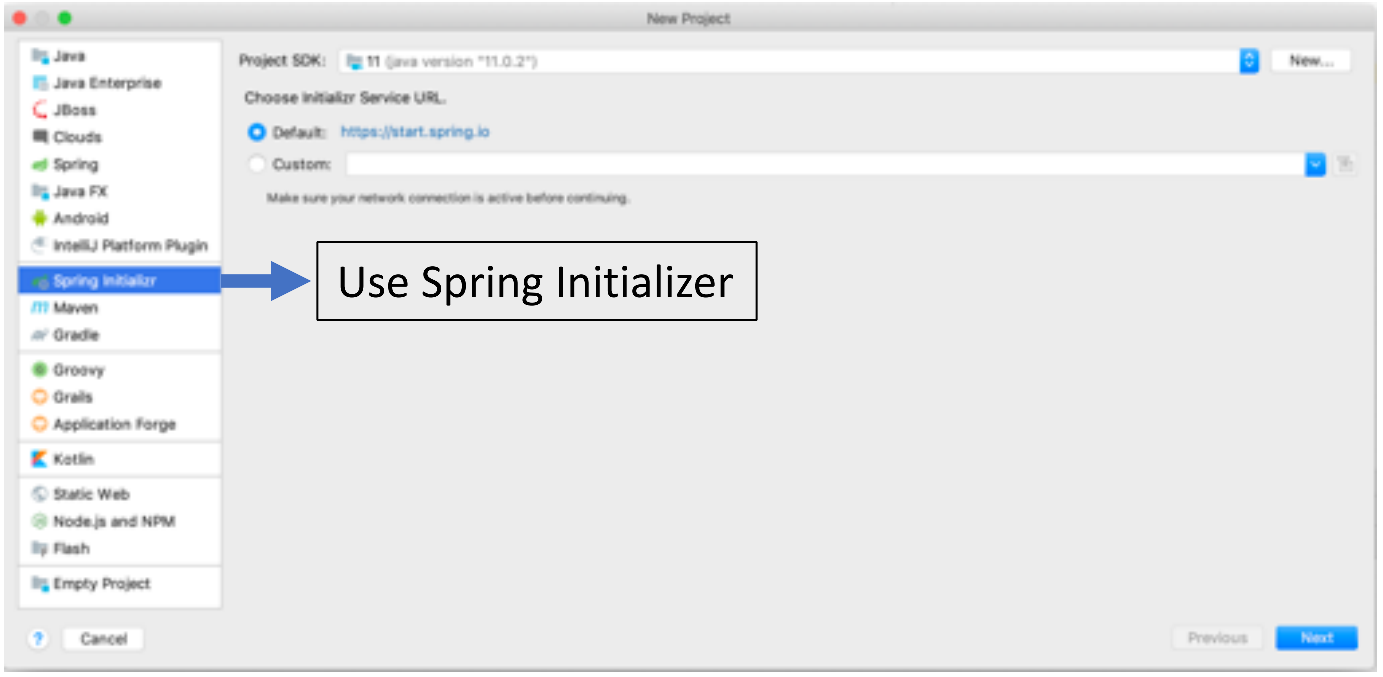
Agenda

BridgeLabz
Employability Delivered

- Firstly we will start quickly with developing a simple Spring App. In this app we will do the following. User curl for all
  - Say simple Hello World
  - Say Hello with Name as Query Parameter
  - Say Hello with Name in Path Variable
  - Say Hello with Name in the Body
- Develop EmployeeApp where we use Hibernate and H2 Database

Create First Spring App

- **Step 1:** Create a New Project



Project SDK: 11 (java version "11.0.2")

Choose Initializr Service URL.

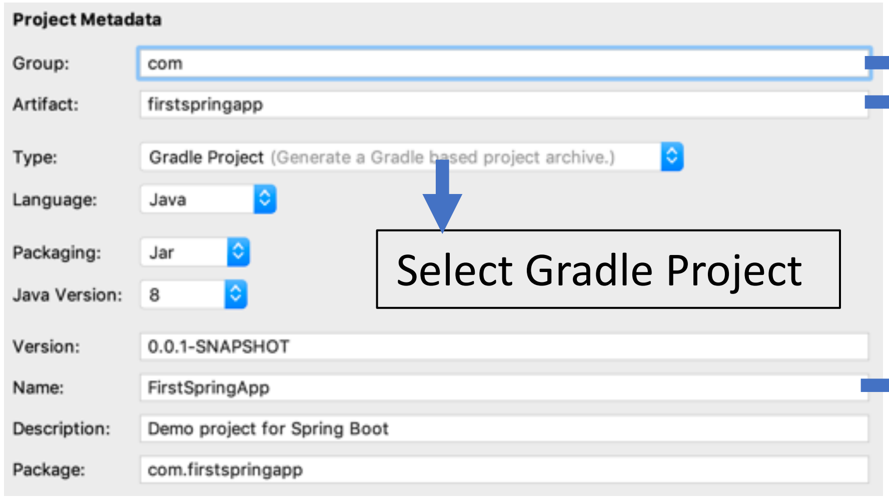
Default: https://start.spring.io

Custom:

Make sure your network connection is active before continuing.

Use Spring Initializer

- **Step 2:** Enter Project Data – Group and Artifact



Project Metadata

Group:	com
Artifact:	firstspringapp
Type:	Gradle Project (Generate a Gradle based project archive.)
Language:	Java
Packaging:	Jar
Java Version:	8
Version:	0.0.1-SNAPSHOT
Name:	FirstSpringApp
Description:	Demo project for Spring Boot
Package:	com.firstspringapp

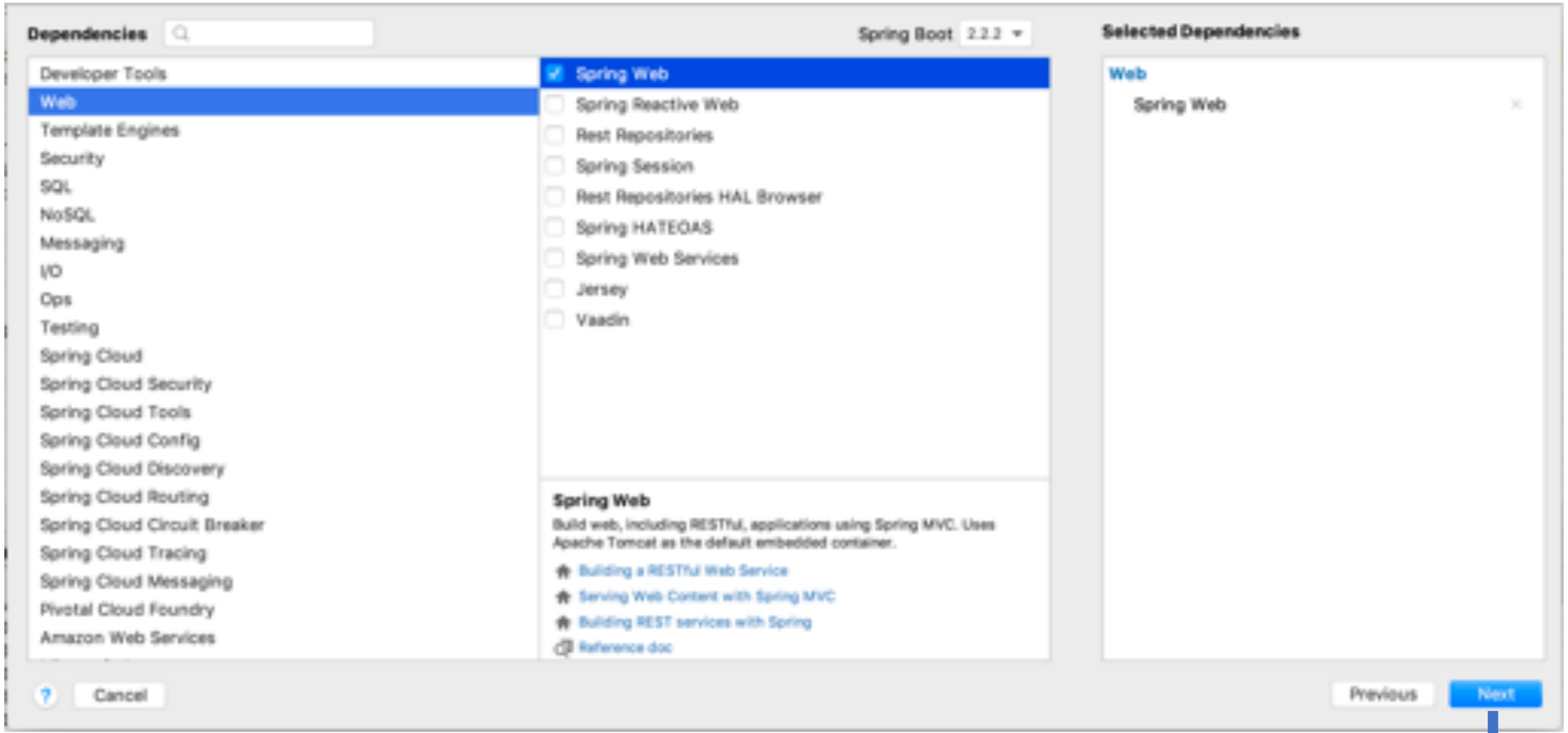
Enter Package

Enter Artifact

Select Gradle Project

Enter App Name

Step 3: Select Spring Dependencies


Dependencies

Spring Boot 2.2.2

Developer Tools
Web
Template Engines
Security
SQL
NoSQL
Messaging
I/O
Ops
Testing
Spring Cloud
Spring Cloud Security
Spring Cloud Tools
Spring Cloud Config
Spring Cloud Discovery
Spring Cloud Routing
Spring Cloud Circuit Breaker
Spring Cloud Tracing
Spring Cloud Messaging
Pivotal Cloud Foundry
Amazon Web Services

- [x] Spring Web
- [ ] Spring Reactive Web
- [ ] Rest Repositories
- [ ] Spring Session
- [ ] Rest Repositories HAL Browser
- [ ] Spring HATEOAS
- [ ] Spring Web Services
- [ ] Jersey
- [ ] Vaadin

Spring Web
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

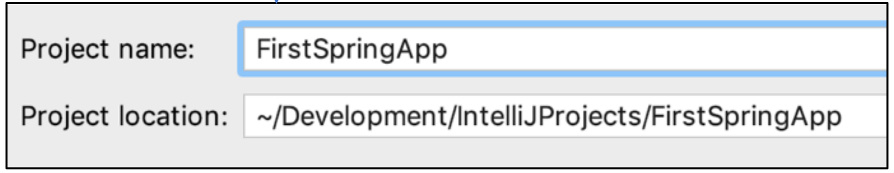
Building a RESTful Web Service
Serving Web Content with Spring MVC
Building REST services with Spring
Reference doc

Selected Dependencies

Web
Spring Web

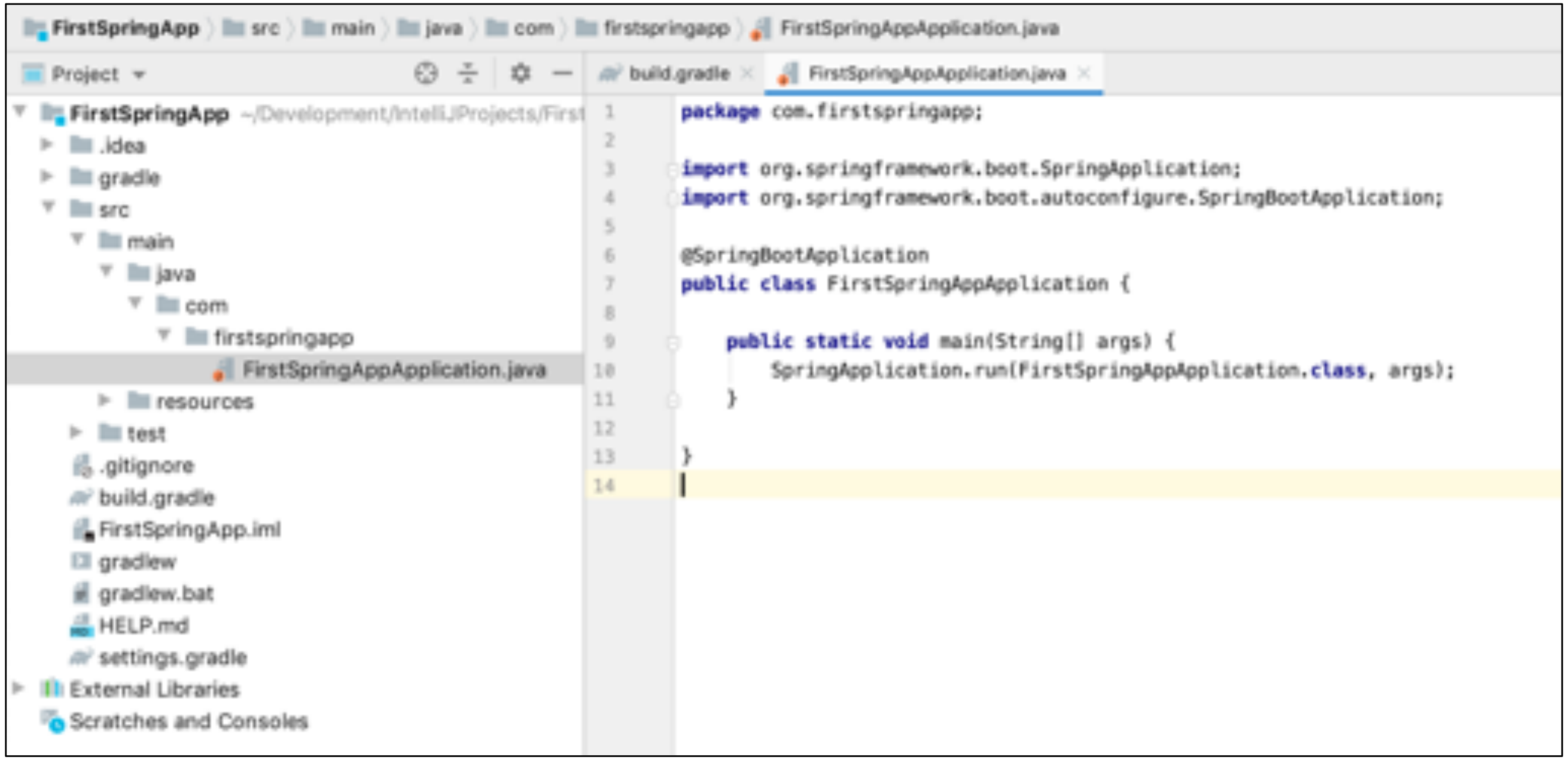
BridgeLabz
Employability Delivered

Step 4: Set Project Name


Project name: FirstSpringApp

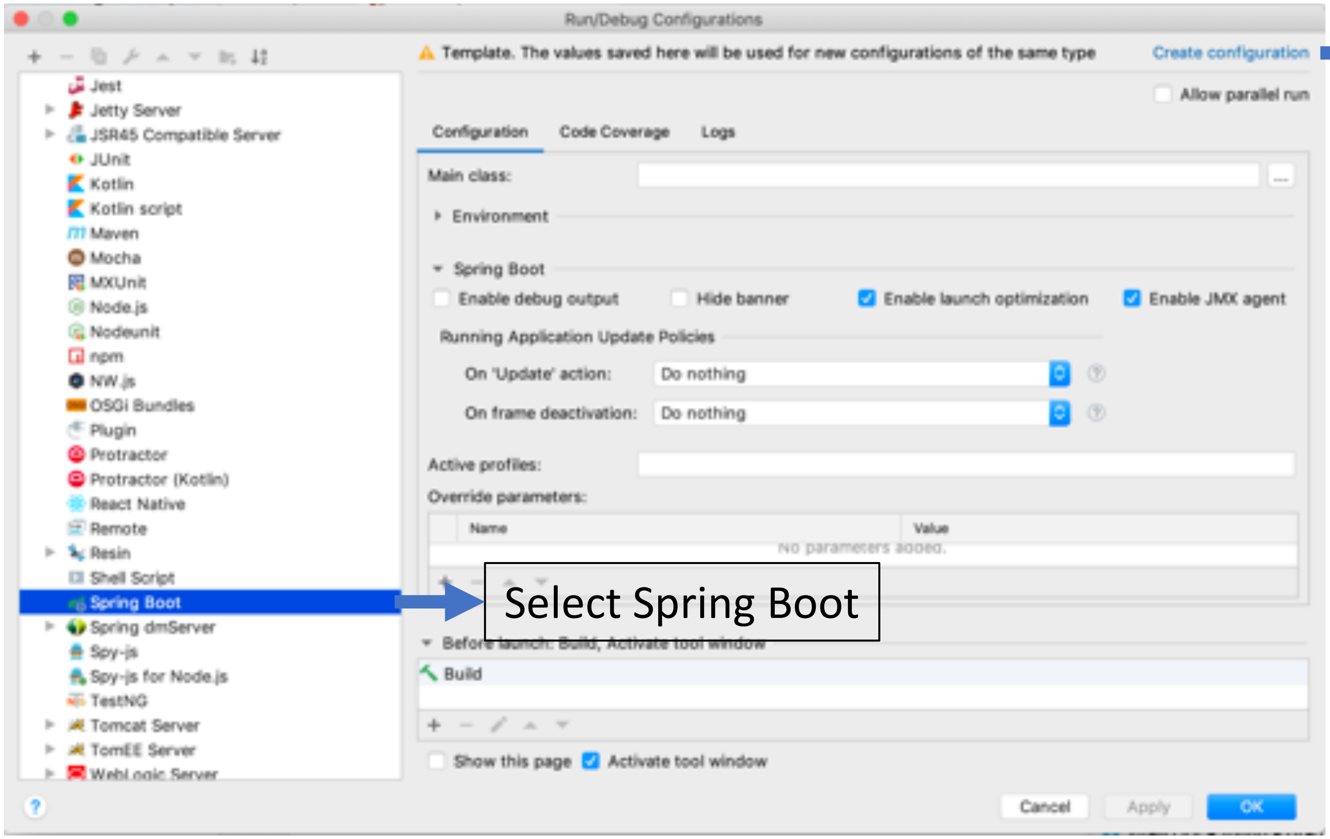
Project location: ~/Development/IntelliJProjects/FirstSpringApp

Step 5: Creates FirstSpringApplication


FirstSpringApp > src > main > java > com > firstspringapp > FirstSpringAppApplication.java
Project
build.gradle
FirstSpringAppApplication.java
FirstSpringApp ~/Development/IntelliJProjects/First
.idea
gradle
src
main
java
com
firstspringapp
FirstSpringAppApplication.java
resources
test
.gitignore
build.gradle
FirstSpringApp.iml
gradlew
gradlew.bat
HELP.md
settings.gradle
External Libraries
Scratches and Consoles

```java
package com.firstspringapp;

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

@SpringBootApplication
public class FirstSpringAppApplication {

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

}
```

Step 6: Setup Spring Boot Configuration



Run/Debug Configurations

Jest
Jetty Server
JSR45 Compatible Server
JUnit
Kotlin
Kotlin script
Maven
Mocha
MXUnit
Node.js
Nodeunit
npm
NW.js
OSGi Bundles
Plugin
Protractor
Protractor (Kotlin)
React Native
Remote
Resin
Shell Script
Spring Boot
Spring dmServer
Spy-js
Spy-js for Node.js
TestNG
Tomcat Server
TomEE Server
WebLogic Server

Template. The values saved here will be used for new configurations of the same type — Create configuration

Allow parallel run

Configuration | Code Coverage | Logs

Main class:

Environment

Spring Boot
Enable debug output · Hide banner · [✓ Enable launch optimization] · [✓ Enable JMX agent]

Running Application Update Policies
On 'Update' action: Do nothing
On frame deactivation: Do nothing

Active profiles:
Override parameters:

Name	Value
No parameters added.	

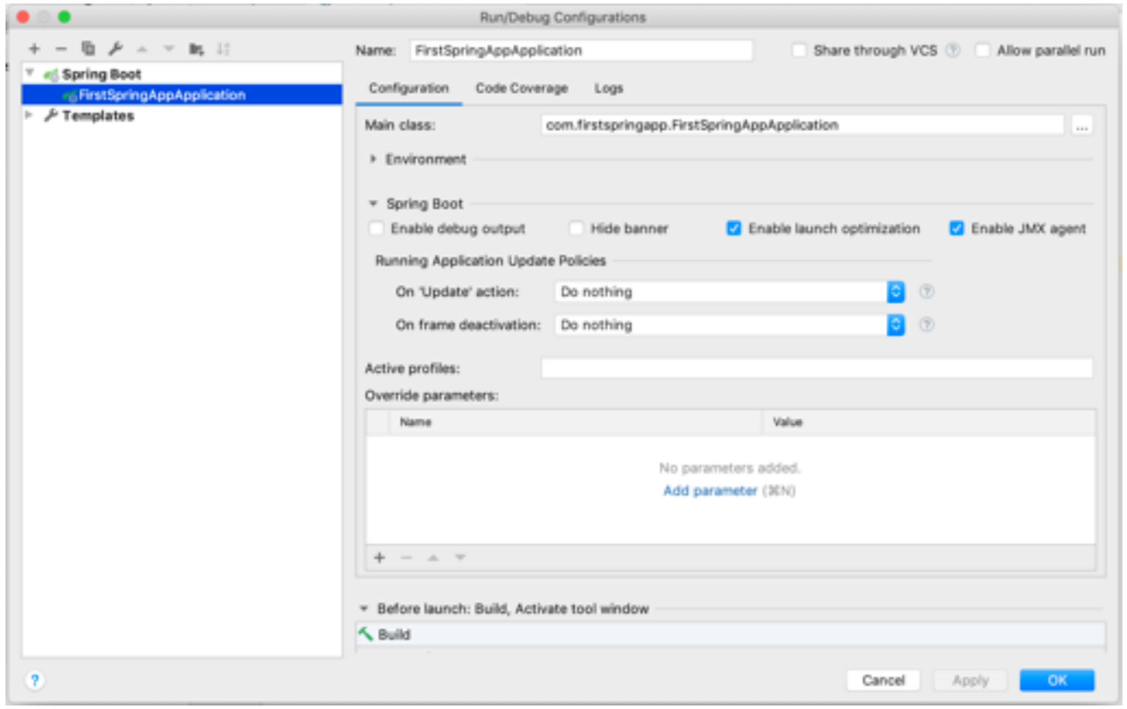
Before launch: Build, Activate tool window
Build

Show this page · [✓ Activate tool window]

Create New Configuration

BridgeLabz
Employability Delivered

Step 7: Create Configuration – Auto Created



Run/Debug Configurations

Spring Boot
FirstSpringAppApplication
Templates

Name: FirstSpringAppApplication · Share through VCS · Allow parallel run

Configuration | Code Coverage | Logs

Main class: com.firstspringapp.FirstSpringAppApplication

Environment

Spring Boot
Enable debug output · Hide banner · [✓ Enable launch optimization] · [✓ Enable JMX agent]

Running Application Update Policies
On 'Update' action: Do nothing
On frame deactivation: Do nothing

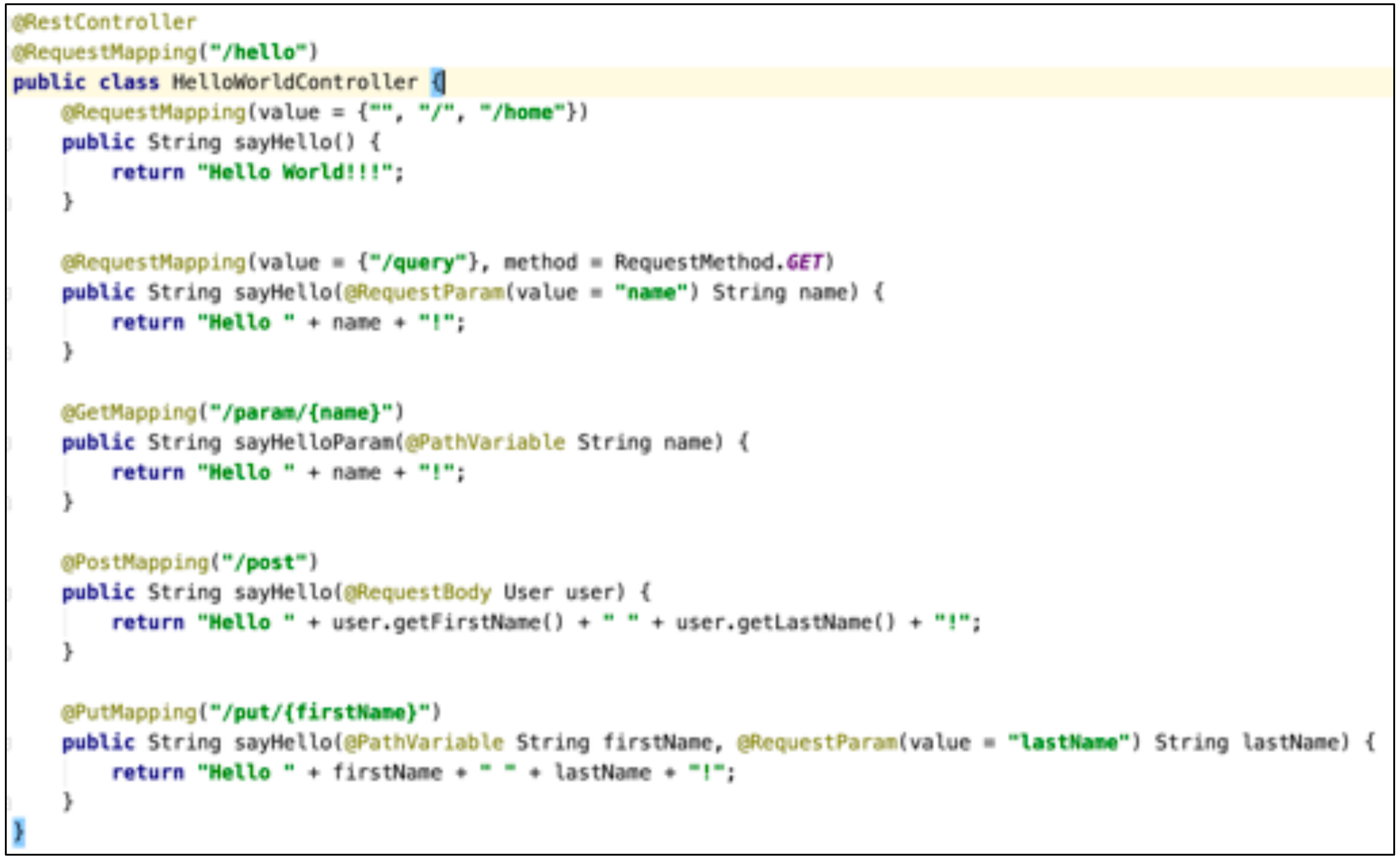
Active profiles:
Override parameters:

Name	Value
No parameters added. Add parameter (⌘N)	

Before launch: Build, Activate tool window
Build

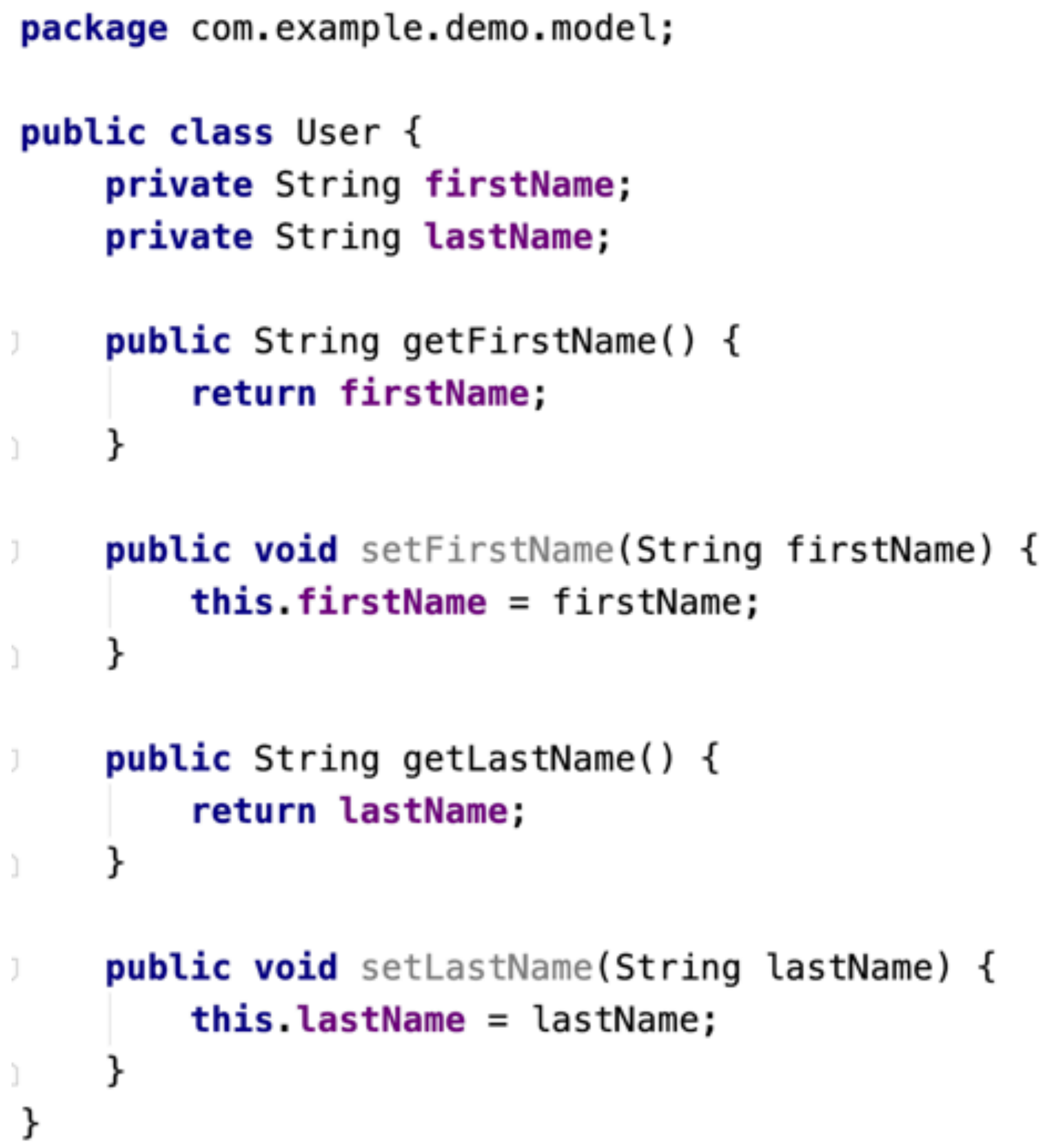
Step 8: Run the application and click http://localhost:8080/

Step 9: Creates HelloWorldController



```java
@RestController
@RequestMapping("/hello")
public class HelloWorldController {
    @RequestMapping(value = {"", "/", "/home"})
    public String sayHello() {
        return "Hello World!!!";
    }

    @RequestMapping(value = {"/query"}, method = RequestMethod.GET)
    public String sayHello(@RequestParam(value = "name") String name) {
        return "Hello " + name + "!";
    }

    @GetMapping("/param/{name}")
    public String sayHelloParam(@PathVariable String name) {
        return "Hello " + name + "!";
    }

    @PostMapping("/post")
    public String sayHello(@RequestBody User user) {
        return "Hello " + user.getFirstName() + " " + user.getLastName() + "!";
    }

    @PutMapping("/put/{firstName}")
    public String sayHello(@PathVariable String firstName, @RequestParam(value = "lastName") String lastName) {
        return "Hello " + firstName + " " + lastName + "!";
    }
}
```

Step 10: Creates User Model



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

public class User {
    private String firstName;
    private String lastName;

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
}
```

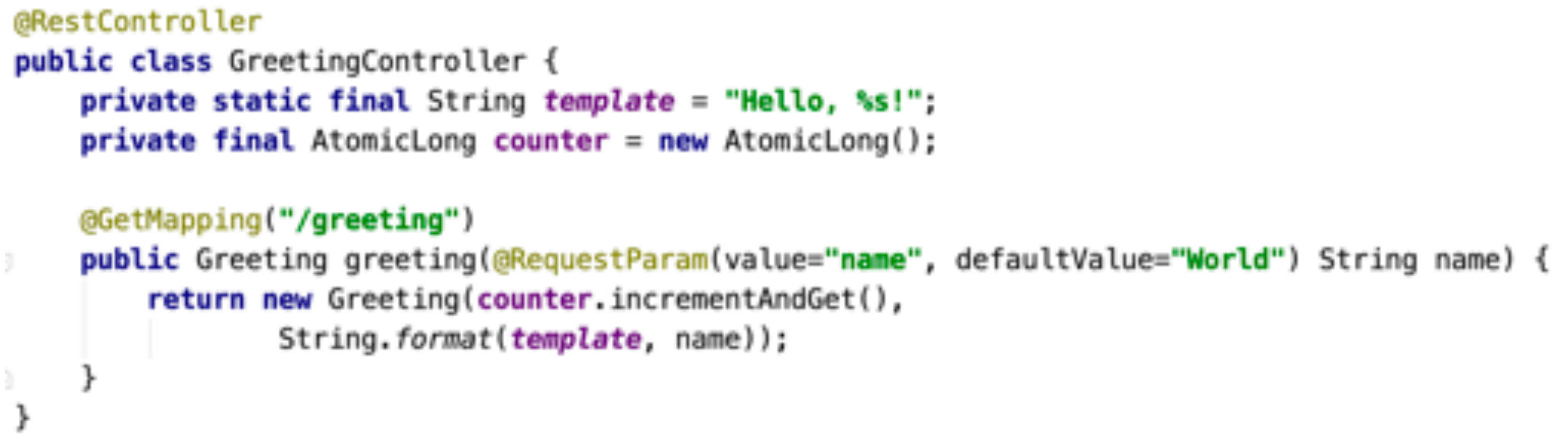
BridgeLabz

Employability Delivered

UC 1

Create Spring App for Hello Messages using different HTTP Methods. Test using curl

GreetingController



```java
@RestController
public class GreetingController {
    private static final String template = "Hello, %s!";
    private final AtomicLong counter = new AtomicLong();

    @GetMapping("/greeting")
    public Greeting greeting(@RequestParam(value="name", defaultValue="World") String name) {
        return new Greeting(counter.incrementAndGet(),
                String.format(template, name));
    }
}
```

UC 2

Using GreetingController return JSON for different HTTP Methods. Test using curl

BridgeLabz

Employability Delivered

Thank You