

Object Oriented Programming

LESSON 12

Spring Framework (Part 1)

Outline

- Introduction
- Spring projects
- Web Project Example



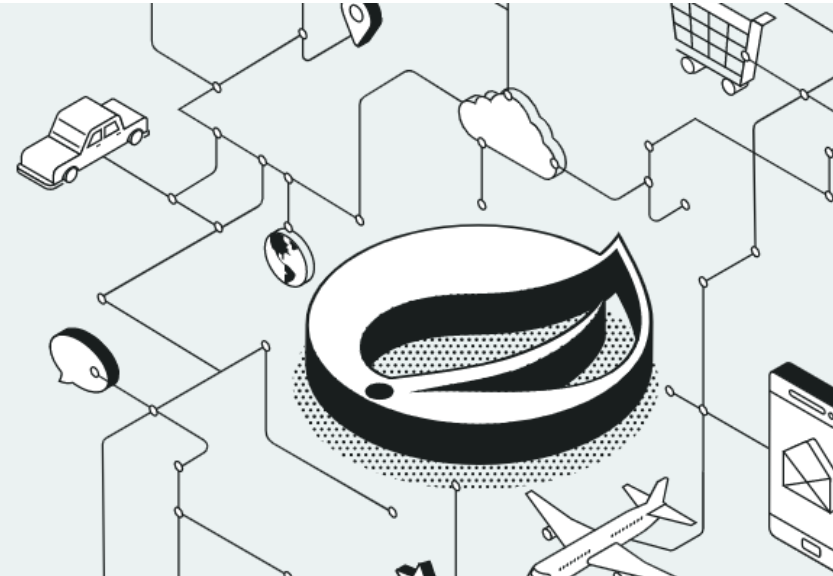
Introduction



3

Why Spring?

Spring makes programming Java quicker, easier, and safer for everybody. Spring's focus on speed, simplicity, and productivity has made it the [world's most popular](#) Java framework.



“We use a lot of the tools that come with the Spring framework and reap the benefits of having a lot of the out of the box solutions, and not having to worry about writing a ton of additional code—so that really saves us some time and energy.”

Introduction



Spring is everywhere



Spring's flexible libraries are trusted by developers all over the world. Spring delivers delightful experiences to millions of end-users every day—whether that's [streaming TV](#), [connected cars](#), [online shopping](#), or countless other innovative solutions. Spring also has contributions from all the big names in tech, including Alibaba, Amazon, Google, Microsoft, and more.

Spring is flexible



Spring's flexible and comprehensive set of extensions and third-party libraries let developers build almost any application imaginable. At its core, Spring Framework's [Inversion of Control \(IoC\)](#) and [Dependency Injection \(DI\)](#) features provide the foundation for a wide-ranging set of features and functionality. Whether you're building secure, reactive, cloud-based microservices for the web, or complex streaming data flows for the enterprise, Spring has the tools to help.

Spring is productive



[Spring Boot](#) transforms how you approach Java programming tasks, radically streamlining your experience. Spring Boot combines necessities such as an application context and an auto-configured, embedded web server to make [microservice](#) development a cinch. To go even faster, you can combine Spring Boot with Spring Cloud's rich set of supporting libraries, servers, patterns, and templates, to safely deploy entire microservices-based architectures into the [cloud](#), in record time.

Spring is fast



Our engineers care deeply about performance. With Spring, you'll notice fast startup, fast shutdown, and optimized execution, by default. Increasingly, Spring projects also support the [reactive](#) (nonblocking) programming model for even greater efficiency. Developer productivity is Spring's superpower. Spring Boot helps developers build applications with ease and with far less toil than other competing paradigms. Embedded web servers, auto-configuration, and "fat jars" help you get started quickly, and innovations like [LiveReload in Spring DevTools](#) mean developers can iterate faster than ever before. You can even start a new Spring project in seconds, with the Spring Initializr at [start.spring.io](#).

What Spring can do



Microservices

Quickly deliver production-grade features with independently evolvable microservices.



Reactive

Spring's asynchronous, nonblocking architecture means you can get more from your computing resources.



Cloud

Your code, any cloud—we've got you covered. Connect and scale your services, whatever your platform.



Web apps

Frameworks for fast, secure, and responsive web applications connected to any data store.



Serverless

The ultimate flexibility. Scale up on demand and scale to zero when there's no demand.



Event Driven

Integrate with your enterprise. React to business events. Act on your streaming data in realtime.



Batch

Automated tasks. Offline processing of data at a time to suit you.



Spring projects



6



Spring Boot

Takes an opinionated view of building Spring applications and gets you up and running as quickly as possible.



Spring Framework

Provides core support for dependency injection, transaction management, web apps, data access, messaging, and more.



Spring Data

Provides a consistent approach to data access – relational, non-relational, map-reduce, and beyond.



Spring Cloud

Provides a set of tools for common patterns in distributed systems. Useful for building and deploying microservices.

Spring projects



7



Spring Cloud Data Flow

Provides an orchestration service for composable data microservice applications on modern runtimes.



Spring Security

Protects your application with comprehensive and extensible authentication and authorization support.



Spring Session

Provides an API and implementations for managing a user's session information.



Spring Integration

Supports the well-known Enterprise Integration Patterns through lightweight messaging and declarative adapters.

Spring projects



8



Spring HATEOAS

Simplifies creating REST representations that follow the HATEOAS principle.



Spring REST Docs

Lets you document RESTful services by combining hand-written documentation with auto-generated snippets produced with Spring MVC Test or REST Assured.



Spring Batch

Simplifies and optimizes the work of processing high-volume batch operations.



Spring AMQP

Applies core Spring concepts to the development of AMQP-based messaging solutions.

Spring projects



9



Spring for Android

Provides key Spring components for use in developing Android applications.



Spring CredHub

Provides client-side support for storing, retrieving, and deleting credentials from a CredHub server running in a Cloud Foundry platform.



Spring Flo

Provides a JavaScript library that offers a basic embeddable HTML5 visual builder for pipelines and simple graphs.



Spring for Apache Kafka

Provides Familiar Spring Abstractions for Apache Kafka.

Spring projects



10



Spring LDAP

Simplifies the development of applications that use LDAP by using Spring's familiar template-based approach.



Spring Mobile

Simplifies the development of mobile web apps through device detection and progressive rendering options.



Spring Roo

Makes it fast and easy to build full Java applications in minutes.



Spring Shell

Makes writing and testing RESTful applications easier with CLI-based resource discovery and interaction.

Spring projects



11



Spring StateMachine

Provides a framework for application developers to use state machine concepts with Spring applications.



Spring Vault

Provides familiar Spring abstractions for HashiCorp Vault



Spring Web Flow

Supports building web applications that feature controlled navigation, such as checking in for a flight or applying for a loan.



Spring Web Services

Facilitates the development of contract-first SOAP web services.

Web Project Example



12

Step 1: Use start.spring.io to create a “web” project. In the “Dependencies” dialog search for and add the “web” dependency as shown in the screenshot. Hit the “Generate” button, download the zip, and unpack it into a folder on your computer.

Web Project Example



13



Project

☐ Maven Project

☒ Gradle Project

Language

☒ Java

☐ Kotlin

☐ Groovy

Spring Boot

☐ 2.4 (SNAPSHOT)

☐ 2.3.1 (SNAPSHOT)

☒ 2.3.0

☐ 2.2.8 (SNAPSHOT)

☐ 2.2.7

☐ 2.1.15 (SNAPSHOT)

☐ 2.1.14

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☐ Jar

☒ War

Java

☐ 14

☐ 11

☒ 8

Dependencies

ADD DEPENDENCIES... CTRL + B

No dependency selected

Select Gradle Project

Select latest stable release

Mostly follow institution structure

Jar for application (*.jar or *.exe)

War for web application

Libraries and Frameworks pre-built for fast development

GENERATE CTRL + G

EXPLORE CTRL + SPACE

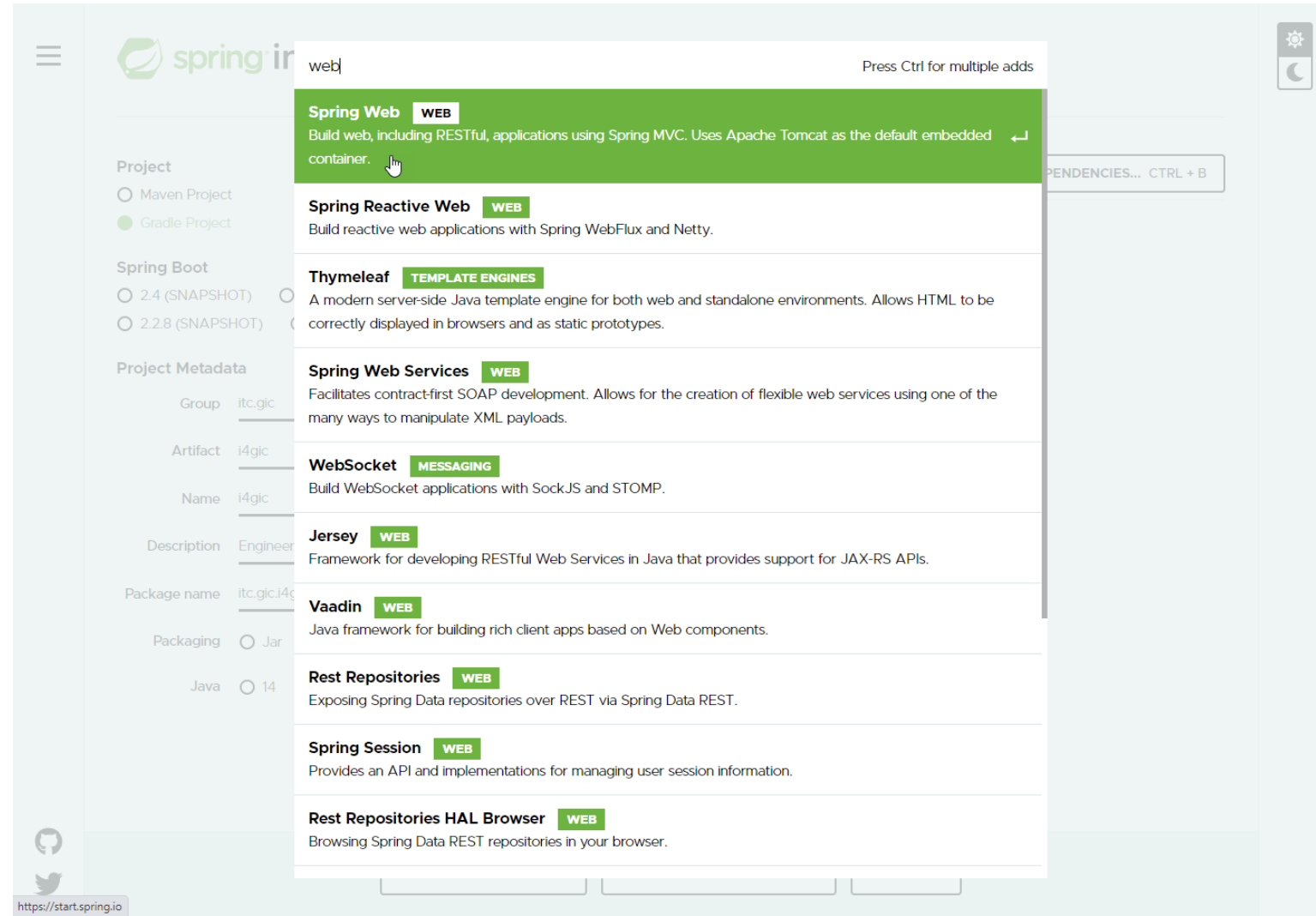
SHARE...

Web Project Example



14

Add **Spring Web** dependency project.



Web Project Example

Click button
GENERATE
to generate
and download
the project.



Project

- ☐ Maven Project
☒ Gradle Project

Language

- ☒ Java ☐ Kotlin
☐ Groovy

Spring Boot

- ☐ 2.4 (SNAPSHOT) ☐ 2.3.1 (SNAPSHOT) ☒ 2.3.0
☐ 2.2.8 (SNAPSHOT) ☐ 2.2.7 ☐ 2.1.15 (SNAPSHOT) ☐ 2.1.14

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☐ Jar ☒ War

Java ☐ 14 ☐ 11 ☒ 8

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Web

WEB

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



GENERATE CTRL + G

EXPLORE CTRL + SPACE

SHARE...



15











Web Project Example



Extract
downloaded
file.

This PC > X-HDD (D:) > ITC > 2020 > I4 > TP22_spring > i4gic >

	Name ^	Date modified	Type	Size
areil photo	 gradle	6/10/2020 8:56 AM	File folder	
	 src	6/10/2020 8:56 AM	File folder	
	 .gitignore	6/10/2020 8:56 AM	Text Document	1 KB
	 build.gradle	6/10/2020 8:56 AM	GRADLE File	1 KB
	 gradlew	6/10/2020 8:56 AM	File	6 KB
	 gradlew.bat	6/10/2020 8:56 AM	Windows Batch File	3 KB
	 HELP.md	6/10/2020 8:56 AM	MD File	2 KB
	 settings.gradle	6/10/2020 8:56 AM	GRADLE File	1 KB

Web Project Example

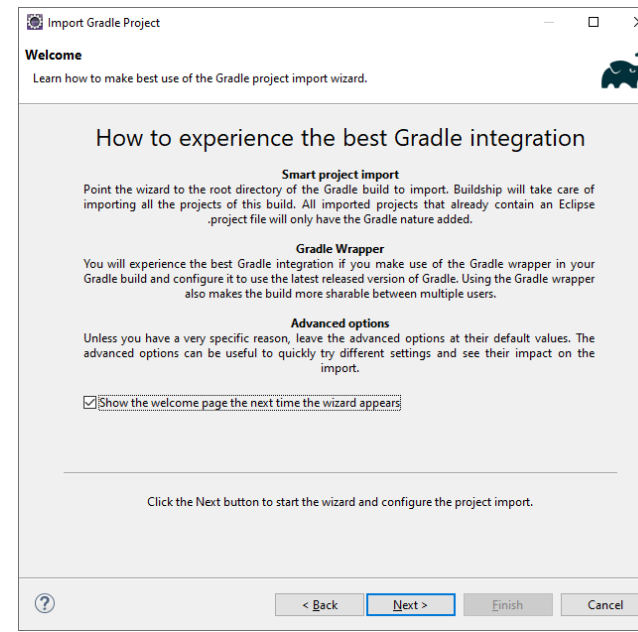
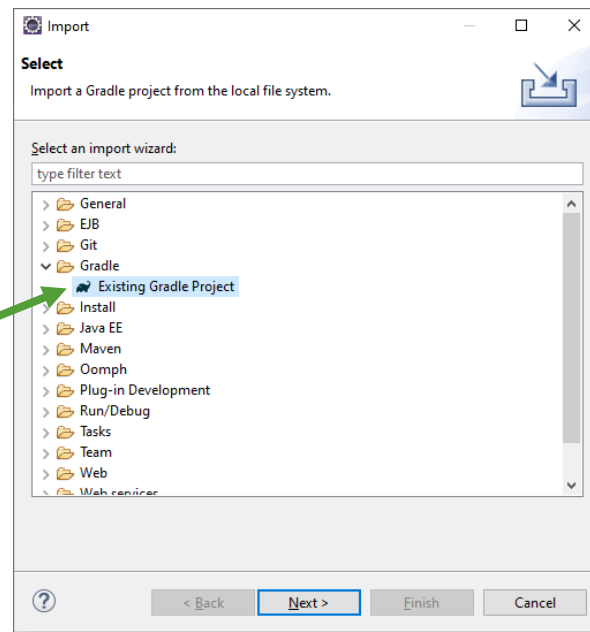
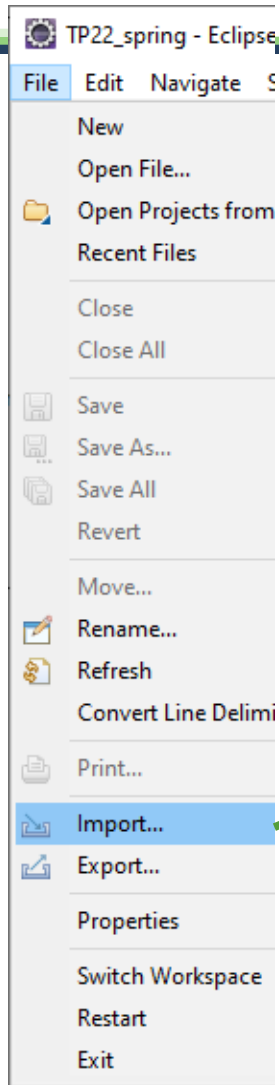


17

- IDE supports [IntelliJ IDEA](#), [Spring Tools](#), [Visual Studio Code](#), or [Eclipse](#), and many more.

- In this example we use **Eclipse**.

1. Open Eclipse
2. File menu → Import... → Existing Gradle Project → Next → Next → Browse...

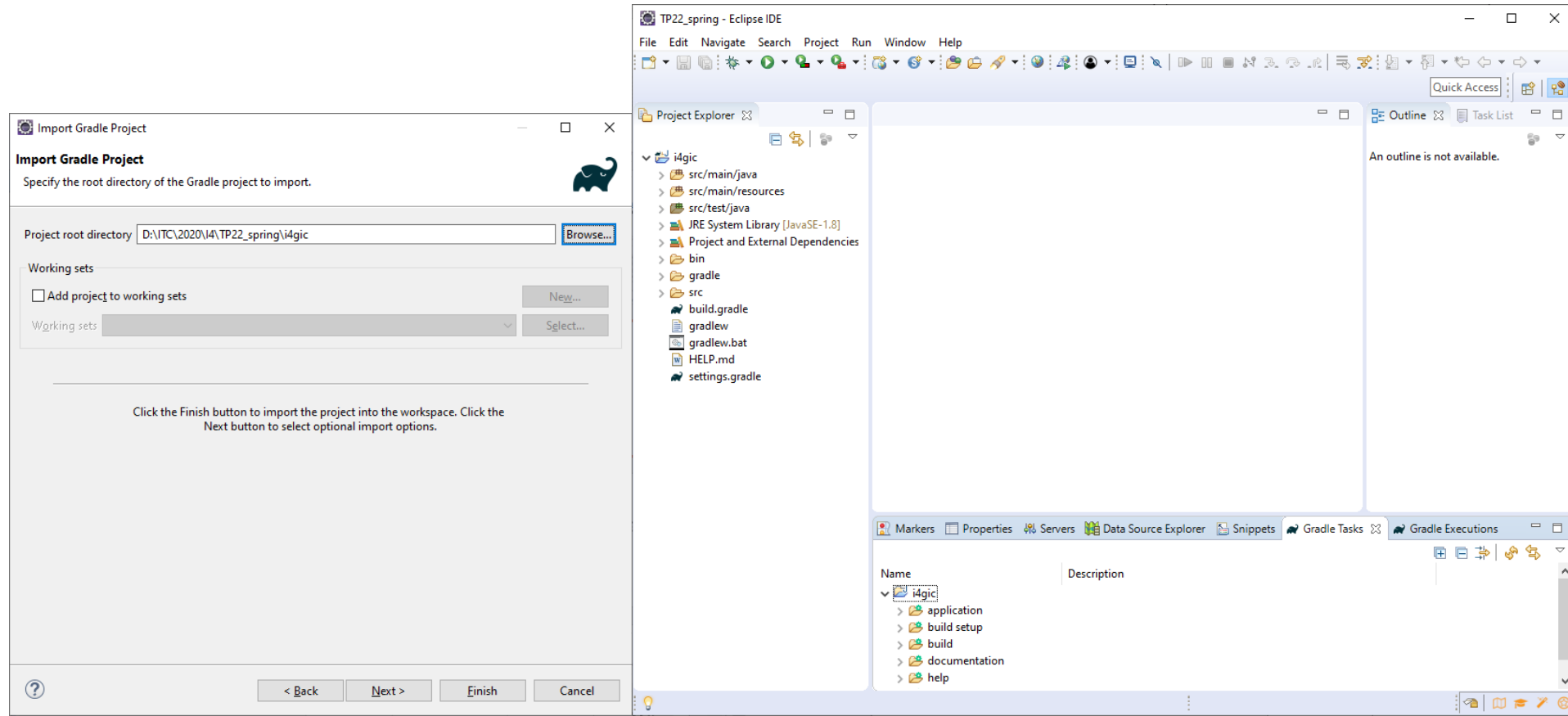


Web Project Example



18

3. Select extracted folder **i4gic**
4. If keep defaults just click **Finish**.



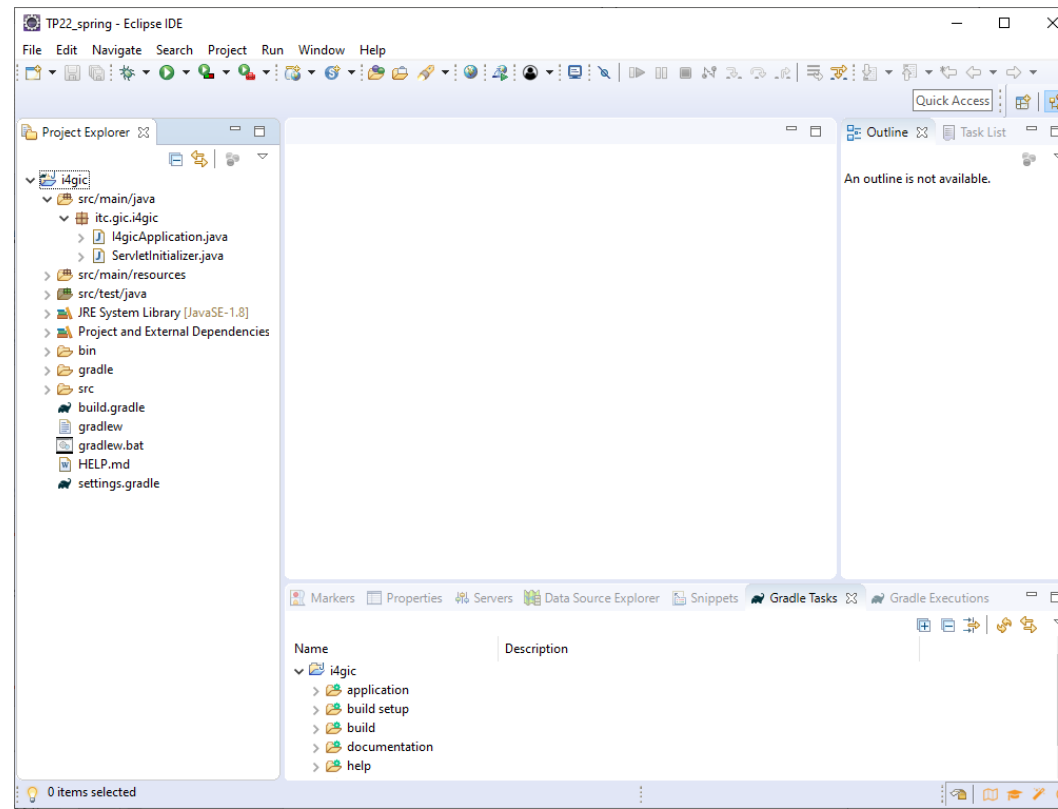
Web Project Example



19

Step 2: Add your code

Locate the **I4gicApplication.java** file in the **src/main/java/itc/gic/i4gic** folder. Now change the contents of the file by adding the extra method and annotations shown in the code below. You can copy and paste the code or just type it.



Web Project Example



20

```
package itc.gic.i4gic;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@RestController
public class I4gicApplication {

    public static void main(String[] args) {
        SpringApplication.run(I4gicApplication.class, args);
    }

    @GetMapping("/hello")
    public String hello(@RequestParam(name="name", defaultValue="World")String name) {
        return String.format("Hello %s!", name);
    }
}
```

The hello() method we've added is designed to take a String parameter called name, and then combine this parameter with the word "Hello" in the code. This means that if you set your name to "Sreng" in the request, the response would be "Hello Sreng!".

Web Project Example



21

```
package itc.gic.i4gic;
```

```
import org.springframework.boot.SpringApplication;
```

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

```
import org.springframework.web.bind.annotation.GetMapping;
```

```
import org.springframework.web.bind.annotation.RequestParam;
```

```
import org.springframework.web.bind.annotation.RestController;
```

```
@SpringBootApplication
```

```
@RestController
```

```
public class I4gicApplication {
```

```
    public static void main(String[] args) {
```

```
        SpringApplication.run(I4gicApplication.class, args);
```

```
    } The @GetMapping("/hello") tells Spring to use our hello() method to  
    answer requests that get sent to the http://localhost:8080/hello address.
```

```
@GetMapping("/hello")
```

```
public String hello(@RequestParam(name="name", defaultValue="World")String name) {
```

```
    return String.format("Hello %s!", name);
```

```
}
```

```
}
```

The **@RestController** annotation tells Spring that this code describes an endpoint that should be made available over the web.

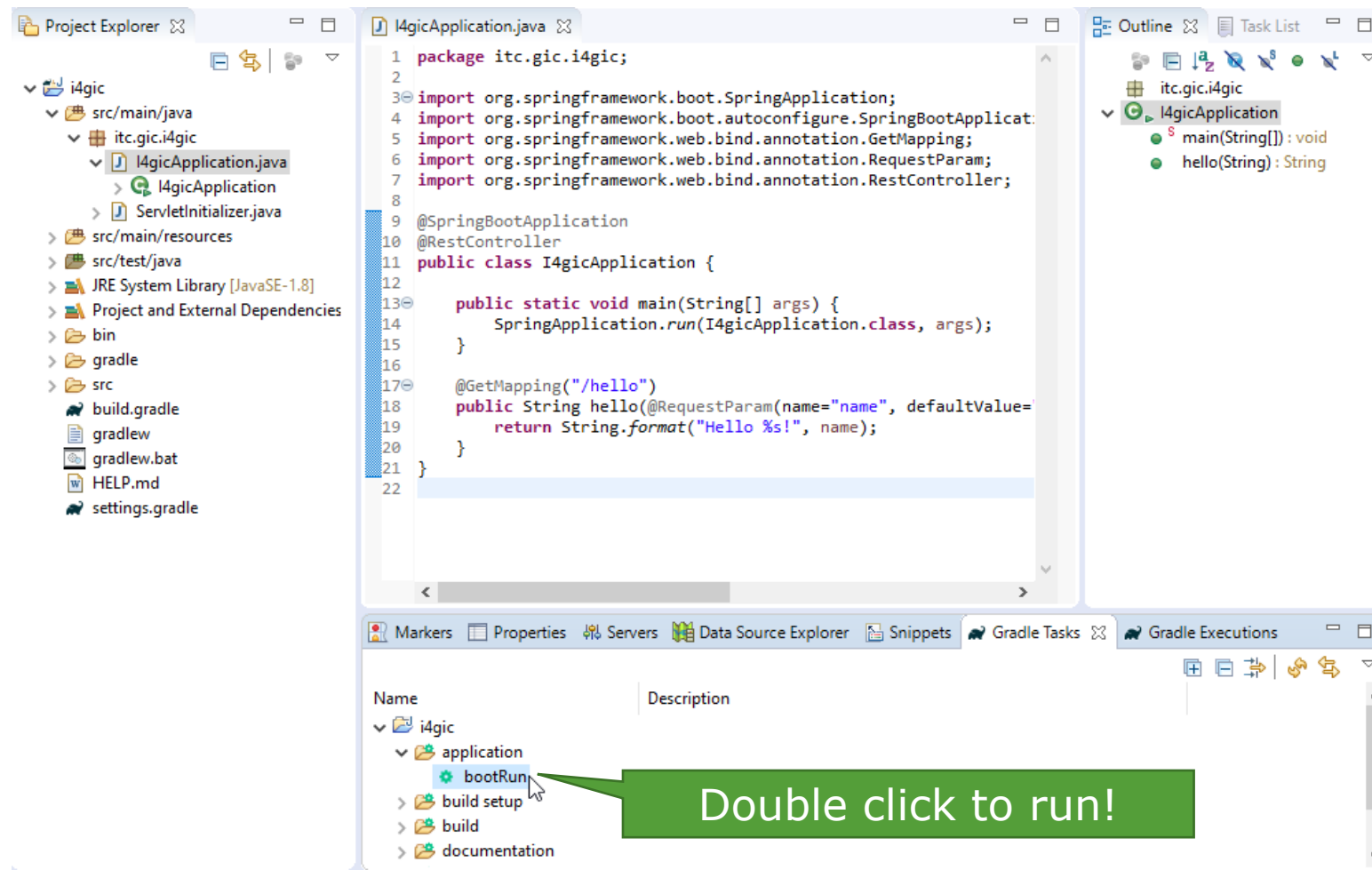
the **@RequestParam** is telling Spring to expect a name value in the request, but if it's not there, it will use the word **"World"** by default.

Web Project Example



22

Step 3: Run web app



The screenshot shows an IDE with the following components:

- Project Explorer:** Shows the project structure with folders like `src/main/java`, `src/main/resources`, `src/test/java`, and files like `build.gradle`, `gradlew`, `gradlew.bat`, `HELP.md`, and `settings.gradle`.
- Editor:** Displays the `I4gicApplication.java` file with the following code:

```
1 package itc.gic.i4gic;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.web.bind.annotation.GetMapping;
6 import org.springframework.web.bind.annotation.RequestParam;
7 import org.springframework.web.bind.annotation.RestController;
8
9 @SpringBootApplication
10 @RestController
11 public class I4gicApplication {
12
13     public static void main(String[] args) {
14         SpringApplication.run(I4gicApplication.class, args);
15     }
16
17     @GetMapping("/hello")
18     public String hello(@RequestParam(name="name", defaultValue="") String name) {
19         return String.format("Hello %s!", name);
20     }
21 }
22
```
- Outline:** Shows the class `I4gicApplication` with methods `main(String[]): void` and `hello(String): String`.
- Gradle Tasks:** Shows a tree structure under `i4gic` with `application` expanded, showing `bootRun` (highlighted with a green box and a callout), `build setup`, `build`, and `documentation`.

A green callout box with the text "Double click to run!" points to the `bootRun` task in the Gradle Tasks panel.

Web Project Example



23

Step 3: Run web app

The screenshot shows an IDE with two windows. The top window, titled 'Gradle Tasks', displays a tree of operations: 'Run build' (13.322 s), 'Load build' (0.044 s), 'Configure build' (0.226 s), 'Calculate task graph' (0.286 s), 'Run tasks' (12.757 s), and ':compileJava' (1.802 s). The bottom window, titled 'Console', shows the output of the 'bootRun' task. It includes the working directory, Gradle version (6.4.1), Java home, and JVM arguments. The console output shows the 'Task :bootRun' command being executed, followed by the Spring Boot logo and version (v2.3.0.RELEASE). The console output then shows the application starting on port 8080, with Tomcat 9.0.35 initialized and the application starting successfully.

```
i4gic - bootRun [Gradle Project] bootRun in D:\ITC\2020\I4\TP22_spring\i4gic (Jun 10, 2020 4:56:23 PM)
Working Directory: D:\ITC\2020\I4\TP22_spring\i4gic
Gradle User Home: C:\Users\DICE\.gradle
Gradle Distribution: Gradle wrapper from target build
Gradle Version: 6.4.1
Java Home: C:\Program Files\Java\jdk1.8.0_181
JVM Arguments: None
Program Arguments: None
Build Scans Enabled: false
Offline Mode Enabled: false
Gradle Tasks: bootRun

> Task :compileJava
> Task :processResources
> Task :classes

> Task :bootRun

:: Spring Boot :: (v2.3.0.RELEASE)

2020-06-10 16:56:28.080 INFO 7532 --- [main] itc.gic.i4gic.I4gicApplication : Starting I4gicApplication on DESKTOP-P9IPQM1 with PID 7532 (D:\ITC\
2020-06-10 16:56:28.084 INFO 7532 --- [main] itc.gic.i4gic.I4gicApplication : No active profile set, falling back to default pro
2020-06-10 16:56:30.069 INFO 7532 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2020-06-10 16:56:30.082 INFO 7532 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-06-10 16:56:30.083 INFO 7532 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.35]
2020-06-10 16:56:30.177 INFO 7532 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2020-06-10 16:56:30.177 INFO 7532 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 2029 ms
2020-06-10 16:56:30.362 INFO 7532 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2020-06-10 16:56:30.533 INFO 7532 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2020-06-10 16:56:30.545 INFO 7532 --- [main] itc.gic.i4gic.I4gicApplication : Started I4gicApplication in 2.954 seconds (JVM running for 3.556)
```

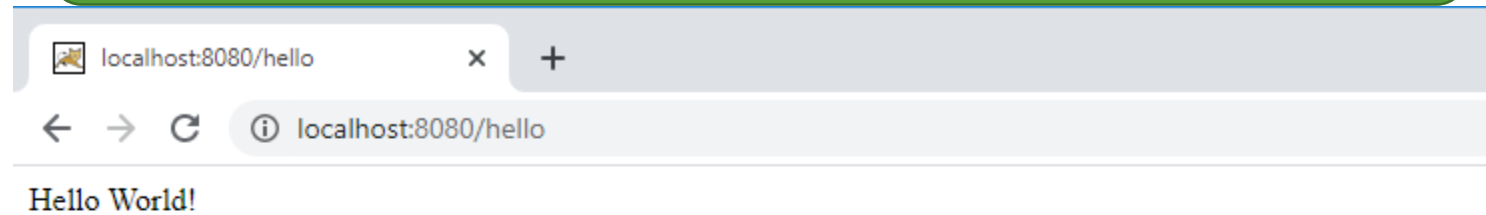
Start Tomcat 9.0.35
at port 8080

Web Project Example



24

Now open your browser and go to address:
`http://localhost:8080/hello`



Try add parameter to URL **?name=your_name** what will be displayed?

References



25

- <https://spring.io/>
- <https://spring.io/quickstart>
- <https://start.spring.io/>
- <https://spring.io/guides>
- <https://spring.io/guides#getting-started-guides>
- <https://spring.io/projects>
- <https://projects.spring.io/spring-roo/#quick-start>