

# Software Engineering

LESSON 05

## Spring Framework (Core)

# Outline

---

- Introduction
- Spring projects
- Spring Framework Architecture
- Web Project Example
- Web Project Example (VS Code)



# Introduction



## 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

# 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



## 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



## 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



## 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



## 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

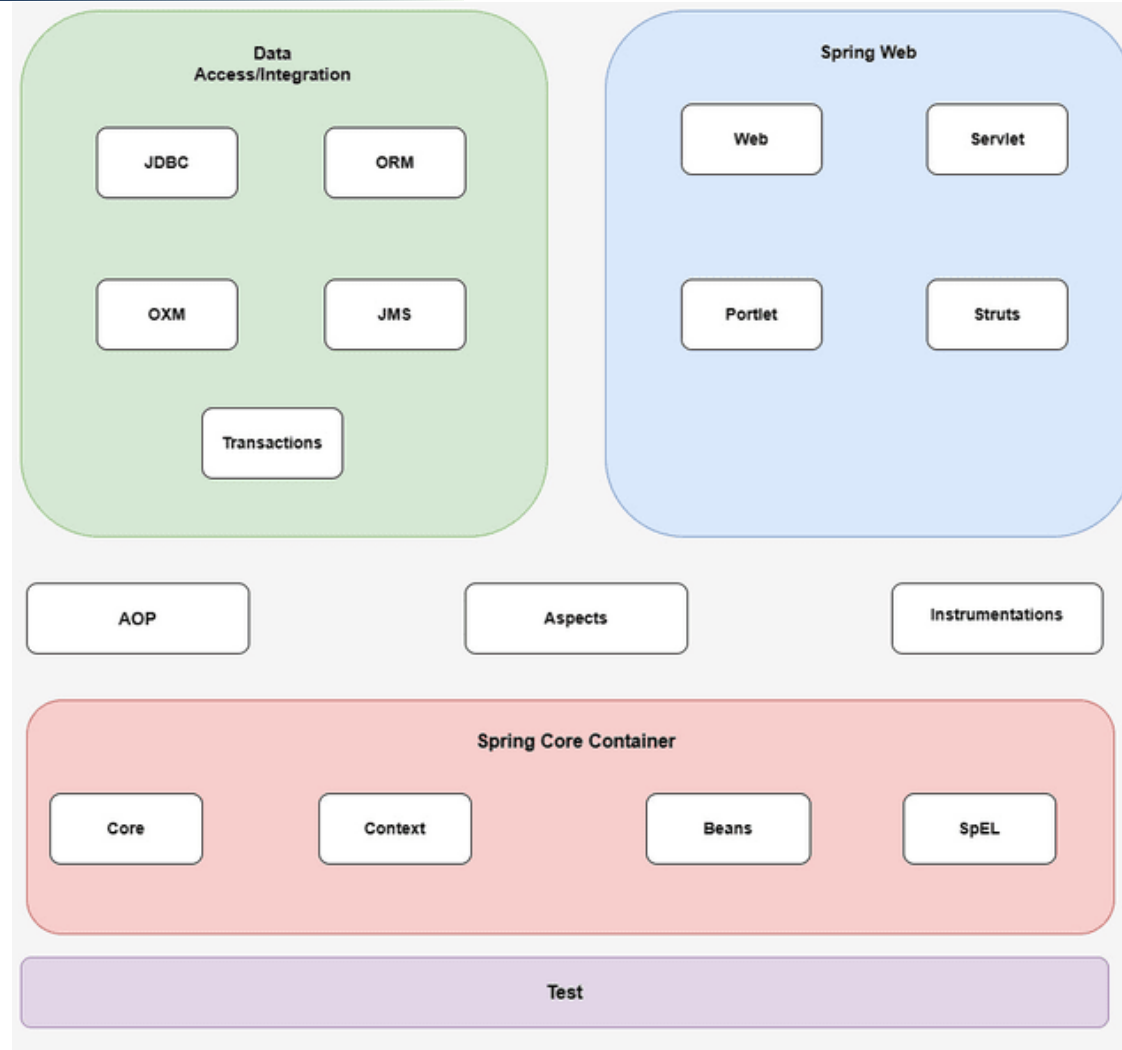
Makes it fast and easy to build full Java applications in minutes.



## Spring Shell

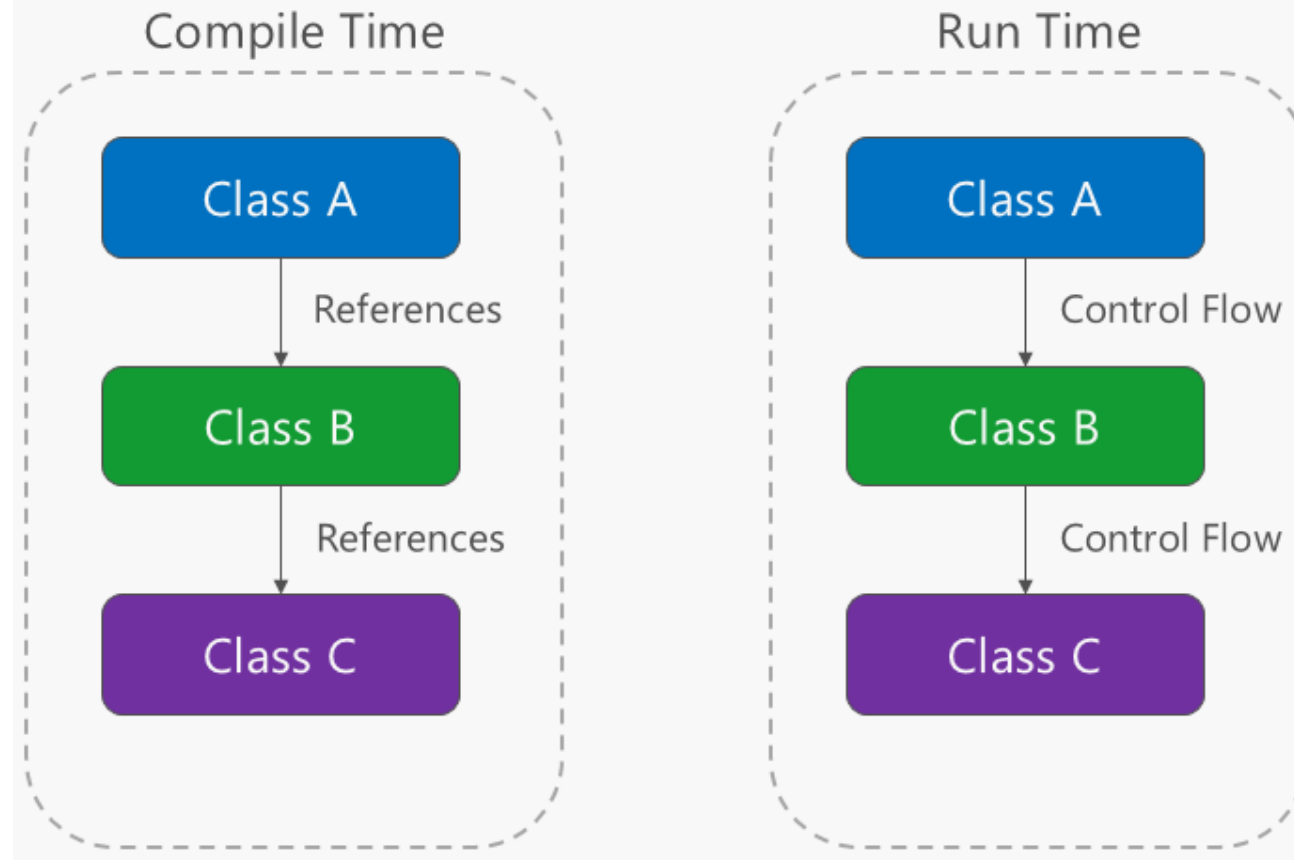
Makes writing and testing RESTful applications easier with CLI-based resource discovery and interaction.

# Spring Framework Architecture



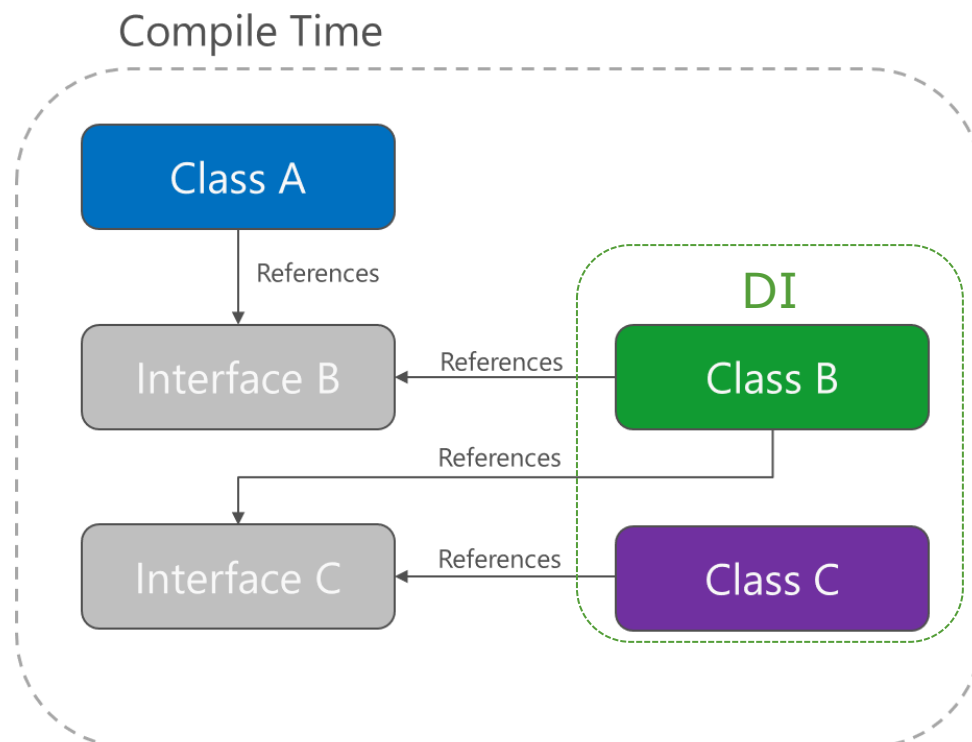
# Inversion of Control (IoC)

## Direct Dependency Graph

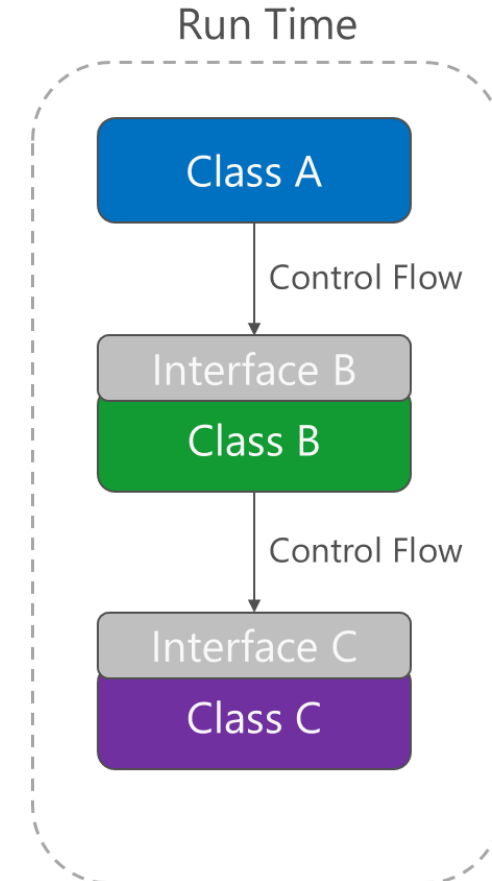


# Inversion of Control (IoC)

## Inverted Dependency Graph



The Spring-Core module is responsible for injecting dependencies through either Constructor or Setter methods.





# Web Project Example

---

Step 1: Use [start.spring.io](https://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

Select Gradle Project

Select latest stable release

Mostly follow institution structure

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

War for web application

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

itc.gic

Artifact

i4gic

Name

i4gic

Description

Engineering year 4

Package name

itc.gic.i4gic

Packaging

☐ Jar

☒ War

Java

☐ 14

☐ 11

☒ 8

Dependencies

ADD DEPENDENCIES... CTRL + B

No dependency selected

GENERATE CTRL + G

EXPLORE CTRL + SPACE

SHARE...

Libraries and Frameworks pre-built for fast development

# Web Project Example

Add **Spring Web** dependency project.

The screenshot shows the Spring Initializr interface. On the left, there's a sidebar with navigation options: Project (Maven Project, Gradle Project), Spring Boot (2.4 (SNAPSHOT), 2.2.8 (SNAPSHOT)), and Project Metadata (Group, Artifact, Name, Description, Package name, Packaging, Java). The main area displays a search bar with 'web' entered. Below the search bar, a list of dependencies is shown, each with a category tag and a description. The first item, 'Spring Web', is highlighted with a green background and a mouse cursor pointing at it. Other items include 'Spring Reactive Web', 'Thymeleaf', 'Spring Web Services', 'WebSocket', 'Jersey', 'Vaadin', 'Rest Repositories', 'Spring Session', and 'Rest Repositories HAL Browser'. A button 'DEPENDENCIES... CTRL + B' is visible on the right side of the list.

Dependency	Category	Description
Spring Web	WEB	Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
Spring Reactive Web	WEB	Build reactive web applications with Spring WebFlux and Netty.
Thymeleaf	TEMPLATE ENGINES	A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.
Spring Web Services	WEB	Facilitates contract-first SOAP development. Allows for the creation of flexible web services using one of the many ways to manipulate XML payloads.
WebSocket	MESSAGING	Build WebSocket applications with SockJS and STOMP.
Jersey	WEB	Framework for developing RESTful Web Services in Java that provides support for JAX-RS APIs.
Vaadin	WEB	Java framework for building rich client apps based on Web components.
Rest Repositories	WEB	Exposing Spring Data repositories over REST via Spring Data REST.
Spring Session	WEB	Provides an API and implementations for managing user session information.
Rest Repositories HAL Browser	WEB	Browsing Spring Data REST repositories in your browser.

# 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...





# Web Project Example



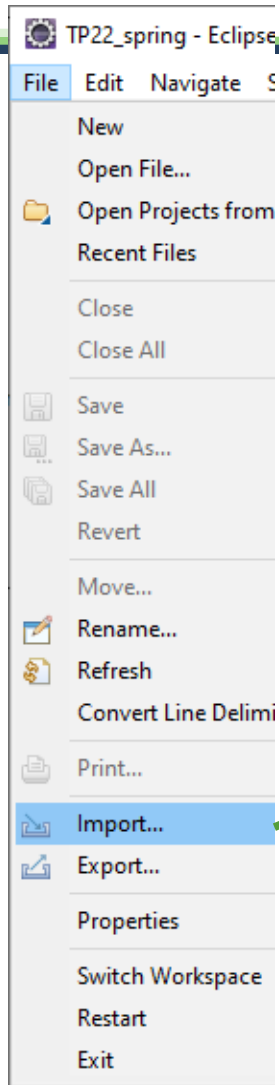
Extract  
downloaded  
file.

This PC > X-HDD (D:) > ITC > 2020 > I4 > TP22\_spring > i4gic >

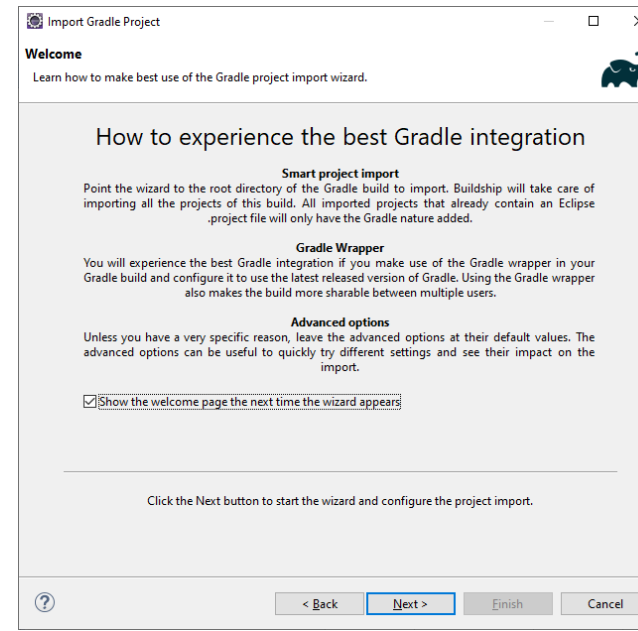
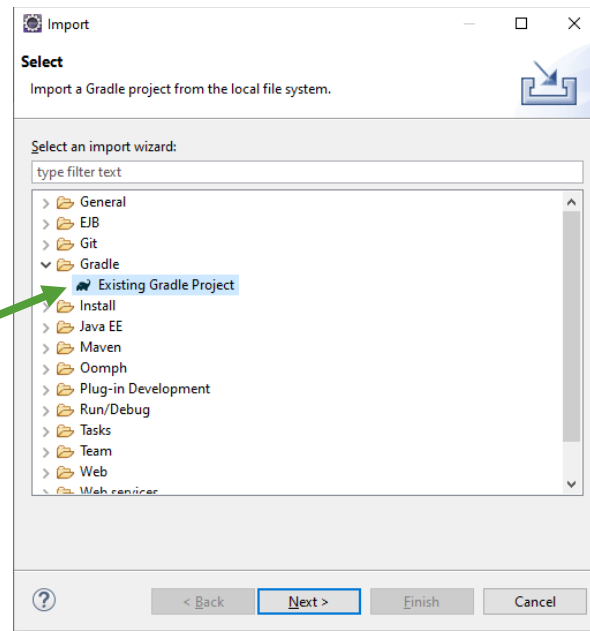
	Name	Date modified	Type	Size
	gradle	6/10/2020 8:56 AM	File folder	
	src	6/10/2020 8:56 AM	File folder	
	.gitignore	6/10/2020 8:56 AM	Text Document	1 KB
	build.gradle	6/10/2020 8:56 AM	GRADLE File	1 KB
	gradlew	6/10/2020 8:56 AM	File	6 KB
	gradlew.bat	6/10/2020 8:56 AM	Windows Batch File	3 KB
	HELP.md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

areil photo

# Web Project Example

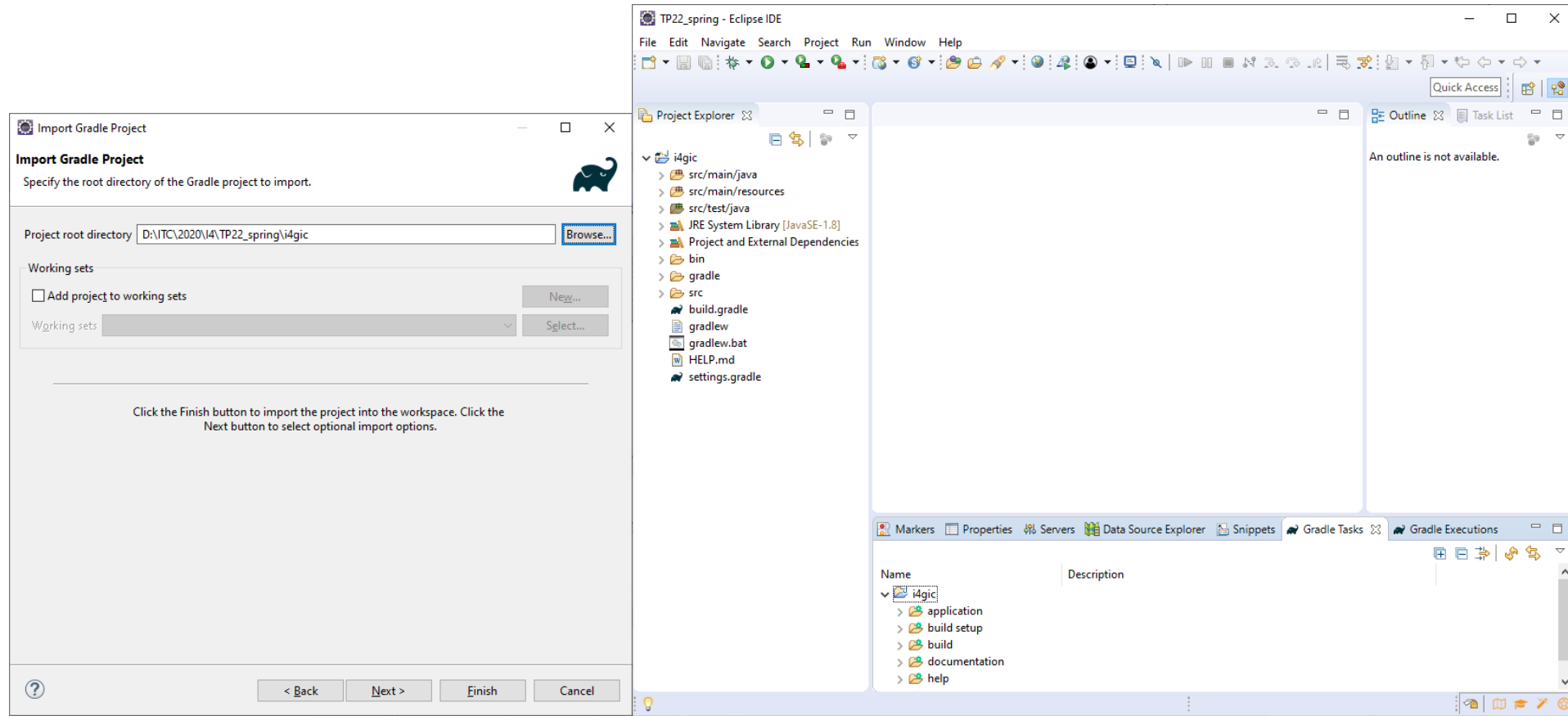


- IDE supports [IntelliJ IDEA](#), [Spring Tools](#), [Visual Studio Code](#), or [Eclipse](#), and many more.
- In this example we use **Eclipse**.
  1. Open Eclipse
  2. File menu → Import... → Existing Gradle Project → Next → Next → Browse...



# Web Project Example

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

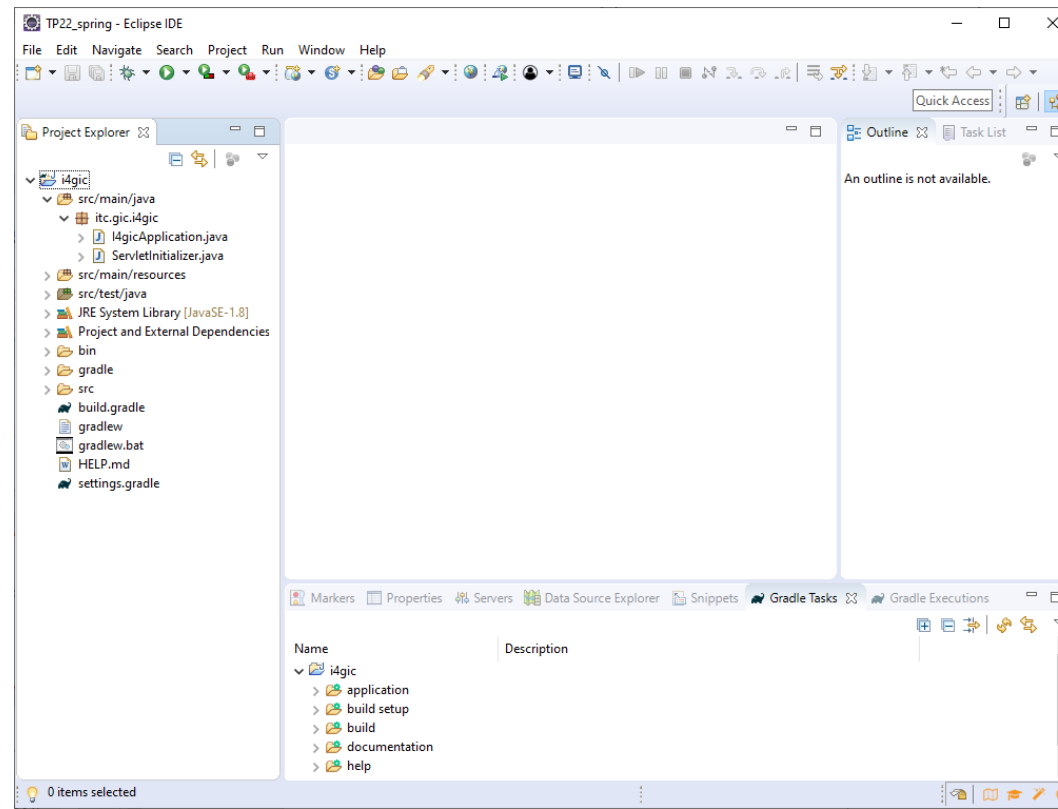




# Web Project Example

## 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

---

```
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

```
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

## Step 3: Run web app

The screenshot displays an IDE interface for a Spring Boot web application project named 'i4gic'. The 'Project Explorer' on the left shows the project structure, including the 'src/main/java' directory containing 'I4gicApplication.java' and 'ServletInitializer.java'. The 'I4gicApplication.java' file is open in the editor, showing 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
```

The 'Outline' panel on the right shows the class structure, including the 'main' method and the 'hello' endpoint. The 'Gradle Tasks' panel at the bottom shows the 'bootRun' task under the 'application' sub-task, which is highlighted with a green callout box saying 'Double click to run!'.

# Web Project Example

## Step 3: Run web app

The screenshot displays an IDE window with two tabs: 'Gradle Tasks' and 'Console'. The 'Gradle Tasks' tab shows a tree of operations with their durations:

Operation	Duration
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
> :compileJava	1.802 s

The 'Console' tab shows the output of the 'bootRun' task. It includes environment variables, task names, and a detailed log of the application startup process, including the initialization of Tomcat 9.0.35.

```
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\I4\TP22_spring\i4gic\build\classes\java\main
2020-06-10 16:56:28.084 INFO 7532 --- [main] itc.gic.i4gic.I4gicApplication : No active profile set, falling back to default profile
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)
```

A green callout box points to the Tomcat startup log entry:

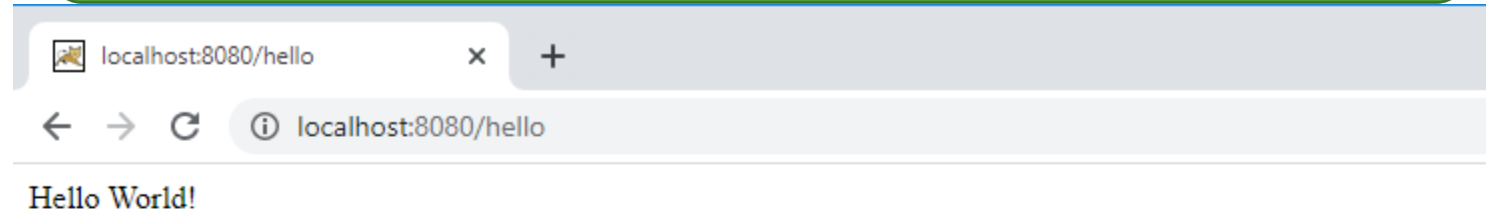
**Start Tomcat 9.0.35 at port 8080**





# Web Project Example

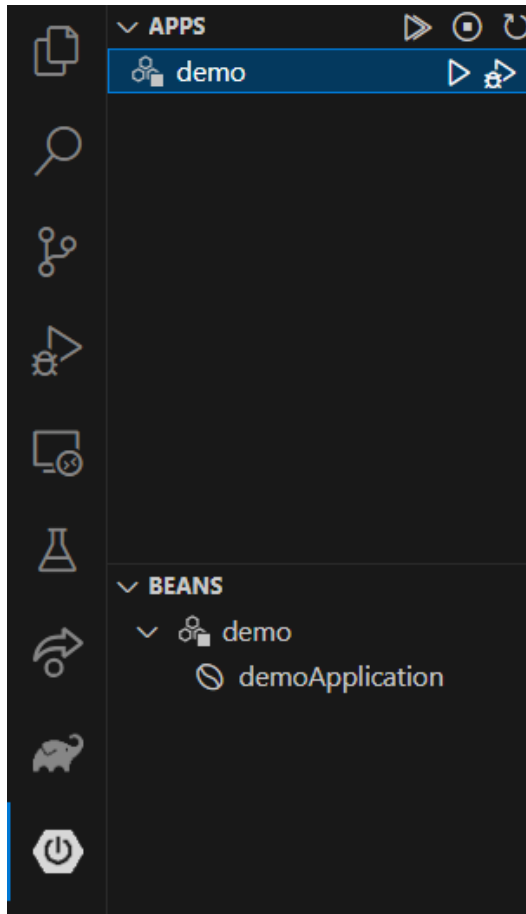
Now open your browser and go to address:  
`http://localhost:8080/hello`



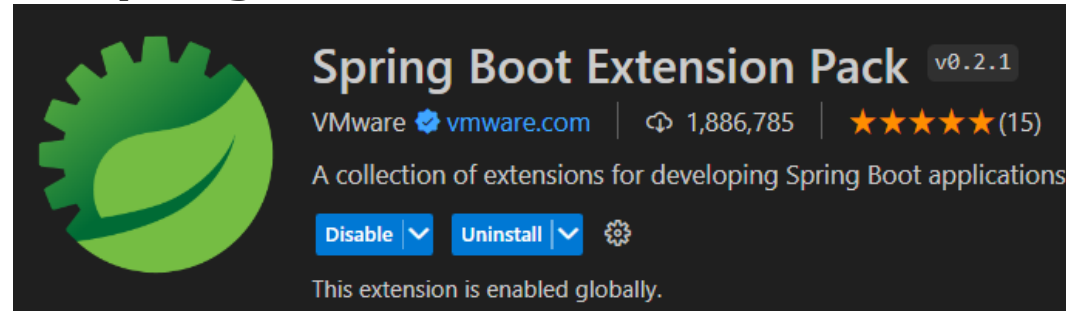
Try add parameter to URL **?name=your\_name** what will be displayed?

# Web Project Example (VS Code)

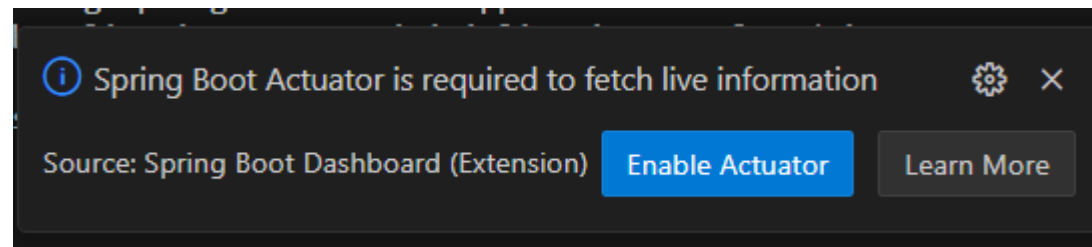
- In case we use **Visual Studio Code**.



1. Open the project folder with Visual Studio Code
2. **"Spring Boot Extension Pack"** is needed



3. After the extension is activated, you run the project by opening Spring Boot Tab on the left.  
If you want to monitor app health, enable Actuator.





# References

---

- <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>
- <https://www.geeksforgeeks.org/spring-framework-architecture/>