

# Spring Boot Introduction



# You will learn how to ...

# You will learn how to ...

- Quickly develop Spring Boot applications

# You will learn how to ...

- Quickly develop Spring Boot applications
- Develop a REST API using Spring Boot

# You will learn how to ...

- Quickly develop Spring Boot applications
- Develop a REST API using Spring Boot
- Create a Spring MVC app with Spring Boot

# You will learn how to ...

- Quickly develop Spring Boot applications
- Develop a REST API using Spring Boot
- Create a Spring MVC app with Spring Boot
- Connect Spring Boot apps to a Database for CRUD development

# You will learn how to ...

- Quickly develop Spring Boot applications
- Develop a REST API using Spring Boot
- Create a Spring MVC app with Spring Boot
- Connect Spring Boot apps to a Database for CRUD development
- Leverage all Java configuration (no xml) and Maven

# Practical Results

# Practical Results

- Introduction to Spring Boot development

# Practical Results

- Introduction to Spring Boot development
- Not an A to Z reference

# Practical Results

- Introduction to Spring Boot development
- Not an A to Z reference
- For complete reference, see [Spring Boot Reference Manual](#)

# Practical Results

- Introduction to Spring Boot development
- Not an A to Z reference
- For complete reference, see [Spring Boot Reference Manual](#)

<https://spring.io/projects/spring-boot>

# The Problem

# The Problem

- Building a Spring application is really HARD!!!

# The Problem

- Building a Spring application is really HARD!!!

Q: What Maven archetype to use?

# The Problem

- Building a Spring application is really HARD!!!

Q: What Maven archetype to use?

Q: Which Maven dependencies do I need?

# The Problem

- Building a Spring application is really HARD!!!

Q: What Maven archetype to use?

Q: Which Maven dependencies do I need?

Q: How do I set up configuration (xml or Java)?

# The Problem

- Building a Spring application is really HARD!!!

Q: What Maven archetype to use?

Q: Which Maven dependencies do I need?

Q: How do I set up configuration (xml or Java)?

Q: How do I install the server? (Tomcat, JBoss etc...)

# The Problem

- Building a Spring application is really HARD!!!

Q: What Maven archetype to use?

And that's  
JUST the basics  
for getting started

Q: Which Maven dependencies do I need?

Q: How do I set up configuration (xml or Java)?

Q: How do I install the server? (Tomcat, JBoss etc...)

# The Problem

# The Problem

- Tons of configuration

# The Problem

- Tons of configuration

```
<!-- Step 1: Configure Spring MVC Dispatcher Servlet -->
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</s
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring-mvc-demo-servlet.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>

<!-- Step 2: Set up URL mapping for Spring MVC Dispatcher Servlet -->
<servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

# The Problem

- Tons of configuration

```
<!-- Step 1: Configure Spring MVC Dispatcher Servlet -->
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</s
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring-mvc-demo-servlet.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>

<!-- Step 2: Set up URL mapping for Spring MVC Dispatcher Servlet -->
<servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>

<!-- Step 3: Add support for component scanning -->
<context:component-scan base-package="com.Luv2code.springdemo" />

<!-- Step 4: Add support for conversion, formatting and validation support -->
<mvc:annotation-driven/>

<!-- Step 5: Define Spring MVC view resolver -->
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/view/" />
    <property name="suffix" value=".jsp" />
</bean>
```

# The Problem

- Tons of configuration

```
<!-- Step 1: Configure Spring MVC Dispatcher Servlet -->
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring-mvc-demo-servlet.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>

<!-- Step 2: Set up URL mapping for Spring MVC Dispatcher Servlet -->
<servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>

<!-- Step 3: Add support for component scanning -->
<context:component-scan base-package="com.luv2code.springsecurity.demo">
<!-- Step 4: Add support for conversion, formatting and localization -->
<mvc:annotation-driven>

<!-- Step 5: Define Spring MVC view resolver -->
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/view/" />
    <property name="suffix" value=".jsp" />

```

```
@Configuration
@EnableWebMvc
@ComponentScan(basePackages="com.luv2code.springsecurity.demo")
public class DemoAppConfig {

    // define a bean for ViewResolver

    @Bean
    public ViewResolver viewResolver() {
        InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
        viewResolver.setPrefix("/WEB-INF/view/");
        viewResolver.setSuffix(".jsp");
        return viewResolver;
    }
}
```

# The Problem

- Tons of configuration

Very error-prone

Easy to make  
a simple mistake

```
<!-- Step 1: Configure Spring MVC Dispatcher Servlet -->
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring-mvc-demo-servlet.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>

<!-- Step 2: Set up URL mapping for Spring MVC Dispatcher Servlet -->
<servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>

<!-- Step 3: Add support for component scanning -->
<context:component-scan base-package="com.luv2code.springsecurity.demo">
<!-- Step 4: Add support for conversion, formatting and localization -->
<mvc:annotation-driven>

<!-- Step 5: Define Spring MVC view resolver -->
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/view/" />
    <property name="suffix" value=".jsp" />

```

```
@Configuration
@EnableWebMvc
@ComponentScan(basePackages="com.luv2code.springsecurity.demo")
public class DemoAppConfig {

    // define a bean for ViewResolver

    @Bean
    public ViewResolver viewResolver() {
        InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
        viewResolver.setPrefix("/WEB-INF/view/");
        viewResolver.setSuffix(".jsp");
        return viewResolver;
    }
}
```

# The Problem

- Tons of configuration

Very error-prone  
Easy to make  
a simple mistake

```
<!-- Step 1: Configure Spring MVC Dispatcher Servlet -->
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherS
        @Configuration
        @EnableWebMvc
        public class WebConfig {
            ...
        }
    
```

There should be  
an easier solution

```

        <bean id="viewResolver"
            class="org.springframework.web.servlet.view.InternalResourceView
            <property name="prefix" value="/WEB-INF/view/" />
            <property name="suffix" value=".jsp" />
        </bean>
    
```

# Spring Boot Solution



# Spring Boot Solution

- Make it easier to get started with Spring development



# Spring Boot Solution

- Make it easier to get started with Spring development
- Minimize the amount of manual configuration



# Spring Boot Solution

- Make it easier to get started with Spring development
- Minimize the amount of manual configuration
  - Perform auto-configuration based on props files and JAR classpath



# Spring Boot Solution

- Make it easier to get started with Spring development
- Minimize the amount of manual configuration
  - Perform auto-configuration based on props files and JAR classpath
  - Help to resolve dependency conflicts (Maven or Gradle)



# Spring Boot Solution

- Make it easier to get started with Spring development
- Minimize the amount of manual configuration
  - Perform auto-configuration based on props files and JAR classpath
  - Help to resolve dependency conflicts (Maven or Gradle)
  - Provide an embedded HTTP server so you can get started quickly



# Spring Boot Solution



- Make it easier to get started with Spring development
- Minimize the amount of manual configuration
  - Perform auto-configuration based on props files and JAR classpath
  - Help to resolve dependency conflicts (Maven or Gradle)
  - Provide an embedded HTTP server so you can get started quickly
    - Tomcat, Jetty, Undertow, ...

# Spring Initializr

# Spring Initializr

<http://start.spring.io>

The screenshot shows the Spring Initializr web application. At the top, it says "SPRING INITIALIZR bootstrap your application now". Below that, there's a main heading "Generate a  with  and Spring Boot .

**Project Metadata**

Artifact coordinates  
Group:   
Artifact:

**Dependencies**

Add Spring Boot Starters and dependencies to your application  
Search for dependencies

**Selected Dependencies**

**Generate Project**

Don't know what to look for? Want more options? [Switch to the full version.](#)

# Spring Initializr

- Quickly create a starter Spring project

<http://start.spring.io>

The screenshot shows the Spring Initializr web application. At the top, it says "SPRING INITIALIZR bootstrap your application now". Below that, there's a main heading "Generate a  with  and Spring Boot ".  
  
On the left, under "Project Metadata", there are fields for "Group" (containing "com.example") and "Artifact" (containing "demo").  
  
On the right, under "Dependencies", there's a "Search for dependencies" input field containing "Web, Security, JPA, Actuator, Devtools...". Below it is a "Selected Dependencies" section with a single item "Selected".  
  
At the bottom right is a green "Generate Project" button.

# Spring Initializr

- Quickly create a starter Spring project
- Select your dependencies

<http://start.spring.io>

The screenshot shows the Spring Initializr interface. At the top, it says "SPRING INITIALIZR bootstrap your application now". Below that, there's a "Generate a" dropdown set to "Maven Project", a "with" dropdown set to "Java", and a "and Spring Boot" dropdown set to "2.0.5".  
  
The "Project Metadata" section has fields for "Group" (set to "com.example") and "Artifact" (set to "demo").  
  
The "Dependencies" section has a "Search for dependencies" input field containing "Web, Security, JPA, Actuator, Devtools...". A "Selected Dependencies" list is empty.  
  
At the bottom right is a green "Generate Project" button.

# Spring Initializr

- Quickly create a starter Spring project
- Select your dependencies
- Creates a Maven/Gradle project

<http://start.spring.io>

The screenshot shows the Spring Initializr web application. At the top, it says "SPRING INITIALIZR bootstrap your application now". Below that, there's a "Generate a" dropdown set to "Maven Project", a "with" dropdown set to "Java", and a "and Spring Boot" dropdown set to "2.0.5".  
  
On the left, under "Project Metadata", there are fields for "Artifact coordinates": "Group" (com.example) and "Artifact" (demo).  
  
On the right, under "Dependencies", there's a search bar with "Web, Security, JPA, Actuator, Devtools..." and a "Selected Dependencies" section containing "Selected Dependencies".  
  
At the bottom center is a green "Generate Project" button.

# Spring Initializr

- Quickly create a starter Spring project
- Select your dependencies
- Creates a Maven/Gradle project
- Import the project into your IDE

<http://start.spring.io>

The screenshot shows the Spring Initializr web application. At the top, it says "SPRING INITIALIZR bootstrap your application now". Below that, there's a header with "Generate a" followed by dropdown menus for "Maven Project", "Java", and "Spring Boot" (set to 2.0.5). To the right of the header is a "Dependencies" section with a search bar containing "Web, Security, JPA, Actuator, Devtools...". In the middle left is a "Project Metadata" section with "Artifact coordinates" fields for "Group" (com.example) and "Artifact" (demo). At the bottom right is a green "Generate Project" button.

SPRING INITIALIZR bootstrap your application now

Generate a  with  and

Project Metadata

Artifact coordinates

Group

Artifact

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Selected Dependencies

Generate Project

Don't know what to look for? Want more options? [Switch to the full version.](#)

# Spring Initializr

- Quickly create a starter Spring project
- Select your dependencies
- Creates a Maven/Gradle project
- Import the project into your IDE
  - Eclipse, IntelliJ, NetBeans etc ...

<http://start.spring.io>

The screenshot shows the Spring Initializr web application. At the top, it says "SPRING INITIALIZR bootstrap your application now". Below that, there's a header with "Generate a" followed by dropdown menus for "Maven Project", "Java", and "Spring Boot" (set to 2.0.5). To the right of this is a "Dependencies" section with a search bar and a list of starters: "Web, Security, JPA, Actuator, Devtools...". The main area is divided into "Project Metadata" and "Dependencies". In "Project Metadata", there are fields for "Artifact coordinates" (Group: com.example, Artifact: demo), and a "Selected Dependencies" section. At the bottom is a green "Generate Project" button.

# Spring Boot Embedded Server

# Spring Boot Embedded Server

- Provide an embedded HTTP server so you can get started quickly

# Spring Boot Embedded Server

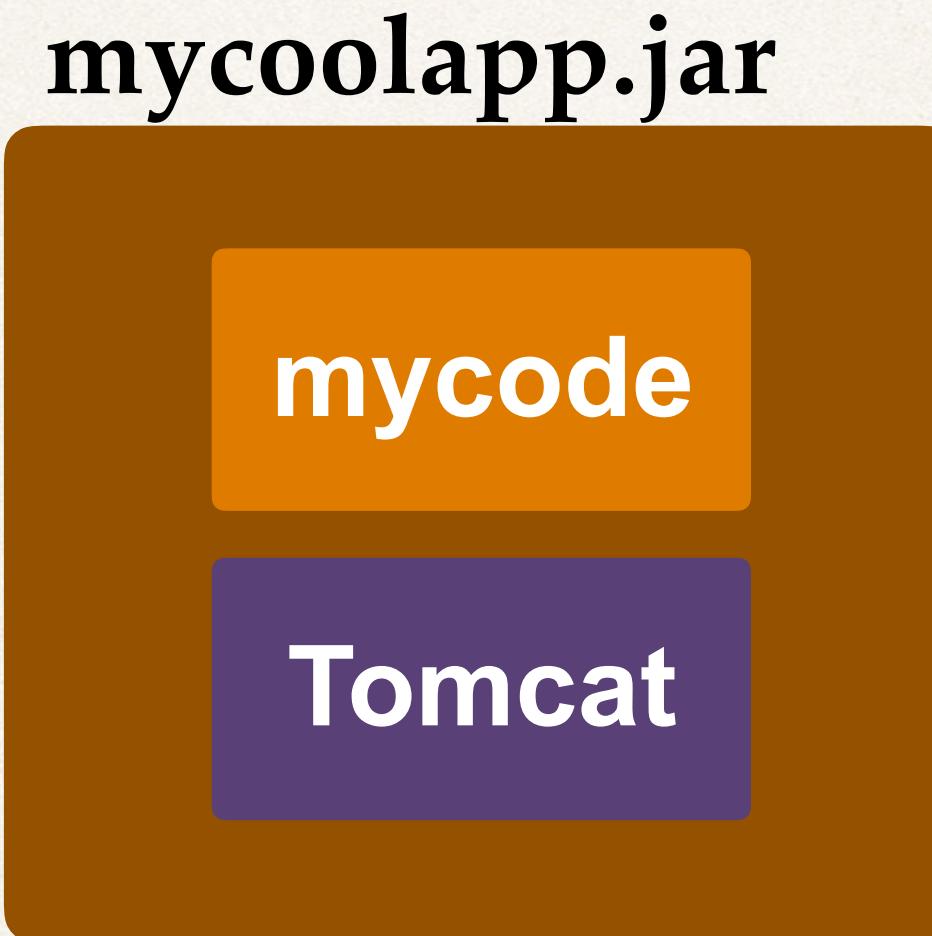
- Provide an embedded HTTP server so you can get started quickly
  - Tomcat, Jetty, Undertow, ...

# Spring Boot Embedded Server

- Provide an embedded HTTP server so you can get started quickly
  - Tomcat, Jetty, Undertow, ...
- No need to install a server separately

# Spring Boot Embedded Server

- Provide an embedded HTTP server so you can get started quickly
  - Tomcat, Jetty, Undertow, ...
- No need to install a server separately



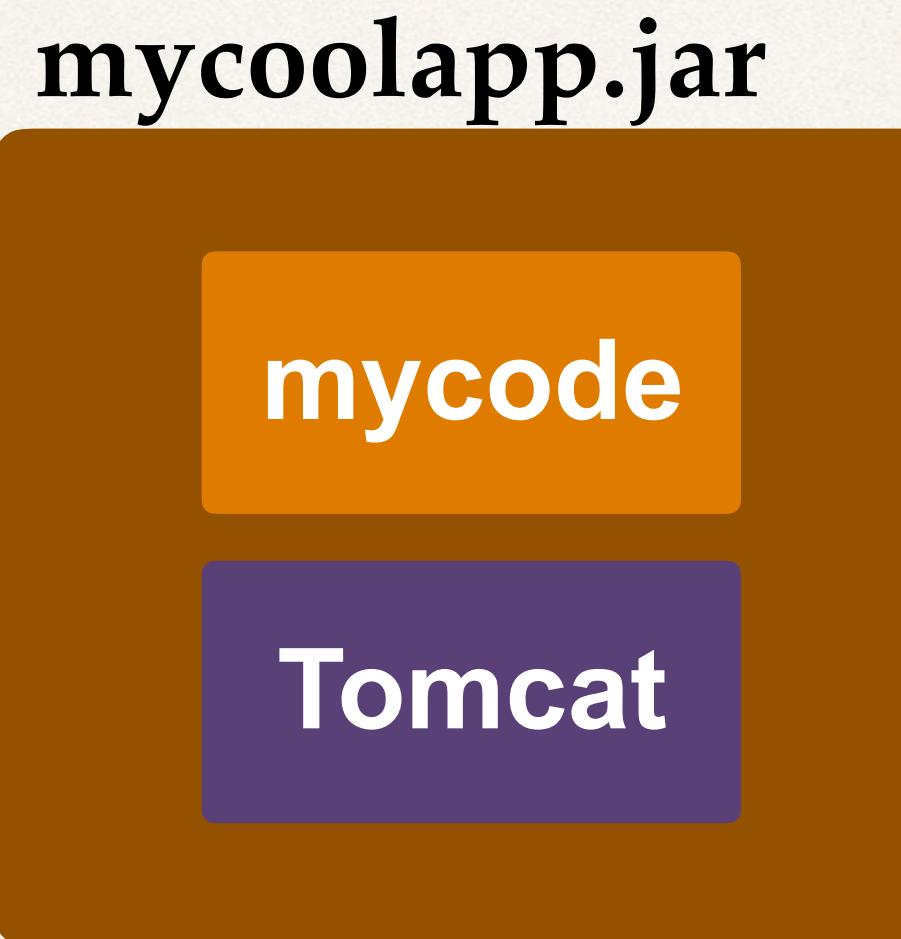
# Spring Boot Embedded Server

- Provide an embedded HTTP server so you can get started quickly
  - Tomcat, Jetty, Undertow, ...
- No need to install a server separately



# Spring Boot Embedded Server

- Provide an embedded HTTP server so you can get started quickly
  - Tomcat, Jetty, Undertow, ...
- No need to install a server separately



Self-contained unit  
Nothing else to install

# Running Spring Boot Apps

# Running Spring Boot Apps

- Spring Boot apps can be run standalone (includes embedded server)

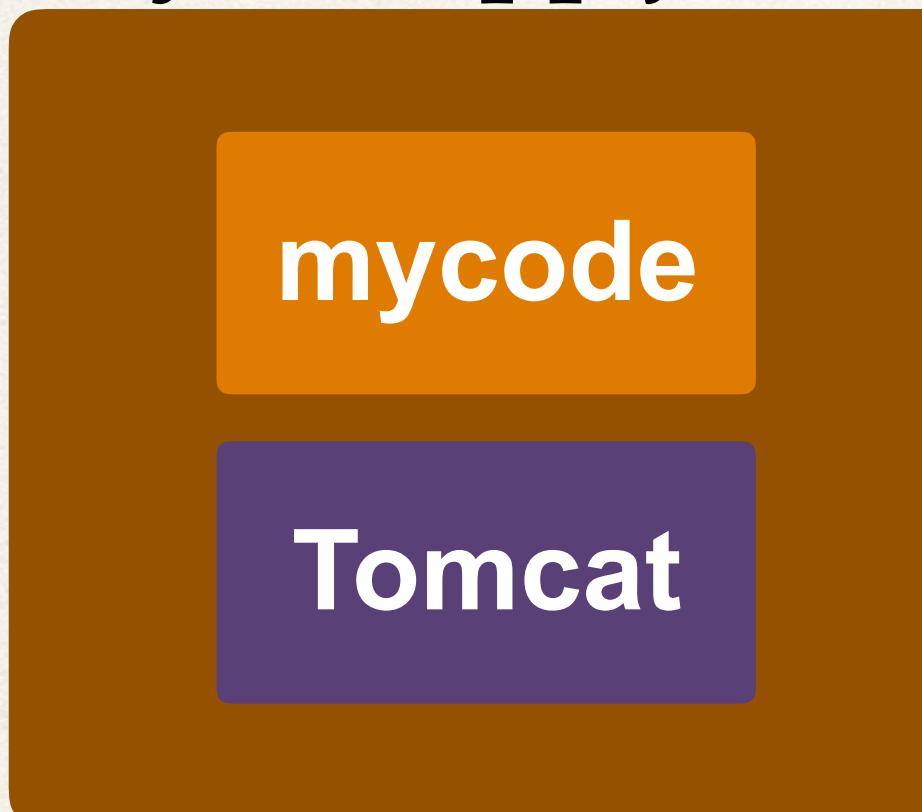
# Running Spring Boot Apps

- Spring Boot apps can be run standalone (includes embedded server)
- Run the Spring Boot app from the IDE or command-line

# Running Spring Boot Apps

- Spring Boot apps can be run standalone (includes embedded server)
- Run the Spring Boot app from the IDE or command-line

**mycoolapp.jar**



# Running Spring Boot Apps

- Spring Boot apps can be run standalone (includes embedded server)
- Run the Spring Boot app from the IDE or command-line

**mycoolapp.jar**

**mycode**

**Tomcat**

```
> java -jar mycoolapp.jar
```

**Name of our JAR file**

# Deploying Spring Boot Apps

# Deploying Spring Boot Apps

- Spring Boot apps can also be deployed in the traditional way

# Deploying Spring Boot Apps

- Spring Boot apps can also be deployed in the traditional way
- Deploy **WAR file** to an external server: Tomcat, JBoss, WebSphere etc ...

# Deploying Spring Boot Apps

- Spring Boot apps can also be deployed in the traditional way
- Deploy **WAR file** to an external server: Tomcat, JBoss, WebSphere etc ...

Tomcat

# Deploying Spring Boot Apps

- Spring Boot apps can also be deployed in the traditional way
- Deploy **WAR file** to an external server: Tomcat, JBoss, WebSphere etc ...



# Deploying Spring Boot Apps

- Spring Boot apps can also be deployed in the traditional way
- Deploy **WAR file** to an external server: Tomcat, JBoss, WebSphere etc ...



# Deploying Spring Boot Apps

- Spring Boot apps can also be deployed in the traditional way
- Deploy **WAR file** to an external server: Tomcat, JBoss, WebSphere etc ...



# Spring Boot FAQ #1

# Spring Boot FAQ #1

Q: Does Spring Boot replace Spring MVC, Spring REST etc ...?

# Spring Boot FAQ #1

Q: Does Spring Boot replace Spring MVC, Spring REST etc ...?

- No. Instead, Spring Boot actually uses those technologies

# Spring Boot FAQ #1

Q: Does Spring Boot replace Spring MVC, Spring REST etc ...?

- No. Instead, Spring Boot actually uses those technologies

Spring Core

Spring AOP

Spring ...

# Spring Boot FAQ #1

Q: Does Spring Boot replace Spring MVC, Spring REST etc ...?

- No. Instead, Spring Boot actually uses those technologies

Spring MVC

Spring REST

Spring ...

Spring Core

Spring AOP

Spring ...

# Spring Boot FAQ #1

Q: Does Spring Boot replace Spring MVC, Spring REST etc ...?

- No. Instead, Spring Boot actually uses those technologies

Spring Boot

Spring MVC

Spring REST

Spring ...

Spring Core

Spring AOP

Spring ...

# Spring Boot FAQ #1

Q: Does Spring Boot replace Spring MVC, Spring REST etc ...?

- No. Instead, Spring Boot actually uses those technologies



Once you do **Spring Boot configs**  
then you make use of  
regular **Spring coding**

@Component  
@Controller  
@Autowired  
etc...

# Spring Boot FAQ #2

# Spring Boot FAQ #2

Q: Does Spring Boot run code faster than regular Spring code?

# Spring Boot FAQ #2

Q: Does Spring Boot run code faster than regular Spring code?

- No.
- Behind the scenes, Spring Boot uses same code of Spring Framework

# Spring Boot FAQ #2

Q: Does Spring Boot run code faster than regular Spring code?

- No.
- Behind the scenes, Spring Boot uses same code of Spring Framework
- Remember, Spring Boot is about making it easier to get started

# Spring Boot FAQ #2

Q: Does Spring Boot run code faster than regular Spring code?

- No.
- Behind the scenes, Spring Boot uses same code of Spring Framework
- Remember, Spring Boot is about making it easier to get started
  - Minimizing configuration etc ...

# Spring Boot FAQ #3

# Spring Boot FAQ #3

Q: Do I need a special IDE for Spring Boot?

# Spring Boot FAQ #3

Q: Do I need a special IDE for Spring Boot?

- No.
- You can use any IDE for Spring Boot apps ... even use plain text editor

# Spring Boot FAQ #3

Q: Do I need a special IDE for Spring Boot?

- No.
- You can use any IDE for Spring Boot apps ... even use plain text editor
- The Spring team provides free *Spring Tool Suite (STS)* [IDE plugins]

# Spring Boot FAQ #3

Q: Do I need a special IDE for Spring Boot?

- No.
- You can use any IDE for Spring Boot apps ... even use plain text editor
- The Spring team provides free *Spring Tool Suite (STS)* [IDE plugins]
- Some IDEs provide fancy Spring tooling support

# Spring Boot FAQ #3

Q: Do I need a special IDE for Spring Boot?

- No.
- You can use any IDE for Spring Boot apps ... even use plain text editor
- The Spring team provides free *Spring Tool Suite (STS)* [IDE plugins]
- Some IDEs provide fancy Spring tooling support
- Not a requirement. Feel free to use the IDE that works best for you 😊