

13:51

Home Insert Draw View Tell me

Text Lasso Insert Eraser Pen Marker Highlighter Ink Colour 0.25 mm 0.35 mm 0.5 mm 0.7 mm 1 mm

Sync Error We're having trouble syncing. Click "Sign in again" to fix this issue. Sign in again

Payment class is creating an instance of User class, and there is one Major problems with this and i.e.

Tight coupling: Now payment class is tightly coupled with User class.

How?

-> Suppose I want to write Unit test cases for Payment "getSenderDetails()" method, but now I can not easily MOCK "User" object, as Payment class is creating new object of User, so it will invoke the method of User class too.

-> Suppose in future, we have different types of User like "admin", "Member" etc., then with this logic, I can not change the user dynamically.

Now, Lets see an example **with** Dependency Injection:

tight coupling below hard for unit testing

14:06

Home Insert Draw View Tell me

Text Lasso Insert Eraser Pen Marker Highlighter Ink Colour 0.25 mm 0.35 mm 0.5 mm 0.7 mm 1 mm

Sync Error We're having trouble syncing. Click "Sign in again" to fix this issue. Sign in again

Lets see an example **without** Dependency Injection:

```

public class Payment {
    User sender = new User();

    void getSenderDetails(String userID){
        sender.getUserDetails(userID);
    }
}
    
```

```

public class User {
    public void getUserDetails(String id) {
        //do something
    }
}
    
```

spring makes it easier by loose coupling using @autowired annotation unit testing now easy

15:38

The screenshot shows a digital whiteboard interface with a toolbar at the top. The main content area has a dark background. At the top, there is a yellow banner with a "Sync Error" message. Below this, the text "Now, Lets see an example with Dependency Injection:" is written in white, with "with" circled in white. Two code snippets are displayed in white boxes. The first snippet is for a class named "Payment" and the second is for a class named "User".

```
@Component
public class Payment {

    @Autowired
    User sender;

    void getSenderDetails(String userID){
        sender.getUserDetails(userID);
    }
}
```

```
@Component
public class User {

    public void getUserDetails(String id) {
        //do something
    }
}
```

@component create entire object of class by spring

@autowired create object of dependency by spring and include here

16:38

The screenshot shows the same digital whiteboard interface. The code snippets from the previous image are still present. In the "Payment" class snippet, the "@Autowired" annotation and the "sender" variable are highlighted with a blue bracket. In the "User" class snippet, the "@Component" annotation is highlighted with a blue bracket. Below the code snippets, there is a text explanation of the annotations.

```
@Autowired
User sender;

void getSenderDetails(String userID){
    sender.getUserDetails(userID);
}
```

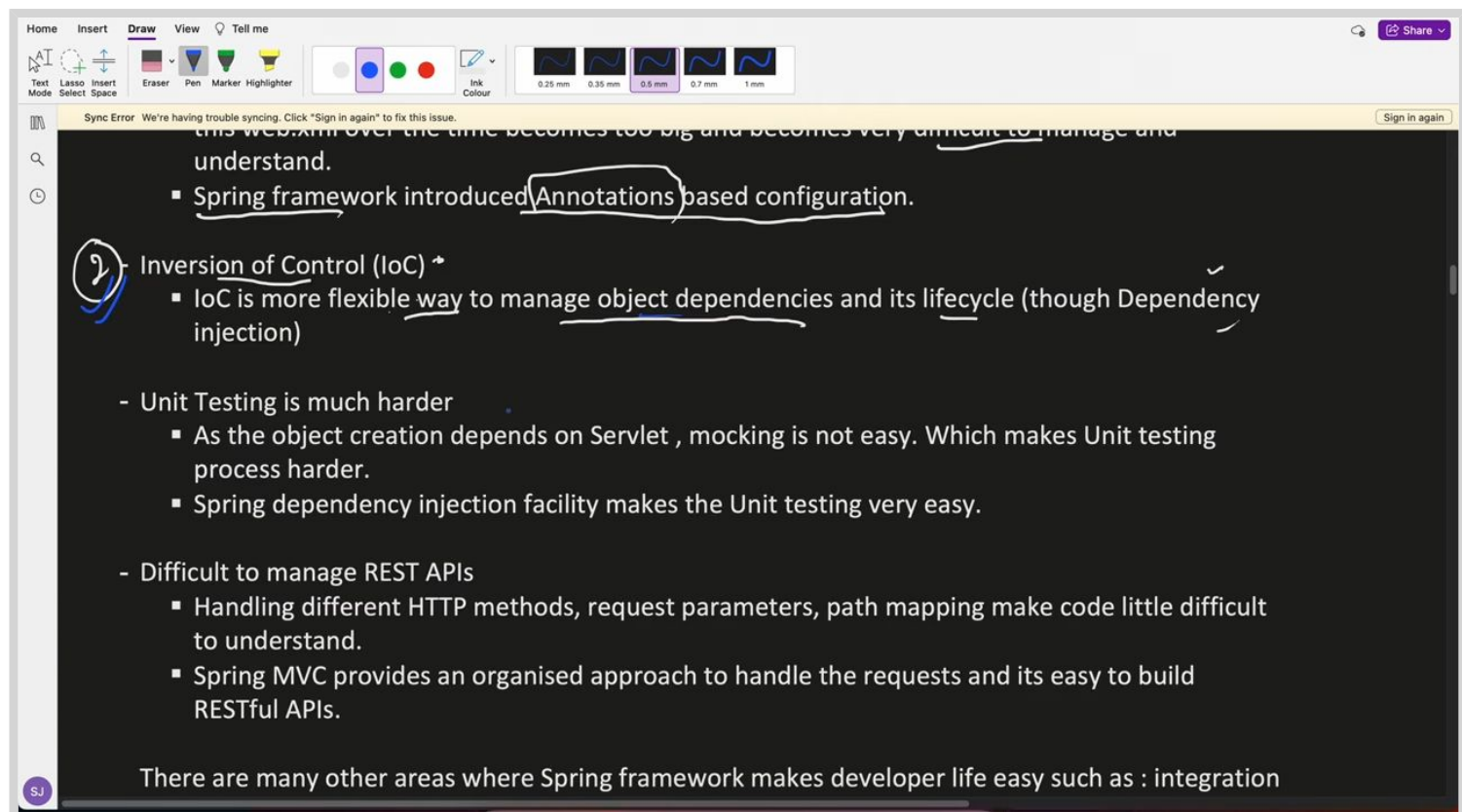
```
@Component
public class User {

    public void getUserDetails(String id) {
        //do something
    }
}
```

@Component: tells Spring that, you have to manage this class or bean.
@Autowired: tells Spring to resolve and add this object dependency.

Rest api become easier with spring

18:05



Sync Error We're having trouble syncing. Click "Sign in again" to fix this issue.

this WEB.XML over the time becomes too big and becomes very difficult to manage and understand.

- Spring framework introduced Annotations based configuration.

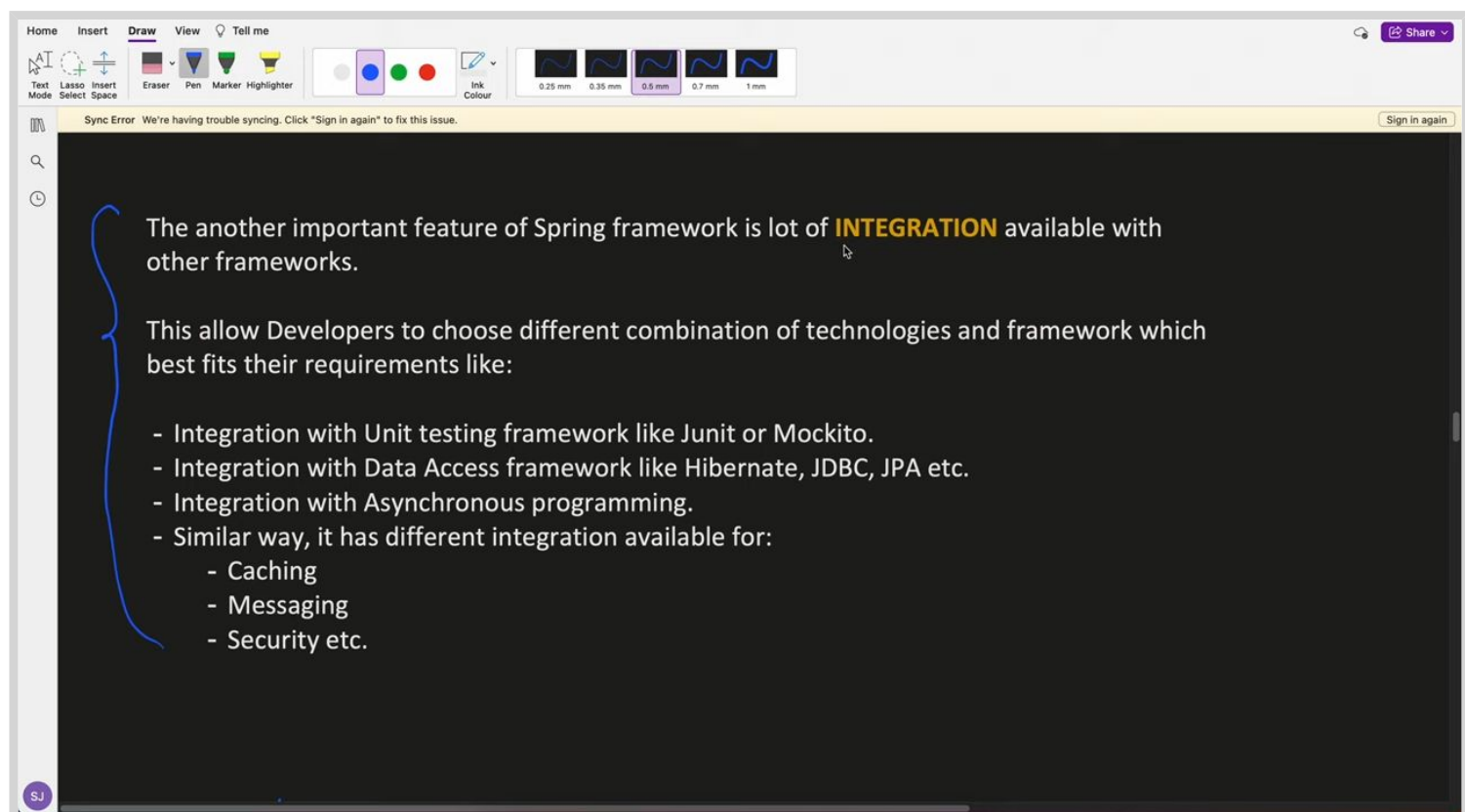
② Inversion of Control (IoC) →

- IoC is more flexible way to manage object dependencies and its lifecycle (though Dependency injection)

- Unit Testing is much harder
 - As the object creation depends on Servlet , mocking is not easy. Which makes Unit testing process harder.
 - Spring dependency injection facility makes the Unit testing very easy.
- Difficult to manage REST APIs
 - Handling different HTTP methods, request parameters, path mapping make code little difficult to understand.
 - Spring MVC provides an organised approach to handle the requests and its easy to build RESTful APIs.

There are many other areas where Spring framework makes developer life easy such as : integration

20:40



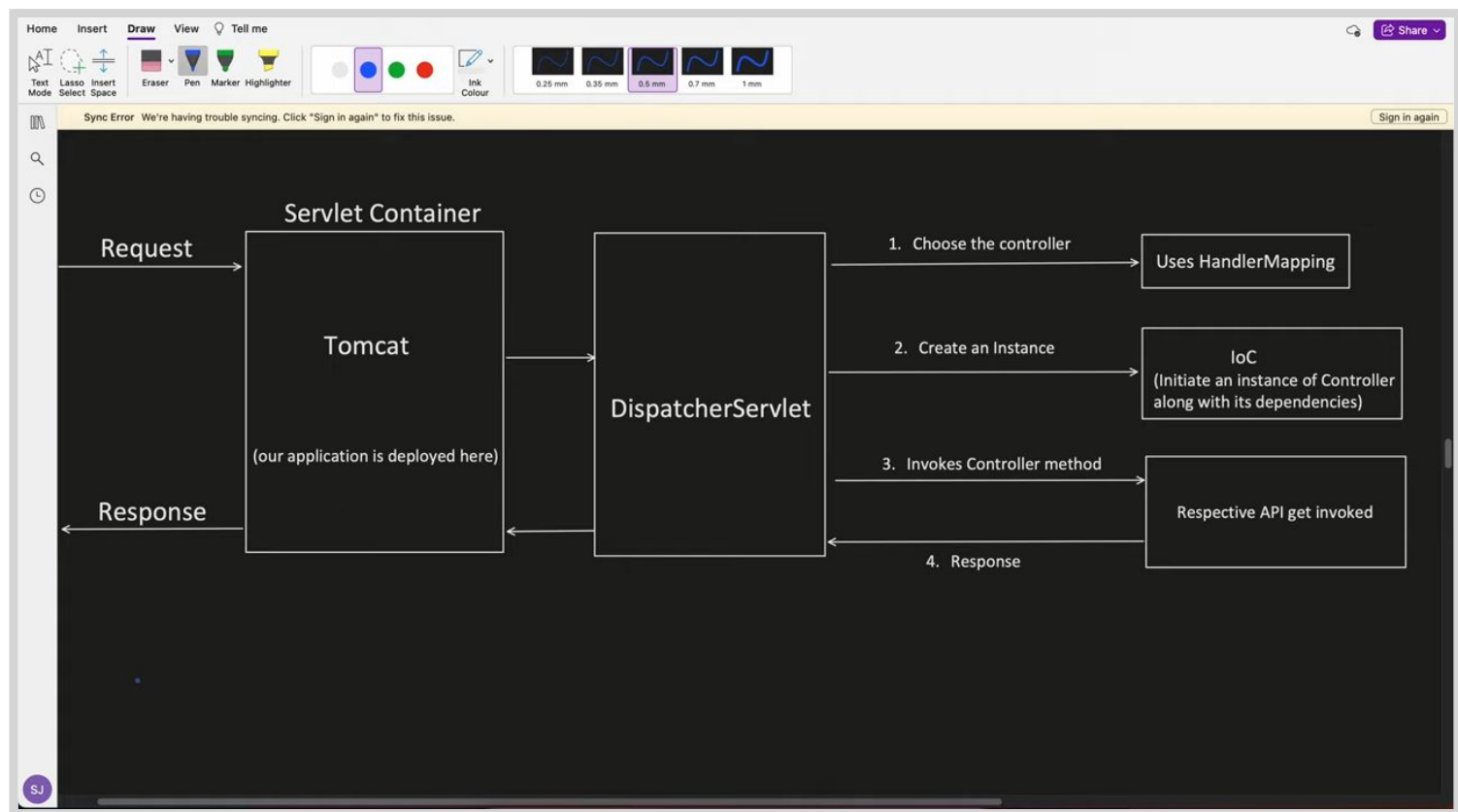
Sync Error We're having trouble syncing. Click "Sign in again" to fix this issue.

The another important feature of Spring framework is lot of **INTEGRATION** available with other frameworks.

This allow Developers to choose different combination of technologies and framework which best fits their requirements like:

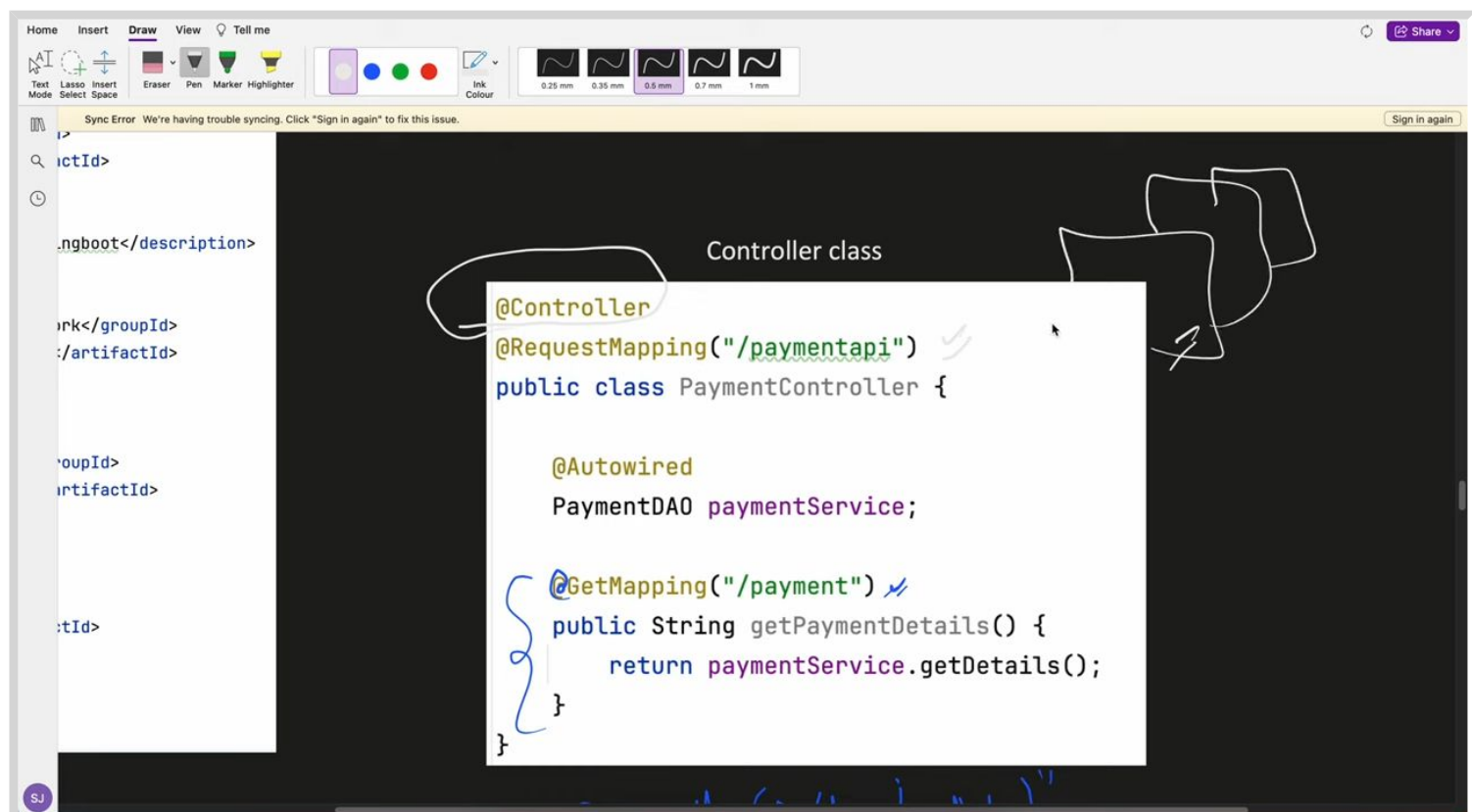
- Integration with Unit testing framework like Junit or Mockito.
- Integration with Data Access framework like Hibernate, JDBC, JPA etc.
- Integration with Asynchronous programming.
- Similar way, it has different integration available for:
 - Caching
 - Messaging
 - Security etc.

21:20



when it comes to ioc which api end point first spring creates all of it required dependencies using ioc controller class this is where request comes int

22:58



in spring we add all the dependencies along with its version in pom.xml

30:25

The screenshot shows a digital drawing application interface. On the left, a code editor displays a `pom.xml` file. The `<dependencies>` section is highlighted with a blue bracket and a clock icon. The dependencies listed are:

- `org.springframework` `spring-webmvc` `6.1.4`
- `javax.servlet` `servlet-api` `2.5`
- `junit` `junit` `4.13.2` `test` scope

On the right, a Java class named `PaymentController` is shown, annotated with `@Controller`, `@RequestMapping("/paymentapi")`, and `@GetMapping("/payment")`. It contains a `getPaymentDetails()` method that returns `paymentService.getDetail`.

config file where all things required to run the application lies here

@configuration says this is configuration class

@EnableWebMvc give all the requirements to run spring application

@Componentscan create beans of the dependencies and manage with in that package com.con....

30:55

The screenshot shows a digital drawing application interface. On the left, a code editor displays a Java class named `AppConfig`, annotated with `@Configuration`, `@EnableWebMvc`, and `@ComponentScan(basePackages = "com.conceptandcoding")`. The class contains a `public class AppConfig` with a comment `// add configuration here if required`.

Dispatcher Servlet class

```
public class MyApplicationInitializer extends  
AbstractAnnotationConfigDispatcherServletInitializer {
```

dispatcher servlet class this class helps to map all the routes with methods inside spring

includes appConfig.class for configuration

when ever request comes at "/" this runs first

32:40

The screenshot shows a digital whiteboard interface with a toolbar at the top. The main content area displays the following Java code for the Dispatcher Servlet class:

```
public class MyApplicationInitializer extends  
AbstractAnnotationConfigDispatcherServletInitializer {  
  
    @Override  
    protected Class<?>[] getRootConfigClasses() {  
        return null;  
    }  
  
    @Override  
    protected Class<?>[] getServletConfigClasses() {  
        return new Class[] { AppConfig.class };  
    }  
  
    @Override  
    protected String[] getServletMappings() {  
        return new String[] { "/" };  
    }  
}
```

we put our code in tomcat server as war file

springboot

1st advantage dependency management of version no need to give version spring boot manages it using springboot starter

34:13

The screenshot shows a digital whiteboard interface with a toolbar at the top. The main content area displays the text:

Spring Boot, solve challenges which exists with Spring MVC.

1. **Dependency Management:** No need for adding different dependencies separately and also th compatible version headache.

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>3.2.3</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>com.conceptandcoding</groupId>
<artifactId>learningspringboot</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>springboot application</name>
<description>project for learning springboot</description>
<properties>
```

like this

35:19

Sync Error We're having trouble syncing. Click "Sign in again" to fix this issue.

```
<artifactId>learningspringboot</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>springboot application</name>
<description>project for learning springboot</description>
<properties>
  <java.version>17</java.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Auto configuration: all the spring mvc conguration explicitly @springbootApplication includes with in it is opionated bcz by default it add need configuration we can customize it it is called auto configured bcoz it added configuration automatically

37:11

Sync Error We're having trouble syncing. Click "Sign in again" to fix this issue. Sign in again

2. Auto Configuration : No need for separately configuring "DispatcherServlet", "AppConfig", "EnableWebMvc", "ComponentScan". Spring boot add internally by-default.

```
@SpringBootApplication
public class SpringbootApplication {

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

it has embedded server(tomcat) for spring mvc we dont have we have add war file in tomcat server but here springboot application adds within

40:46

Home Insert Draw View Tell me Share

Sync Error We're having trouble syncing. Click "Sign in again" to fix this issue. Sign in again

3. Embedded Server:

In traditional Spring MVC application, we need to build a WAR file, which is a packaged file containing your application's classes, JSP pages, configuration files, and dependencies. Then we need to deploy this WAR file to a servlet container like Tomcat.

But in Spring boot, Servlet container is already embedded, we don't have to do all this stuff. Just run the application, that's all.

41:51

HomeInsertDrawViewTell me

Text ModeLasso SelectInsert Space

EraserPenMarkerHighlighter

Ink Colour

0.25 mm0.35 mm0.5 mm0.7 mm1 mm

Share

Sync Error We're having trouble syncing. Click "Sign in again" to fix this issue.

Sign in again

So, what is Spring boot?

- It provides a quick way to create a production ready application.

- It is based on Spring framework.

- It support **"Convention over Configuration"**.
Use default values for configuration, and if developer don't want to go with convention(the way something is done), they can override it.

- It also help to run an application as quick as possible.

SJ

