infogain

SPRING BOOT

Part1

Objective

infogain
www.infogain.com

- Introduction to Spring Boot
- What are the Primary Goals of Spring Boot?
- Ways of Creating Spring Boot Application?
- What is Spring Boot Starter?
- Create Rest Full application using Spring Boot
- Create Spring MVC application using Spring Boot

What is Spring Boot?

- It is not a framework. it is a tool that allows you to create spring based application within no time.
- it helps you to build ,package and deploy the spring application with minimal or absolutely no configurations.
- It provides a set of Starter Pom's which one can use to add required dependencies and also facilitate auto configuration.

Spring Boot Primary Goals

- Allows you to focus more on business features and less on infrastructure.
- Spring Boot dynamically wires up beans and settings and applies them to your application context on startup of your application.
- To avoid XML Configuration completely
- To avoid defining more Annotation Configuration(It combined some existing Spring Framework Annotations to a simple and single Annotation)
- To avoid writing lots of import statements
- To provide some defaults to quick start new projects within no time.

Different ways of Creating Spring Boot Application

- Creating a Spring Boot App
  - Using Spring Boot CLI Tool
  - Using Spring STS IDE
  - Using Spring Initializr (Website http://start.spring.io/)

Dependency Management

- Each release of Spring Boot provides a curated list of dependencies it supports.
- In practice, you do not need to provide a version for any of these dependencies in your build configuration as Spring Boot is managing that for you.
- When you upgrade Spring Boot itself, these dependencies will be upgraded as well in a consistent way.

- Note
- Each release of Spring Boot is associated with a base version of the Spring Framework so it is highly recommend you to not specify its version on your own.

6

Dependency Management (contd..)

- Maven users can inherit from the spring-boot-starter-parent project to obtain sensible defaults.
- It declare our project as child project of **spring starter-parent project**. The idea is that configuration defined in the
- parent project is inherited to the child project
- **The parent project provides the following features:**
  - Java 1.8 as the default compiler level.
  - UTF-8 source encoding.
  - A Dependency Management section, inherited from the **spring-boot-dependencies pom**, that manages the versions of common dependencies
  - Sensible resource filtering for application.properties and application.yml.

Dependency Management (contd..)

infogain
www.infogain.com

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.3.0.RELEASE</version>
</parent>
```

Spring Boot Dependency Management

BOM - bill of materials

Example - spring-core 4.2.3 works well with logback-core 1.1.3

Spring Boot Starter

- **Spring Boot Starter**
  - Spring Boot starters are jar dependencies which provide a quick way to configure your application, without manually including a lot of related dependencies.
  - It contain a lot of the dependencies that you need to get a project up and running quickly.
  - The starter POMs are convenient dependency descriptors that can be added to your application's Maven.
  - You get a one-stop-shop for all the Spring and related technology that you need, without having to hunt through sample code and copy paste the jar's.

List of some Important Spring Starter POM

Name	Description
spring-boot-starter	Core starter, including auto-configuration support, logging
spring-boot-starter-test	Starter for testing Spring Boot applications with libraries including JUnit, Hamcrest and Mockito
spring-boot-starter-social-twitter	Starter for using Spring Social Twitter
spring-boot-starter-data-jpa	Starter for using Spring Data JPA with Hibernate
spring-boot-starter-web	Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container
spring-boot-starter-social-facebook	Starter for using Spring Social Facebook

How Does Spring Boot Work?

Java

Main method entry point

Spring Application

Spring context

Spring environment

Initializers

Embedded Server

Default is Tomcat

Auto configured

Spring Boot Working...


```
public static void main( … )


@SpringBootApplication



@Configuration
@EnableAutoConfiguration
@ComponentScan


SpringApplication.run( … );
```

- Starts Java and then the application
- A convenience annotation that wraps commonly used annotations with Spring Boot
- Spring configuration on startup
- Auto configures frameworks
- Scans project for Spring components
- Starts Spring, creates spring context, applies annotations and sets up container

SpringApplication

- The SpringApplication class provides a convenient way to bootstrap a Spring application that will be started from a main() method.
- It Create an appropriate ApplicationContext instance (depending on your classpath)

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

Spring Boot Working…

- **@SpringBootApplication: Adds all of the following:**
  - **@Configuration :** Tags the class as a source of bean definitions for the application context.
  - **@EnableAutoConfiguration :** Tells Spring Boot to start adding beans based on class path settings, other beans, and various property settings.
  - **@EnableWebMvc** : For a Spring MVC app, but Spring Boot adds it automatically when it sees spring-webmvc on the class path. This flags the application as a web application and activates key behaviors such as setting up a DispatcherServlet.
  - **@ComponentScan :** Tells Spring to look for other components, configurations, and services in the specified package allowing it to find the controllers.

Starting Spring BOOT

- **SpringApplication.run(App.class,args);**
  - This is a magical line which takes two argument one is the class name annoted with @SpringBootAppliction and another is command line argument.

- **What @SpringBootAppliction does?**
  - Sets up the default configuration
  - Starts Spring application context
  - Performs class path scan
  - Starts the Tomcat server

Create a RestController

infogain
www.infogain.com

- **@RestController**
- public class HelloController {
- @RequestMapping("/")
- public String greeting() {
- return "Greetings from Spring Boot!";
- }
- }

Create a simple web application

- **@RestController :**
  - Indicates that class act as Spring MVC Rest controller and ready to handle web requests.
  - Combines @Controller and @ResponseBody, two annotations that results in web requests returning data rather than a view.
  - **@RestController = @Controller + @ResponseBody,**

- **@RequestMapping :**
  - Annotation is used to map web requests onto specific request handler classes and/or handler methods.

Create an Application class


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

Spring MVC Using SpringBoot

Create a MVC Controller


```
@Controller
public class HelloController {
 @RequestMapping("/view")
    public ModelAndView  greet()
        {
        return new ModelAndView("greeting", "msg", "Welcome to Spring Boot");
            }
         }
```

- **@Controller :** Indicates that class act as Spring MVC controller and ready to handle web requests.
- **ModelAndView**: Encapsulate model and view information in a single object

Overriding Template Path and Template Technology

- By default Spring Boot use themleaf as template which is placed inside **src/main/resources/templates** folder
- **Overriding Default Setting First Way**
  - Adding following entry inside **application.properties** file
  - spring.mvc.view.prefix: /WEB-INF/jsp/
  - spring.mvc.view.suffix: .jsp
- **Second Way**
  - By declaring bean of InternalResourceViewResolver in our Application.java config
  - @Bean
  - public InternalResourceViewResolver internalResourceViewResolver() {
  - InternalResourceViewResolver resolver = new InternalResourceViewResolver();
  - resolver.setPrefix("WEB-INF/jsp/");
  - resolver.setSuffix(".jsp");
  - resolver.setViewClass(JstlView.class);
  - return resolver;
  - }

Add following dependencies inside pom.xml file

infogain
www.infogain.com

- <!-- JSTL -->
- <dependency>
- <groupId>javax.servlet</groupId>
- <artifactId>jstl</artifactId>
- </dependency>

- <!-- Need this to compile JSP -->
- <dependency>
- <groupId>org.apache.tomcat.embed</groupId>
- <artifactId>tomcat-embed-jasper</artifactId>
- </dependency>

greeting.jsp

- **greeting.jsp**
- <html>
- <body>
- ${msg}
- </body>
- </html>

infogain

Thank You