**Introduction to Spring Boot Framework**

Use Cases Manual

Sandbox Link [Spring Boot](https://share.percipio.com/cd/J676mja_Z)

Application Configuration with Spring Boot

This guide will introduce you to the various application configuration options that are present in Spring Boot.

**You will learn**

* What is application configuration?
* Why do we need application configuration?
* What is application.properties?
* What are some of the important things that can customized in application.properties?
* How do you externalize configuration using Spring Boot?
* How can you add custom application properties using Spring Boot?

**Tools you will need**

* Maven 3.0+ is your build tool
* Your favorite IDE. We use Eclipse.
* JDK 1.8+

Need for Application Configuration

**Configuration for applications vary from one environment to another**

* You would want to connect to a different database or queues
* You would want to connect with different services
* You would want to configure less logging in production
* You might want to have different custom configuration

**Application Configuration with Spring Boot application.properties**

Spring Boot allows you to configure your application configuration using a file named **application.properties**

application.properties can reside anywhere in the classpath of the application.

You can also configure the port to run the server on using server.port

server.port = 9080

In application.properties, we can use the “logging.level” prefix to set logging levels.

logging.level.some.package.path=DEBUG

logging.level.some.other.package.path=ERROR

**Application Configuration using @Value**

Let’s create a very simple example.

@RestController

public class WelcomeResource {

@Value("${welcome.message}")

private String welcomeMessage;

@GetMapping("/welcome")

public String retrieveWelcomeMessage() {

// Complex Method

return welcomeMessage;

}

}

Notes

**@Value("${welcome.message}") private String welcomeMessage;** - Pick up the value for welcome.message from application configuration and assign it to this member variable.

**@GetMapping("/welcome") public String retrieveWelcomeMessage**() - Expose a simple service using the configuration

Let’s configure a value for welcome message in the application.properties.

welcome.message=Welcome message from property file! Welcome to Optum

When you launch <http://localhost:8080/welcome> , you would see a page with this message

Welcome message from property file! Welcome to Optum

We can make the application.properties more creative by using other properties as variables too.

app.name=Optum

welcome.message=Welcome message from property file! Welcome to ${app.name}

**Application Configuration using Type-safe Configuration Properties**

The problem with @Value is that you would have your configuration values distributed throughout your application. A better option would be to have a centralized approach.

You can define a configuration component using @ConfigurationProperties.

@Component

@ConfigurationProperties("basic")

public class BasicConfiguration {

private boolean value;

private String message;

private int number;

Notes:

**@Component @ConfigurationProperties("basic")** - Defines that this java bean contains configuration properties. All property names will start with basic..

**private boolean value** - A boolean value. Can be configured as basic.value.

**private String message** - A String value. Can be configured as basic.message.

We would need to use the BasicConfiguration in a service to expose the values

@Autowired

private BasicConfiguration configuration;

@RequestMapping("/dynamic-configuration")

public Map dynamicConfiguration() {

// Not the best practice to use a map to store differnt types!

Map map = new HashMap();

map.put("message", configuration.getMessage());

map.put("number", configuration.getNumber());

map.put("key", configuration.isValue());

return map;

}

Notes

**@Autowired private BasicConfiguration configuration** - Its very easy to use BasicConfiguration. Autowire it in when ever you need the value for a property from it.

**@RequestMapping("/dynamic-configuration") public Map dynamicConfiguration() {** - Define a simple service to expose the configured values.

The values can be configured in application.properties

basic.value: true

basic.message: Dynamic Message

basic.number: 100

When you browse to <http://localhost:8080/dynamic-configuration>

, you see the following response.

{"number":200,"message":"Dynamic Message","key":true}