**Application Frameworks**

**Lab Session 7 - Java: Maven and Spring Boot**

**Objective**: Teach the configurations on Java and Maven and build a simple Java application with maven

1. Checking Java version and configure the environment variables.

Check existing java version:

**javac -version**

If this is not working even though Java is installed, you need to configure the following environment variables.

**JAVA\_HOME**: Path to the parent directory of the Java installation.

**PATH**: Path to the ‘bin’ directory of the Java installation.

1. Setup maven and configure the environment variables.

Maven is distributed as a zip file and can be configured at any place.

<http://redrockdigimark.com/apachemirror/maven/maven-3/3.3.9/binaries/apache-maven-3.3.9-bin.zip>

Configuring maven:

Following environment variables should be configured for working with maven.

**MAVEN\_HOME** or **M2\_HOME**: Path to the home directory of maven (root folder of extracted zip file).

**PATH**: Path to the **‘bin’** directory of the extracted location.

Note: This require the Java environment configurations to be configured as a prerequisite.

Check whether maven is working:

**mvn -v**

1. Create a simple maven project with “quickstart” Maven Artifact.

<<Remove>>

mvn archetype:generate -DarchetypeArtifactId=maven-archetype-quickstart -DgroupId=com.mycompany.app -DartifactId=my-app

1. Setting the manifest file for **.jar**

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-jar-plugin</artifactId>

<configuration>

<archive>

<manifest>

<addClasspath>true</addClasspath>

<mainClass>com.mycompany.app.App</mainClass>

</manifest>

</archive>

</configuration>

</plugin>

</plugins>

</build>

1. Implement and run a “Hello World” project with maven.

Use the main method of the application in the App.java file and write a simple print line statement.

System.out.println(“Hello Maven”);

Execute the following command in command line to compile and package the application.

**mvn clean compile package**

Execute the following command to run the application using the **jar** file.

**java -jar <project\_name>.jar** (In the example the project\_name is **my-app**)

1. Setting up Spring-Boot Project

* For setting up spring-boot projects main manifest setting is not needed and should be removed
* Setting the parent POM

<parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>1.5.2.RELEASE</version>  
 </parent>

* Setting spring-boot-starter-web dependency.

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>

* Setting spring boot maven build plugin for support features

<build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>

1. Create a model class named Greeting with **id** and **content** fields in the model package.

<<Remove>>

public class Greeting {  
 private final long id;  
 private final String content;  
 public Greeting(long id, String content) {  
 this.id = id;  
 this.content = content;  
 }  
 public long getId() {  
 return id;  
 }  
 public String getContent() {  
 return content;  
 }  
}

1. Modify the main class to become a Spring Boot application.

<<Remove>>

import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class Application {  
  
 public static void main(String[] args) {  
 SpringApplication.run(Application.class, args);  
 }  
}

1. Create a GET method to respond with Greeting model. Id field should contain a random or an incremental value. (Use POSTMAN to test the implemented method)
2. Update the method to accept a query parameter “name” and response with the content “Hello, <<name provided>>”. Set a default value also when the request doesn’t contain the name field.

<<Remove>>

import java.util.concurrent.atomic.AtomicLong;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class GreetingController {  
  
 private static final String template = "Hello, %s!";  
 private final AtomicLong counter = new AtomicLong();  
  
 @RequestMapping("/greeting")  
 public Greeting greeting(@RequestParam(value="name", defaultValue="World") String name) {  
 return new Greeting(counter.incrementAndGet(),  
 String.format(template, name));  
 }  
}

1. Add a service class named GreetingServiceImpl and use the service to generate the message and call to the class through an interface in the controller.

<<Remove All>>

***// GreetingService Interface***

**import** com.mycompany.app.model.Greeting;

**public interface** GreetingService {

Greeting generateMessage(String name);

}

***// GreetingServiceImpl Class***

**import** java.util.concurrent.atomic.AtomicLong;

**import** org.springframework.stereotype.Service;

**import** com.mycompany.app.model.Greeting;

**import** com.mycompany.app.service.GreetingService;

@Service

**public class** GreetingServiceImpl **implements** GreetingService {

**private static final** String ***template*** = **"Hello, %s!"**;

**private final** AtomicLong **counter** = **new** AtomicLong();

**public** Greeting generateMessage(String name) {

**return new** Greeting(**counter**.incrementAndGet(), String.*format*(***template***, name));

}

}

***// Greeting Controller Class***

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.mycompany.app.service.GreetingService;

@RestController

**public class** GreetingController {

@Autowired

**private** GreetingService **greetingService**;

@RequestMapping(**"/greeting"**)

**public** Greeting greeting(@RequestParam(value = **"name"**, defaultValue = **"World"**) String name) {

**return greetingService**.generateMessage(name);

}

}