Making Coffee

- What is the most efficient, hassle-free way to make coffee?
- What would you prefer for better productivity and results?

Slide Note

Spring Core Application

DailyNews Inc. wants a web application to manage news details, including the following information:

News title, author, description, content, and date

The web application will support the following functionalities:

1. Add a new story by accepting input from some view pages.
2. Display a list of existing stories stored in the database.
3. Delete existing news stored in the database.

Developer Jim has been tasked with working on the objectives above. Help Jim implement the tasks using Java-based configurations.

- List the steps that need to be implemented while developing the Spring core application.

Slide Note

Steps to Develop the Spring Core Application

- How many bean classes will you create?
- How can you distinguish which bean has business, database, or other logic?
- Is it possible for a developer to remember all the dependencies and settings needed to build an application?
- Will it be easy for the developer to manage the version of each dependency?
- How easy will it be to connect to the database where you write the configuration steps?

Slide Note

Think and Tell

- Is adding various dependencies, configuration files, and servers time-consuming?
- How can a developer improve productivity while developing back-end applications using Spring?

Slide Note

Build the Skeleton of Spring Boot Application

Slide Note

Learning Objectives

- Explore Spring Boot
- Create a Spring Boot application
- Implement a Spring Boot application
- Add business logic to a Spring Boot application

Slide Note

Introduction to Spring Boot

What is Spring Boot?

- Spring Boot makes it easy to create standalone, production-grade Spring-based applications that you can run.
- A standalone application is a software program that can be executed independently, like the main method in a Java program.
- You can develop Spring applications without Spring Boot. But this involves a lot of configuration, which is time-consuming.
- Spring Boot makes setting up the Spring framework easier by giving you a set of rules. If, as a developer, you follow those rules, Spring Boot will do all the configuration for you.
- So, as a programmer, you need to focus and write your application code instead of the framework's configuration code.

Slide Note

Why Use Spring Boot?

- The following features and benefits of Spring Boot make it popular among developers:
  - It is very easy to develop Spring-based applications with Java.
  - It reduces a lot of development time and increases productivity.
  - It avoids writing lots of repetitive code, annotations, and XML configurations.
  - It eases dependency management.
  - It provides embedded HTTP servers like Tomcat, Jetty, etc., to easily develop and test our web applications.
  - It provides many plugins to work with embedded and in-memory databases easily.

Slide Note

Creating a Spring Boot Application

Slide Note

Creating a Spring Boot Project

- Spring Initializr is a web application that can generate a Spring Boot project structure for developers.
- The image at left shows the Initializr setup for the sample project:
  - Maven is chosen as the build tool by default.
  - The Spring Boot version we will choose is 2.7.2.
  - It shows values for group and artifact, respectively, that can be changed as per the project.
  - Jar and version 11 of Java are the packages for this project.
  - Click on Generate.

Slide Note

Spring Boot Starter Parent

- The parent tag specifies the `spring-boot-starter-parent`, which is a project starter.
- It is a special starter project that provides default configurations for our application and a complete dependency tree to quickly build a Spring Boot project.
- It allows us to keep the Java version and other related properties consistent.
- It controls the versions of dependencies to avoid conflict.
- It also provides default configurations for many Maven plugins that we do not have to specify individually.

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.7.2</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>
```

Slide Note

Spring Boot Dependencies

- In the pom.xml, there are two dependencies: `spring-boot-starter` and `spring-boot-starter-test`.
- They are the basic dependencies required to write a simple Spring Boot application.
- Starters are a one-stop shop for all the Spring dependencies.
- `spring-boot-starter` brings in dependencies like Spring-core, autoconfiguration, etc.
- `spring-boot-starter-test` brings in dependencies required for testing, like JUnit, Mockito, and Hamcrest. We do not have to add these dependencies individually.
- So, we shall not add any individual dependencies.

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

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

```
Dependencies
  org.springframework.boot:spring-boot-starter:2.7.2
    org.springframework.boot:spring-boot:2.7.2
    org.springframework.boot:spring-boot-autoconfigure:2.7.2
    org.springframework.boot:spring-boot-starter-logging:2.7.2
    jakarta.annotation:jakarta.annotation-api:1.3.5
    org.springframework:spring-core:5.3.22
    org.yaml:snakeyaml:1.30
```

Slide Note

Packaging the Project

- The `spring-boot-maven-plugin` is included in pom.xml. This plugin is responsible for creating the executable Spring Boot jar file.
- This jar file is different from the traditional Java jar file. You can execute this jar to run your application.
- This JAR file, also known as Uber JAR or Fat JAR, contains all the dependencies along with the application – a specialty of Spring Boot.
- The project is configured to build a jar or war (as appropriate) using the packaging element.

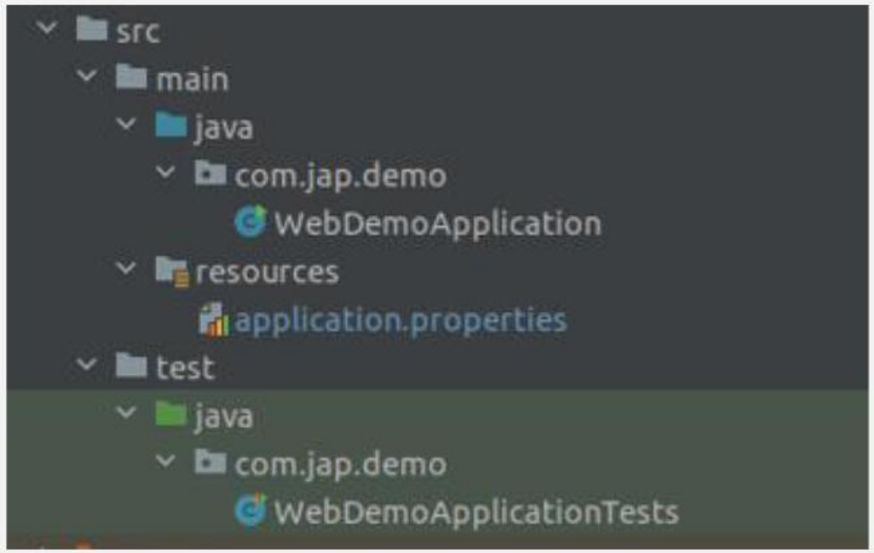
```
<plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
```

```
<groupId>com.jap</groupId>
<artifactId>demo</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>demo</name>
<packaging>jar</packaging>
<description>Demo project for Spring Boot</description>
```

Slide Note

Spring Boot Structure

- Within the `src` folder, there are the `main` and `test` folders.
- Within the main folder are the `java` and `resources` folders.
  - The Java folder has all the Java classes.
  - Within the resources folder, there is a file called application.properties that contains all project configurations.



Slide Note

The @SpringBootApplication annotation tells Spring Boot to scan the package where the main class is, scan other packages that are inside the root package, identify the annotated classes, and configure them automatically as Spring beans.

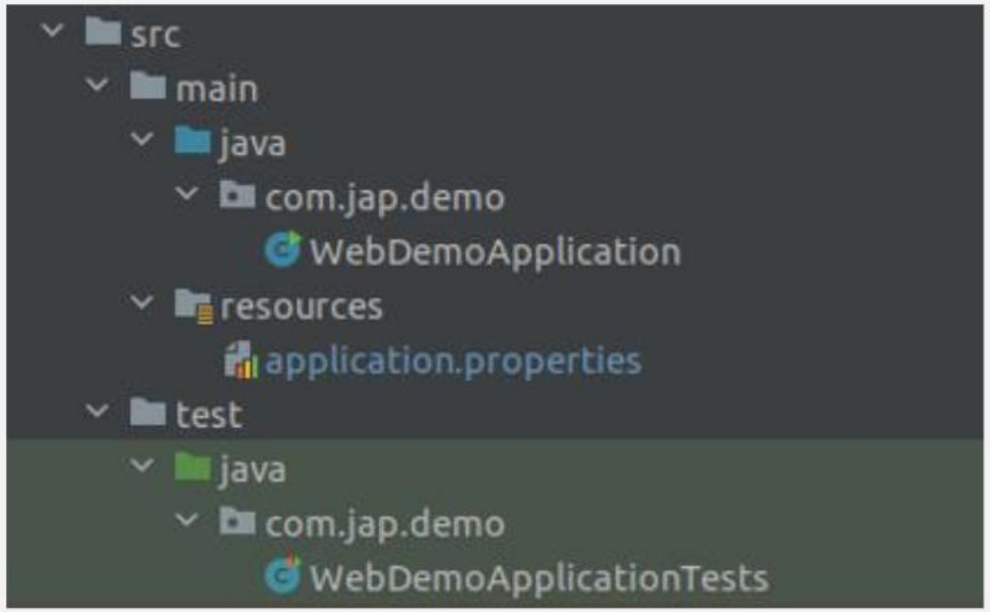
Hello World in Spring Boot

- The DemoApplication class contains the main method, and it is annotated with `@SpringBootApplication`.
- `com.jap.demo` is called the root package, as the main method is inside it.
- All other packages are within this root package.
- `@SpringBootApplication` annotation instructs Spring Boot to scan the package at the main class location, scan other packages within the root package, identify the annotated classes, and configure them automatically as Spring beans.
- Within the main method, the `SpringApplication` class is used to bootstrap and launch a Spring application as a standalone application.
- This class automatically creates the ApplicationContext instance and loads the bean.

```java
@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
        System.out.println("Hello world");
    }

}
```



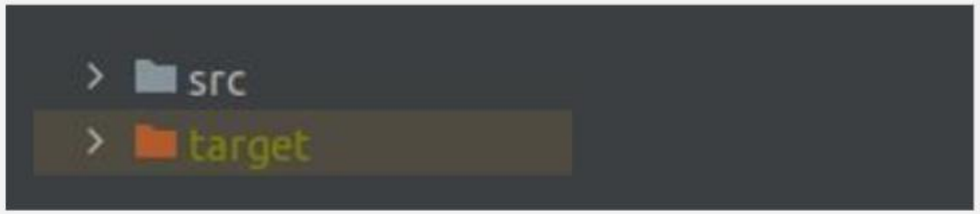
Slide Note

Implementing a Spring Boot Application

Slide Note

Package and Run a Spring-boot Application

- Create the jar file by running the command `mvn package` in the terminal.
- After writing this command, the `spring-boot-maven-plugin` will generate the jar file within the target folder.
- The jar file name is the same as the name given in the pom.xml file.
- The command to run the jar in Spring is **mvn spring-boot:run.**




```
seconds (JVM running for 0.749)
Hello world
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.776 s
[INFO] Finished at: 2022-10-03T17:56:04+05:30
[INFO] ------------------------------------------------------------------------
```

Slide Note

The DailyNews Inc. wants a web application to manage the news details, but their team of developers is facing deadline issues because they spend so much time
adding dependencies and managing configuration files rather than focusing on the end objective.

The Spring Boot Application

DailyNews Inc. wants a web application to manage news details. However, the developers' team faces deadline issues as they spend much time adding dependencies and managing configuration files instead of focusing on the end objective.
Developer Jim has been tasked with implementing the objectives above for the solution.
Help Jim select the correct technology and develop the back-end application.
Start with the first stage to help Jim create a simple Spring Boot application using Spring Initializr, and then run the application using the IntelliJ IDE.
Click here for the solution

DEMO

Slide Note

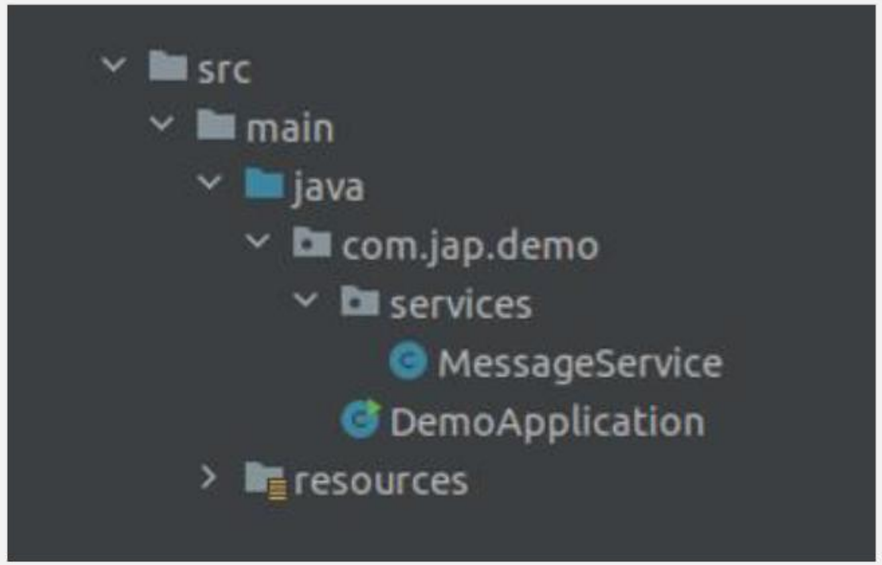
Adding Business Logic to Spring Boot Application

Slide Note

Adding a Service Layer

- Now that we have created a bare-bones Spring Boot application, let's introduce some functionality or logic to our application.
- In Spring Boot, the business logic is written within the service classes.
- Create a service package that will hold all the service classes.
- A service package should be created inside the root package, i.e., `com.jap.demo`.

Slide Note




```java
import org.springframework.stereotype.Service;

@Service
public class MessageService {

    public String helloWorld(){
        return "Hello World";
    }
}
```

Creating a Service Class

- This package has a class `MessageService` that will publish the message "Hello World".
- The class is annotated with the `@Service` annotation.
- All classes that contain business logic must be annotated with `@Service`.

Slide Note

Accessing the Service Class

```java
@SpringBootApplication
public class DemoApplication {
    private static MessageService messageService;

    public static void main(String[] args) {
        ApplicationContext context =SpringApplication.run(DemoApplication.class, args);
        messageService = context.getBean( s: "messageService", MessageService.class);
        String message = messageService.helloWorld();
        System.out.println(message);

    }
}
```

- The Spring Boot application creates the ApplicationContext object, through which the other objects of the application can be accessed.
- By using the context `getBean()` method, we are accessing the `MessageService` class.
- And we are calling the method of rhe `MessageService` class.

Slide Note

Quick Check

Which of the following can be used for dependency management in Spring Boot?

1. Maven
2. Spring
3. Gradle
4. Ant

Slide Note

Quick Check : Solution

Which of the following can be used for dependency management in Spring Boot?

1. **Maven**
2. Spring
3. Gradle
4. Ant