Hello World – Using Java Spring Framework

To develop a Java Spring-based "Hello World" application using ApplicationContext approach with Maven and Spring Tool Suite (STS), follow these steps:

**Prerequisites**

* **Java Development Kit (JDK)** installed
* **Spring Tool Suite (STS)** installed
* **Maven** installed

**Step 1: Create a Maven Project in STS**

1. Open **Spring Tool Suite**.
2. Click on **File** > **New** > **Maven Project**.
3. Select the **Create a simple project (skip archetype selection)** checkbox and click **Next**.
4. Enter **Group Id** (e.g., com.example) and **Artifact Id** (e.g., hello-world-spring) and click **Finish**.

**Step 2: Add Spring Dependencies**

1. Open the pom.xml file.
2. Add the following dependencies inside the <dependencies> tag:

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.20</version>

</dependency>

</dependencies>

**Step 3: Create Spring Configuration Class**

1. Create a new package com.example.config.
2. Create a new Java class AppConfig in the com.example.config package.

package com.example.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class AppConfig {

@Bean

public HelloWorld helloWorld() {

return new HelloWorld();

}

}

**Step 4: Create the HelloWorld Bean**

1. Create a new package com.example.
2. Create a new Java class HelloWorld in the com.example package.

package com.example;

public class HelloWorld {

public String sayHello() {

return "Hello, World!";

}

}

**Step 5: Create the Main Application Class**

1. Create a new Java class MainApp in the com.example package.

package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new AnnotationConfigApplicationContext(com.example.config.AppConfig.class);

HelloWorld helloWorld = context.getBean(HelloWorld.class);

System.out.println(helloWorld.sayHello());

}

}

**Step 6: Run the Application**

1. Right-click on the MainApp class.
2. Select **Run As** > **Java Application**.

You should see the output:

**Hello, World!**

**Project Structure**

hello-world-spring

|-- src

| |-- main

| |-- java

| |-- com

| |-- example

| |-- config

| |-- App

java

Copy code

| |-- AppConfig.java

| |-- HelloWorld.java

| |-- MainApp.java

|-- pom.xml

**Detailed Explanation**

1. **pom.xml**: The pom.xml file manages dependencies. We added spring-context to include core Spring functionality.
2. **AppConfig.java**: This is the configuration class annotated with @Configuration. The @Bean annotation indicates that a method produces a bean to be managed by the Spring container.
3. **HelloWorld.java**: This is the bean class with a simple method sayHello() returning "Hello, World!".
4. **MainApp.java**: The main class to bootstrap the Spring application. We use AnnotationConfigApplicationContext to load the Spring context from the AppConfig class and retrieve the HelloWorld bean.

By following these steps, you've created a basic Spring application using a Bean configuration approach, managed with Maven in Spring Tool Suite (STS).

**Additional Tips**

* **STS Shortcuts**: Use STS shortcuts for creating packages and classes quickly.
* **Run Configurations**: If needed, you can create custom run configurations in STS for different environments.
* **Maven Commands**: Use Maven commands like mvn clean install to build the project and ensure all dependencies are correctly resolved.

To create a Java Spring-based "Hello World" application using the Bean XML configuration approach with Maven and Spring Tool Suite (STS), follow these steps:

**Prerequisites**

* **Java Development Kit (JDK)** installed
* **Spring Tool Suite (STS)** installed
* **Maven** installed

**Step 1: Create a Maven Project in STS**

1. Open **Spring Tool Suite**.
2. Click on **File** > **New** > **Maven Project**.
3. Select the **Create a simple project (skip archetype selection)** checkbox and click **Next**.
4. Enter **Group Id** (e.g., com.example) and **Artifact Id** (e.g., hello-world-spring-xml) and click **Finish**.

**Step 2: Add Spring Dependencies**

1. Open the pom.xml file.
2. Add the following dependencies inside the <dependencies> tag:

xml

Copy code

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.20</version>

</dependency>

</dependencies>

**Step 3: Create the Spring Bean XML Configuration**

1. Right-click on the src/main/resources directory, select **New** > **Other** > **XML File** and name it applicationContext.xml.
2. Add the following content to applicationContext.xml:

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="helloWorld" class="com.example.HelloWorld"/>

</beans>

**Step 4: Create the HelloWorld Bean**

1. Create a new package com.example.
2. Create a new Java class HelloWorld in the com.example package.

package com.example;

public class HelloWorld {

public String sayHello() {

return "Hello, World!";

}

}

**Step 5: Create the Main Application Class**

1. Create a new Java class MainApp in the com.example package.

package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

HelloWorld helloWorld = (HelloWorld) context.getBean("helloWorld");

System.out.println(helloWorld.sayHello());

}

}

**Step 6: Run the Application**

1. Right-click on the MainApp class.
2. Select **Run As** > **Java Application**.

You should see the output:

**Hello, World!**

**Project Structure**

hello-world-spring-xml

|-- src

| |-- main

| |-- java

| |-- com

| |-- example

| |-- HelloWorld.java

| |-- MainApp.java

| |-- resources

| |-- applicationContext.xml

|-- pom.xml

**Detailed Explanation**

1. **pom.xml**: The pom.xml file manages dependencies. We added spring-context to include core Spring functionality.
2. **applicationContext.xml**: This is the Spring configuration file where beans are defined. The helloWorld bean is configured here with the class com.example.HelloWorld.
3. **HelloWorld.java**: This is the bean class with a simple method sayHello() returning "Hello, World!".
4. **MainApp.java**: The main class to bootstrap the Spring application. We use ClassPathXmlApplicationContext to load the Spring context from the applicationContext.xml file and retrieve the HelloWorld bean.

This is the example of a basic Spring application using a Bean XML configuration approach, managed with Maven in Spring Tool Suite (STS).

<https://www.youtube.com/watch?v=OQIsKgfkYcE>

<https://github.com/Krishnamurtyp/spring-framework-hello-world>