Spring MVC App - Java Config

Java Configuration

Java Configuration

- Instead of configuring Spring MVC app using XML

Java Configuration

- Instead of configuring Spring MVC app using XML
  - web.xml

Java Configuration

- Instead of configuring Spring MVC app using XML
  - web.xml
  - spring-mvc-demo-servlet.xml

Java Configuration

- Instead of configuring Spring MVC app using XML
  - web.xml
  - spring-mvc-demo-servlet.xml

Java Configuration

- Instead of configuring Spring MVC app using XML
  - web.xml
  - spring-mvc-demo-servlet.xml

- Configure the Spring MVC app with Java code

Java Configuration

- Instead of configuring Spring MVC app using XML
  - web.xml
  - spring-mvc-demo-servlet.xml

- Configure the Spring MVC app with Java code

No XML!

Development Process

Step-By-Step

Development Process

Step-By-Step

1. Add Maven dependencies for Spring MVC Web App

Development Process

Step-By-Step

1. Add Maven dependencies for Spring MVC Web App

2. Create Spring App Configuration (@Configuration)

Development Process

Step-By-Step

1. Add Maven dependencies for Spring MVC Web App

2. Create Spring App Configuration (@Configuration)

3. Create Spring Dispatcher Servlet Initializer

Development Process

Step-By-Step

1. Add Maven dependencies for Spring MVC Web App

2. Create Spring App Configuration (@Configuration)

3. Create Spring Dispatcher Servlet Initializer

4. Develop our Spring controller

Development Process

Step-By-Step

1. Add Maven dependencies for Spring MVC Web App

2. Create Spring App Configuration (@Configuration)

3. Create Spring Dispatcher Servlet Initializer

4. Develop our Spring controller

5. Develop our JSP view page

Step 1: Add Maven dependencies for Spring MVC Web App

Step 1: Add Maven dependencies for Spring MVC Web App

spring-

Step 1: Add Maven dependencies for Spring MVC Web App

spring-

jstl

Step 1: Add Maven dependencies for Spring MVC Web App

spring-

javax.servlet-api

jstl

Step 1: Add Maven dependencies for Spring MVC Web App

spring-

javax.servlet-api

jstl

javax.servlet.jsp-api

Step 1: Add Maven dependencies for Spring MVC Web App

File: pom.xml

```
<!-- Spring MVC support -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>...</version>
</dependency>
```

Step 1: Add Maven dependencies for Spring MVC Web App

File: pom.xml

```
<!-- Spring MVC support -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>...</version>
</dependency>
```

Will load all supporting dependencies: spring-core, logging etc …

Step 1: Add Maven dependencies for Spring MVC Web App

File: pom.xml

```
<!-- Spring MVC support -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>...</version>
</dependency>
```

Step 1: Add Maven dependencies for Spring MVC Web App

File: pom.xml

```
<!-- Spring MVC support -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>...</version>
</dependency>

<!-- Servlet, JSP and JSTL support -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>...</version>
</dependency>

<dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>javax.servlet.jsp-api</artifactId>
    <version>...</version>
</dependency>

<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>...</version>
```

Add Servlet, JSP and JSTL support

Customize Maven Build

File: pom.xml

```
<build>

    <pluginManagement>
        <plugins>









        </plugins>
    </pluginManagement>
```

Customize Maven Build

File: pom.xml

```
<build>

    <pluginManagement>
      <plugins>









      </plugins>
    </pluginManagement>
```

Need to customize Maven build

Since we are not using web.xml

Customize Maven Build

Need to customize Maven build

Since we are not using web.xml

File: pom.xml

```
<build>

    <pluginManagement>
        <plugins>

            <plugin>
                <!-- Add Maven coordinates (GAV) for: maven-war-plugin -->
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>3.2.0</version>
            </plugin>

        </plugins>
    </pluginManagement>
```

Customize Maven Build

File: pom.xml

```
<build>

    <pluginManagement>
        <plugins>

            <plugin>
                <!-- Add Maven coordinates (GAV) for: maven-war-plugin -->
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>3.2.0</version>
            </plugin>

        </plugins>
    </pluginManagement>
```

Need to customize Maven build

Since we are not using web.xml

Must add Maven WAR plugin

XML config to Java config

XML config to Java config

web.xml

spring-mvc-demo-servlet.xml

XML config to Java config

web.xml

spring-mvc-demo-servlet.xml

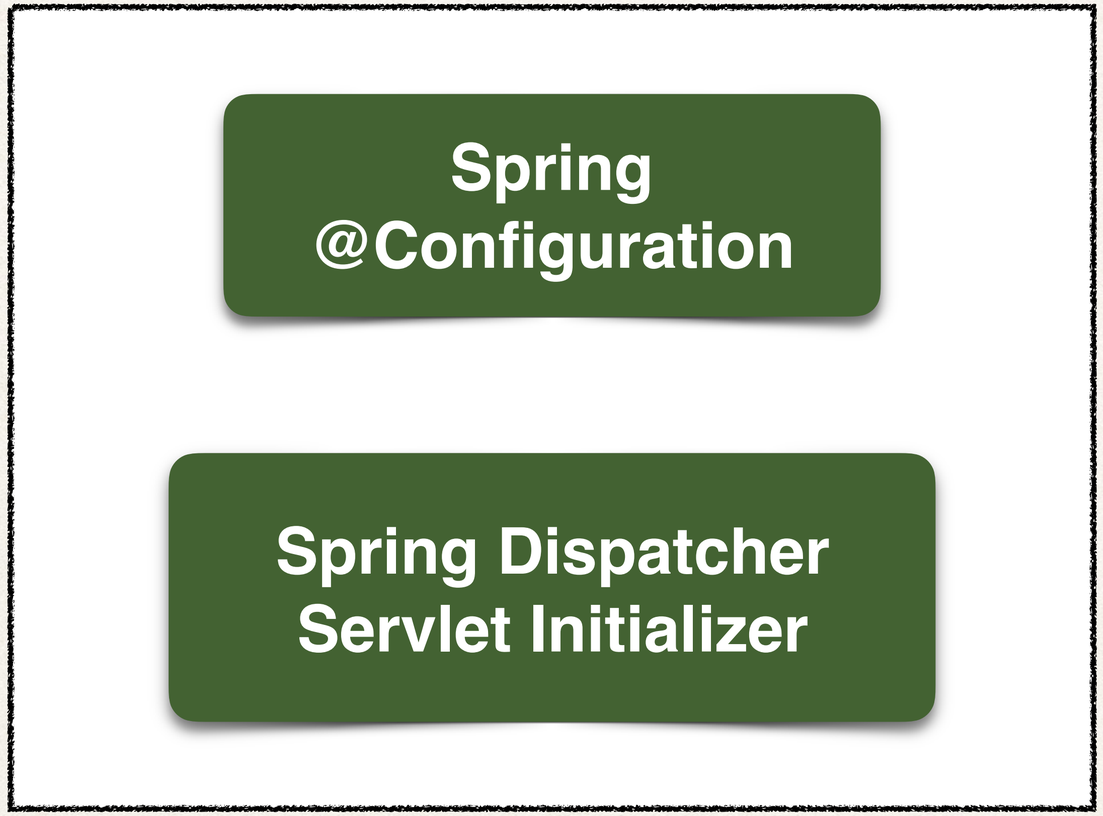
Java Config

Spring
@Configuration

Spring Dispatcher
Servlet Initializer

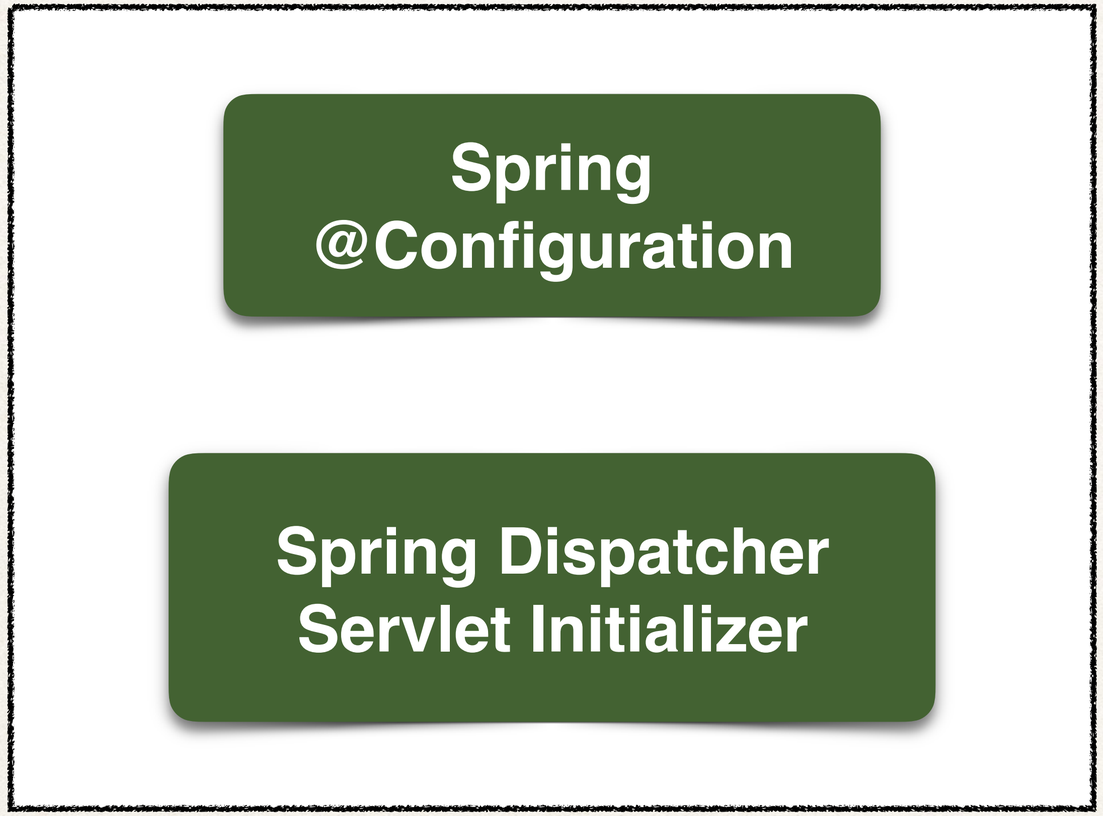
XML config to Java config

Java Config



XML config to Java config

Java Config



No XML!

Flash Back to XML config (the old way)

File: spring-mvc-demo-servlet.xml

```
<beans>









</beans>
```

Flash Back to XML config (the old way)

File: spring-mvc-demo-servlet.xml

```
<beans>

</beans>
```

Just an FYI

Flash Back to XML config (the old way)

File: spring-mvc-demo-servlet.xml

```
<beans>

    <!-- Add support for component scanning -->
    <context:component-scan base-package="com.luv2code.springdemo" />

</beans>
```

Just an FYI

Flash Back to XML config (the old way)

File: spring-mvc-demo-servlet.xml

```
<beans>

    <!-- Add support for component scanning -->
    <context:component-scan base-package="com.luv2code.springdemo" />

    <!-- Add support for conversion, formatting and validation support -->
    <mvc:annotation-driven/>









</beans>
```

Just an FYI

Flash Back to XML config (the old way)

File: spring-mvc-demo-servlet.xml

Just an FYI

```
<beans>

    <!-- Add support for component scanning -->
    <context:component-scan base-package="com.luv2code.springdemo" />

    <!-- Add support for conversion, formatting and validation support -->
    <mvc:annotation-driven/>

    <!-- Define Spring MVC view resolver -->
    <bean
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/view/" />
        <property name="suffix" value=".jsp" />
    </bean>

</beans>
```

Enabling the MVC Java Config

Enabling the MVC Java Config

```
@EnableWebMvc
```

Enabling the MVC Java Config

@EnableWebMvc

- Provides similar support to **<mvc:annotation-driven />** in XML.

Enabling the MVC Java Config

@EnableWebMvc

- Provides similar support to **<mvc:annotation-driven />** in XML.

- Adds conversion, formatting and validation support

Enabling the MVC Java Config

@EnableWebMvc

- Provides similar support to **<mvc:annotation-driven />** in XML.
- Adds conversion, formatting and validation support
- Processing of @Controller classes and @RequestMapping etc ... methods

Step 2: Create Spring App Configuration

File: DemoAppConfig.java

```
@Configuration
@EnableWebMvc
@ComponentScan(basePackages="com.luv2code.springsecurity.demo")
public class DemoAppConfig {









}
```

Step 2: Create Spring App Configuration

File: DemoAppConfig.java

```
@Configuration
@EnableWebMvc
@ComponentScan(basePacka
public class DemoAppConfig {

}
```

provides similar support to
<mvc:annotation-driven />

Step 2: Create Spring App Configuration

File: DemoAppConfig.java

```
@Configuration
@EnableWebMvc
@ComponentScan(basePackages="com.luv2code.springsecurity.demo")
public class DemoAppConfig {









}
```

Step 2: Create Spring App Configuration

File: DemoAppConfig.java

```
@Configuration
@EnableWebMvc
@ComponentScan(basePackages="com.luv2code.springsecurity.demo")
public class DemoAppConfig {

    // define a bean for ViewResolver

}
```

Step 2: Create Spring App Configuration

File: DemoAppConfig.java

```
@Configuration
@EnableWebMvc
@ComponentScan(basePackages="com.luv2code.springsecurity.demo")
public class DemoAppConfig {

    // define a bean for ViewResolver

    @Bean
    public ViewResolver viewResolver() {

        InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();

        viewResolver.setPrefix("/WEB-INF/view/");
        viewResolver.setSuffix(".jsp");

        return viewResolver;
    }

}
```

View Resolver Configs - Explained

```
InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();

viewResolver.setPrefix("/WEB-INF/view/");
viewResolver.setSuffix(".jsp");
```

View Resolver Configs - Explained

```
InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();

viewResolver.setPrefix("/WEB-INF/view/");
viewResolver.setSuffix(".jsp");
```

/WEB-INF/view/ show-student-list .jsp