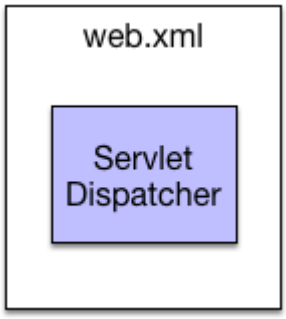
El concepto de Spring Web Initializer es poco conocido. En muchas ocasiones cuando damos de alta una aplicación con Spring MVC necesitamos si o sí dar de alta un despachador a nivel del web.xml.



spring web initilizer

Sin embargo a partir de la versión 4 de Spring podemos inicializar Spring MVC sin hacer uso de web.xml. Vamos a ver como hacerlo. El primer paso es configurar las dependencias de Maven.

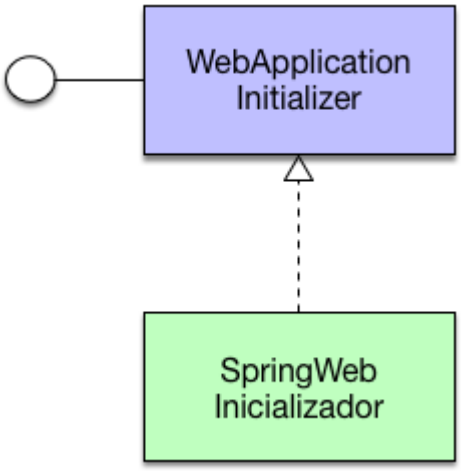
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>com.arquitecturajava</groupId>
        <artifactId>springweb</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>war</packaging
        <dependencies>

                <!--
```

```
https://mvnrepository.com/artifact/org.springframework/spring-core -->
                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-core</artifactId>
                        <version>4.3.13.RELEASE</version>
                </dependency>
                <!--
https://mvnrepository.com/artifact/org.springframework/spring-context
-->
                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-context</artifactId>
                        <version>4.3.13.RELEASE</version>
                </dependency>

                <!--
https://mvnrepository.com/artifact/org.springframework/spring-webmvc -
->
                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-webmvc</artifactId>
                        <version>4.3.13.RELEASE</version>
                </dependency>
                        <dependency>
                        <groupId>javax.servlet</groupId>
                        <artifactId>javax.servlet-api</artifactId>
                        <version>3.1.0</version>
                        <scope>provided</scope>
                </dependency>
        </dependencies>
        <build>
```

```
		<plugins>
			<plugin>
				<artifactId>maven-compiler-
plugin</artifactId>
				<configuration>
					<source>1.8</source>
					<target>1.8</target>
				</configuration>
			</plugin>
			<plugin>
<groupId>org.apache.maven.plugins</groupId>
				<artifactId>maven-war-
plugin</artifactId>
				<version>2.6</version>
				<configuration>
<failOnMissingWebXml>false</failOnMissingWebXml>
				</configuration>
			</plugin>
		</plugins>

	</build>
</project>
```

Spring Web Initializer

El siguiente paso es crear una clase que implemente el interface de WebApplicationInitializer

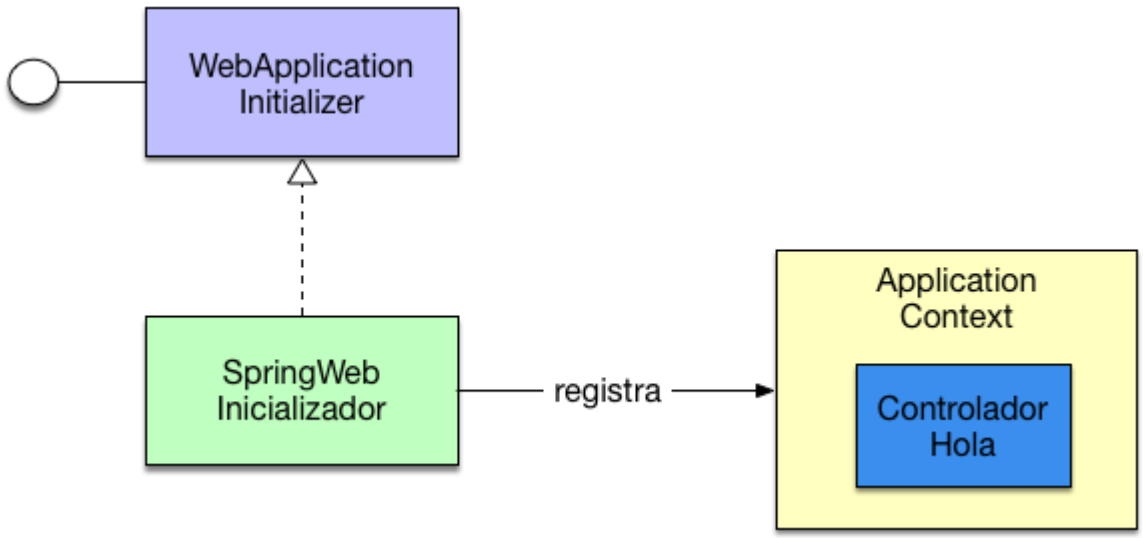


spring web initilizer interface

```
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRegistration;

import org.springframework.web.WebApplicationInitializer;
import
org.springframework.web.context.support.AnnotationConfigWebApplication
Context;
import org.springframework.web.servlet.DispatcherServlet;

import com.arquitecturajava.config.ConfiguracionSpring;

public class SpringWebInicializador implements
WebApplicationInitializer {

        public void onStartup(ServletContext contenedor) throws
ServletException {
```

```
                AnnotationConfigWebApplicationContext contexto = new
AnnotationConfigWebApplicationContext();
                contexto.register(ConfiguracionSpring.class);
                contexto.setServletContext(contenedor);

                ServletRegistration.Dynamic servlet =
contenedor.addServlet("dispatcher", new DispatcherServlet(contexto));
                servlet.setLoadOnStartup(1);
                servlet.addMapping("/");
        }

}
```

El interface solo implementa un método onStartup que se encarga de inicializar la aplicacion web. Para ello registra un contexto basado puramente en anotaciones.



spring web initializer y contexto

Este contexto registra un Controlador y un ViewResolver veamos el código del contexto creado con anotaciones:

```
package com.arquitecturajava.config;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdap
ter;
import
org.springframework.web.servlet.view.InternalResourceViewResolver;

@Configuration
@ComponentScan("com.arquitecturajava.*")
@EnableWebMvc
public class ConfiguracionSpring  extends WebMvcConfigurerAdapter {
        @Bean
         public InternalResourceViewResolver
getInternalResourceViewResolver() {
         System.out.println("llega");
         InternalResourceViewResolver resolver = new
InternalResourceViewResolver();
         resolver.setPrefix("/WEB-INF/jsp/");
         resolver.setSuffix(".jsp");
         return resolver;

        }
}
```

Spring con @ComponentScan

En este caso estamos registrando un bean con un ViewResolver para JSP y ademas a través

de la anotación de @ComponentScan estamos registrando clases adicionales que estén anotadas como beans de Spring dentro de com.arquitecturajava. En este ejemplo tan sencillo solo afecta al controlador.

```
package com.arquitecturajava.controllers;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class HolaController {

        @RequestMapping("/hola")
        public String hola() {
                return "hola";
        }
}
```

Una vez configurados los beans nos queda por ver el código del fichero hola.jsp

```
<htm>
<body>
hola
<body>
</html>
```

Acabamos de configurar una aplicación con Spring Web Initializer . Al no hacer uso de XML reduciremos de forma drástica los problemas de configuración.

Otros artículos relacionados:

1. Spring Boot @Lazy Components
2. Spring @Qualifier utilizando @Autowired
3. Spring MVC Flash Attributes
4. Spring MVC