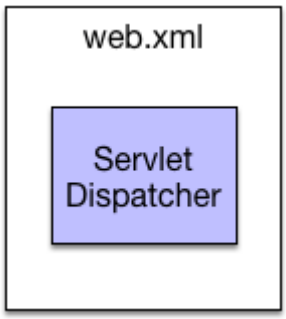
Acabamos de ver como usar Java equals y hashcode

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.

```
&amp;lt;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"&amp;gt;
        &amp;lt;modelVersion&amp;gt;4.0.0&amp;lt;/modelVersion&amp;gt;
&amp;lt;groupId&amp;gt;com.arquitecturajava&amp;lt;/groupId&amp;gt;
        &amp;lt;artifactId&amp;gt;springweb&amp;lt;/artifactId&amp;gt;
        &amp;lt;version&amp;gt;0.0.1-SNAPSHOT&amp;lt;/version&amp;gt;
        &amp;lt;packaging&amp;gt;war&amp;lt;/packaging
        &amp;lt;dependencies&amp;gt;

                &amp;lt;!--
```

```
https://mvnrepository.com/artifact/org.springframework/spring-core --
&amp;gt;
                &amp;lt;dependency&amp;gt;
&amp;lt;groupId&amp;gt;org.springframework&amp;lt;/groupId&amp;gt;
                        &amp;lt;artifactId&amp;gt;spring-
core&amp;lt;/artifactId&amp;gt;
&amp;lt;version&amp;gt;4.3.13.RELEASE&amp;lt;/version&amp;gt;
                &amp;lt;/dependency&amp;gt;
                &amp;lt;!--
https://mvnrepository.com/artifact/org.springframework/spring-context
--&amp;gt;
                &amp;lt;dependency&amp;gt;
&amp;lt;groupId&amp;gt;org.springframework&amp;lt;/groupId&amp;gt;
                        &amp;lt;artifactId&amp;gt;spring-
context&amp;lt;/artifactId&amp;gt;
&amp;lt;version&amp;gt;4.3.13.RELEASE&amp;lt;/version&amp;gt;
                &amp;lt;/dependency&amp;gt;

                &amp;lt;!--
https://mvnrepository.com/artifact/org.springframework/spring-webmvc -
-&amp;gt;
                &amp;lt;dependency&amp;gt;
&amp;lt;groupId&amp;gt;org.springframework&amp;lt;/groupId&amp;gt;
                        &amp;lt;artifactId&amp;gt;spring-
webmvc&amp;lt;/artifactId&amp;gt;
&amp;lt;version&amp;gt;4.3.13.RELEASE&amp;lt;/version&amp;gt;
                &amp;lt;/dependency&amp;gt;

                        &amp;lt;dependency&amp;gt;
&amp;lt;groupId&amp;gt;javax.servlet&amp;lt;/groupId&amp;gt;
                        &amp;lt;artifactId&amp;gt;javax.servlet-
```

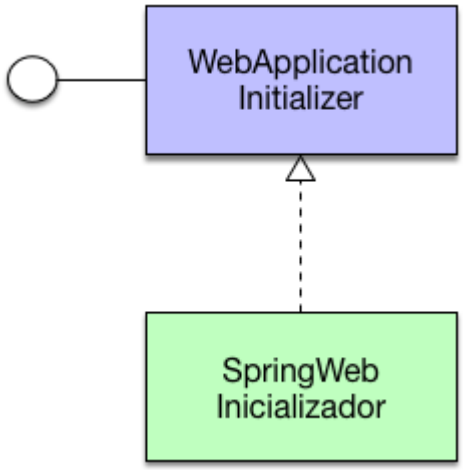
```
api&amp;lt;/artifactId&amp;gt;
&amp;lt;version&amp;gt;3.1.0&amp;lt;/version&amp;gt;
&amp;lt;scope&amp;gt;provided&amp;lt;/scope&amp;gt;
                &amp;lt;/dependency&amp;gt;
        &amp;lt;/dependencies&amp;gt;

        &amp;lt;build&amp;gt;
                &amp;lt;plugins&amp;gt;
                        &amp;lt;plugin&amp;gt;
                                &amp;lt;artifactId&amp;gt;maven-
compiler-plugin&amp;lt;/artifactId&amp;gt;
                                &amp;lt;configuration&amp;gt;
&amp;lt;source&amp;gt;1.8&amp;lt;/source&amp;gt;
&amp;lt;target&amp;gt;1.8&amp;lt;/target&amp;gt;
                                &amp;lt;/configuration&amp;gt;
                        &amp;lt;/plugin&amp;gt;
                        &amp;lt;plugin&amp;gt;
&amp;lt;groupId&amp;gt;org.apache.maven.plugins&amp;lt;/groupId&amp;gt
;
                                &amp;lt;artifactId&amp;gt;maven-war-
plugin&amp;lt;/artifactId&amp;gt;
&amp;lt;version&amp;gt;2.6&amp;lt;/version&amp;gt;
                                &amp;lt;configuration&amp;gt;
&amp;lt;failOnMissingWebXml&amp;gt;false&amp;lt;/failOnMissingWebXml&a
mp;gt;
                                &amp;lt;/configuration&amp;gt;
                        &amp;lt;/plugin&amp;gt;
                &amp;lt;/plugins&amp;gt;

        &amp;lt;/build&amp;gt;
&amp;lt;/project&amp;gt;
```

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 {
```

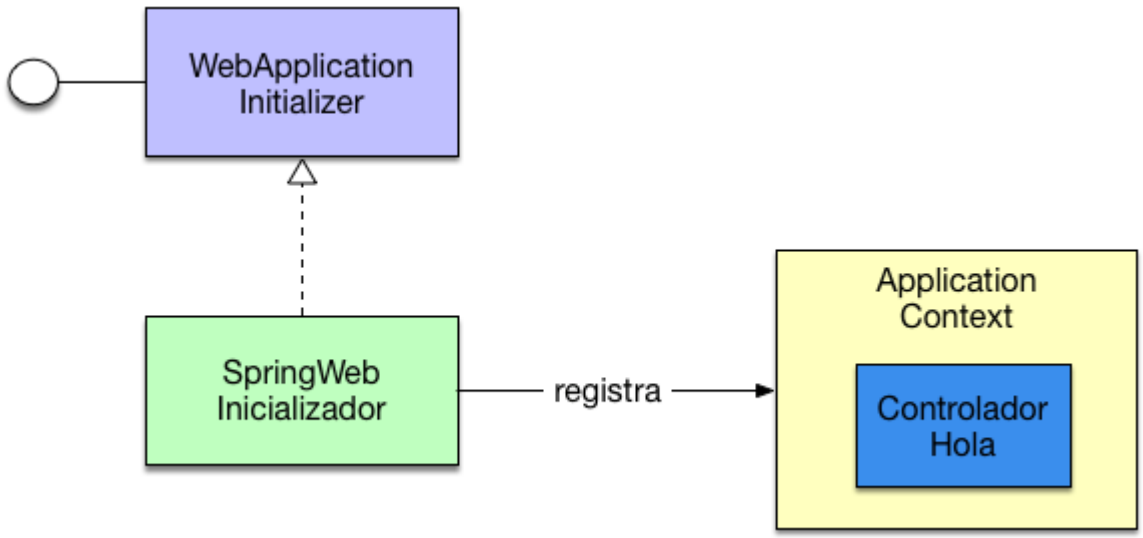
```
        public void onStartup(ServletContext contenedor) throws
ServletException {

                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

```
&amp;lt;htm&amp;gt;
&amp;lt;body&amp;gt;
hola
&amp;lt;body&amp;gt;
&amp;lt;/html&amp;gt;
```

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