Vamos a construir un ejemplo de Spring 5 Hello World. Todavía no tenemos una versión final de Spring 5 , pero no queda mucho para ello. Por lo tanto tenemos que empezar a echar un vistazo a las novedades que trae. Para ello nada mejor que construir un ejemplo de hola mundo , vamos con ello. El primer paso es instalarnos las dependencias del framework con Maven. Eso sí al no estar disponible una versión final tendremos que conectarnos al repositorio de Maven del propio Spring.

```
<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>Spring5HelloWorld</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-context</artifactId>
                        <version>5.0.0.RC3</version>
                </dependency>

                <!--
https://mvnrepository.com/artifact/org.springframework/spring-web -->
                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-web</artifactId>
                        <version>5.0.0.RC3</version>
```

```
		</dependency>
		<dependency>
			<groupId>io.projectreactor</groupId>
			<artifactId>reactor-core</artifactId>
			<version>3.0.6.RELEASE</version>
		</dependency>
		<dependency>
			<groupId>io.projectreactor.addons</groupId>
			<artifactId>reactor-test</artifactId>
			<version>3.0.6.RELEASE</version>
		</dependency>

		<dependency>
			<groupId>javax.servlet</groupId>
			<artifactId>javax.servlet-api</artifactId>
			<version>3.1.0</version>
		</dependency>
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-webmvc</artifactId>
			<version>5.0.0.RC3</version>
		</dependency>

	</dependencies>
<repositories>
		<repository>
			<id>spring</id>
			<name>Spring Repo</name>
			<url>http://repo.spring.io/milestone/</url>
		</repository>
```

```
        </repositories>
        <build>
                <plugins>
                        <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>
```

Como podemos ver en este caso vamos a utilizar la RC 3 de Spring 5.

Spring 5 Hello World

El primer paso para configurar un ejemplo de Hello World es registrar un WebApplicationListener

```
package com.arquitecturajava;

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;

public class SpringInicializador implements WebApplicationInitializer
{

        public void onStartup(ServletContext container) throws
ServletException {

                AnnotationConfigWebApplicationContext ctx = new
AnnotationConfigWebApplicationContext();
                ctx.register(SpringConfigurador.class);
                ctx.setServletContext(container);

                ServletRegistration.Dynamic servlet =
container.addServlet("dispatcher", new DispatcherServlet(ctx));

                servlet.setLoadOnStartup(1);
                servlet.addMapping("/");
                servlet.setAsyncSupported(true);
        }

}
```

Este WebApplicationListener tiene la peculiaridad que registra un DispatcherServlet Asíncrono. Ya que recordemos que con las nuevas arquitecturas de microservicios estos temas van a ir a más. El siguiente paso es registrar un configurador que defina los packages que vamos a escanear.

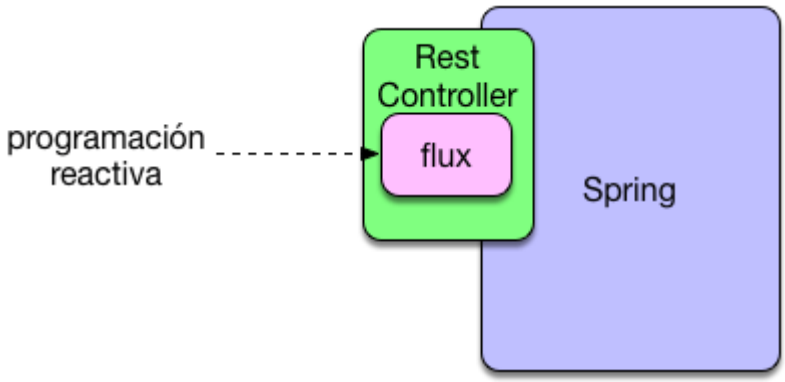
```
package com.arquitecturajava;

import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

@Configuration
@ComponentScan("com.arquitecturajava")
public class SpringConfigurador {

}
```

Por último vamos a dar de alta un @RestController y devolver un sencillo listado de noticias que en este caso son cadenas de texto simple

```
package com.arquitecturajava;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import reactor.core.publisher.Flux;

@RestController
```

```
public class ControladorNoticias {
        @GetMapping("/noticias")
        Flux<String> noticias() {
                return Flux.just("noticia1","noticia2");
        }
}
```

Como podemos ver hay cosas que cambian concretamente retornamos una lista de tipo Flux que es el framework de programación Reactiva de Spring 5.



Nos queda ejecutar las aplicación y ver como nos imprime las noticias al invocar la URL.

Otros artículos relacionados

1. Spring AOP y Aspectos
2. ¿Qué es Spring Boot?
3. Utilizando Spring Profiles

4. Spring 5 Referencia