¿Qué es Vertx?. Vertx es un framework Java que corre en la JVM y permite construir aplicaciones . Hasta aquí todo parece lo de siempre. Sin embargo se diferencia del resto en que su enfoque es similar al de Node.js y a la plataforma de JavaScript de servidor. La programación esta orientada de forma reactiva con un enfoque especial en Microservicios. Esto es algo que en el resto de los frameworks clásicos no tiene. Vamos a construir el ejemplo de hola mundo,para ello necesitaremos crear un proyecto Maven con Eclipse y añadir una dependencia:

```
<dependency>
 <groupId>io.vertx</groupId>
 <artifactId>vertx-web</artifactId>
 <version>3.2.0</version>
 </dependency>
```

Realizado este primer paso , el siguiente será construir una clase Java que va a recibir peticiones HTTP.

```
package com.arquitecturajava;

import io.vertx.core.AbstractVerticle;
import io.vertx.core.Future;
import io.vertx.core.http.HttpServer;
import io.vertx.core.http.HttpServerResponse;
import io.vertx.ext.web.Router;

public class AplicacionWeb extends AbstractVerticle {

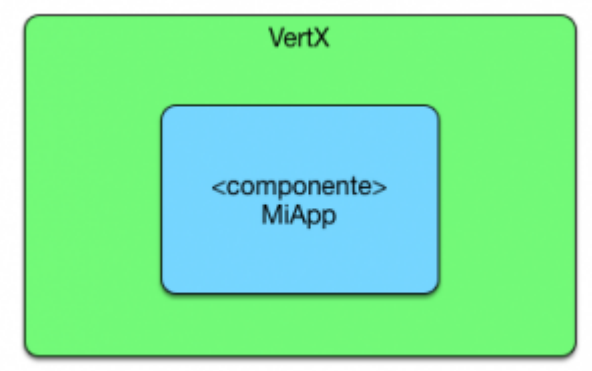
 @Override
 public void start(final Future&lt;Void&gt; futuro) {
 HttpServer server = vertx.createHttpServer();
```

```
 Router router = Router.router(vertx);

 router.route().handler(routingContext -&gt; {
 HttpServerResponse response = routingContext.response();
 response.putHeader("content-type", "text/html");
 response.end("&lt;html&gt;hola vert x&lt;/html&gt;");
 });

 server.requestHandler(router::accept).listen(8080);
 }
}
```

Como se puede observar la clase AplicacionWeb extiende de AbstractVerticle un componente del framework. Un Verticle es un componente diseñado para ejecutarse dentro de Vertx. En este caso hemos construido la aplicación web de hola mundo pero en vez de usar Node.js hemos usado el nuevo framework. Vamos a desplegar esta aplicación como si se tratase de un Microservicio.



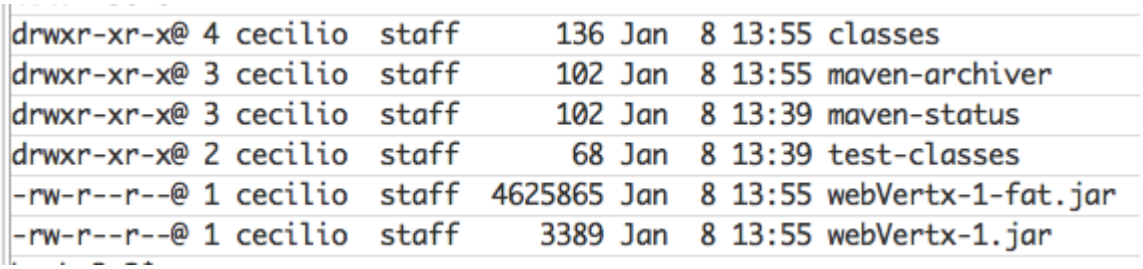
Microservicios y jars

Dos de las características fundamentales de los Microservicios es que son elementos independientes tanto en funcionalidad como en despliegue. Vamos a usar Maven para convertir nuestra pequeña clase en un elemento que se pueda desplegar solo. Para ello añadiremos un plugin a nuestro fichero pom.xml.

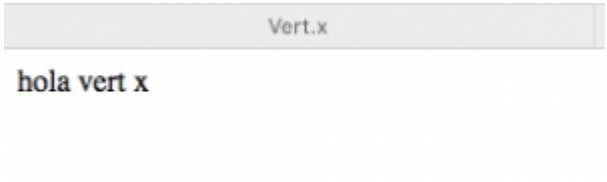
```
<plugin>
 <groupId>org.apache.maven.plugins</groupId>
 <artifactId>maven-shade-plugin</artifactId>
 <version>2.3</version>
 <executions>
 <execution>
 <phase>package</phase>
 <goals>
 <goal>shade</goal>
 </goals>
 <configuration>
 <transformers>
 <transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResour
ceTransformer">
```

```
 <manifestEntries>
 <Main-Class>io.vertx.core.Starter</Main-Class>
 <Main-Verticle>com.arquitecturajava.AplicacionWeb</Main-Verticle>
 </manifestEntries>
 </transformer>
 </transformers>
 <artifactSet />
 <outputFile>${project.build.directory}/${project.artifactId}-
${project.version}-fat.jar</outputFile>
 </configuration>
 </execution>
 </executions>
 </plugin>
```

Maven (Shade) se encargará de empaquetar en un único jar nuestra clase y todas las dependencias que contenga a esto es a lo que comunmente se le denomina fatjar. Ejecutamos mvn package y empaquetamos.



Desde la consola ejecutamos : java -jar webVertx-1-fat.jar y el programa se ejecutará arrancando el servidor web en el puerto 8080 que nos devolverá el mensaje de hola sin la necesidad de tener un servidor de aplicaciones.



Otros artículos relacionados: MicroServicios , Java Lambda