Diplomado en Programación Java

UAPA - MESCYT

Programación Web Java

Facilitador: Ing. Eudris Cabrera Rodríguez

Fecha de Realización: 15 Diciembre del 2018

Santiago de los Caballeros, República Dominicana

Introducción

¿Qué es una aplicación web?

'Por definición, se trata de algo más que un 'sitio web'.

Se trata de una aplicación cliente / servidor que utiliza un navegador Web como su programa cliente, y por consiguiente constituye un servicio interactivo mediante la conexión con los servidores a través de Internet (o Intranet).

Una aplicación web presenta contenido adaptado dinámicamente en función de parámetros de la petición, los comportamientos de los usuarios seguidos, y consideraciones de seguridad.

Una aplicación Web Java puede ser representada como una jerarquía de directorios y archivos, que a su vez contiene:

- **Componentes Web (Servlets, JavaServer Pages, entre otros)**
- **Recursos estáticos (páginas html e imágenes).**
- **Clases Java.**
- **Librerías (Archivos Jars).**
- **Un archivo descriptor de despliegue (web.xml).**

Una aplicación web de Java se puede implementar como un archivo **".war"**.
El archivo **".war"** es un archivo zip que contiene todo el contenido de la aplicación web correspondiente.

Las aplicaciones java web normalmente no se ejecutan directamente en el servidor, sino que se ejecutan dentro de un **contenedor** en el servidor. El contenedor proporciona un entorno de ejecución para aplicaciones web en Java.

El contenedor es para aplicaciones web en Java lo que la **JVM (Java Virtual Machine)** es para las aplicaciones Java que se ejecutan locales. El contenedor en sí se ejecuta en la JVM.

En general, Java distingue dos contenedores: El **contenedor web** y el **contenedor Java EE**.

Un contenedor Web apoya la ejecución de Servlets Java y Java Server Pages.

Un contenedor compatible con Java EE provee funcionalidades adicionales, tales como, gestor de ejecución de los Enterprise JavaBeans, interfaz de conexión entre el servidor Java

EE y aplicaciones clientes.

Contenedores web típicos en el mundo Java son **Tomcat** o **Jetty**.

La especificación **Servlet** es el fundamento de las aplicaciones web Java. Muchos frameworks de alto nivel incluyendo **JAX-RS**, **JavaServer Faces (JSF)**, y **Spring MVC** están basados en **servlets** y **JavaServer Pages**.

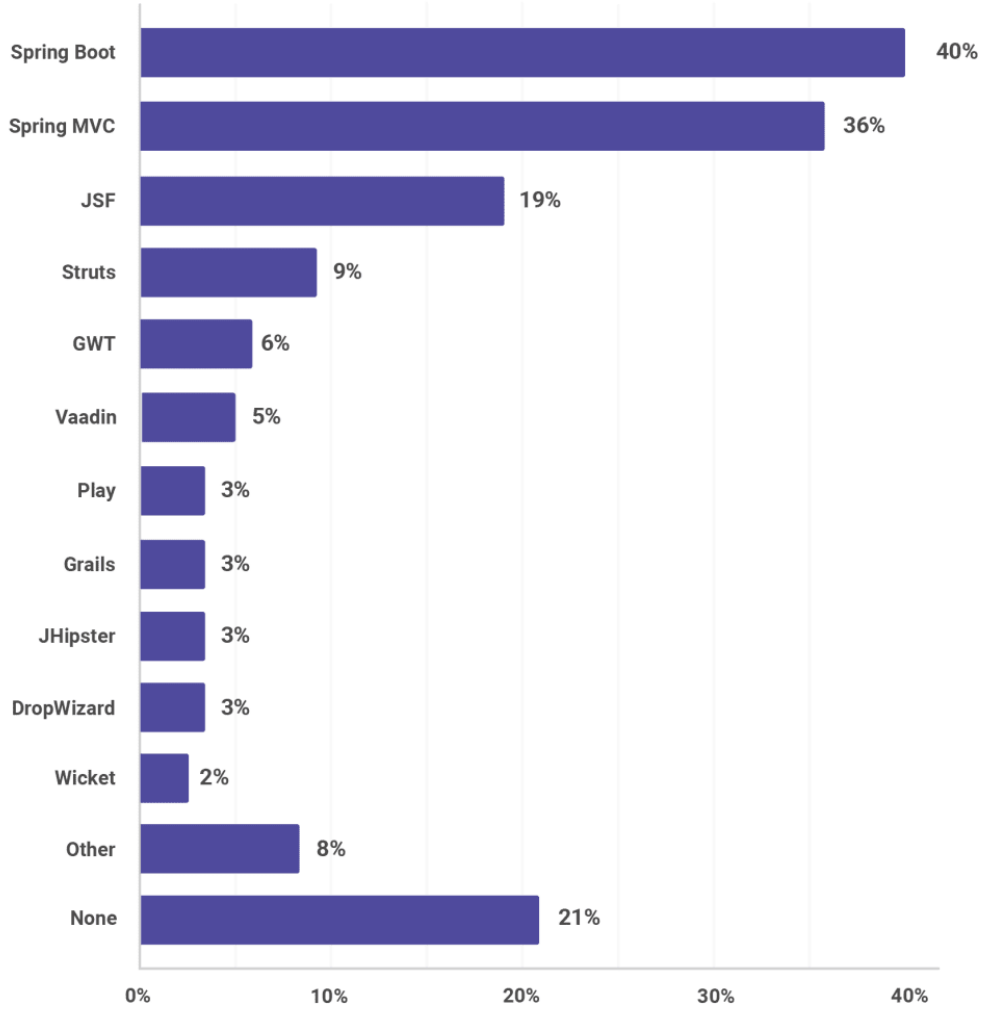
Los frameworks web populares en Java son : Spring Framework (varias vertientes) y Java Server Faces. Otros como **Vaadin**, **Grails** y **Play** tienen cierta presencia en los nichos de desarrollo web en Java, pero no han logrado escalar a los niveles esperados.

Estos frameworks web por lo general se ejecutan en un contenedor web.

Nota: Cuando hablamos de **framework** nos referimos a una estructura conceptual y tecnológica que sirve de base para desarrollar proyectos de software de forma más fácil y organizada, ya que agrupa otros componentes y herramientas para agilizar el proceso de desarrollo de un software.

Popularidad de los frameworks web en Java

1. https://zeroturnaround.com/webframeworksindex/
2. https://snyk.io/blog/jvm-ecosystem-report-2018



Guía del laboratorio

Este laboratorio está enfocado en el desarrollo de aplicaciones Web con Java utilizando el framework **Spring Boot**.

Reseña General

El desarrollo web evoluciona muy rápido y con frecuencia vemos que una herramienta que era bastante útil y usada, varios años después pierde la relevancia que tenía en años anteriores. El ecosistema Java no ha estado exento de esa realidad.

Por años, Java EE y Spring Framework se disputaban los mayores porcentaje entre el nicho de desarrolladores web Java, pero las necesidades de poder crear aplicaciones web más rápido y que corrieran en entornos ligeros trajo a escena un sin números de herramientas para

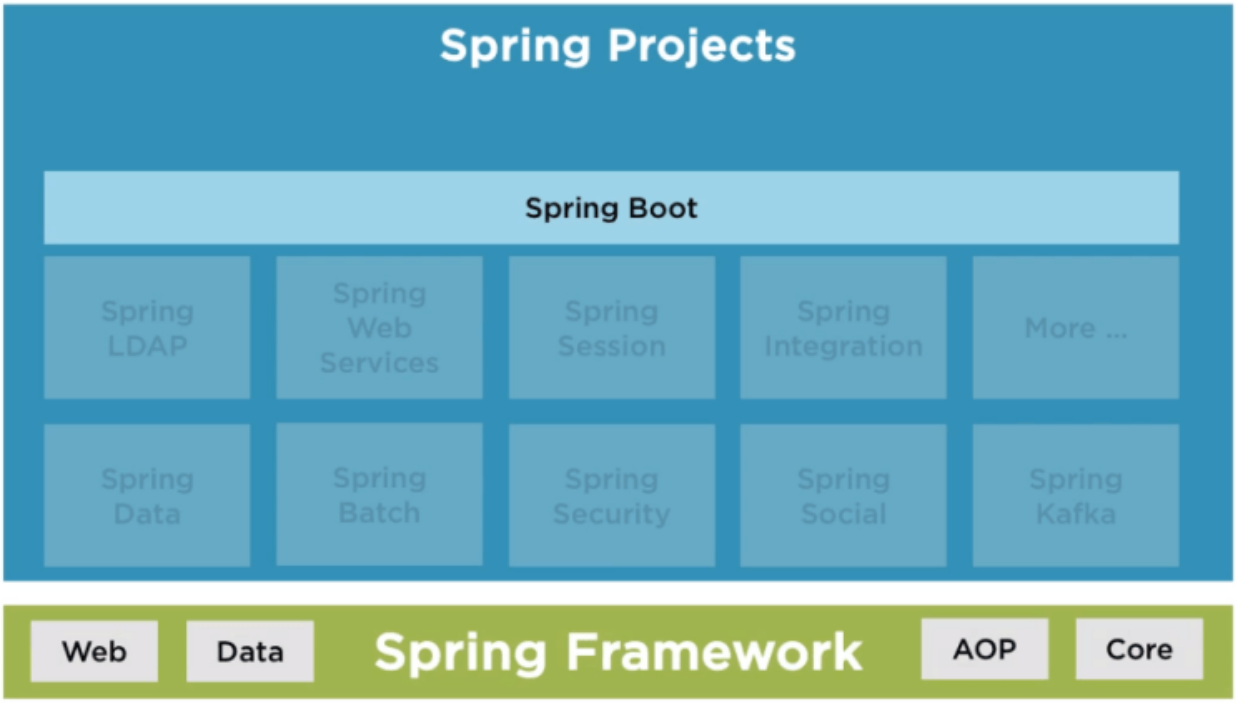
solucionar dicha problemática. Entre ellas Spark Java, Dropwizard, Spring Boot entre otras.

De las anteriores, Spring Boot ha sido la herramienta que más rápido ha escalado.

Qué es Spring Boot ?

Spring Boot es un proyecto construido usando como base el Spring Framework. Proporciona una forma más sencilla y rápida de inicializar, configurar, ejecutar aplicaciones simples y aplicaciones web complejas.

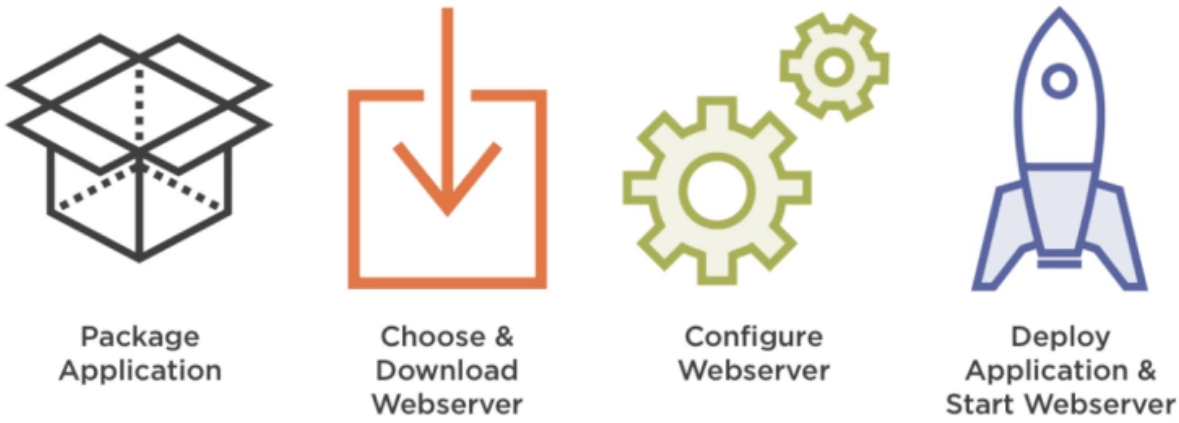
Spring Boot está diseñado para ponerlo en funcionamiento lo más rápido posible, con una configuración inicial mínima de Spring. Spring Boot tiene una opinión crítica de la creación de aplicaciones listas para la producción.



En el Spring Framework, necesita configurar todo por tí mismo. Por lo tanto, puede tener muchos archivos de configuración, como los descriptores XML. **Ese es uno de los principales problemas que Spring Boot resuelve para ti.**

El proceso tradicional para iniciar una aplicación web basada en Java

1. En primer lugar, necesita empaquetar su aplicación.
2. Elija qué tipo de servidores web desea usar y descárguelo. Hay muchas soluciones diferentes por ahí.
3. Es necesario configurar este servidor web específico.
4. Después de eso, debe organizar la implementación e iniciar su servidor web.



Con Spring Boot, necesitas el siguiente proceso:

1. Paquete de su aplicación.
2. Ejecútalo con un comando simple como **java -jar my-application.jar**

Realmente, es así de simple.

Spring Boot se encarga del resto al iniciar y configurar un servidor web incorporado e implementa su aplicación allí.

Características Notables

Configuración automática: configura su aplicación en función del entorno a su alrededor, así como sugerencias de lo que proporcionan los desarrolladores.

Independiente(Standalone): Literalmente, es completamente independiente. Por lo tanto, no es necesario implementar su aplicación en un servidor web o en ningún entorno especial. Su única tarea es hacer clic en el botón o dar el comando, y se iniciará.

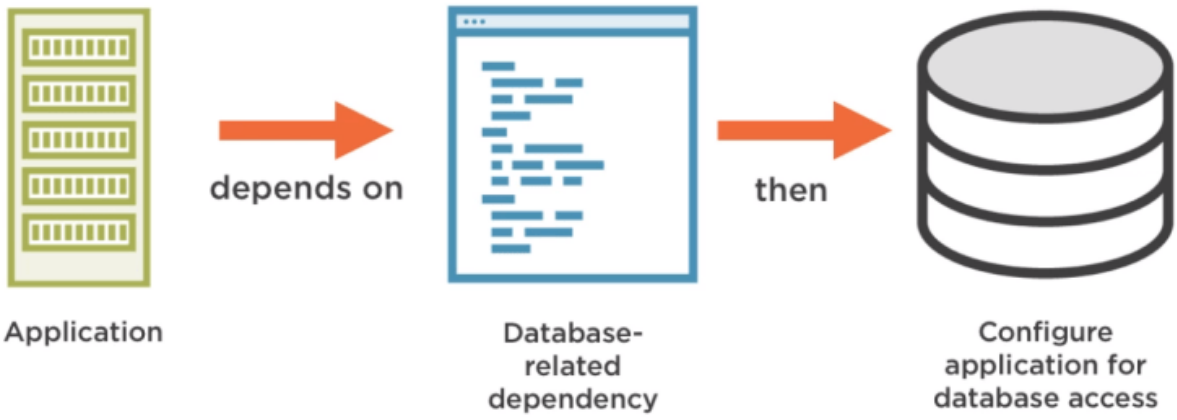
Dogmático(Opinionated): Esto significa que el marco elige cómo hacer las cosas por sí mismo.

Configuración automática inteligente

La autoconfiguración inteligente intenta configurar automáticamente su aplicación. Es

contextualmente consciente e inteligente. Veamos un ejemplo de acuerdo con una característica de la base de datos.

Si agrega una dependencia al pom.xml, que se relaciona con una base de datos, el marco asume que probablemente le gustaría usar una base de datos. Luego, configura automáticamente su aplicación para el acceso a la base de datos.



Además, si la dependencia aparece para una base de datos muy específica, por ejemplo, Oracle o MySQL. Puede hacer una suposición más cierta y probablemente configurará ese acceso a la base de datos específica exactamente lo que necesita exactamente.

Configurar la autoconfiguración es extremadamente sencillo. Solo necesita agregar la anotación @EnableAutoConfiguration a su aplicación Spring Boot.

Requerimientos

Java 1.8+ (Recomendado Java 8)

Netbeans 8.1+

Apache Maven 3.2.1+

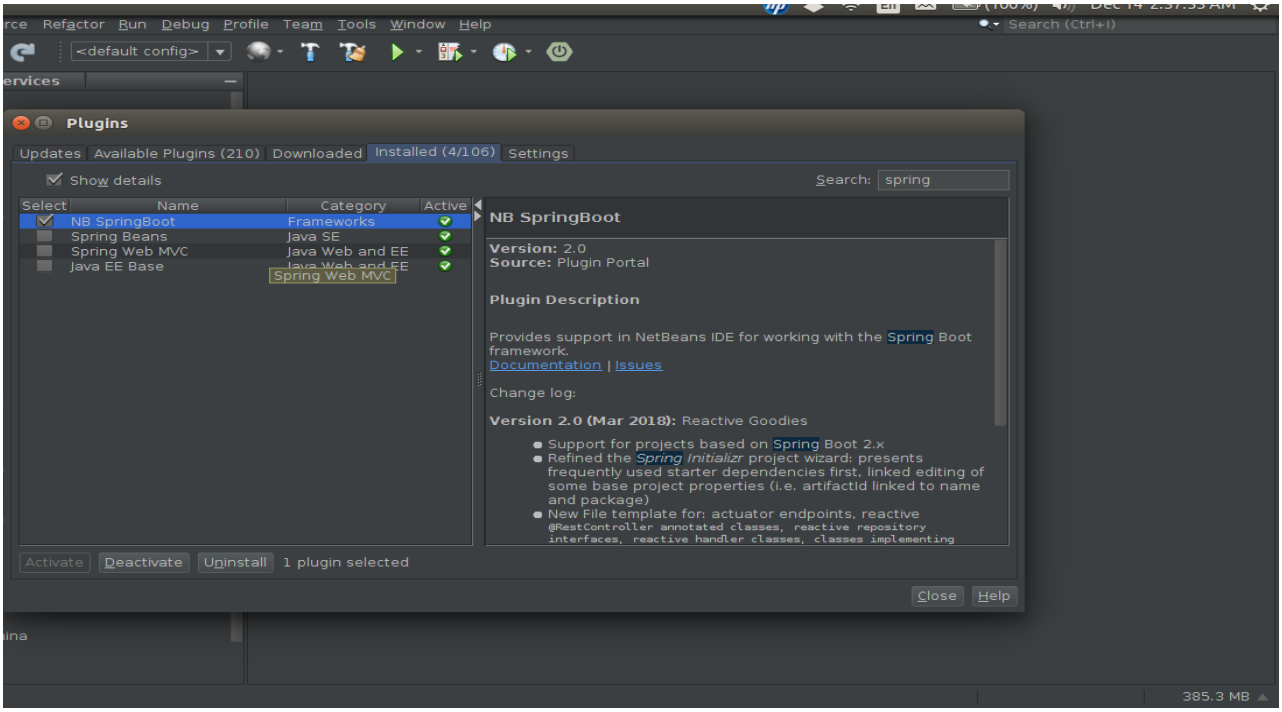
Mysql Server 5.0+ (Opcional)

Parte I:

1. **Instalar plugin de SpringBoot en netbeans.**
   Propósito:
   Instalar plugin de Spring Boot en netbeans para agregar soporte a Spring Boot.

Tiempo estimado: 5 – 10 minutos.

Pulsar **Tools > Plugins**
En el campo de búsqueda escribir NB Spring Boot, seleccionar y pulsar el botón de instalar.



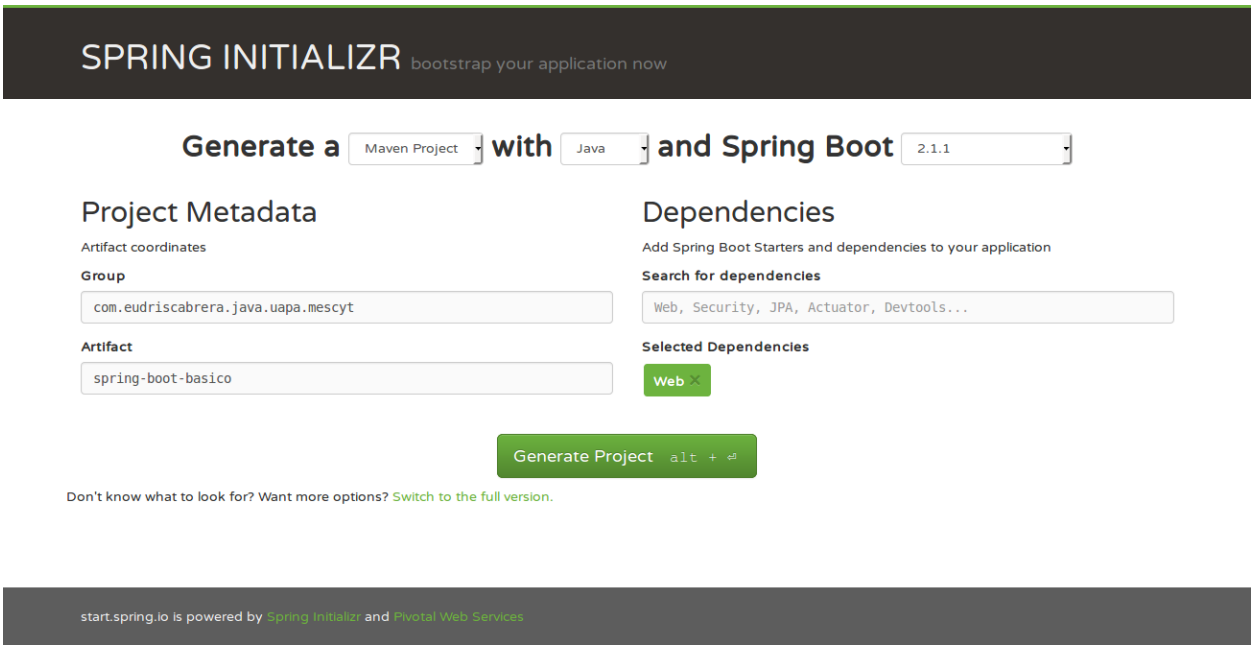
Luego de instalar el plugin de Spring Boot debe verificar que fue instalado correctamente.

2. Entrar a Spring Initializr y definir una aplicación básica

Propósito:
Usar el inicializador de Spring Boot https://start.spring.io para generar una aplicación básica.

Tiempo estimado: 5 – 10 minutos.

Información del proyecto
Tipo de Proyecto: **Maven**
Versión de Spring Boot : **2.2.1**
Grupo : **com.eudriscabrera.java.uapa.mescyt**
artefacto: **spring-boot-basico**
Dependencias: **Web**

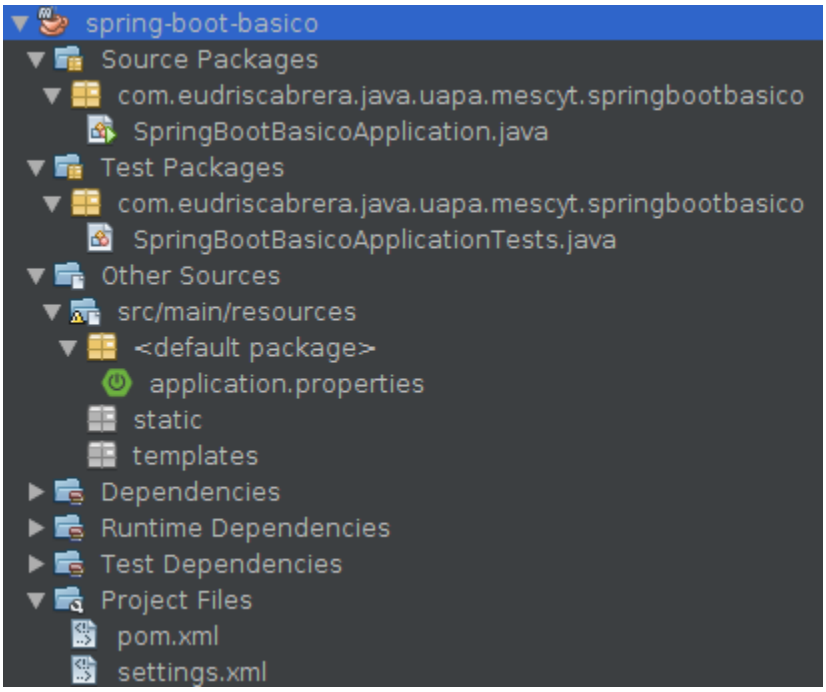


Pulsar botón de **generar proyecto** y descargar el archivo.

3. Recorrido por la Aplicación de ejemplo
Propósito: En esta sección se abrirá en netbeans la aplicación generada anteriormente . Un recorrido por la aplicación será llevado a cabo para proveer un entendimiento de la arquitectura de la aplicación.

Tiempo estimado: 15 – 30 minutos.
3.1 Descomprima el proyecto generado con el inicializador de Spring Boot en el directorio de trabajo. Esto creará un directorio llamado "spring-boot-basico".

3.2 En NetBeans IDE, Seleccione "**File**", "**Open Project**...", seleccione el directorio descomprimido y haga clic en "**Open Project**". La estructura del proyecto es mostrada a continuación**:**



3.3 Dependencias del proyecto: Expanda "Project Files" y haga doble clic en el archivo "pom.xml".
En este archivo se especifica la dependencia de Spring Boot para crear aplicaciones Web.

```
<dependencies>
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-test</artifactId>
                <scope>test</scope>
        </dependency>
  </dependencies>
```

La segunda dependencia nos permite crear pruebas unitarias (https://es.wikipedia.org/wiki/Prueba_unitaria) en la aplicación.

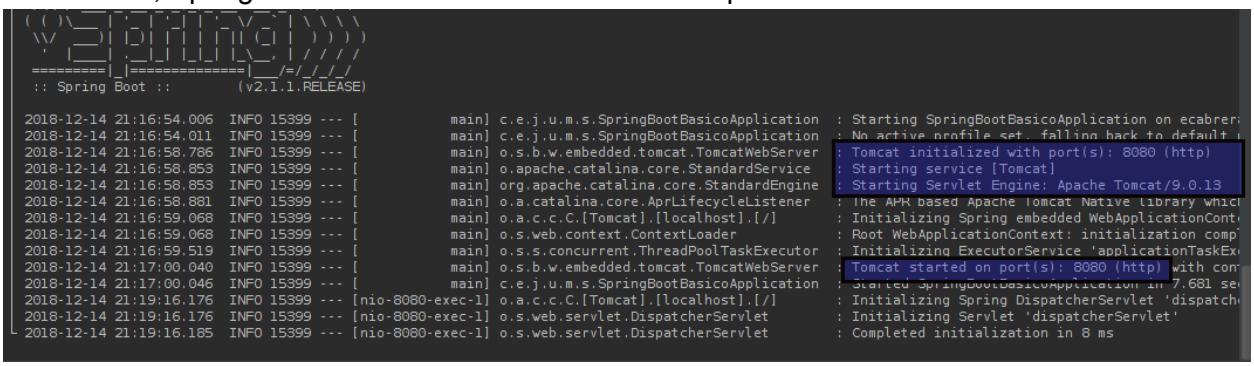
3.4. Clase para arrancar la aplicación.
Para la lanzar una aplicación Spring Boot es necesario colocar la anotación **@SpringBootApplication** en una clase y utilizar el método run() de la clase **SpringAplication SpringApplication.run(Class<?> clase, String ...args)**.

```java
package com.eudriscabrera.java.uapa.mescyt.springbootbasico;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringBootBasicoApplication {

	public static void main(String[] args) {
		SpringApplication.run(SpringBootBasicoApplication.class, args);
	}

}
```

3.5 Correr la aplicación

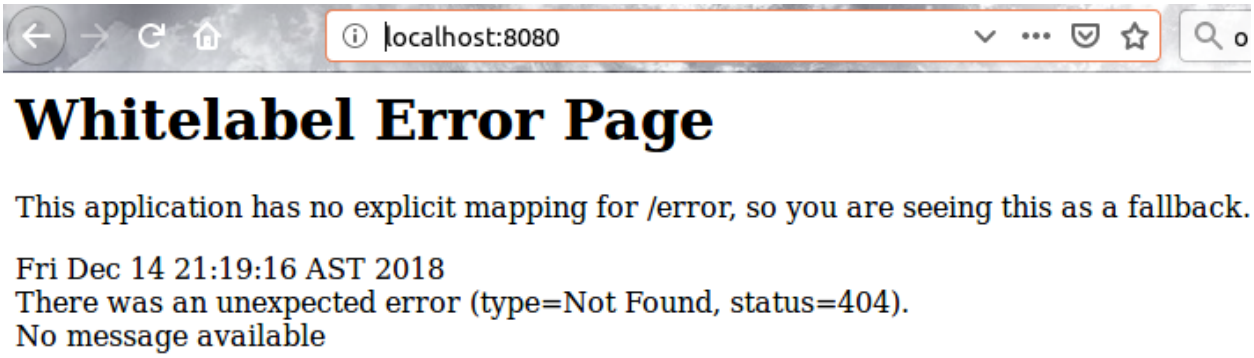
En Netbeans IDE, seleccione el proyecto "spring-boot-basico", click derecho y hacer click en la opción "run".

Por defecto, Spring Boot inicia un servidor tomcat en el puerto **8080**.



Para acceder a nuestra aplicación escribir en la barra de navegación el url http://localhost:8080/

Como todavía no se ha definido ningún archivo html presentará el siguiente mensaje.



4.0 Flujo de navegación en Spring Boot

Propósito: Agregarle contenido a nuestra aplicación y estudiar el flujo de navegación de una aplicación basada en Spring Boot.

Tiempo estimado: 15 – 30 minutos.

4.0 Página básica con parámetro

Propósito: Agregarle contenido a nuestra aplicación y estudiar el flujo de navegación de una aplicación basada en Spring Boot.

Tiempo estimado: 15 – 30 minutos.

4.1 Crear un archivo **index.html** en el directorio **static**. Todos los recursos estáticos **(html,css, javascript)** se colocan en ese directorio.

Contenido del archivo **index.html**

```
<!DOCTYPE html>
<html>
        <head>
        <title>Demo Spring Boot </title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        </head>
        <body>
        <p>Reciba su saludo <a href="/saludo">aqu&iacute;</a></p>
        </body>
</html>
```

4.2 Crear un controlador para el flujo de la aplicación, para esto, crear una clase llamada SaludoController.
Luego creamos un método con las siguientes características

```
public String saludos(){
  return "saludos.html";
}
```

Para que nuestra clase se convierta en un controlador se debe colocar la anotación
@Controller
y al método saludo tenemos que agregarle la anotación @RequestMapping

@Controller

```
public class SaludoController {

	@RequestMapping("/saludo")
	public String saludos(){
	return "saludos.html";
	}
}
```

Cuando alguien pulse el url "/saludo" el controlador lo redireccionará a la página saludos.html que vamos a crear a continuación.

Crear el archivo saludos.html en el directorio static

Contenido del archivo **saludos.html**

```
<!DOCTYPE html>
<html>
	<head>
	<title>Pagina de saludos</title>
	<meta charset="UTF-8">
	<meta name="viewport" content="width=device-width, initial-scale=1.0">
	</head>
	<body>
	<p>Enhorabuena !</p>
	</body>
</html>
```

Luego de hacer los pasos anteriores, limpiar, compilar y correr la aplicación nuevamente

4.3 Modificar los archivos del punto 4.2 para que permita pasar un parámetro de una página a otra.

Mover el archivo saludos.html al directorio templates y hacer las modificaciones al contenido de acuerdo al siguiente:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
	<head>
	<title>Pagina de saludos</title>
	<meta charset="UTF-8">
	<meta name="viewport" content="width=device-width, initial-scale=1.0">
	</head>
	<body>

	<p th:text="'Enhorabuena, ' + ${nombre} + '!'" />
```

```
        </body>
</html>
```

Esto indica que vamos a usar un manejador de plantilla llamado **thymeleaf** y que estamos pasando el parámetro nombre.

Debemos agregar la dependencia de **thymeleaf** a nuestro proyecto

```
 <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

En el método saludo de nuestro controller debemos hacer los siguientes cambios:

```
 @RequestMapping("/saludo")
 public String saludos(@RequestParam(value = "nombre",required = false, defaultValue =
"Eudris Cabrera") String nombre, Model model){
        model.addAttribute("nombre", nombre);

        return "saludos";
}
```

Definir un parámetro llamado nombre y asignarle un valor por defecto.

Luego de hacer los pasos anteriores, limpiar, compilar y correr la aplicación nuevamente.