



Spring

Sesión Conceptual 02

{desafío}
latam_







Inicio

{desafío}
latam_



15 minutos

*/**Crear proyecto controlador-vista que permita el despliegue de contenido estático.**/*

Objetivo



Desarrollo

{desafío}
latam_



90 minutos

/*Spring Framework*/

¿Qué es un Framework?

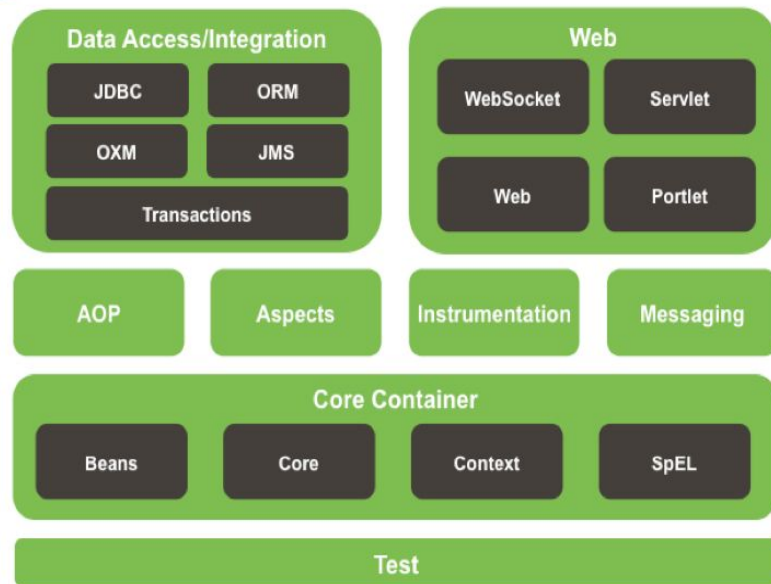
- Entorno de trabajo que provee al desarrollador:
 - Herramientas
 - Módulos
 - Librerías
 - Metodologías
 - Conceptos
- Permite reutilizar código reduciendo los tiempos para la generación de aplicaciones.

¿Qué es Spring Framework?

- Crear sistemas informáticos de alto rendimiento, livianos y reutilizables.
- Aportando herramientas para agilizar, estandarizar y resolver problemas.
- Sistema modular: posee módulos que agrupan distintas funcionalidades.

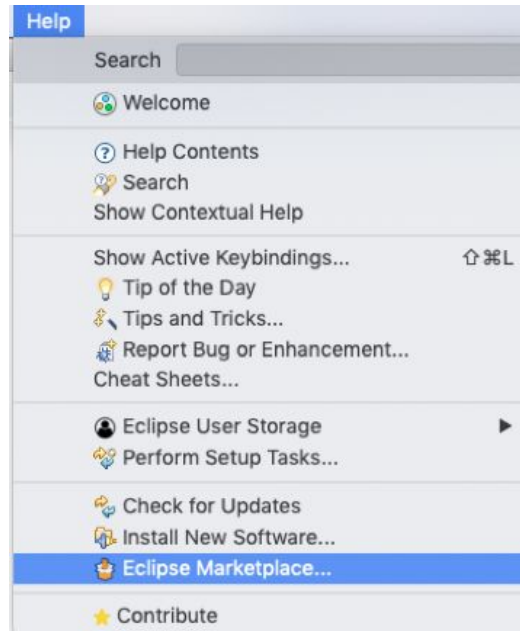


Spring Framework Runtime

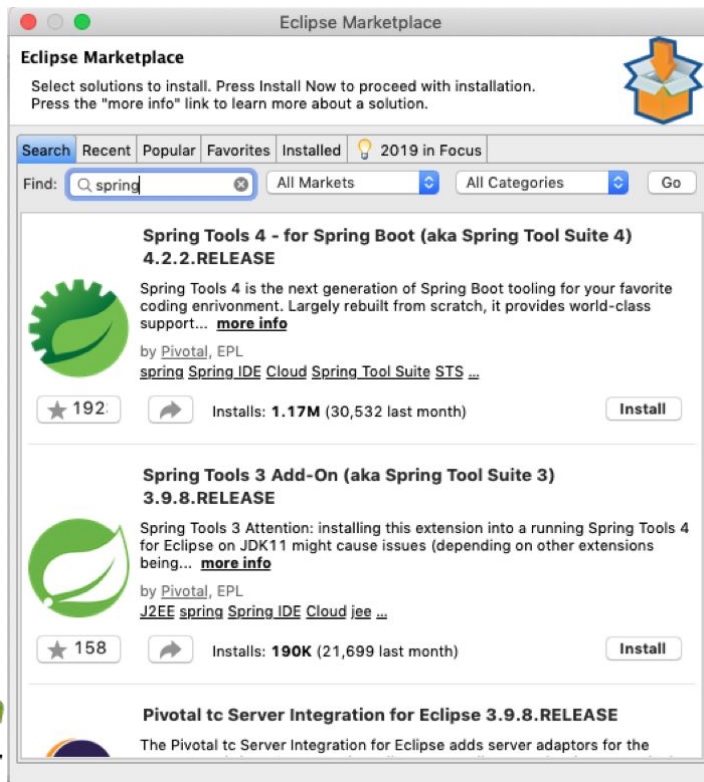


Maneras para crear proyectos spring:

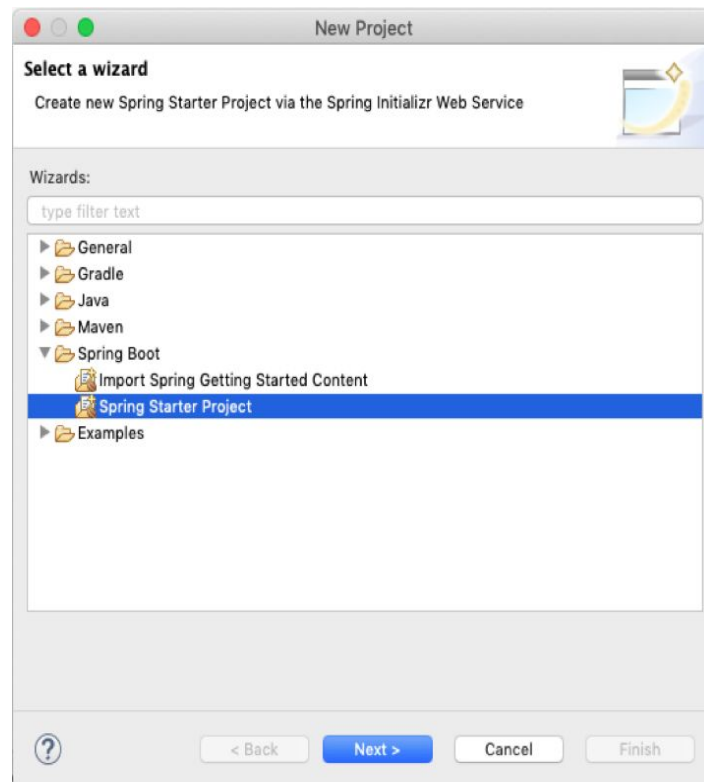
- **Primera opción :** agregar spring tools mediante Eclipse Marketplace.



Buscando Spring Tool 4



Verificando la incorporación de Spring Tool 4



- **Segunda opción: descargando STS.**



Spring Tools 4 for Eclipse

The all-new Spring Tool Suite 4.
Free. Open source.

Download STS4

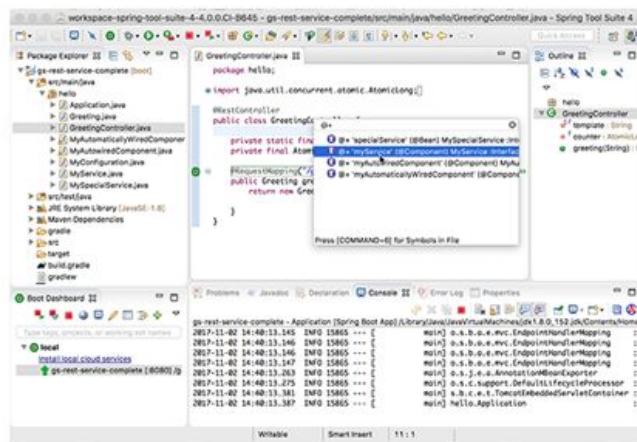
Linux 64-bit

Download STS4

macOS 64-bit

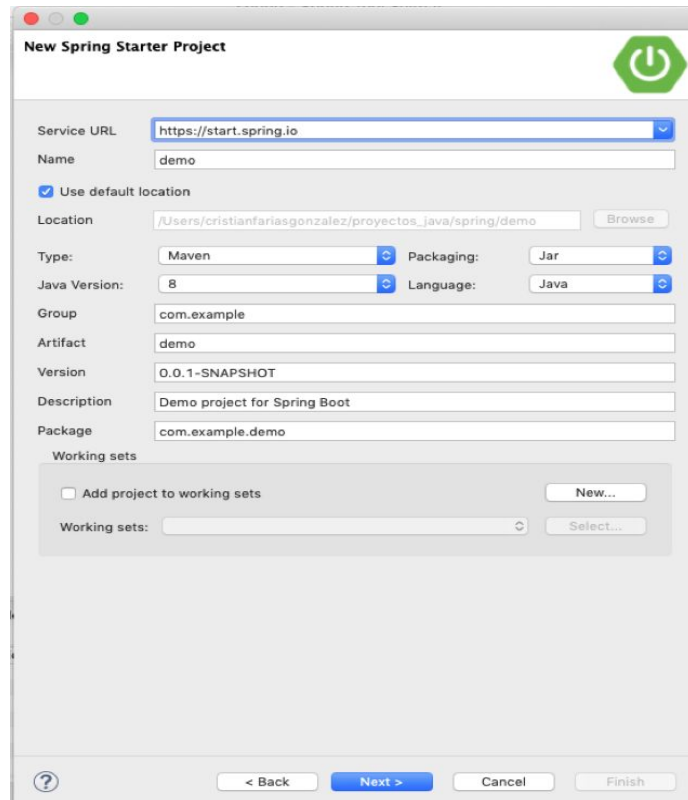
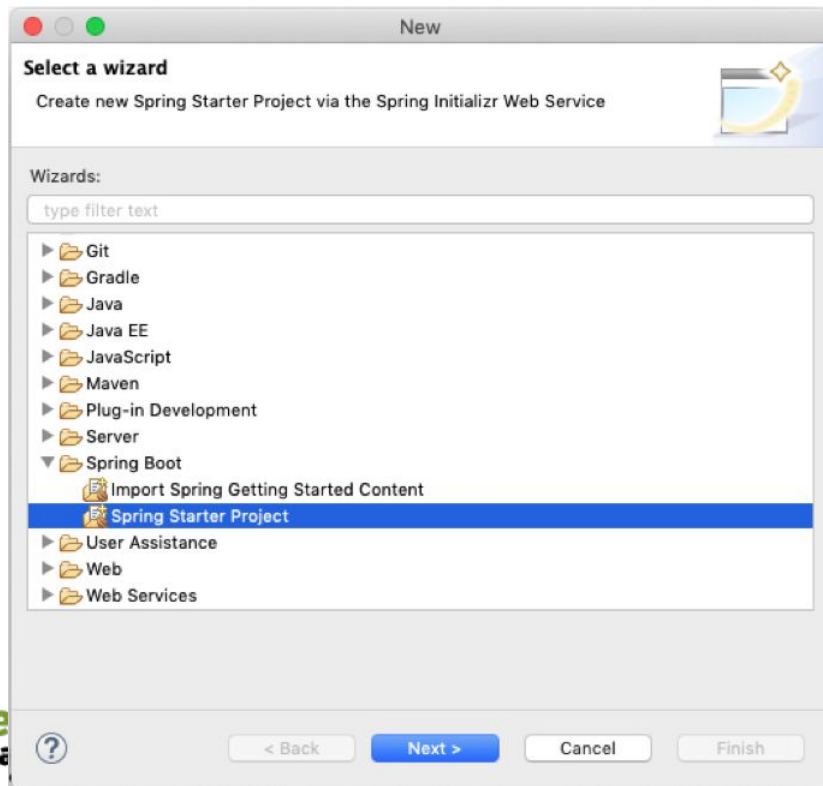
Download STS4

Windows 64-bit



Vista desde STS para crear un nuevo proyecto

Creando un nuevo proyecto



New Spring Starter Project Dependencies

Spring Boot Version:

Frequently Used:

☒ Web

Available:

Type to search dependencies

- ▶ NoSQL
- ▶ Ops
- ▶ Pivotal Cloud Foundry
- ▶ SQL
- ▶ Security
- ▶ Template Engines
- ▼ **Web**
 - ☒ Web
 - ☐ Reactive Web
 - ☐ Rest Repositories
 - ☐ Rest Repositories HAL Browser
 - ☐ HATEOAS
 - ☐ Web Services
 - ☐ Jersey
 - ☐ REST Docs

Selected:


X Web

{desafío}
latam_

Agregando
dependencias

- **Tercera opción:** descargar una herramienta spring lista.

← → ↻ 🏠 <https://start.spring.io>

**Spring Initializr**
Bootstrap your application

Project

Language

Spring Boot

Project Metadata

Dependencies
[See all](#)

Maven Project | Gradle Project

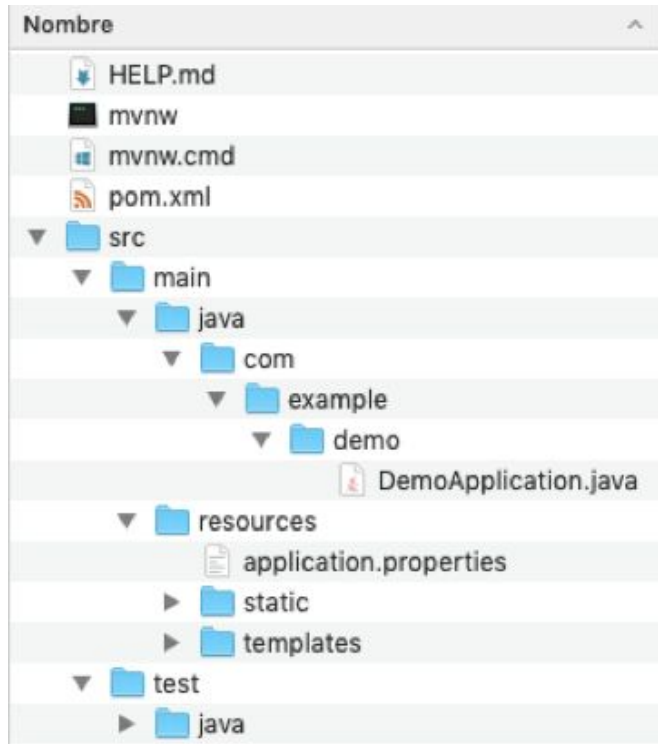
Java | Kotlin | Groovy

2.2.0 M3 | 2.2.0 (SNAPSHOT) | 2.1.6 (SNAPSHOT) | **2.1.5** | 1.5.21

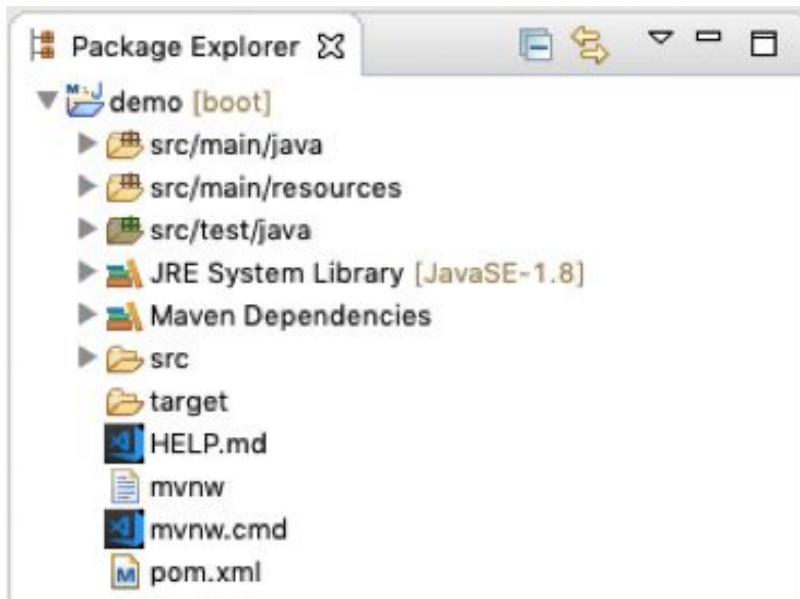
Group

Artifact

Search dependencies to add



Agregamos las
dependencias
necesarias y
generamos el
proyecto



Estructura
creada con
STS

- Automáticamente la generación de proyectos spring, inserta las dependencias necesarias para el proyecto.

```
<?xml version="1.0" encoding="UTF-8"?>
<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>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.5.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.example</groupId>
  <artifactId>demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>demo</name>
  <description>Demo project for Spring Boot</description>
```

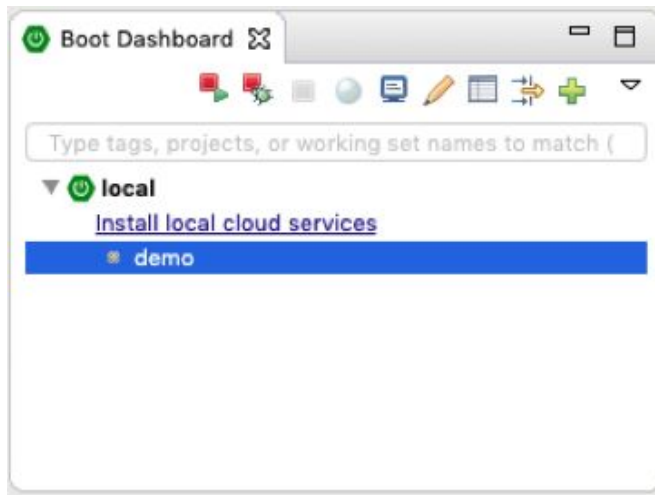
```
<properties>
  <java.version>1.8</java.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
</project>
```

Curiosidades del archivo POM.xml

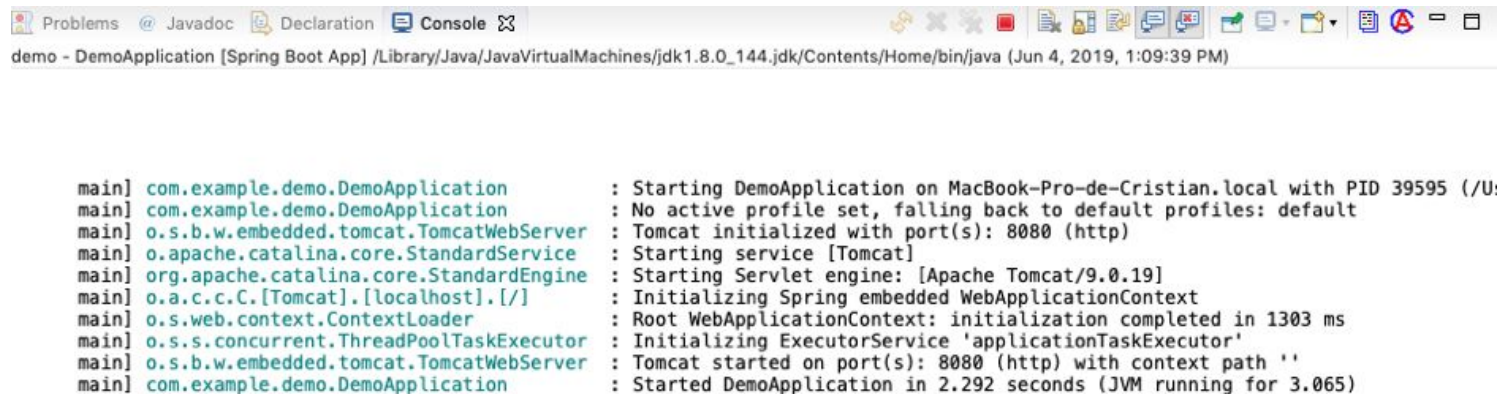
Etiqueta <parent>	Etiqueta <properties>	Etiqueta <plugin>
El proyecto que generamos, hereda todas las librerías y aplicaciones propuestas en el proyecto que hace referencia.	Propiedades que se le dan a la aplicación.	Incorporar artefactos empaquetados y listos para utilizar en cualquiera de los módulos de la aplicación.

Iniciando proyecto spring con STS

Boot DashBoard: aparece el título del directorio principal.



Información al levantar el servicio.



The screenshot shows an IDE console window with the following tabs: Problems, Javadoc, Declaration, and Console. The title bar of the console window reads: "demo - DemoApplication [Spring Boot App] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Jun 4, 2019, 1:09:39 PM)". The console output displays the startup sequence of the application, including the initialization of Tomcat and the Spring embedded web application context.

```
main] com.example.demo.DemoApplication : Starting DemoApplication on MacBook-Pro-de-Cristian.local with PID 39595 (/U:
main] com.example.demo.DemoApplication : No active profile set, falling back to default profiles: default
main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.19]
main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 1303 ms
main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
main] com.example.demo.DemoApplication : Started DemoApplication in 2.292 seconds (JVM running for 3.065)
```

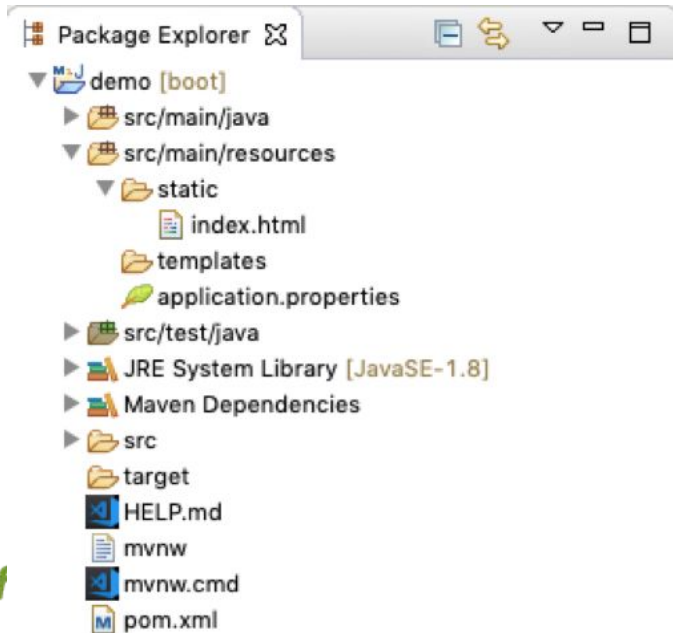
Inicia Spring embebido con la extensión de aplicación web.

Accediendo a la aplicación:

No hay nada que haga conexión aún con el servicio de spring y el servidor.



Incorporaremos un archivo HTML para mostrar una página estática.



Agregaremos un título

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title></title>
</head>
<body>
    <h1>Página de Inicio</h1>
</body>
</html>
```

Al reconocer un archivo de index.html.



Página de Inicio

Sí se agregaron nuevas funcionalidades que generan contenido dinámico, ahí sí debe ser reiniciada la aplicación.

/*Beans*/

¿Qué es IoC (Inversión de control)?

Permite inversión de dependencia sobre objetos.	Cambia el concepto de que el programador tenga que incorporar y crear clases.
Cambio en la forma de cómo se controla la aplicación.	Se encarga spring de incorporarlos en tiempo de ejecución del proyecto.
Posee un contenedor de IoC.	Se encarga de hacer la inyección de dependencias de las clases y objetos.
Núcleo de la aplicación.	Se encarga de relacionarlos, configurarlos y manejar su ciclo de vida

¿Qué es un Beans?

- Objeto (Plain Old Java Object, POJO) que almacena datos de nuestro programa. debe cumplir con lo siguiente:
 - Tener un constructor por defecto (sin argumentos).
 - Tener todas sus atributos privados.
 - Tener para cada atributo su método getter y setter.

¿Cómo manejo un Beans?

La primera forma de manejarlos por un archivo xml.

Ejemplo:

Persona que tiene dos atributos nombre y edad.

```
package com.beansEjemplos.beansxml;

public class Persona {
    private String nombre;
    private int edad;
    public Persona(){}
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public int getEdad() {
        return edad;
    }
    public void setEdad(int edad) {
        this.edad = edad;
    }
    public String toString() {
        return "Nombre: " + this.nombre + ", Edad:" +
this.edad;
    }
}
```

Creamos nuestro archivo beans.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd">
  <bean id="Persona"
class="com.beansEjemplos.beansxml.Persona">
    <property name="nombre" value="Cristian XML" />
    <property name="edad" value="29" />
  </bean>
</beans>
```

Beans hace la llamada a la ruta a la que está conectada y al nombre de la clase.



Estructura de archivos

Creamos el método Main y conectamos todo.

```
package com.beansEjemplos;
import org.springframework.context.ApplicationContext;
import
org.springframework.context.support.ClassPathXmlApplicationContext;
import com.beansEjemplos.beansxml.Persona;
public class BaseEjemplosApplication {
    public static void main(String[] args) {
        ApplicationContext appContext = new
        ClassPathXmlApplicationContext("com/beansEjemplos/beansxml/beans.xml");
        Persona p = appContext.getBean(Persona.class);
        System.out.println(p.toString());
    }
}
```

Obtendremos la siguiente salida:



The screenshot shows an IDE's console window with the following content:

```
<terminated> beansEjemplos - BeansEjemplosApplication [Spring Boot App] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Jun 16, 2019, 5:29:22 PM)
17:29:22.900 [main] DEBUG org.springframework.context.support.ClassPathXmlApplicationContext - Refreshing org.springframework.context.support.ClassPathXmlApplicationContext
17:29:23.063 [main] DEBUG org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loaded 1 bean definitions from class path resource [beans.xml]
17:29:23.089 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of single bean named 'usuario' with scope 'singleton'
Nombre: Cristian XML, Edad:29
```


Anotaciones en Spring

Palabras reservadas que ayudan a manipular la aplicación, categorizando componentes de manera funcional.

Spring Stereotypes: anotaciones para categorizar los componentes importantes.

@Component	Indica que un objeto es de tipo componente.
@Controller	Indica que un objeto es de tipo controlador.
@Repository	Indica que es un objeto de tipo repositorio y se encarga de implementar un almacén de datos.
@Service	Encargado de gestionar la operación de negocios, agrupa varias llamadas simultáneas a repositorios.

Tipo de anotaciones importantes:

@Configuration	Indica que la clase posee la configuración principal del proyecto.
@EnableAutoConfiguration	Indica que se aplicará la configuración automática.
@ComponentScan	Ayuda a localizar elementos etiquetados con otras anotaciones.
@SpringBootApplication	Engloba las tres anotaciones anteriores, @Configuration, @EnableAutoConfiguration y @ComponentScan.
@Autowired	Spring realiza la inyección de dependencia sobre el atributo seleccionado.
@PathVariable	Indica con qué variable de la URL se relaciona el parámetro que se está usando la anotación.
@Bean	Indica que la clase seleccionada es un bean.

Incorporando anotaciones

Generamos nuestro archivo de configuración (AppConfig.java).
Con el Ejemplo anterior haremos lo mismo pero con anotaciones.

```
package com.beansejemplo2.beans;
import org.springframework.context.annotation.Bean;
import
org.springframework.context.annotation.Configuration;
@Configuration
public class AppConfig {
    @Bean
    public Persona persona() {
        return new Persona();
    }
}
```

Diferencia con la declaración de los Beans en XML, se reemplaza el archivo XML del ejercicio anterior a esta pequeña declaración.

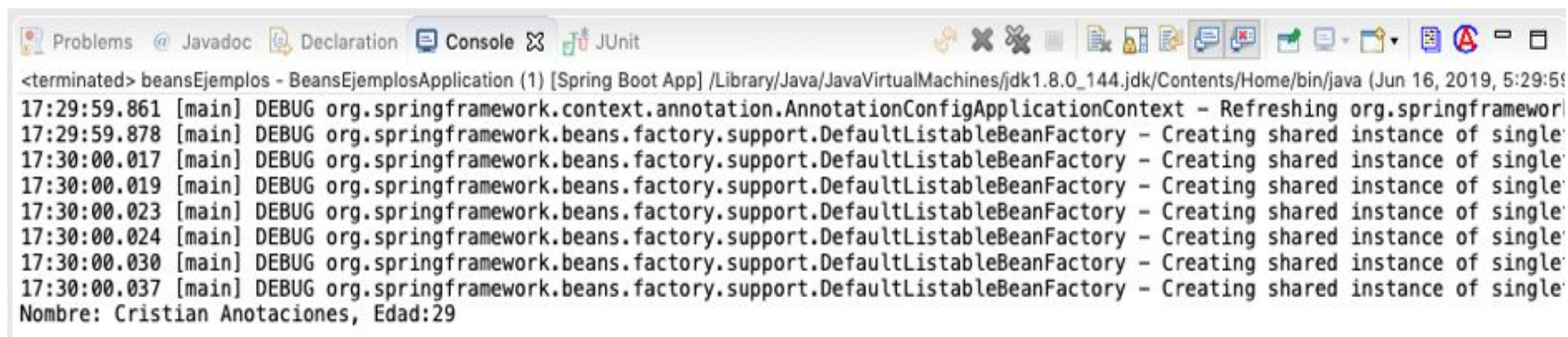
Cambiar el Main, ahora se llama a la clase AppConfig.

```
package com.beansejemplo2.beans;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import
org.springframework.context.annotation.AnnotationConfigApplicationContext;
import com.beansejemplo2.beans.Persona;
import com.beansejemplo2.beans.AppConfig;
@SpringBootApplication
public class Ejemplo2Application {
    public static void main(String[] args) {
        ApplicationContext appContext = new
AnnotationConfigApplicationContext(AppConfig.class);
        Persona p = appContext.getBean(Persona.class);
        System.out.println(p.toString());
    }
}
```

La clase Persona también fue modificada, para agregar valores a esta con la notación @Value.

```
package com.beansejemplo2.beans;
import org.springframework.beans.factory.annotation.Value;
public class Persona {
    @Value("Cristian Anotaciones")
    private String nombre;
    @Value("29")
    private int edad;
    Persona(){}
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public int getEdad() {
        return edad;
    }
    public void setEdad(int edad) {
        this.edad = edad;
    }
    public String toString() {
        return "Nombre: " + this.nombre + ", Edad:" + this.edad;
    }
}
```

Resultando usando anotaciones.



The screenshot shows an IDE's console window with the following tabs: Problems, Javadoc, Declaration, Console, and JUnit. The console output is as follows:

```
<terminated> beansEjemplos - BeansEjemplosApplication (1) [Spring Boot App] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Jun 16, 2019, 5:29:59)
17:29:59.861 [main] DEBUG org.springframework.context.annotation.AnnotationConfigApplicationContext - Refreshing org.springframework.context.annotation.AnnotationConfigApplicationContext
17:29:59.878 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'org.springframework.context.annotation.AnnotationConfigApplicationContext'
17:30:00.017 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'org.springframework.context.annotation.AnnotationConfigApplicationContext'
17:30:00.019 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'org.springframework.context.annotation.AnnotationConfigApplicationContext'
17:30:00.023 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'org.springframework.context.annotation.AnnotationConfigApplicationContext'
17:30:00.024 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'org.springframework.context.annotation.AnnotationConfigApplicationContext'
17:30:00.030 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'org.springframework.context.annotation.AnnotationConfigApplicationContext'
17:30:00.037 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creating shared instance of singleton bean 'org.springframework.context.annotation.AnnotationConfigApplicationContext'
Nombre: Cristian Anotaciones, Edad:29
```



**Estructura
resultante del
proyecto**

Observaciones

- La diferencia entre la estructura de beans mediante xml y la de beans mediante anotaciones son el archivo que utilizan para conectar las respectivas clases de Persona.
- El primero se basa en un archivo XML, el segundo se basa en un archivo Java.

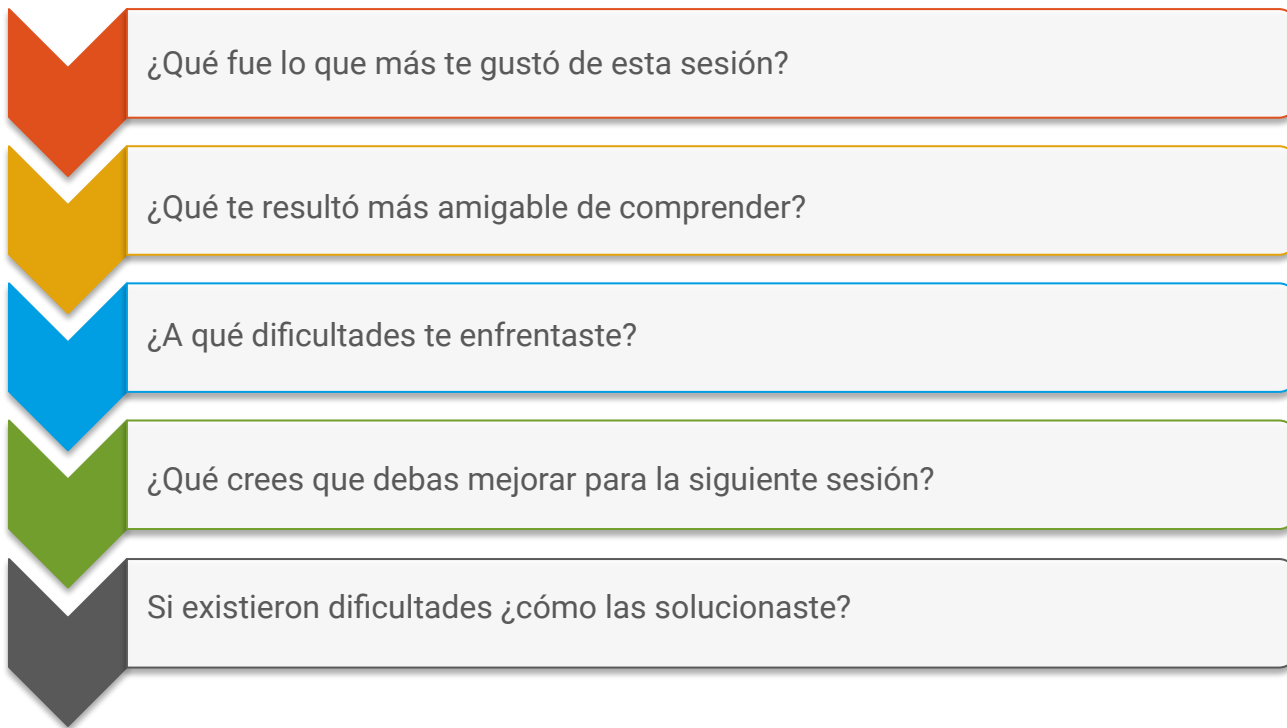


Cierre

{desafío}
latam_



15 minutos



¿Qué fue lo que más te gustó de esta sesión?

¿Qué te resultó más amigable de comprender?

¿A qué dificultades te enfrentaste?

¿Qué crees que debas mejorar para la siguiente sesión?

Si existieron dificultades ¿cómo las solucionaste?



*Academia de
talentos digitales*

www.desafiolatam.com



/DesafioLatam



/DesafioLatam



/DesafioLatam



/DesafioLatam