



Spring

Sesión Conceptual 02

{desafío}
latam_







Inicio

{desafío}
latam_



5 minutos

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

Objetivo



Desarrollo

{desafío}
latam_



50 minutos

/*Logs*/

¿Qué es un Log?

- Archivo de texto plano que almacena de manera cronológica acontecimientos o cambios que han afectado a la aplicación.
- Permite detectar dónde y cuándo se produjeron errores.

Niveles de información

DEBUG	Mensajes de depuración,útiles para obtener el comportamiento mientras se desarrolla la aplicación. (color verde).
INFO	Mensajes que muestran información del programa durante la ejecución. (color verde).
WARN	Mensajes de alerta por situaciones anómalas de la ejecución, pero no afectan el funcionamiento. (color amarillo).
ERROR	Mensajes que muestran una situación de errores en la ejecución, el programa puede ser afectado.(color rojo).
FATAL	Mensajes Críticos, hace que el programa aborte la ejecución y se detenga. (color rojo).

Generando logs

- Utilizaremos la librería slf4j.
- Creamos un proyecto Spring, exportamos las dependencias web, debemos agregar configuración en la aplicación, editamos el archivo de propiedades de la aplicación, en este archivo agregamos información básica del log.

```
logging.file=app.log  
logging.file.max-size=10MB  
logging.level.root=INFO  
logging.level.org.springframework.web=DEBUG
```

Cuatro configuraciones básicas

Logging.file	Donde se van a almacenar estos logs.
Logging.file.max-size	Tamaño máximo que tendrá el archivo.
Logging.level.roo	Nivel máximo que almacenará el logger para la aplicación base.
Logging.level.org.springframework.web	Lo mismo que el anterior, pero es para la aplicación web.

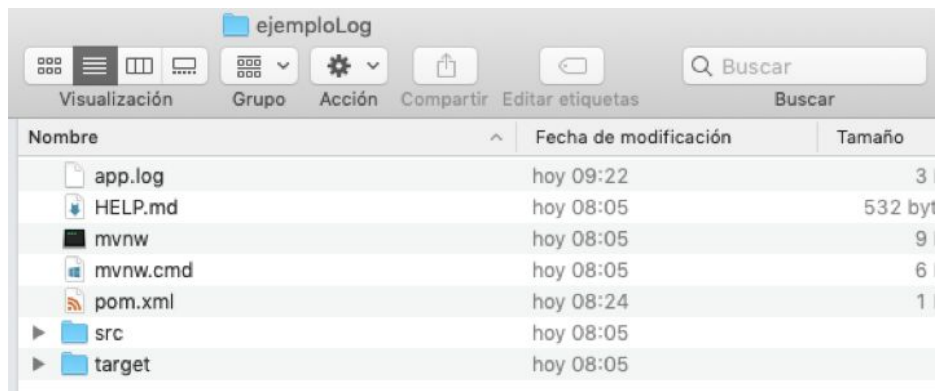
Niveles de Logging

Debug	Informar eventos específicos útiles para depurar una aplicación.
Info	Mensajes útiles para registrar el progreso de la aplicación.
Warn	Alerta de situaciones que posiblemente perjudiquen el funcionamiento normal de nuestra aplicación.
Error	Informar eventos de error que nos podrían permitir mantener la aplicación en ejecución.
Fatal	Eventos de error muy graves que obligan a la aplicación a detenerse.

- Logger:

```
package com.log.ejemploLog;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class App {
    private final static Logger logger = LoggerFactory.getLogger(App.class);
    public static void main(String[] args) {
        SpringApplication.run(App.class, args);
        logger.debug("debug log");
        logger.info("info log");
        logger.warn("warning log");
        logger.error("error log");
    }
}
```

- Raíz de nuestro proyecto.



- Revisando el contenido del Log.

```
2019-06-05 09:22:34.020 INFO 54359 --- [main] com.log.ejemploLog.App : Starting App on MacBook-Pro-de-Cristian.local with PID 54359 (/Users/cristianfariasgonzalez/proyectos_java/spring/ejemploLog/target/classes started by cristianfariasgonzalez in /Users/cristianfariasgonzalez/proyectos_java/spring/ejemploLog)
2019-06-05 09:22:34.023 INFO 54359 --- [main] com.log.ejemploLog.App : No active profile set, falling back to default profiles: default
2019-06-05 09:22:34.987 INFO 54359 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2019-06-05 09:22:35.015 INFO 54359 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2019-06-05 09:22:35.016 INFO 54359 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.19]
2019-06-05 09:22:35.099 INFO 54359 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2019-06-05 09:22:35.099 INFO 54359 --- [main] o.s.web.context.ContextLoader : Published root WebApplicationContext as ServletContext attribute with name [org.springframework.web.context.WebApplicationContext.ROOT]
2019-06-05 09:22:35.099 INFO 54359 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 1036 ms
2019-06-05 09:22:35.230 DEBUG 54359 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Patterns [/=/favicon.ico] in 'faviconHandlerMapping'
2019-06-05 09:22:35.386 INFO 54359 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2019-06-05 09:22:35.400 DEBUG 54359 --- [main] s.w.s.m.m.a.RequestMappingHandlerAdapter : ControllerAdvice beans: 0 @ModelAttribute, 1 @InitBinder, 1 RequestBodyAdvice, 1 ResponseBodyAdvice
2019-06-05 09:22:35.460 DEBUG 54359 --- [main] s.w.s.m.m.a.RequestMappingHandlerMapping : 2 mappings in 'requestMappingHandlerMapping'
2019-06-05 09:22:35.474 DEBUG 54359 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Patterns [/webjars/**, /**] in 'resourceHandlerMapping'
2019-06-05 09:22:35.484 DEBUG 54359 --- [main] m.m.a.ExceptionHandlerExceptionResolver : ControllerAdvice beans: 0 @ExceptionHandler, 1 ResponseBodyAdvice
2019-06-05 09:22:35.645 INFO 54359 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2019-06-05 09:22:35.650 INFO 54359 --- [main] com.log.ejemploLog.App : Started App in 2.029 seconds (JVM running for 2.713)
2019-06-05 09:22:35.652 INFO 54359 --- [main] com.log.ejemploLog.App : info log
2019-06-05 09:22:35.652 WARN 54359 --- [main] com.log.ejemploLog.App : warning log
2019-06-05 09:22:35.653 ERROR 54359 --- [main] com.log.ejemploLog.App : error log
```

/*Proyecto Controlador-Vista*/

¿Qué es un MVC?

- Modelo Vista Controlador.
- Forma de organizar un software.
- Separa los datos de la aplicación.

Estructuras de Archivos

- **Vistas** -> ./src/main/webapp/WEB-INF/views/
- **Contenido estático** -> ./src/main/resources/static/
- **Controladores** -> ./src/main/java/<nombre.del.paquete>/controller/
- **Modelos** -> ./src/main/java/<nombre.del.paquete>/model/
- **Servicios** -> ./src/main/java/<nombre.del.paquete>/service/

Enunciado

- Crear un proyecto web que liste información desde un archivo propuesto (data.txt).

```
PRODUCTO UNO  
PRODUCTO DOS  
PRODUCTO TRES
```

Solución:

Configuramos el archivo pom.xml agregando una nueva dependencia que nos permitirá manejar la conexión con los archivos html.

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

Archivo POM.xml:

```
<?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.proyectoFinal.productos</groupId>
  <artifactId>productos</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>productos</name>
  <description>Proyecto que lista productos</description>
  <properties>
    <java.version>1.8</java.version>
  </properties>
```

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
<build>
```

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
</project>
```

Creamos un controlador, este se encargará de resolver las respuestas y entregar un resultado.

```
package com.proyectoFinal.productos;
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.ArrayList;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
@Controller
public class HomeController {
    private final static Logger logger =
LoggerFactory.getLogger(HomeController.class);
    @RequestMapping("/")
    public String Main(Model modelo) {
        String nombre = "src/main/resources/static/data.txt";
        ArrayList<String> p = new ArrayList<String>();
        try {L
```

```
        FileReader fr = new FileReader(nombre);
        BufferedReader br = new BufferedReader(fr);
        String data = br.readLine();
        while (data != null) {
            p.add(data);
            data = br.readLine();
        }
        br.close();
        fr.close();
    } catch (Exception e) {
        logger.error("Error leyendo el fichero "+ nombre + ": " + e);
    }
    modelo.addAttribute("nombre1",p.get(0));
    modelo.addAttribute("nombre2",p.get(1));
    modelo.addAttribute("nombre3",p.get(2));
    return "main";
}
}
```

Archivo HomeController

- **@Controller:** Anotación que denomina a la clase como tipo controlador.
- **@RequestMapping("/"):** Anotación que resuelve la llamada desde el navegador.
- **Main(Model modelo):** Nombre que recibe el método de este controlador, el atributo Model.
- **Modelo.addAttribute("nombre",valor):** Corresponde al método que agrega un atributo al modelo.
- **Return "main":** Resuelve la respuesta que será enviada al navegador, en este caso al archivo main.html.

Creamos el archivo main.html:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Productos</title>
</head>
<body>
<h1>Página de productos</h1>
<p th:text="'Producto 1: ' + ${nombre1} + '!'" />
<p th:text="'Producto 2: ' + ${nombre2} + '!'" />
<p th:text="'Producto 3: ' + ${nombre3} + '!'" />
</body>
</html>
```


Ejecutamos nuestra aplicación, desde el método Main:

```
package com.proyecto.productos;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Resultado



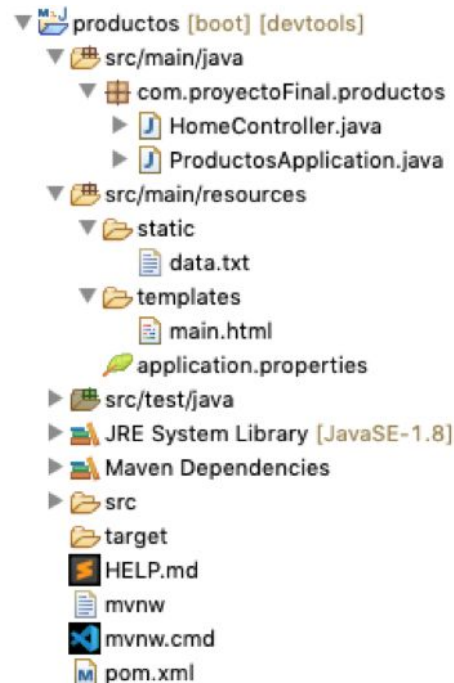
Página de productos

Producto 1: PRODUCTO UNO!

Producto 2: PRODUCTO DOS!

Producto 3: PRODUCTO TRES!

Estructura final del proyecto





Quiz

{desafío}
latam_



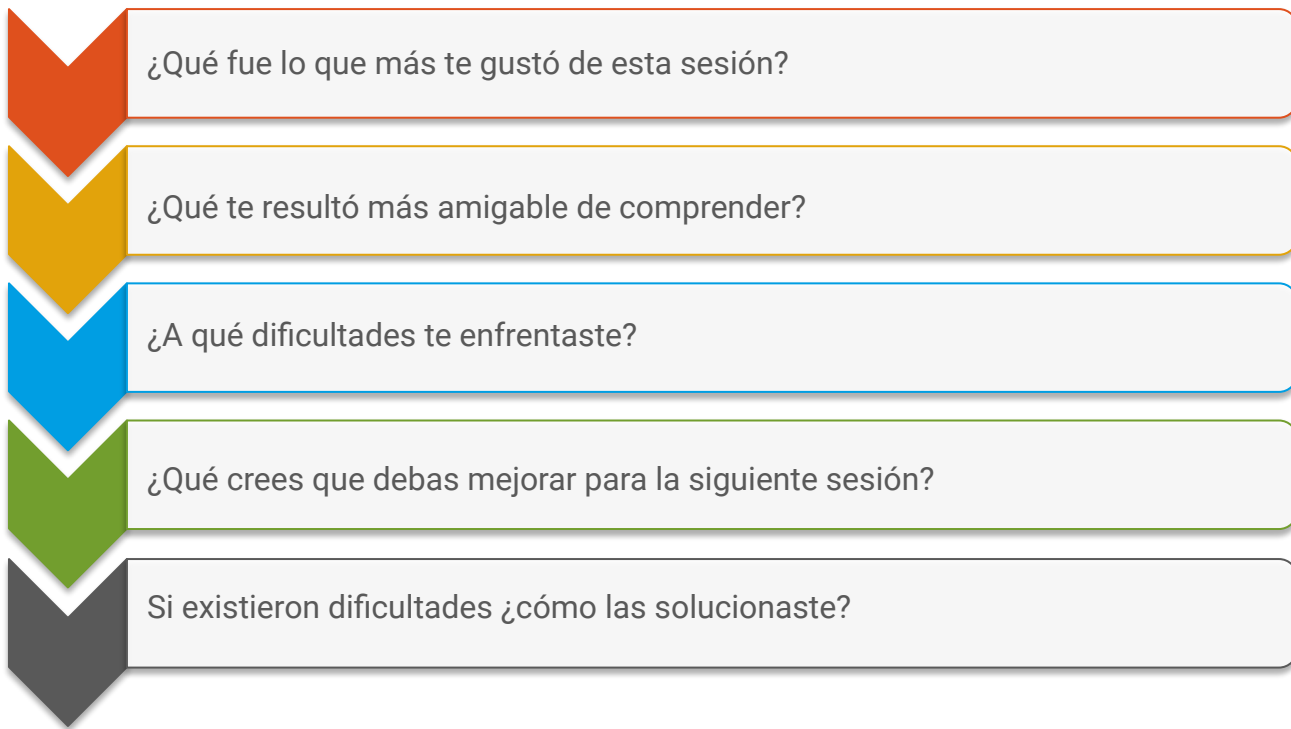


Cierre

{desafío}
latam_



5 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