Clase grabada

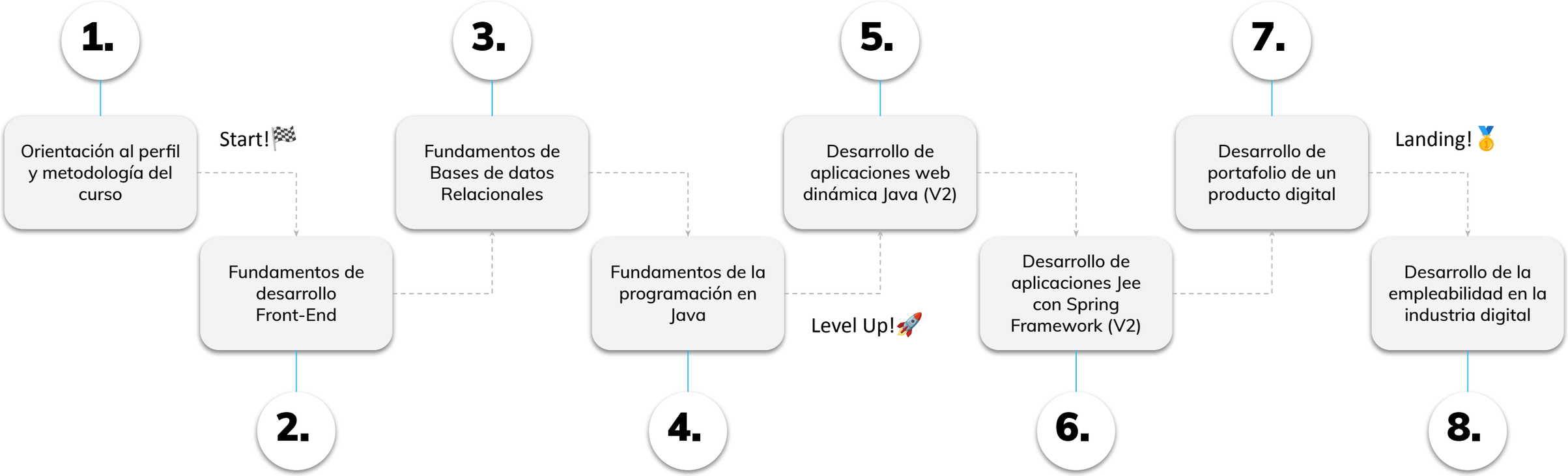
El gestor de Proyectos

Plan formativo: Desarrollo de Aplicaciones Full Stack Java Trainee V2.0

English Always | alkemy

HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



English Always | alkemy

REPASO CLASE ANTERIOR

En la clase anterior trabajamos 📚:

- ✔ *Administración de aplicaciones en Tomcat Manager*

English Always | alkemy

LEARNING PATHWAY

¿Sobre qué temas trabajaremos?

6.1

El gestor de Proyectos

Gestor de Proyectos

Dependencias

Qué es un gestor de proyectos y sus beneficios.
Características generales de Maven.
Un overview al Modelo de Objeto de Proyecto (POM).
Creación de un proyecto: Java, SpringMVC, SpringBoot.

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?

- *Aprender el concepto de gestor de proyectos*
- *Conocer Maven y sus características*
- *Comprender el Modelo de Objeto Proyecto (POM)*

Gestor de proyectos

¿Qué es un gestor de proyectos?

Un gestor de proyectos es una aplicación utilizada tanto por empresas como por equipos independientes para **gestionar los diferentes elementos y etapas por los que puede pasar un proyecto**. Estos ayudan a declarar y visualizar por cuáles fases ha de pasar un proyecto, qué tareas han de realizarse y quien ha de realizarlas.

¿Qué es un gestor de proyectos?

Los gestores de proyectos son muy utilizados en el ámbito del desarrollo software permitiendo aplicar de forma más eficiente una metodología y ayudando a mantener el control sobre los diferentes elementos de un proyecto de software.

¿Qué es un gestor de proyectos?

En general, el uso de un gestor de proyectos de software agiliza y mejora la gestión de proyectos, permitiendo una mayor eficiencia, colaboración, control y seguimiento de las actividades. Ayuda a minimizar los riesgos y a maximizar las posibilidades de éxito en la ejecución de proyectos.

Beneficios

Los gestores de proyectos en general ofrecen una serie de beneficios que son aplicables a la mayoría de las herramientas de gestión de proyectos, sin importar cuál se utilice. Estos beneficios incluyen:

- **Gestión de dependencias**: Facilitan la gestión de las dependencias de un proyecto, lo que significa que ayudan a administrar las bibliotecas y componentes necesarios para el desarrollo del software. Esto asegura que las versiones correctas de las dependencias estén disponibles y se integren sin problemas.

- **Automatización de tareas**: Automatizan tareas repetitivas y procesos de construcción, lo que reduce la posibilidad de errores humanos y garantiza que las tareas se realicen de manera coherente.

Beneficios

- **Estandarización**: Ofrecen un marco de trabajo estándar y un ciclo de vida de proyecto definido que facilita la colaboración y garantiza que todos los miembros del equipo sigan un proceso coherente.

- **Gestión de dependencias transitorias**: Manejan las dependencias transitivas, es decir, las dependencias de las dependencias, simplificando aún más la gestión de las bibliotecas y componentes necesarios.

- **Generación de informes y documentación:** Facilitan la generación de informes sobre el estado del proyecto y la creación de documentación, lo que ayuda a mantener a los desarrolladores y usuarios informados.

Beneficios

- **Personalización**: Permiten la personalización de la configuración del proyecto, lo que significa que puedes adaptar las herramientas a las necesidades específicas de tu proyecto, definiendo tareas, plugins y reglas de construcción personalizadas.

- **Compatibilidad**: Pueden ser utilizados en una variedad de lenguajes y plataformas, lo que los hace versátiles y adecuados para una amplia gama de proyectos.

- **Comunidad y soporte**: Suelen contar con comunidades activas y una amplia base de usuarios, lo que significa que hay recursos, plugins y documentación disponibles. Además, a menudo tienen un ecosistema de extensiones y complementos que pueden adaptarse a necesidades específicas.

Beneficios

En resumen, los gestores de proyectos ofrecen ventajas clave para simplificar y mejorar el proceso de desarrollo de software, independientemente de la herramienta específica utilizada. Estos beneficios contribuyen a una gestión eficiente de proyectos y a la optimización de los flujos de trabajo de desarrollo.

Maven

Maven™

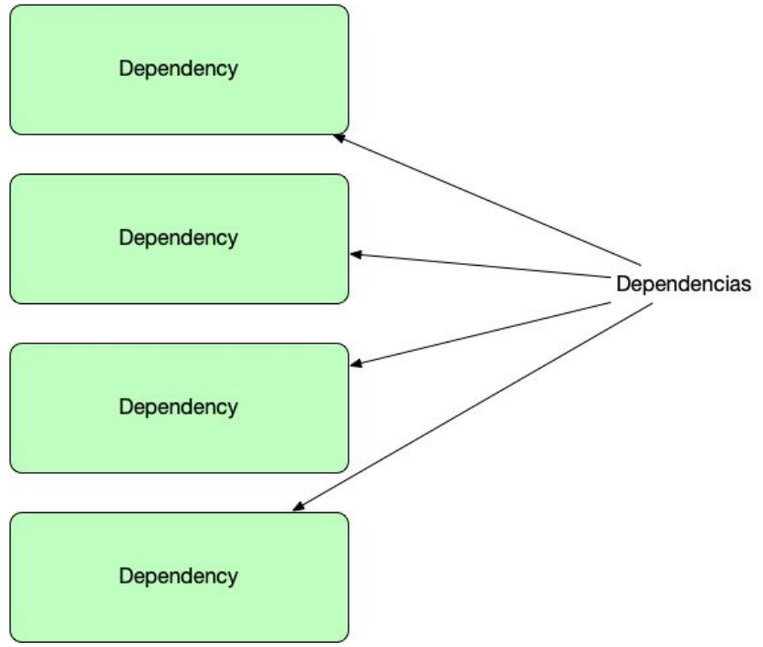
Maven es una herramienta ampliamente utilizada en el desarrollo de aplicaciones Java. Aunque no es un gestor de proyectos en el sentido tradicional, se puede considerar como una herramienta que facilita la gestión de proyectos de software.

Maven se utiliza principalmente para la construcción, gestión de dependencias y generación de informes en proyectos Java. Proporciona una estructura y una convención para organizar el código fuente, las bibliotecas de terceros y los recursos del proyecto.

Características generales de Maven

Algunas de las funcionalidades de Maven que están relacionadas con la gestión de proyectos son las siguientes:

1. **Gestión de dependencias**: Maven permite especificar las dependencias de tu proyecto en un archivo de configuración llamado "pom.xml". Maven descarga automáticamente las bibliotecas y las incorpora al proyecto, lo que facilita la gestión de las dependencias externas. También resuelve conflictos de versiones y garantiza la coherencia entre las dependencias.

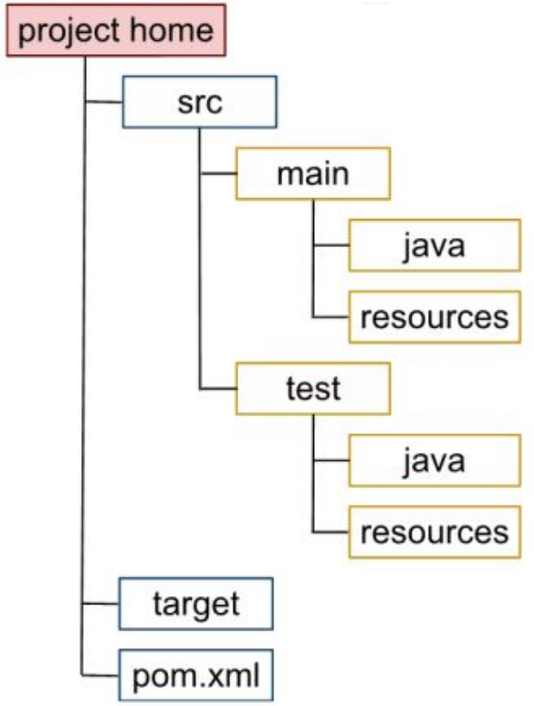


Características generales de Maven

2. **Construcción y compilación**: Maven se encarga de la construcción del proyecto, incluyendo la compilación del código fuente, la ejecución de pruebas unitarias y la generación de artefactos, como archivos JAR o WAR. Con un solo comando, Maven gestiona la construcción completa del proyecto y sus componentes, lo que facilita la integración y la automatización del proceso.

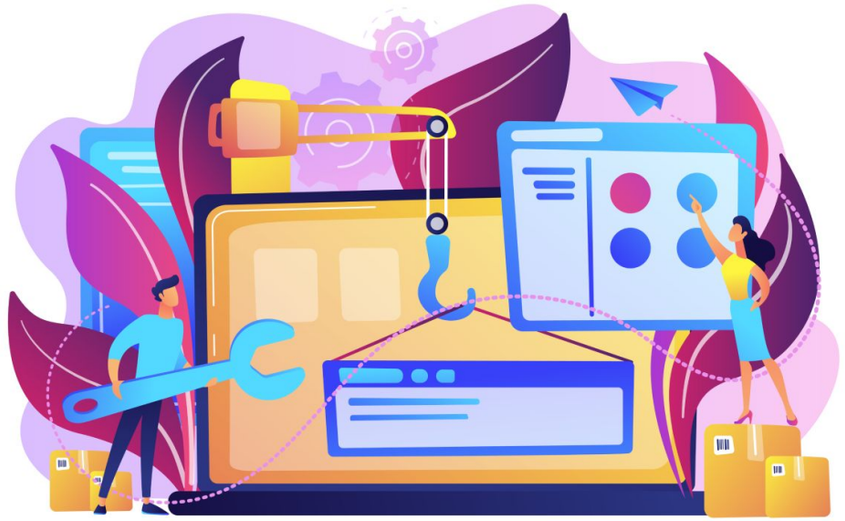
Características generales de Maven

3. **Estructura del proyecto**: Maven sigue una estructura de directorios estándar para organizar el código fuente, los recursos y las pruebas del proyecto. Esto proporciona una convención que facilita la navegación y la comprensión del proyecto para los desarrolladores y los miembros del equipo. Además, permite una fácil integración con herramientas de desarrollo y control de versiones.



Características generales de Maven

4. **Generación de informes**: Maven incluye la capacidad de generar informes detallados sobre el proyecto, como el informe de cobertura de pruebas, el informe de análisis estático del código o el informe de dependencias. Estos informes proporcionan información valiosa para la toma de decisiones y la evaluación de la calidad del proyecto.



Modelo de Objeto de Proyecto (POM)

Modelo de Objeto de Proyecto (POM)

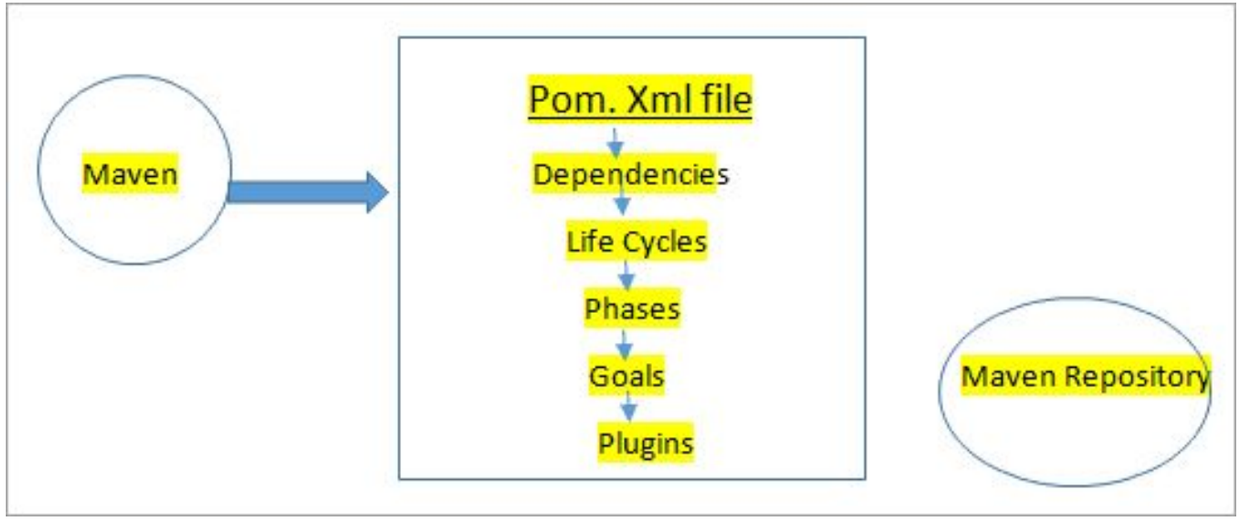
Un modelo de objeto de proyecto o POM es la unidad de trabajo fundamental en Maven. Es un archivo XML que **contiene información sobre el proyecto y los detalles de configuración** utilizados por Maven para construir el proyecto. Contiene valores predeterminados para la mayoría de los proyectos.

Ejemplos de esto es el directorio de compilación, que es **target**; el directorio fuente, que es **src/main/java**; el directorio fuente de prueba, que es **src/test/java**; etcétera.

Al ejecutar una tarea u objetivo, Maven busca el POM en el directorio actual. Lee el POM, obtiene la información de configuración necesaria y luego ejecuta el objetivo.

Modelo de Objeto de Proyecto (POM)

Algunas de las configuraciones que se pueden especificar en el POM son las dependencias del proyecto, los complementos o los objetivos que se pueden ejecutar, los perfiles de compilación, etc. También se puede especificar otra información, como la versión del proyecto, la descripción, los desarrolladores, las listas de correo y demás.



Modelo de Objeto de Proyecto (POM)

Los requisitos mínimos para un POM son los siguientes:

- `project` raíz
- `modelVersion`- debe establecerse en 4.0.0
- `groupId`- el id del grupo del proyecto.
- `artifactId`- la identificación del artefacto (proyecto)
- `version`- la versión del artefacto bajo el grupo especificado

```
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
    <groupId>com.example</groupId>
    <artifactId>projectArtifact</artifactId>
    <version>0.0.1-SNAPSHOT</version>
```

Creación de un proyecto Java: Spring Framework

Spring Framework

Spring es un framework alternativo al stack de tecnologías estándar en aplicaciones JavaEE. Popularizó ideas como la inyección de dependencias o el uso de objetos convencionales (POJOs) como objetos de negocio.

Spring Framework

Spring es el framework más popular para el desarrollo de aplicaciones empresariales en Java, para crear código de alto rendimiento, liviano y reutilizable.

Su finalidad es estandarizar, agilizar, manejar y resolver los problemas que puedan ir surgiendo en el trayecto de la programación.

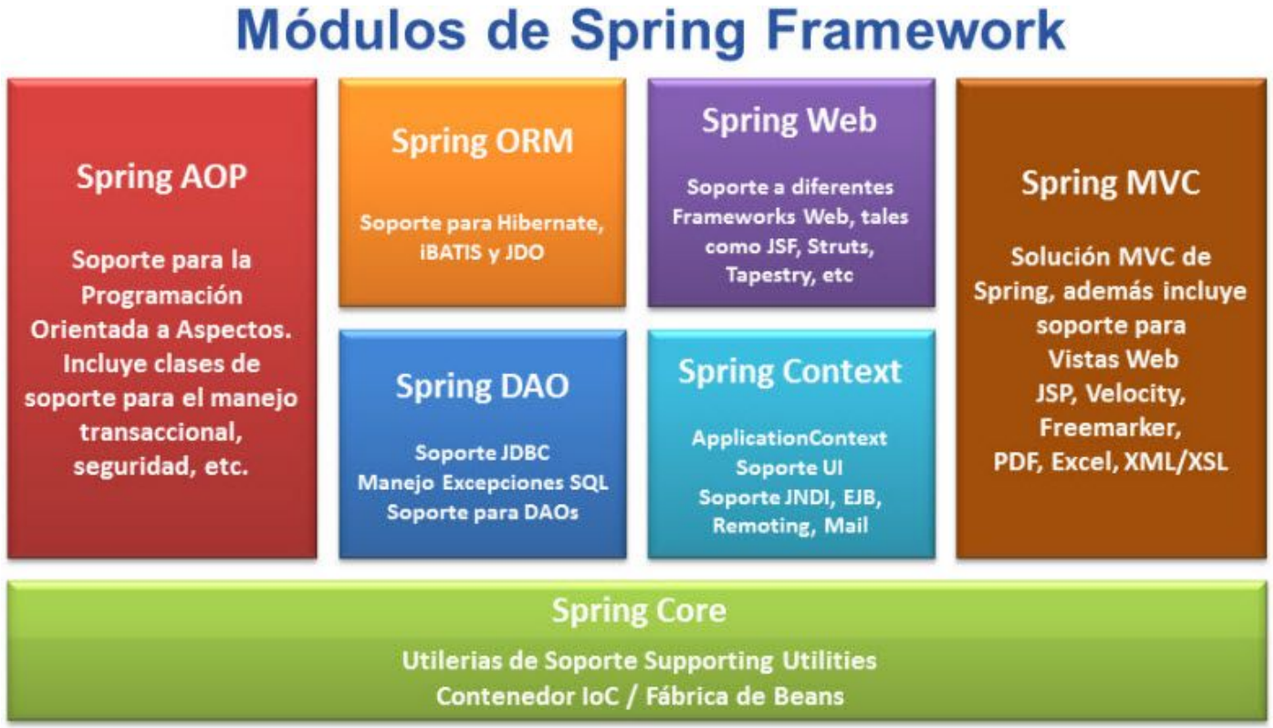
Spring Framework

Spring, ofrece como elemento clave la inyección de dependencias a nuestro proyecto, pero existen otras funcionalidades también muy útiles:

- **Core container**: proporciona inyección de dependencias e inversión de control.
- **Web**: nos permite crear controladores Web, tanto de vistas MVC como aplicaciones REST. Esto facilita en gran medida la programación basada en MVC (Modelo Vista Controlador)
- **Acceso a datos**: abstracciones sobre JDBC, ORMs como Hibernate, sistemas OXM (Object XML Mappers), JSM y transacciones.
- **Instrumentación**: proporciona soporte para la instrumentación de clases.
- **Pruebas de código**: contiene un framework de testing, con soporte para JUnit y TestNG y todo lo necesario para probar los mecanismos de Spring.

Creación de un proyecto Java: Spring Framework

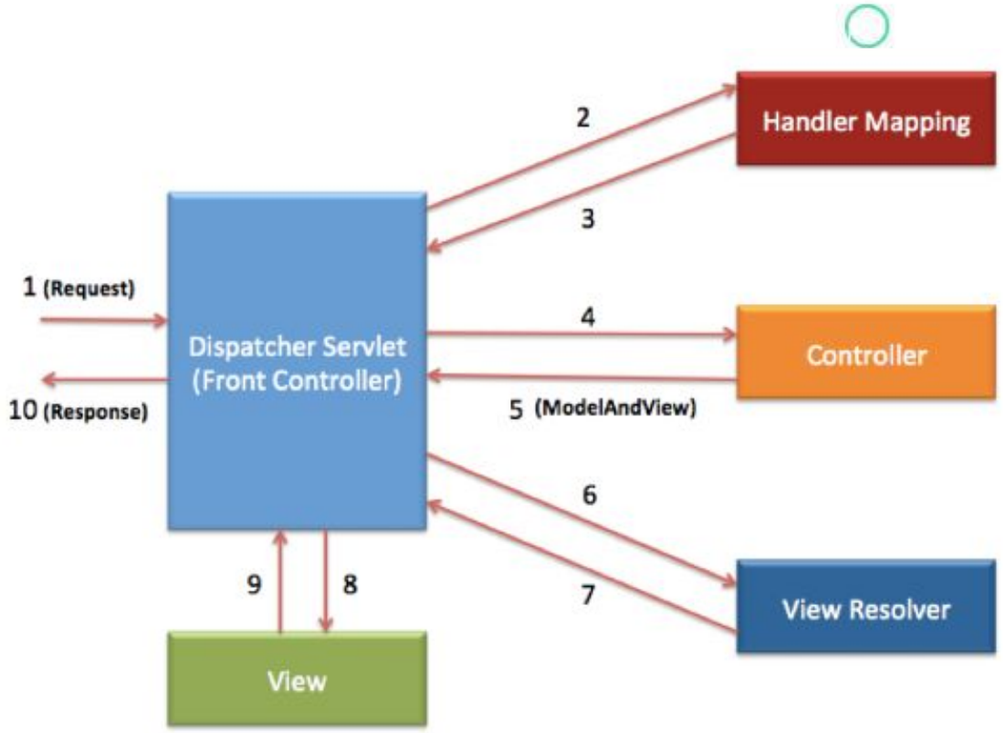
Estos módulos son opcionales, por lo que podemos utilizar los que necesitemos.



Spring MVC

Spring MVC

Spring Web MVC es un sub-proyecto Spring que está dirigido a facilitar y optimizar el proceso de creación de aplicaciones web utilizando el patrón **Modelo Vista Controlador**.



Spring MVC

MVC es un **patrón de diseño** que se estructura mediante tres componentes: **modelo, vista y controlador**.

Este patrón tiene como principio la **separación de cada uno de los componentes en diferentes objetos**, esto significa que los componentes no se pueden combinar dentro de una misma clase.

Sirve para clasificar la información, la lógica del sistema y la interfaz que se le presenta al usuario.

Spring MVC

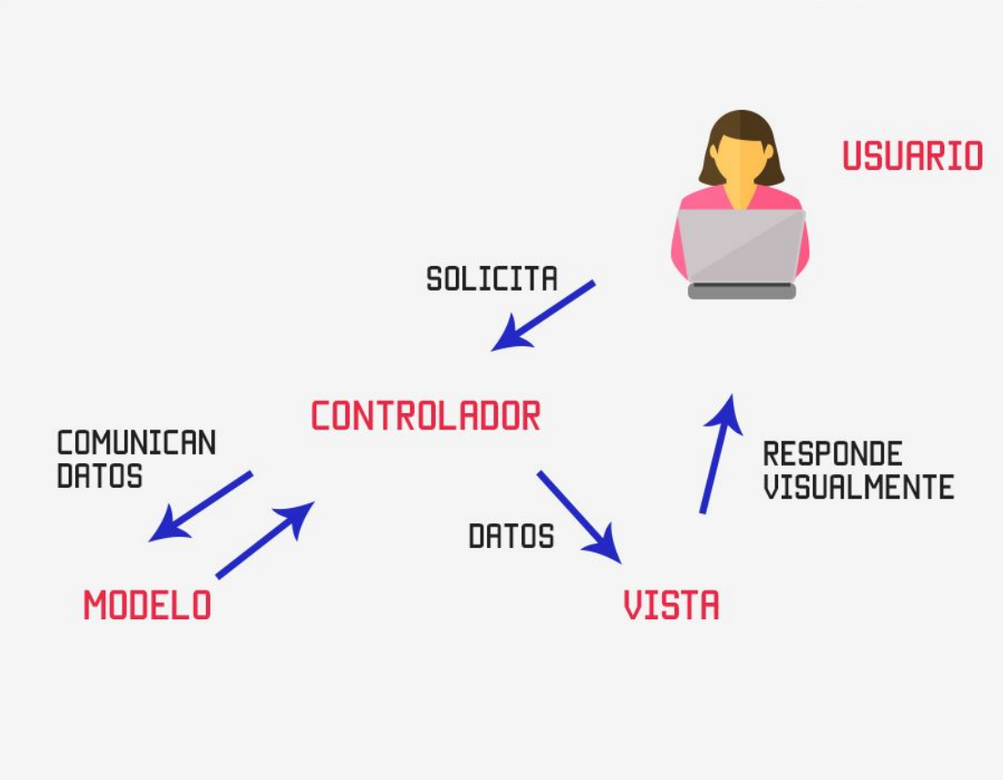
Modelo: Esta capa representa todo lo que tiene que ver con el acceso a datos: guardar, actualizar, obtener datos, etc. Básicamente son las clases Java y parte de la lógica de negocio. No contiene ninguna lógica que describa cómo se deben presentar los datos a un usuario.

Vista: Este componente presenta los datos del modelo al usuario. La vista sabe cómo acceder a los datos del modelo, pero no sabe que significa esta información o que puede hacer el usuario para manipularla.

Controlador: Este componente se encarga de gestionar, atender y procesar las instrucciones que se reciben. El controlador es el encargado de conectar el modelo con las vistas.

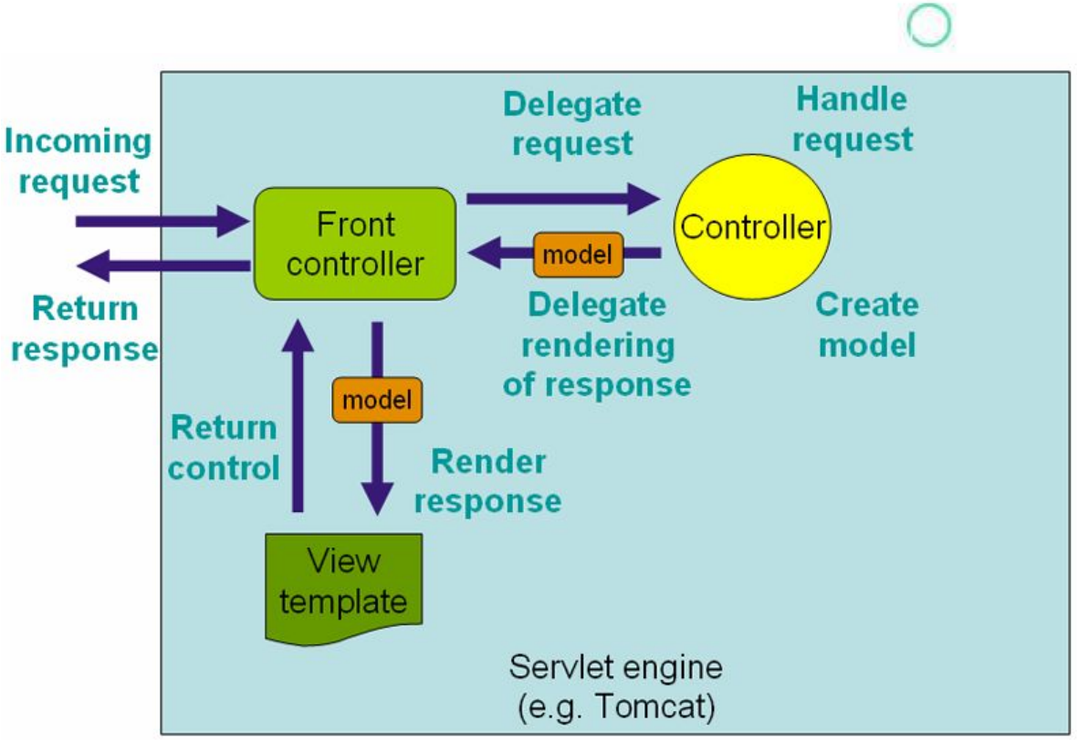
Spring MVC

El controlador recibe eventos generados por el usuario desde las vistas y se encarga de direccionar la petición al modelo, recibir los resultados y entregarlos a la vista para que pueda mostrarlos.



Spring MVC

Spring MVC se basa en este patron de diseño para el manejo de las peticiones http y sus respuestas.
A continuación, se describe el flujo de procesamiento típico para una petición HTTP en Spring MVC. Spring es una implementación del patrón de diseño "front controller”



Spring Boot

Spring Boot

Es una herramienta que nace con la finalidad de simplificar aún más el desarrollo de aplicaciones basadas en el framework Spring. Spring Boot busca que el desarrollador se centre únicamente en el desarrollo de la solución, olvidándose por completo de la compleja configuración que actualmente tiene Spring Core para poder funcionar.

Spring Boot

Spring Boot **es un marco de desarrollo de aplicaciones de código abierto** basado en el ecosistema de Spring Framework. **Está diseñado para simplificar el desarrollo de aplicaciones** empresariales, especialmente aplicaciones basadas en Java, ofreciendo una serie de beneficios y características clave

Spring Boot

Características:

1. **Configuración automática**: Spring Boot utiliza anotaciones y convenciones para la configuración, lo que significa que muchas configuraciones se realizan automáticamente, eliminando la necesidad de archivos de configuración XML.

2. **Arranque rápido**: Spring Boot proporciona un servidor integrado (por ejemplo, Tomcat, Jetty o Undertow) para que puedas ejecutar tu aplicación sin necesidad de implementarla en un servidor externo.

3. **Actuadores**: Spring Boot ofrece una serie de actuadores que permiten supervisar y administrar la aplicación en tiempo de ejecución. Puedes acceder a métricas, información de estado, registro, etc., a través de endpoints dedicados.

Spring Boot

Características:

4. **Perfiles de aplicación**: Puedes definir perfiles de aplicación para gestionar la configuración en diferentes entornos (desarrollo, prueba, producción, etc.).

5. **Manejo de dependencias**: Spring Boot gestiona las dependencias de manera eficiente, lo que facilita la incorporación de bibliotecas y componentes en tu proyecto.

6. **Spring Boot CLI**: Ofrece una línea de comandos para ayudar a desarrolladores a crear, compilar y ejecutar aplicaciones Spring Boot rápidamente.

Spring Boot

Beneficios de Spring Boot:

- **Facilidad de configuración**: Spring Boot se centra en la "opinión sobre la configuración", lo que significa que proporciona configuraciones predeterminadas sensatas para la mayoría de los casos de uso. Esto reduce la necesidad de configuración manual y facilita el inicio rápido de proyectos.

- **Desarrollo rápido**: Spring Boot facilita la creación rápida de aplicaciones empresariales gracias a su capacidad de generación de proyectos iniciales (por ejemplo, utilizando Spring Initializer) y su configuración automática. Esto permite a los desarrolladores concentrarse en la lógica de la aplicación en lugar de preocuparse por detalles de configuración.

Spring Boot

Beneficios de Spring Boot:

- **Integración sencilla**: Spring Boot ofrece integración con una variedad de tecnologías y marcos, como bases de datos, seguridad, sistemas de mensajería y más. Esto simplifica la integración de estas tecnologías en tu aplicación.

- **Microservicios**: Spring Boot es ampliamente utilizado en el desarrollo de microservicios debido a su capacidad para crear aplicaciones independientes y empaquetarlas en contenedores, lo que facilita la implementación y escalabilidad de microservicios.

Spring Boot

Beneficios de Spring Boot:

- **Automatización de tareas**: Spring Boot incluye una serie de herramientas y características para la automatización de tareas comunes, como la construcción de proyectos, la generación de documentación API, la administración de propiedades y más.

- **Amplia comunidad y soporte**: Spring Boot cuenta con una comunidad activa y una amplia documentación, lo que facilita la obtención de ayuda y recursos en línea. Además, es respaldado por Pivotal (anteriormente SpringSource), lo que garantiza un alto nivel de soporte y desarrollo continuo.

Evaluación Integradora ✨

¿Listos para un nuevo desafío? En esta clase comenzamos a construir nuestro....

Trabajo Integrador del Módulo 💪🏽

Iremos completándolo progresivamente clase a clase.

English Always | alkemy

Ejercicio N° 1

Dependencias

Administrando aplicaciones

Manos a la obra: 🙌

¡Finalmente vamos a unificar lo aprendido! Vas a crear un proyecto Spring con Java.

Además, recomendamos

Consigna: ✍️

Paso 1: Configuración del entorno

Asegúrate de tener instalada una versión de Java compatible en tu sistema y, si aún no lo has hecho, instala Spring Boot en tu entorno de desarrollo.

Paso 2: Creación de la aplicación

Crea una clase Java en el paquete de tu elección. Esta clase será la clase principal de tu aplicación. Aquí tienes un ejemplo de cómo podría verse:

Tiempo🕐: *30 minutos*

Administrando aplicaciones

Paso 2: Creación de la aplicación

Crea una clase main Java en el paquete de tu elección. Esta clase será la clase principal de tu aplicación. Debes incluir la anotación necesaria para que tu proyecto sea del tipo Spring Boot Application.

Aquí tienes un ejemplo de un proyecto WalletApplication.

```java
import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class WalletApplication {

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

}
```

¿Alguna consulta?

RESUMEN

¿Qué logramos en esta clase?

- ✓ *Comprender el concepto de gestor de proyectos*
- ✓ *Conocer Maven y el POM*
- ✓ *Aprender las particularidades de Spring Framework, Spring MVC y Spring Boot*

English Always | alkemy

#WorkingTime

Continuemos ejercitando

¡Antes de cerrar la clase! Te invitamos a: 👇👇👇

1. Repasar nuevamente la grabación de esta clase
2. Revisar el material compartido en la plataforma de Moodle (lo que se vio en clase y algún ejercicio adicional)
   a. *Lectura Modulo 6, Lección 1: páginas 1 - 5*
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

¡Muchas Gracias!

Nos vemos en la próxima clase 👋🏻

English Always | alkemy

Momento:

Time-out!

⏰5 min.