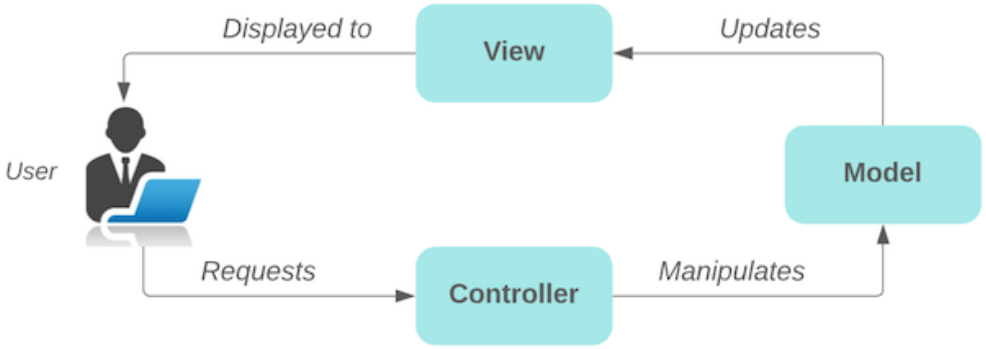
TRABAJO DE CAMPO 3

1. Grafique el funcionamiento del patrón MVC para Spring Framework



2. Describa en qué consiste la inyección de dependencias
    - La inyección de dependencias es una técnica en la que consiste en suministrar objetos a una clase en lugar de ser la propia clase la que cree dichos objetos. Estos objetos están fuertemente enlazados y necesitan las clases para poder funcionar, de ahí el concepto de dependencia.

3. Describa los estereotipos de Spring
    - @Component: Es el estereotipo principal, indica que la clase anotada es un componente (o un Bean de Spring).
    - @Repository: Se encarga de dar de alta un bean para que implemente el patrón repositorio que es el encargado de almacenar datos en una base de datos o repositorio de información que se necesite.
    - @Service: Se encarga de gestionar las operaciones de negocio más importantes a nivel de la aplicación y aglutina llamadas a varios repositorios de forma simultánea. Su tarea fundamental es la de agregador.
    - @Controller: Realiza las tareas de controlador y gestión de la comunicación entre el usuario y el aplicativo. Para ello se apoya habitualmente en algún motor de plantillas o librería de etiquetas que facilitan la creación de páginas.

4. Mencione los proyectos que tiene Spring.io, relacionados a Spring framework, y que aplicamos en el ejemplo construido en clase.

Spring Boot

Spring Boot facilita la creación de aplicaciones independientes basadas en Spring de grado de producción que puede "simplemente ejecutar".

Tomamos una visión obstinada de la plataforma Spring y las bibliotecas de terceros para que pueda comenzar con el mínimo esfuerzo. La mayoría de las aplicaciones Spring Boot necesitan una configuración mínima de Spring.

Si está buscando información sobre una versión específica o instrucciones sobre cómo actualizar desde una versión anterior, consulte la sección de notas de la versión del proyecto en nuestra wiki.

Características

- Cree aplicaciones independientes de Spring
- Incruste Tomcat, Jetty o Undertow directamente (no es necesario implementar archivos WAR)
- Proporcione dependencias 'de inicio' obstinadas para simplificar la configuración de su compilación
- Configure automáticamente las bibliotecas de Spring y de terceros siempre que sea posible
- Proporcione funciones listas para producción, como métricas, comprobaciones de estado y configuración externa.
- Absolutamente sin generación de código y sin requisitos de configuración XML

Spring Framework

Spring Framework proporciona un modelo integral de programación y configuración para aplicaciones empresariales modernas basadas en Java, en cualquier tipo de plataforma de implementación.

Un elemento clave de Spring es el soporte de infraestructura a nivel de aplicación: Spring se enfoca en la "plomería" de

aplicaciones empresariales para que los equipos puedan enfocarse en la lógica de negocios a nivel de aplicación, sin vínculos innecesarios con entornos de implementación específicos.

Política de soporte y migración

Para obtener información sobre los requisitos mínimos, orientación sobre la actualización de versiones anteriores y políticas de soporte, consulte la página wiki oficial de Spring Framework.

Características

- Tecnologías principales : inyección de dependencias, eventos, recursos, i18n, validación, enlace de datos, conversión de tipos, SpEL, AOP.
- Probando : objetos simulados, marco TestContext, Spring MVC prueba, WebTestClient.
- Acceso a datos : transacciones, soporte DAO, JDBC, ORM, Marshalling XML.
- Marcos web Spring MVC y Spring WebFlux .
- Integración : comunicación remota, JMS, JCA, JMX, correo electrónico, tareas, programación, caché.
- Idiomas : Kotlin, Groovy, lenguajes dinámicos.

Spring Data JPA

Spring Data JPA, parte de la familia Spring Data más grande, facilita la implementación de repositorios basados en JPA. Este módulo trata sobre el soporte mejorado para capas de acceso a datos basadas en JPA. Facilita la creación de aplicaciones impulsadas por Spring que utilizan tecnologías de acceso a datos.

Implementar una capa de acceso a datos de una aplicación ha sido engorroso durante bastante tiempo. Se debe escribir demasiado código repetitivo para ejecutar consultas simples, así como para realizar la paginación y la auditoría. Spring Data JPA tiene como objetivo mejorar significativamente la implementación de capas de acceso a datos al reducir el esfuerzo a la cantidad que realmente se necesita. Como desarrollador, escribe las interfaces de su repositorio, incluidos los métodos de búsqueda

personalizados, y Spring proporcionará la implementación automáticamente.

5. Mencione las anotaciones relacionadas a ORM de JPA

CLASE

- @Entity: indica que la clase decorada es una entidad
- @MappedSuperClass: aplicado sobre una clase, indica que se mapeará como cualquier otra clase, pero que se aplicará únicamente a sus subclases, puesto que esta entidad no tiene una tabla asociada.
- @Table: especifica el nombre de la tabla relacionada con la entidad. Admite el parámetro “name”: nombre de la tabla

PROPIEDAD

- @Column: indica el nombre de la columna en base de datos. Admite el parámetro:
- name: especifica el nombre de la columna
- @Enumerated: indica que los valores de la propiedad van a estar dentro del rango de un objeto enumerador. Admite el parámetro:
- value: indica el tipo valor que se va a utilizar en la persistencia en Base de Datos. Puede utilizarse el enumerador EnumType.
- @Id: aplicado sobre una propiedad, indica que es la clave primaria de una entidad. La propiedad a la que hace referencia puede ser de los siguientes tipos: cualquier tipo primitivo, String, java.util.Date, java.sql.Date, java.math.BigDecimal, java.math.BigInteger
- @GeneratedValue: especifica la estrategia de generación de la clave primaria. Admite el parámetro:
- strategy: indica la estrategia a seguir para la obtención de un nuevo identificador. Permite utilizar el enum GenerationType.

- @Lob: aplicado sobre una propiedad, indica que es un objeto grande (Large OBject). Por ejemplo, en el caso de String, si no se especifica @Lob, se le asigna un tamaño máximo de 255 caracteres.
- @NotEmpty: validación de restricción, indica que la propiedad no puede tener un valor vacío.

TIPOS DE RELACIÓN

- @OneToOne especifica un valor único que contiene una relación uno-a-uno con otro objeto. Dispone de los parámetros:
- cascade: indica el tipo de operación de cascada a realizar. Permite utilizar el enumerador CascadeType.
- fetch: indica la forma en que se consultarán las entidades asociadas. Permite utilizar el enumerador FetchType.
- @OneToMany: especifica una relación uno-a-muchos. Dispone de los parámetros:
- cascade: especifica el tipo de cascada. Permite utilizar el enumerador CascadeType.
- mappedBy: especifica la propiedad de la entidad hija (en el extremo muchos) que sirve para enlazar con la entidad principal (en el extremo uno)
- @ManyToOne: especifica una relación muchos-a-uno
- @ManyToMany especifica una relación muchos-a-muchos en la propiedad decorada. Si la relación es bidireccional, se debe especificar el extremo propietario (el que posee la clave principal) mediante el parámetro mappedBy. En casos de relación bidireccional, JPA creará una tabla relacional por cada sentido. Para evitar esto, se usa el decorador @JoinTable.

ENUMERADORES UTILIZADOS

- CascadeType. Tipos de operación de cascada:
    - REMOVE
    - REFRESH
    - PERSIST

        - MERGE
        - DETACH
        - ALL
    - GenerationType. Tipos de estrategia para la generación de identificadores:
    - AUTO: el proveedor de persistencia escoge el método más adecuado según el modelo de base de datos
    - IDENTITY: el proveedor de persistencia escoge un identificador basándose en una columna identity de la tabla
    - SEQUENCE: utiliza una secuencia de la base de datos
    - TABLE: utiliza una tabla auxiliar para asegurarse de que la clave es verdaderamente única
    - FetchType. Tipos de obtención de entidades relacionadas:
    - LAZY
    - EAGER
    - EnumType. Tipo de enumerador. Indica cómo va a ser el tipo de persistencia en la base de datos:
    - ORDINAL: formato predeterminado. En base de datos, se almacenará el ordinal correspondiente al valor del enum.
    - STRING: en base de datos, se almacenará el valor String correspondiente al enum. Usando este tipo de persistencia, se evitan errores en caso de que los valores del enum cambien o se intercalan valores nuevos, que puedan alterar el orden asignado.

6. Describa las características de Spring Data JPA.
    - Soporte sofisticado para construir repositorios basados en Spring y JPA.
    - Soporte para Querydsl predicados y, por lo tanto, consultas JPA con seguridad de tipos
    - Auditoría transparente de la clase de dominio
    - Soporte de paginación, ejecución dinámica de consultas, capacidad para integrar código de acceso a datos personalizado

- Validación de @Queryconsultas anotadas en el momento del arranque
- Soporte para mapeo de entidades basado en XML
- Configuración de repositorio basada en Java Config introduciendo @EnableJpaRepositories.

7. Utilizando el framework Spring, implementar un CRD, usando JPA, Thymeleaf: Adjunte en archivo zip -> Saul