S05.T01: Spring boot API Rest + Aplicació Web

Objectius

- Protocol HTTP/REST
- Jpa
- CRUD amb Spring
- MySQL, Thymeleaf

Descripció

En aquesta tasca faràs un CRUD (Create, Read, Update, Delete) que pugui ser cridat com a API Rest i, també, com aplicació web.

Aprendràs a usar correctament els verbs HTTP i a gestionar els codis de resposta.

Recursos

CRUD Spring API REST + JPA + PostgreSQL:
https://www.bezkoder.com/spring-boot-jpa-crud-rest-api/

CRUD Spring API REST + JPA + H2:
https://www.bezkoder.com/spring-boot-jpa-h2-example/

CRUD Spring API REST + JPA + MySQL:
https://amoelcodigo.com/crud-java-sprig-mysql/
https://www.youtube.com/watch?v=1BYxZCFjfyU
https://javadesde0.com/crud-rest-con-spring-boot-y-jpa/

CRUD Spring API REST + JPA + MongoDB:
https://www.geeksforgeeks.org/spring-boot-crud-operations-using-mongodb/#:~:text=CRUD%20stands%20for%20Create%2C%20Read,we%20perform%20on%20persistence%20storage.

https://www.youtube.com/watch?v=k5PeywcbVYc

ResponseEntity
https://www.youtube.com/watch?v=-I2XC7pQi5k
https://www.baeldung.com/spring-response-entity
https://medium.com/@sebastian.alejandro.hv/java-spring-usando-responseentity-ef327164d514
https://bushansirgur.in/spring-responseentity-example/

SpringBoot Thymeleaf JPA y MySQL - Curso Básico
https://www.youtube.com/playlist?list=PLxxZ0339925GvBiF39ieLkMjyvMlqihB2

Spring Boot CRUD Application with Thymeleaf and H2
https://www.baeldung.com/spring-boot-crud-thymeleaf

Spring Boot CRUD Web Application with Thymeleaf, Spring MVC, Spring Data JPA, Hibernate, MySQL
https://www.javaguides.net/2020/05/spring-boot-crud-web-application-with-thymeleaf.html

Swagger
https://howtodoinjava.com/swagger2/swagger-spring-mvc-rest-example/

Nivell 1: Aplicació Web d'un CRUD amb MySQL

Accedint a la pàgina https://start.spring.io/, genera un projecte Spring boot amb les següents característiques:

Project (gestor de dependències)
Maven o Gradle

Language
Java

Spring Boot
La darrera versió estable

Project Metadata
Group
cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n01
Artifact
S05T01N01GognomsNom
Name
S05T01N01GognomsNom
Description
S05T01N01GognomsNom
Package name
cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n01

Packagin
Jar

Java
Mínim versió 11

Dependències:
Spring Boot DevTools
Spring Web
Spring Data JPA
MySQL Driver
Thymeleaf

Tenim una entitat anomenada **Sucursal**, que disposa de les següents propietats:

- Integer pk_SucursalID
- String nomSucursal
- String paisSucursal

També tenim una DTO anomenada SucrusalDTO, que tindrà les mateixes propietats que l'entitat Sucursal, afegint-ne una:

- String tipusSucursal.

Aquesta propietat, en funció del país de la sucursal, haurà d'indicar si és "UE" o "Fora UE". Per a fer això, pots tenir una llista privada a la pròpia DTO (per exemple: List<String> països), amb els països que formen part de la UE.

Aprofitant l'especificació **JPA**, hauràs de persistir l'entitat **Sucursal** a una base de dades **MySql**, seguint el patró **MVC**.

El consell és que **SucursalDTO** la facis servir al Controller y la Vista, i **Sucursal** al Repository. La capa de serveis serà l'encarregada de fer la traducció entre les dues.

Per a això, depenent del Package principal, crearàs una estructura de packages, on ubicaràs els classes que necessitis:

- `cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n01.controllers`
- `cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n01.model.domain`
- `cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n01.model.dto`
- `cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n01.model.services`
- `cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n01.model.repository`

La classe ubicada al paquet controllers (**SucursalController**, per exemple), haurà de ser capaç de donar resposta a les següents peticions per actualitzar i consultar informació:

```
http://localhost:9000/sucursal/add
http://localhost:9000/sucursal/update
http://localhost:9000/sucursal/delete/{id}
http://localhost:9000/sucursal/getOne/{id}
http://localhost:9000/sucursal/getAll
```

Com pots veure, a l'arxiu application.properties, hauràs de configurar que el port a usar sigui el 9000.

La vista haurà d'estar desenvolupada amb **Thymeleaf**.

Per tal de fer-la més atractiva, pots usar opcionalment alguna llibreria CSS que t'ho faciliti, com bootstrap, tailwind, material, etc.

Molt important

A més de l'enllaç a Git de la tasca resolta, hauràs d'incloure almenys dos enllaços, diferents dels recursos que t'hem proporcionat al campus, que t'hagin servit o ho haguessin pogut fer, per resoldre la totalitat de la tasca o algunes parts.

Nivell 2: API Rest d'un CRUD amb MySQL

Accedint a la pàgina https://start.spring.io/, genera un projecte Spring boot amb les següents característiques:

Project (gestor de dependències)
Maven o Gradle

Language
Java

Spring Boot
La darrera versió estable

Project Metadata
Group
cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n02
Artifact
S05T01N02GognomsNom
Name
S05T01N02GognomsNom
Description
S05T01N02GognomsNom
Package name
cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n02

Packagin
Jar

Java
Mínim versió 11

Dependències:
Spring Boot DevTools
Spring Web
Spring Data JPA
MySQL Driver

Tenim una entitat anomenada **FlorEntity**, que disposa de les següents propietats:

- Integer pk_FlorID
- String nomFlor
- String paisFlor

També tenim una DTO anomenada **FlorDTO**, que tindrà les mateixes propietats que l'entitat Sucursal, afegint-ne una:

- String tipusFlor.

Aquesta propietat, en funció del país de la sucursal, haurà d'indicar si és "UE" o "Fora UE". Per a fer això, pots tenir una llista privada a la pròpia DTO (per exemple: List<String> països), amb els països que formen part de la UE.

Aprofitant l'especificació **JPA**, hauràs de persistir l'entitat **FlorEntity** a una base de dades **MySql**, seguint el patró **MVC**.

El consell és que **FlorDTO** la facis servir al Controller, i **FlorEntity** al Repository. La capa de serveis serà l'encarregada de fer la traducció entre les dues.

Per a això, depenent del Package principal, crearàs una estructura de packages, on ubicaràs els classes que necessitis:

- `cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n02.controllers`
- `cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n02.model.domain`
- `cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n02.model.dto`
- `cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n02.model.services`
- `cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n02.model.repository`

La classe ubicada al paquet controllers (**FlorController**, per exemple), haurà de ser capaç de donar resposta a les següents peticions per actualitzar i consultar informació:

```
http://localhost:9001/flor/add
http://localhost:9001/flor/update
http://localhost:9001/flor/delete/{id}
http://localhost:9001/flor/getOne/{id}
http://localhost:9001/flor/getAll
```

Com pots veure, a l'arxiu application.properties, hauràs de configurar que el port a usar sigui el 9001.

Important: Hauràs de tenir en compte les bones pràctiques de disseny de les API, fent servir correctament els codis d'error i les respostes en cas d'invocacions incorrectes. (Pots consultar informació sobre ResponseEntity).

Has d'incloure **swagger** perquè quasevol desenvolupador/a pugui tenir una idea ràpida dels recursos de que disposa l'API.

Molt important

A més de l'enllaç a Git de la tasca resolta, hauràs d'incloure almenys dos enllaços, diferents dels recursos que t'hem proporcionat al campus, que t'hagin servit o ho haguessin pogut fer, per resoldre la totalitat de la tasca o algunes parts.

Nivell 3: API Rest conectada a una altra API Rest

Accedint a la pàgina https://start.spring.io/, genera un projecte Spring boot amb les següents característiques:

Project (gestor de dependències)
Maven o Gradle

Language
Java

Spring Boot
La darrera versió estable

Project Metadata
Group
cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n03
Artifact
S05T01N03GognomsNom
Name
S05T01N03GognomsNom
Description
S05T01N03GognomsNom
Package name
cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n03

Packagin
Jar

Java
Mínim versió 11

Dependències:
Spring Boot DevTools
Spring Web

Usant **RestTemplate** o **WebClient**, t'hauràs de connectar a l'API que has fet al nivell 2, per cridar i testar totes les peticions que permet aquesta API.

Tingues en compte, que en aquesta tasca del nivell 3, no tens cap referència a cap base de dades, ni necessites fer servir JPA, ja que el teu respository accedirà a l'API del nivell 2.

No et cal crear una Vista, ja que aquest nivell 3 està previst com una API Rest, però hauràs de crear totes les capes fins al controlador com qualsevol altre aplicació:

- `cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n03.controllers`
- `cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n03.model.domain`
- `cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n03.model.dto`
- `cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n03.model.services`
- `cat.itacademy.barcelonactiva.cognoms.nom.s05.t01.n03.model.repository`

La classe controladora, haurà de ser capaç d'atendre les següents peticions:

```
http://localhost:9002/flor/clientFlorsAdd
http://localhost:9002/flor/clientFlorsUpdate
http://localhost:9002/flor/clientFlorsDelete/{id}
```

`http://localhost:9002/flor/clientFlorsGetOne/{id}`
`http://localhost:9002/flor/clientFlorsAll`

Com pots veure, a l'arxiu application.properties, hauràs de configurar que el port a usar sigui el 9002.

Per a provar el nivell 3, hauràs de tenir en marxa l'API del nivell 2. No tindràs problemes ja que l'API del nivell 3 treballa amb el port 9002 i la del nivell 2 amb el port 9001.

Has d'incloure **swagger** perquè quasevol desenvolupador/a pugui tenir una idea ràpida dels recursos de que disposa l'API.

Molt important

A més de l'enllaç a Git de la tasca resolta, hauràs d'incloure almenys dos enllaços, diferents dels recursos que t'hem proporcionat al campus, que t'hagin servit o ho haguessin pogut fer, per resoldre la totalitat de la tasca o algunes parts.