# 5Comienzo de proyecto:

Vamos a usar una bbdd embebida [h2]

## Librerías que ha usado:

Lombook

H2

Thymeleaf

Spring Web

Spring Data JPA

## Básicos del ejemplo:

HomeController

## Application.properties:

Spring.datasource.url=jdbc:h2:mem:”xx “:DATABASE\_TO\_LOWER=TRUE

[si no pones nada en “xx” cada pag genera una bbdd propia y no una común]

[DATABASE\_TO\_LOWER, opcion para descapitalizar la bbdd]

Spring.h2.console.enable=true

[habilita la consola de la bbdd, localhost:puerto/console-h2]

Spring.h2.console.path=/h2

[cambiamos la url a localhost:puerto/h2]

Spring.jpa.show-sql=true

Spring.jpa.defer-datasource-initialitation=true

Spring.datasource.url=jdbc:h2:”Ruta”:”xx“;DATABASE\_TO\_LOWER=TRUE;auto\_server=true

Spring.sql.init.mode=never

Spring.jpa.hibernate.ddl-auto=update

Spring.jpa.show-sql=true

## Generación de entity: [student]

Creación genérica de clases de entidad, para empezar a usar la bbdd.

@Entity, la hemos inyectado gracias a SpringDataJPA, pero hay otras dependencias validas.

@Id

## Generación de repositories, interfaces…:

(Repository) extends CrudRepository<Student(Entity), Long(Id)>

[Nos ahorra todo lo de getAll, save, getById…]

Crear Servicies (@Service).

## Generacion de tablas en .java:

Anotaciones de @table(name=”x”), y cada atributo(@column(porperties))

O scheme.sql, data.sql y creación de tablas sql normales.