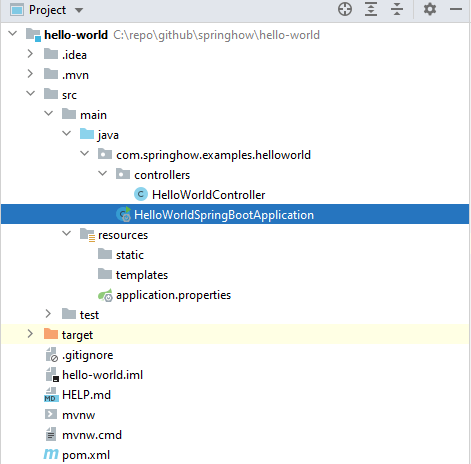
**Migration related info(JAVA SPRINGBOOT FUNDA)**

1. For setting up a project in spring boot we need to specify build tool plugins. It can be either apache’s mavel or gradle. It is similar to package.json and has info about packages and all.
2. We do everything in src/main/java and folder structure is like this –



1. Models or entity:

Here we can create Models like we do in mongodb:

Id of a model -

@Entity

@Table(name = "employee")

public class Employee {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    @Column(name = "emp\_id", nullable = false)

    private Long id;

Why use the @Id annotation?

* The @Id annotation is one of the two mandatory annotations needed when creating an entity with JPA. The other one being @Entity. @Id does two things for us:  
  1) signifies that this field will be the **unique identifier** for this class when mapped to a database table  
  2) the presence of @Id lets the persistence layer know that all other fields within this class are to be mapped to database rows

Why use @GeneratedValue?

* By marking the @Id field with @GeneratedValue we are now enabling id generation. Which means that the persistence layer will generate an Id value for us and handle the auto incrementing. Our application can choose 1 of 4 generations strategies:  
  1) AUTO  
  2) TABLE  
  3) SEQUENCE  
  4) IDENTITY
* If not strategy is specified then AUTO is assumed

What is strategy = GenerationType**.IDENTITY** actually doing?

* When we specify the generation strategy as GenerationType.IDENTITY we are telling the persistence provider(hibernate) to let the database handle the auto **incrementing of the id**. If you were to use postgres as an underling database and specified the strategy as IDENTITY, hibernate would execute this:

create table users ( id bigserial not null, primary key (id) )

* Notice that they type of the id is bigserial, what is bigserial? As per the postgres [documentation](https://www.postgresql.org/docs/9.1/datatype-numeric.html), bigserial is a large auto incrementing **integer**.

Conclusion

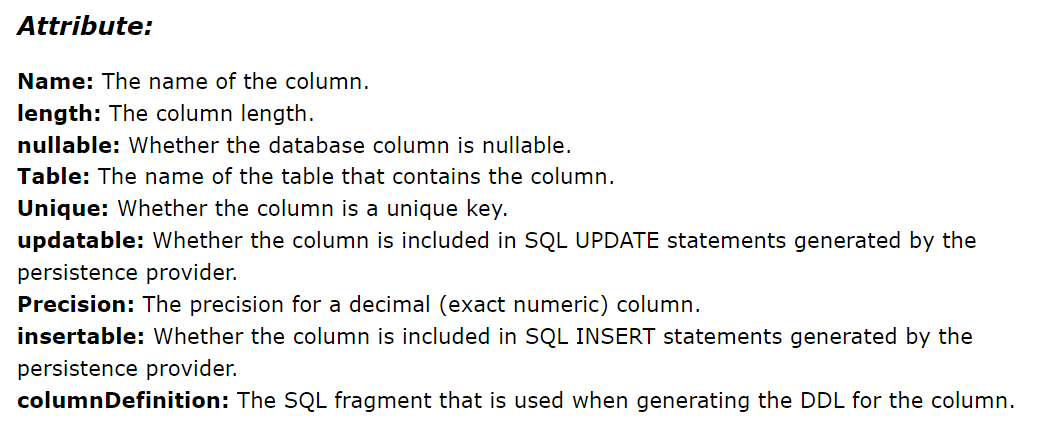
* By specifying:

@Id @GeneratedValue(strategy = GenerationType.IDENTITY) private Integer id;

* you have told the underlying persistence layer to use the **id field as a unique identifier** within the database. **Also told the persistence layer** to let the database handle the auto incrementing of the id with GenerationType.IDENTITY.

1. For setup of mariaDB with spring boot (Example) [**see this**](https://kipalog.com/posts/SpringBoot-MariaDB-CRUD-RestAPI-Example#:~:text=Configuration%20SpringBoot%20to%20connect%20to,%2C%20retrieve%2C%20update%2C%20detele%20entities)
2. General funda -

* Attributes of a column : (Schema key type as in node)



* @Column(nullable = false) is the JPA way of declaring a column to be not-null and is intended for indicating database schema details. {required: true as in node}
* @Lob - Variable length data type for storing  large objects - read [**here**](https://www.baeldung.com/hibernate-lob)
* **Relationships**

Using @OneToMany with @JoinColumn – It just maps the parent to the other documents. There is no way to fetch a parent from a document.

Using @OneToMany with attribute **mappedBy** - It maps parent to doc + doc to their parent.

To implement it :

Create onetomany relationship in parent and use mapped by - property name

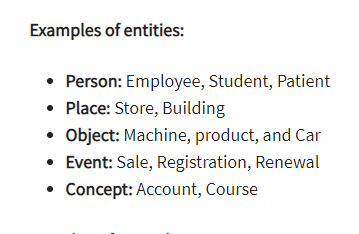
Create many to one in the child table and create the property .

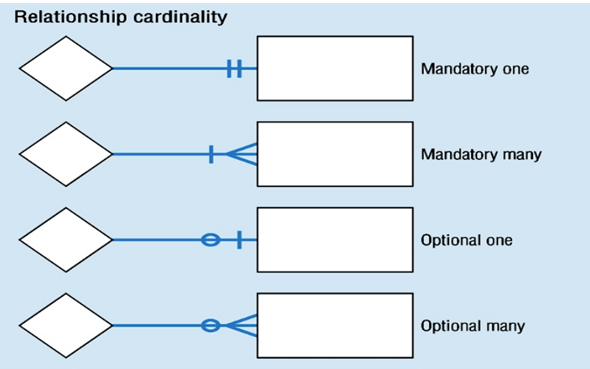
* **fetch** attribute in a column: It is defined for relationships, like if a table has some property linked to many other tables(eg - University -> Students), then when we fetch the first table, we can define whether to look for that mapped property as -

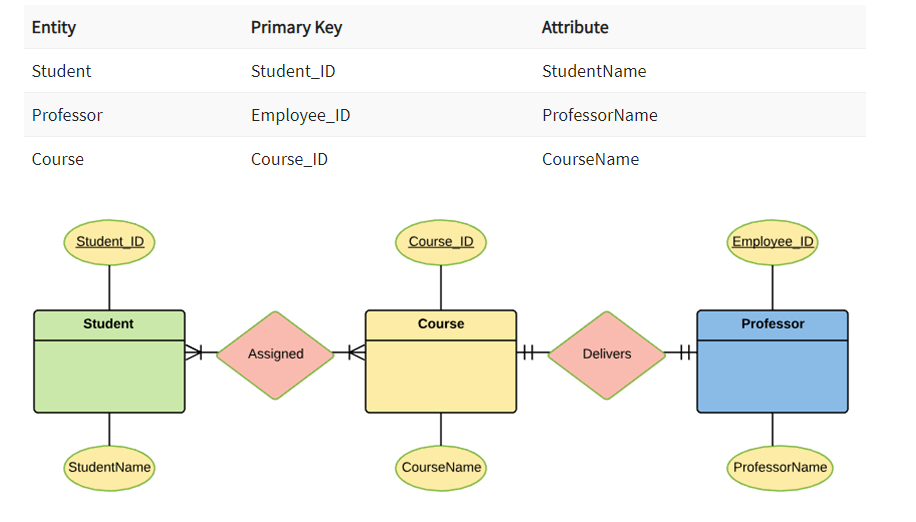
1. **@OneToMany(fetch = fetchType.EAGER) -** That property is fetched while fetching university
2. **fetch = FetchType.LAZY -** It is fetched when needed

If not clear - [**Read this**](https://stackoverflow.com/questions/2990799/difference-between-fetchtype-lazy-and-eager-in-java-persistence-api)

**ER Diagrams**

* It is a tool to visualize Entity relationship models. These models are the logical representation of Entities in the database and their relationship among themselves.
* It is like a GUI non functional representation. It uses symbols like rectangles, ovals and diamonds.
* 
* 





**Primary keys are underlined. One to many many to one work in a similar fashion as studied in 12th. See arrows for understanding**

https://www.guru99.com/er-diagram-tutorial-dbms.html