**25. Hibernate Advanced Mappings - Eager vs Lazy Loading**

**Eager vs Lazy Loading**:

When working with an ORM, data fetching/loading can be classified into two types: eager and lazy.

1. Eager will retrieve everything
2. Lazy will retrieve on request

**Eager Loading**:

Eager Loading is a design pattern in which data initialization occurs on the spot.

* Eager loading will load all dependent entities
* Load instructor and all of their courses at once

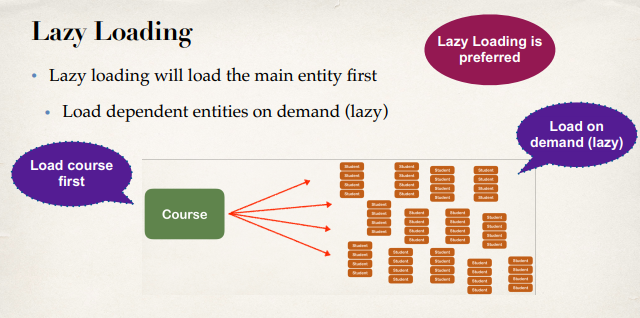
If our database contains a lot of data then eager loading actually impact the performance of our application. It will slow down our application.

**Lazy Loading**:

Lazy Loading is a design pattern which is used to defer initialization of an object as long as it’s possible.

* Lazy loading will load the main entity first
* Load dependent entities on demand (lazy)
* In our app, if we are searching for a course by keyword
* Only want a list of matching courses

Eager loading would still load all students for each course that’s not good. We only want the titles or the descriptions of the courses but not all of the students.

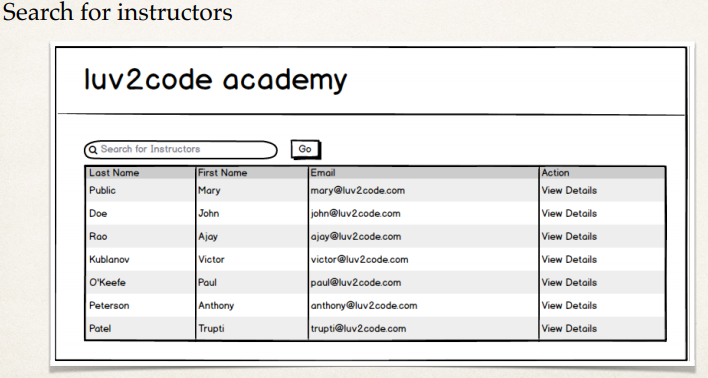


So the best practice is only load data when absolutely needed. Prefer Lazy loading instead of Eager loading.

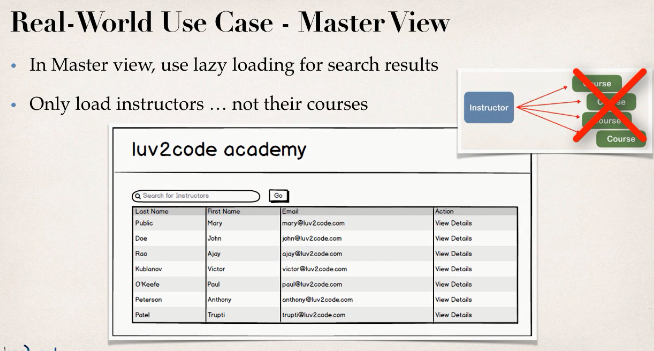
**Real-World Use Case**:

1. Search for instructors

In our application we have a list of instructor. We also have a search option where we can search for Instructor. We also have an action list where we can click and view the details of a given Instructor.

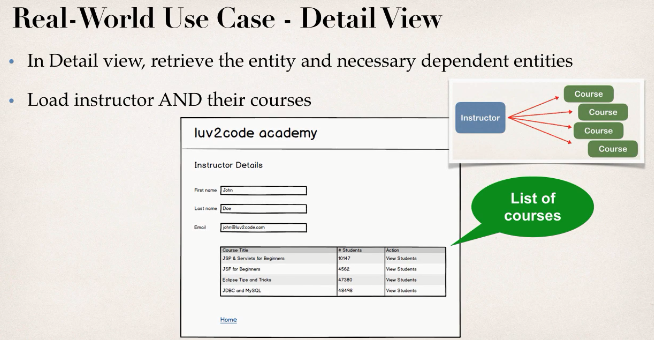


1. In Master view, use lazy loading
   1. In Master view, use lazy loading for search results
   2. Only load instructors … not their courses



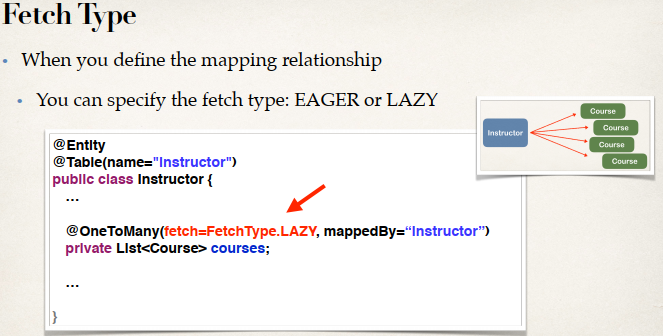
I simply want a high-level view of the instructors but I don’t want their courses at this moment. If I want details information click the “view details” link and go about details. This made our query faster in our databases application.

1. In Detail view, retrieve the entity and necessary dependent entities
   1. In Detail view, retrieve the entity and necessary dependent entities
   2. Load instructor AND their courses



**Fetch Type**:

When you define the mapping relationship between two entities or classes, we can specify the fetch type: EAGER or LAZY.



In the above example we have an Instructor and an Instructor have a list of courses. Here we will have the “**@OneToMany**” and then setup the “**FachType**”. In this case the Instructor is loaded first and then it will load the courses on demand. It will load the courses in a lazy fashion at a later time.

**Default Fetch Types**:

The default Face Type of hibernate is given bellow.

|  |  |
| --- | --- |
| **Mapping** | **Default Face Type** |
| @OneToOne | FetchType.EAGER |
| @OneToMany | FetchType.LAZY |
| @ManyToOne | FetchType.EAGER |
| @ManyToMany | FetchType.LAZY |

**Overriding Default Fetch Type**:

We can override the default Fetch Type. We have to specifying the fetch type and overrides with the default type.



**More about Lazy Loading**:

1. When we use lazy load, the data is only retrieved on demand
2. However, this requires an open Hibernate session
   1. We need a connection to database to retrieve data
3. If the Hibernate session is closed
   1. And we attempt to retrieve lazy data
   2. Hibernate will throw an exception
4. To retrieve lazy data, we will need to open a Hibernate session
5. Retrieve lazy data using
   1. **Option 1**: session.get() and call appropriate getter method(s)
   2. **Option 2**: Hibernate query with HQL
6. Many other techniques available but the two above are most common

25. Hibernate Advanced Mappings - Eager vs Lazy Loading