# Video 1:

Jpa

Data jpa

## Lịch sử java webservice

2 trường phái

* Jee , j2ee , jakataree
  + Thời kì đầu dung jdbc để kết nối csdl
  + Cao cấp hơn là jpa
* Spring framework (2004 trở đi)
  + Thời gian đầu dung jdbc nhưng nó custom lại được gọi là spring jdbc template
  + Sau này spring jdbc template có một số vấn đề thì nó apply jpa vào
  + Sau này nhà phát triền spring framework lại thấy jpa có nhiều nhược điểm
  + Họ lại tiếp tục dựa vào jpa để phát triển spring data jpa
* Jpa: là một library riêng biệt, không thuộc spring framework
  + Câu hỏi? vậy jpa có thể tích hợp vào những framework khác không?
* Data jpa : mới là của spring framework

## Nguyên nhân tại sao phải ra đời jpa

Nhiều nguyên nhân to bự -> sinh ra khái niệm ORM (object relational mapping)

Những nguyên nhân con kèm theo

Vấn đề 1:

* Khi mà đổ dữ liệu từ thằng resultset về đối tượng entity. Ít fields thì không sao nhưng nếu nhiều fields thì rất dễ nhầm lẫn. Hay là khi đổ dữ liệu về db thì cũng vậy.

Vấn đề 2:

* Jdbc phụ thuộc vào database
* Ví dụ khi chuyển từ mysql sang sql server hay oracle thì phải xây lại 90% logic vì cú pháp khác nhau

Giải pháp:

* Tự động.
* Entity sẽ là thằng ánh xạ từ db. Từ tên, kiểu dữ liệu
* Trong trường hợp đổ dữ liệu từ resultset: khi mà đến column name thì sử dụng một số kĩ thuật để kiểm tra xem là entity có name nào và datatype giống không nếu giống nó map data vào luôn
* Định hình ra cơ chế orm: xây dựng tất cả java bean dưới dạng table. Table có j java bean y hệt. Và sẽ truy vấn trực tiếp vào thằng java bean này. Ví dụ dung jdbc và chuyển từ mysql sang sqlserver phải làm lại 90% thì làm theo cơ chế orm chỉ phải làm lại 30%.
* Orm nó giống như là người biết nhiều thứ tiếng
* Native query là truy vấn dựa vào db

# Video 1.2

## Ánh xạ giữa JDBC và JPA

|  |  |
| --- | --- |
| JDBC | JPA |
| Load driver  Set connection (url, username, password) | EntityManager Factory (load driver và khởi tạo connection) |
| Statement: statement, prepared  Statement  Get data: excute query  Xử lý data:  1 build câu query  2 là run excute update | EntityManager  Query: Get data:   * getResultList: ds * getSingleResult: single   CRUD thì entity manager cung cấp những method riêng   * c: entitymanager.persist * u: entitymanager.merge * d: entitymanager .remove   Sql build tự động |
|  |  |

## Transaction

Bản chất transaction của sql

Quản lý commit và rollback

## Phân tách môi trường nói sơ qua

Có thể có nhiều file properties nhưng khi chạy chỉ lấy 1 file

Chạy local sẽ lấy file local properties

Chạy production sẽ lấy file pro properties

# Video 2

## Phân tách môi trường

Quy trình chuẩn

* local: máy tính cá nhân của mỗi người
* SIT: có server riêng ( chỉ có ae dev tự test tự fix )
* UAT: gần như là hoàn chỉnh nhưng chỉ để test
  + Tester
  + Dev
  + Ba
  + Team lead
  + Pm
* Production: môi trường dành cho khách hang sử dụng

Sử dụng spring.profiles.active={tên sau application-…}

## Cách lấy value ở file properties

@PropertySource

Dung annotation @Value

# Video 3

## Cơ chế khi làm việc với JPA

* Jpql: làm việc với entity 100%
  + Một số câu sql thuần nó không nằm trên từ điển của jpa -> không biên dịch được
* SQL native

2 thằng trên đều là ngôn ngữ riêng của jpa

Persisten sẽ convert nn jpa sang nn của sql mà chúng ta sử dụng

## Giới thiệu @Entity

* Khai báo trước class thì sẽ biến 1 java class là một entity đại diện cho 1 table ở sql

## Giới thiệu @Table

* Name: tương đương với table nào

## Giới thiệu @Column(name = “”)

* Tương đương với column nào trong table

## Giới thiệu @Id

* Primary key, notnull

## Giới thiệu @GeneralValue

* Auto increment

## Giới thiệu @ManyToMany

* Tạo quan hệ nhiều nhiều
* Tự động

## Giới thiệu @OneToMany , @ManyToOne

* Tạo quan hệ một nhiều

## Giới thiệu @JoinColumn

* Map tên giống dưới db

## spring.jpa.show-sql=true

* Console ra cho chúng ta câu sql

## spring.jpa.properties.hibernate.format\_sql=true

* Format câu sql nhìn gọn gàng hơn

## spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL55Dialect

* Giúp chúng ta chuyển đổi câu sql về db chúng ta mong muốn