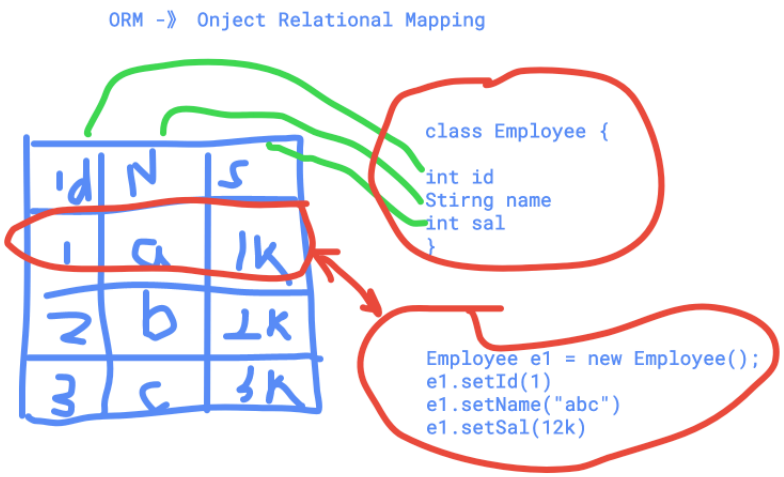
Backend Overview

1. Hibernate

Hibernation ORM (or simple Hibernate) is an object - relational mapping tool for the java programming language. It provides a framework for maaping an **object-oriented domai**n model to a **relational database**.



Advantage of Hibernate:

- Open source and lightweight
- fast performance: first level cache, second level cache
- database independent query.
- automatic table creation
- Simplifies complex join....

HQL: hibernate query language,

Cascade type: one to many : stu -> courses

fetch type: eager loading, lazy loading

Cache: session, sessionFactary level

...

2. Spring boot

The Spring Framework is an application framework and inversion of control container for the java platform

Advantage:

- Spring boot simplified the configuration and dependency management
- Auto-configuration
- lots of default configuration
- embeded tomcat server
- Starters in pom.xml file which makes easy to manage the dependency

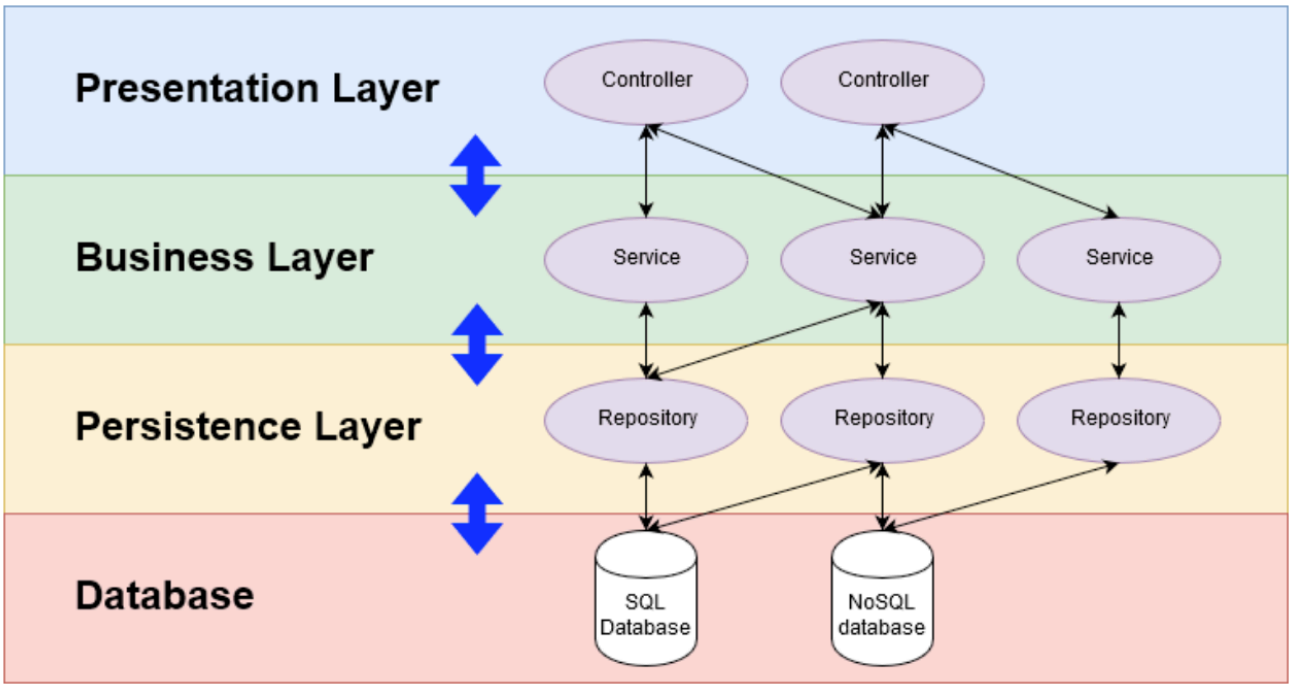
3. Backend Architecture

Three Layer architecture

- Controller: API, TransitionManager
- Service: Cache, business logic
- Repository: Spring data JPA, Hibernate

  ----------app above ----------

  ----------database below -------
- Database



Advantage:

- Fast development, each layer can be developed simultaneously by different team
- Improve scalability: any layer can be scaled independently
- improved security: the presentation layer and data layer can not communicate directly

4. Code Demo

POM.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.4.1</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.shaohua</groupId>
```

```
    <artifactId>springbootcrudexample</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>springbootcrudexample</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
```

```
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

Application.properties

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/crud_example?
useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&ser
verTimezone=UTC
spring.datasource.username=****
spring.datasource.password=****
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
server.port=9190
```

Entity: Product

```
package com.shaohua.springbootcrudexample.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Table;

@Data
@AllArgsConstructor // Product(int id, String name, int quantity, int price)
@NoArgsConstructor // Product()
@Entity
```

```
@Table(name = "PRODUCT_TBL")
public class Product {


    @Id
    @GeneratedValue
    private int id;
    private String name;
    private int quantity;
    private double price;

//    public Product(int id, String name, int quantity, int price) {
//        this.id = id;
//        this.name = name;
//        this.quantity = quantity;
//        this.price = price;
//    }
//
//    public Product() {
//
//    }


    // lombok, getter, setter, constructor

//    public int getId() {}
//    public void setId(int id) {}
//    other getter and setter for name, quanlity, price
}
```

ProductContoller

```
package com.shaohua.springbootcrudexample.controller;

import com.shaohua.springbootcrudexample.entity.Product;
import com.shaohua.springbootcrudexample.service.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
```

```java
import java.util.List;

@RestController
public class ProductController {

    @Autowired
    private ProductService service;

    @PostMapping("/addProduct")
    public Product addProduct(@RequestBody Product product) {
        return service.saveProduct(product);
    }

    @PostMapping("/addProducts")
    public List<Product> addProducts(@RequestBody List<Product> products) {
        return service.saveProducts(products);
    }

    @GetMapping("/products")
    public List<Product> findAllProducts() {
        return service.getProducts();
    }

    @GetMapping("/productById/{id}")
    public Product findProductById(@PathVariable int id) throws Exception {
        if (id == 100) {
            throw new Exception();
        }
        return service.getProductById(id);
    }

    @GetMapping("/product/{name}")
    public Product findProductByName(@PathVariable String name) {
        return service.getProductByName(name);
    }

    @PutMapping("/update")
    public Product updateProduct(@RequestBody Product product) {
        return service.updateProduct(product);
    }

    @DeleteMapping("/delete/{id}")
    public String deleteProduct(@PathVariable int id) {
        return service.deleteProduct(id);
    }
```

```
    @ExceptionHandler(Exception.class)
    public String exceptionHandler(Exception ex) {
        return "something wrong";
    }

}
```

ProductService

```
package com.shaohua.springbootcrudexample.service;

import com.shaohua.springbootcrudexample.entity.Product;
import com.shaohua.springbootcrudexample.repository.ProductRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.*;

@Service
public class ProductService {

    @Autowired
    private ProductRepository repository;

    public Product saveProduct(Product product) {
       return repository.save(product);
    }

    public List<Product> saveProducts(List<Product> products) {
        return repository.saveAll(products);
    }

    public List<Product> getProducts(){
        return repository.findAll();
    }

    public Product getProductById(int id) {
        return repository.findById(id).orElse(null);
    }
```

```
    public Product getProductByName(String name) {
        return repository.findByName(name);
    }

    public String deleteProduct(int id) {
        repository.deleteById(id);
        return "product removed " + id;
    }

    public Product updateProduct(Product product) {
        Product existingProduct =
repository.findById(product.getId()).orElse(null);
        existingProduct.setName(product.getName());
        existingProduct.setQuantity(product.getQuantity());
        existingProduct.setPrice(product.getPrice());
        return repository.save(product);
    }
}
```

Repository

```
import com.shaohua.springbootcrudexample.entity.Product;
import org.springframework.data.jpa.repository.JpaRepository;

public interface ProductRepository extends JpaRepository<Product, Integer> {
    Product findByName(String name);
}
```

SpringBootApplication

```
package com.shaohua.springbootcrudexample;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
```

```java
public class SpringbootcrudexampleApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringbootcrudexampleApplication.class, args);
    }

}
```

API:

Get:

http:// localhost:9091/productById/{id}

http:// localhost:9091/product/{id}

Post

Put:

Delete:

Homework: sammary