Hibernate 学习笔记

https://www.tutorialspoint.com/hibernate/

概述

框架实际上就是一个半成品，

- 一个 Java 领域的持久化框架

  持久化：
  - 狭义：把对象永久保存到数据库
  - 广义：包括数据库的各种操作
    - 保存
    - 更新
    - 删除
    - 查询
    - 加载：根据特定的 OID，把一个对象从数据库中加载到内存中。（为了能找到所需的对象，需要为每个对象分配一个唯一的标识号，在关系型数据库中称之为**主键**； 在对象术语中叫做对象标识 Object identifier-***OID***）
- 一个 ORM 框架
-

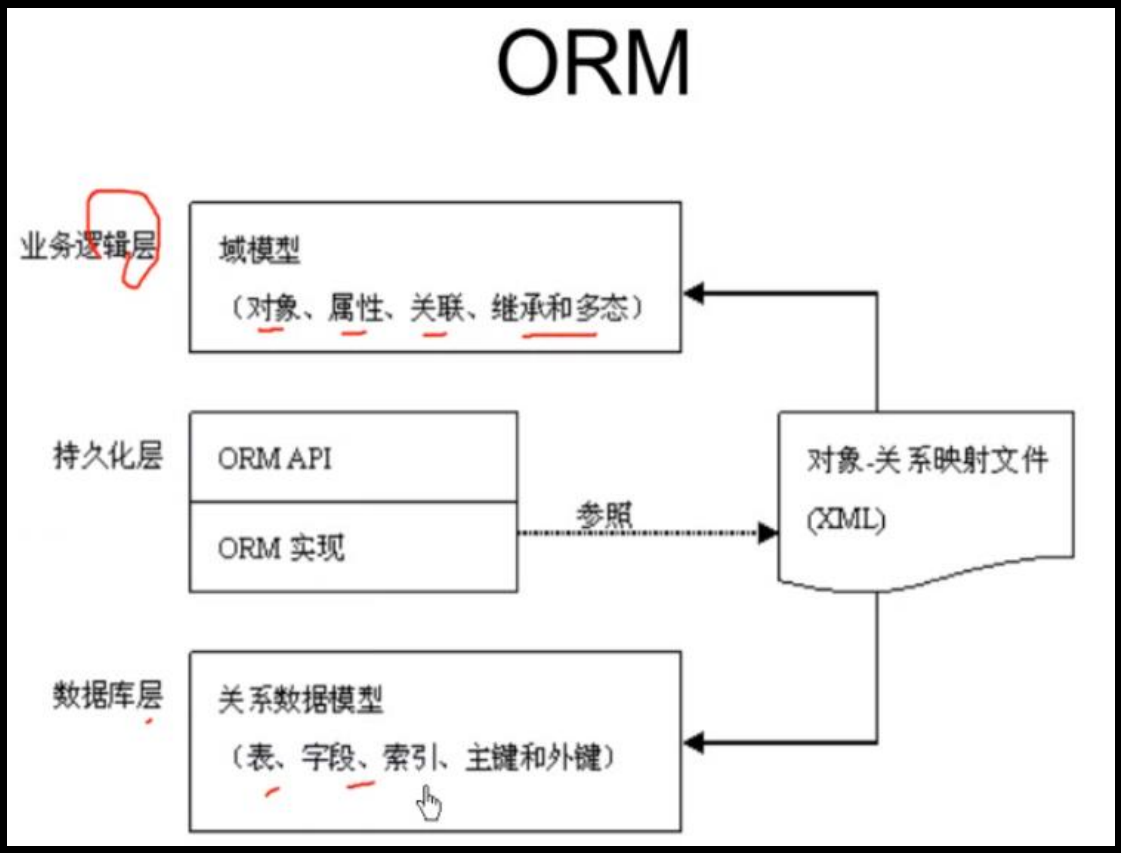
  hibernate 是 JPA 规范，mybatis 不是，从 JAVA 发展的趋势来看，建议用 hibernate。

ORM

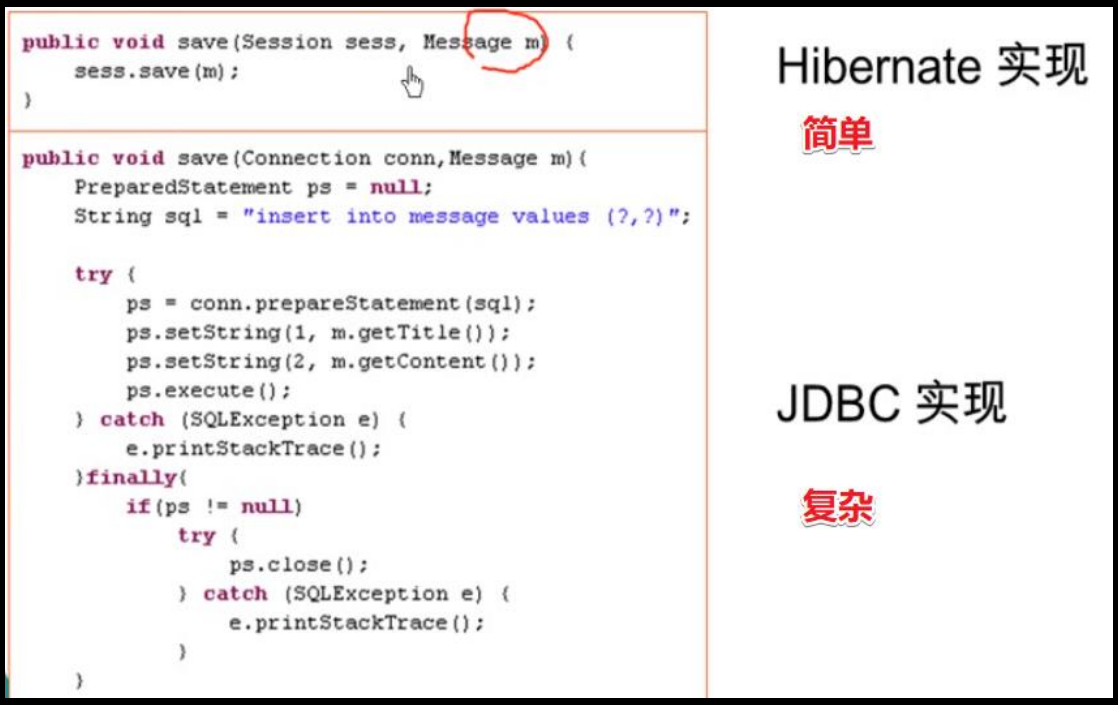
- ORM: Object/Relation Mapping 对象/关系映射
  - Orm 主要解决对象-关系的映射：

面向对象概念	面向关系概念
类	表
对象	表的行记录（）
属性	表的列（字段）

- ORM 的思想：将关系型数据库中表中的记录映射为对象，以对象的形式展现，及**把对数据库的操作转化为对对象的操作。**
- ORM 采用元数据来描述对象-关系映射的细节，元数据通常采用 XML 格式，并且存放在专门的**对象-关系映射文件**中。



- ORM 框架是对 JDBC 的封装。
- 流行的 ORM 框架：
  - Hibernate
    - 成熟优秀，完成对象的持久化操作
    - 允许开发者采用面向对象的方式来操作关系型数据库。
  - MyBatis
    - 灵活度高，运行速度快！
    - 开发速度慢，不支持纯粹的面向对象操作。
  - TopLink
  - OJB

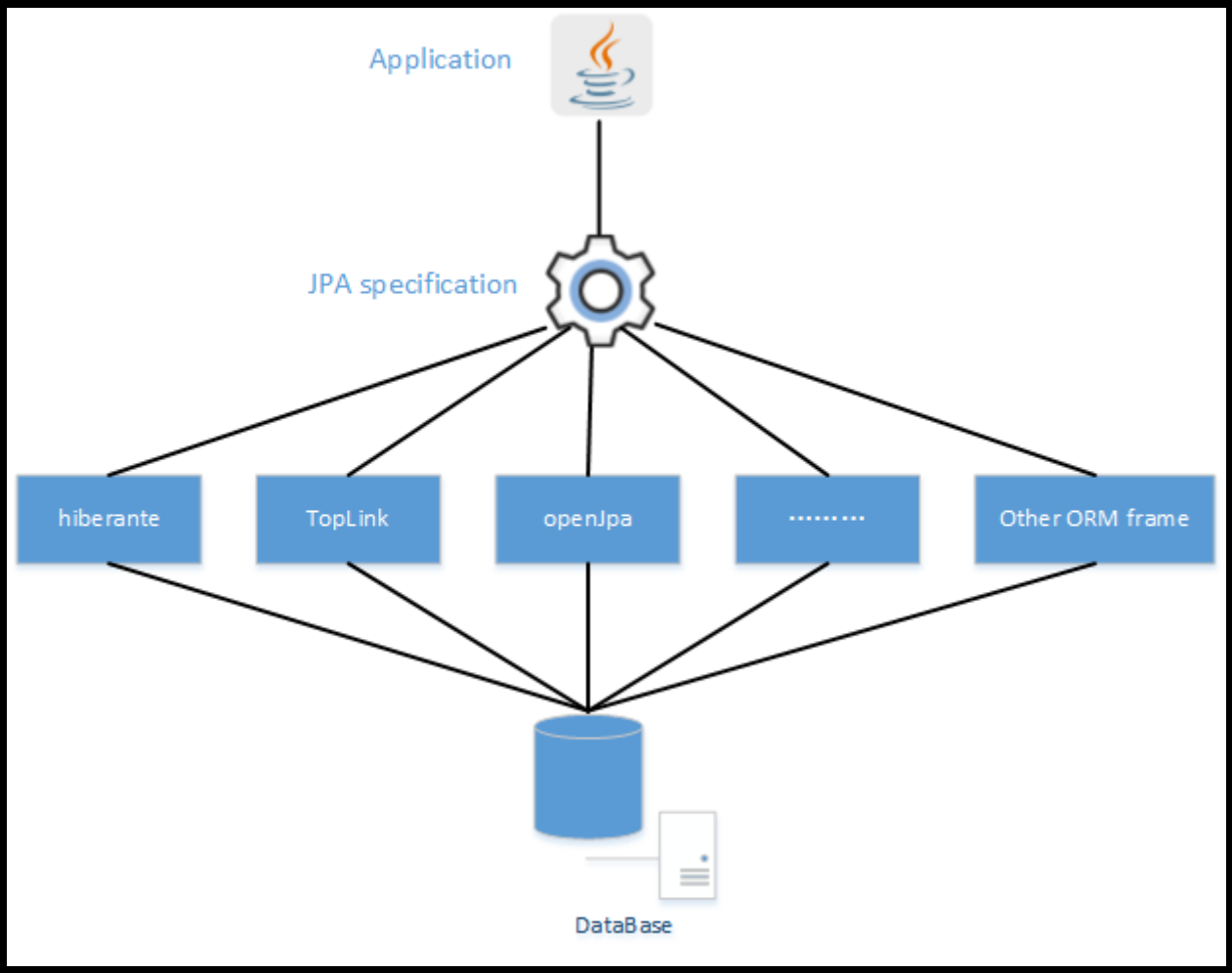


Spring Data JPA, JPA,Hibernate

JPA 规范本质上就是一种 ORM 规范，注意不是 ORM 框架——因为 JPA 并未提供 ORM 实现，它只是制订了一些规范，提供了一些编程的 API 接口，但具体实现则由服务厂商来提供实现，JBoss 应用服务器底层就以 Hibernate 作为 JPA 的实现。

JPA 规范给开发者带来了福音：**开发者面向 JPA 规范的接口，但底层的 JPA 实现可以任意切换**：觉得 Hibernate 好的，可以选择 Hibernate JPA 实现；觉得 TopLink 好的，可以选择 TopLink JPA 实现……这样开发者可以避免为使用 Hibernate 学习一套 ORM 框架，为使用 TopLink 又要再学习一套 ORM 框架。

下图是 JPA 和 Hibernate、TopLink 等 ORM 框架之间的关系：



Spring Data JPA 是在 JPA 规范的基础下提供了 Repository 层的实现，但是使用那一款 ORM 需要你自己去决定。

实际使用

在实际的工程中，推荐采用 **Spring Data JPA + ORM(如： Hibernate)进行开发**，这样在**切换不同的 ORM 提供了方便**，同时也使得 Repository 变得简单。程序低耦合。

spring-boot 中使用

需要在 pom.xml 中添加如下代码

```
<dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-jpa</artifactId>
</dependency>

<!--spring-boot-starter-data-jpa 包含 spring-data-jpa、spring-orm 和 Hibernate 来支持 JPA-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
```

```
</dependency>
```

Hibernate 插件安装

https://blog.csdn.net/yaosilani/article/details/81900856

http://download.jboss.org/jbosstools/oxygen/stable/updates/

HelloWorld

Step1　新建一个 Java project（不一定是 Java web），新建 lib，加包：

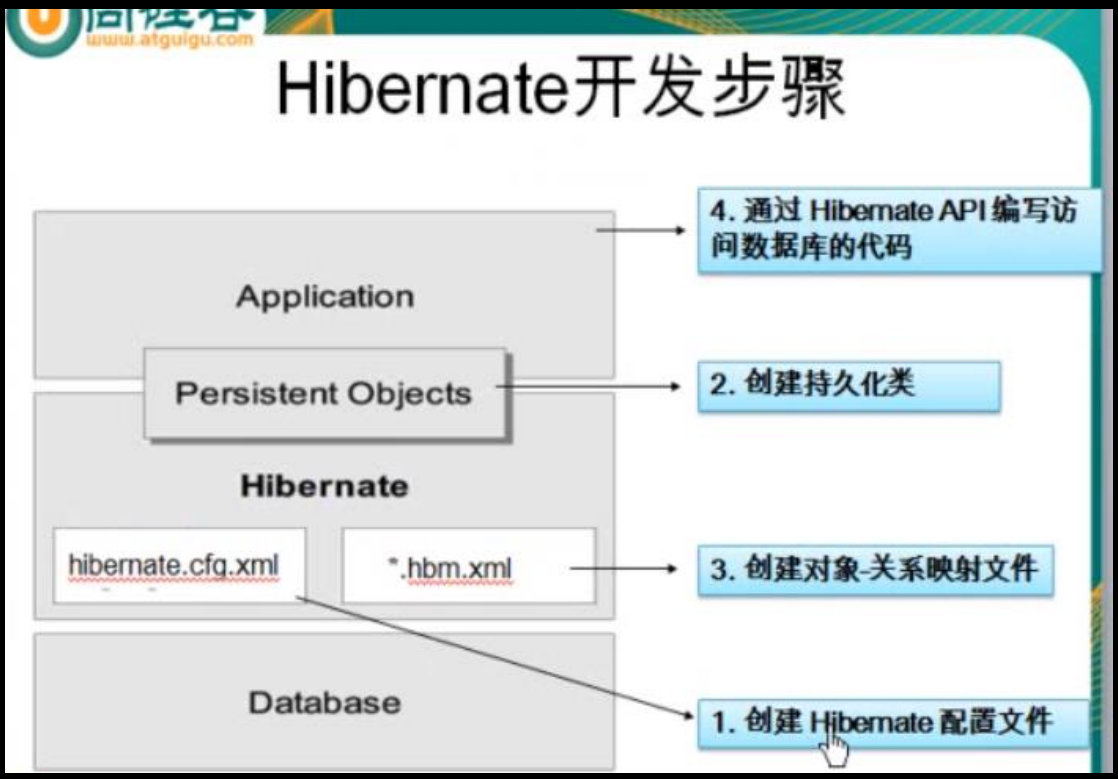
Step2　新建配置文件 hibernate Configuation File（如果点击没反应，降低 hibernate 版本）

hibernate.cfg.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>

    <!-- 配置连接数据库的基本信息 -->
        <property name="connection.usename">root</property>
        <property name="connection.password">xt222483</property>
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.url">jdbc:mysql:///Hibernate</property>
        <!-- jdbc:mysql://localhost/Hibernate  中localhost是默认的，所以上面 -->

        <!-- 配置hibernate的基本信息 -->
        <!-- hibernate所使用的数据库方言 -->
        <property name="dialect">org.hibernate.dialect.MySQLInnoDBDialect</property>


        <!-- 执行操作时是否在控制台打印SQL -->
        <property name = "format_sql">true</property>

        <!-- 指定自动生成数据表的策略 -->
        <property name = "hbm2ddl.auto">update</property>
    </session-factory>
</hibernate-configuration>

```

Step3

Hibernate 开发步骤



Spring Boot+Hibernate

依赖：

```
<dependency>
	<groupId>org.springframework.boot</groupId>
	<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
	<groupId>mysql</groupId>
	<artifactId>mysql-connector-java</artifactId>
</dependency>

jpa 就是 hibernate 的模式
```

配置：

```
- config:
spring.datasource.url=jdbc:mysql://localhost:3306/databasename
spring.datasource.username=root
spring.datasource.password=password
spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.jpa.properties.hibernate.hbm2ddl.auto=update #该属性有不同选项
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect
spring.jpa.show-sql=true
```

创建 PO：

```

@Entity(name = "city")
public class City {
	@Id
```

```
    @Column(columnDefinition = "id")
    private int id;

    @Column(name = "name")
    private String name;

    @Column(name = "District")
    private String District;

    @Column(name = "Population")
    private int Population;

    public City() {
    }
}
```

创建 Repository：

```
@Repository("cityRepository")
@Table(name = "city")
public interface CityRepository extends JpaRepository<City,Long> {
    @Query("select c from city c where c.id=:id")
    City findCityByID(@Param("id") int id );
}
```

使用：

```
@Controller
public class CityController {
    @Qualifier("cityRepository")
    @Autowired
    private     CityRepository cityRepository;

    @GetMapping(value = "/city")
    public String findCityById(@Param("id") int id, Model model){
        City city = cityRepository.findCityByID(id);
        System.out.println(city.getName());
        model.addAttribute("city",city);
        return "QueryCity";
    }
}
```