



全栈开发 & 系统设计串讲

小王子，韩立，Lance 等FLAG老师



课程介绍

课程主要分为三个章节和部分。

了解全栈开发以及基础知识准备

实现一个项目来帮助理解全栈开发

系统设计以及云服务实践和思考

课前申明



版权归上岸教育所有，任何不经同意的录象，转发行为将被视为侵犯版权。

课前须知



我假设你们已经知道了...

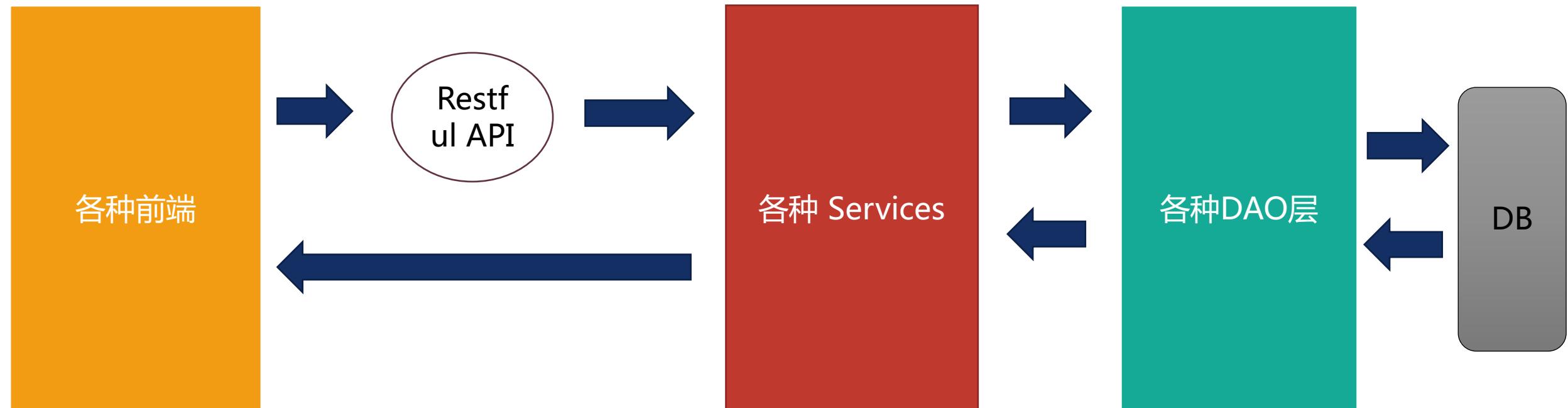
JS的基础语法

网络及http传输的基础知识

Model-View-Controller

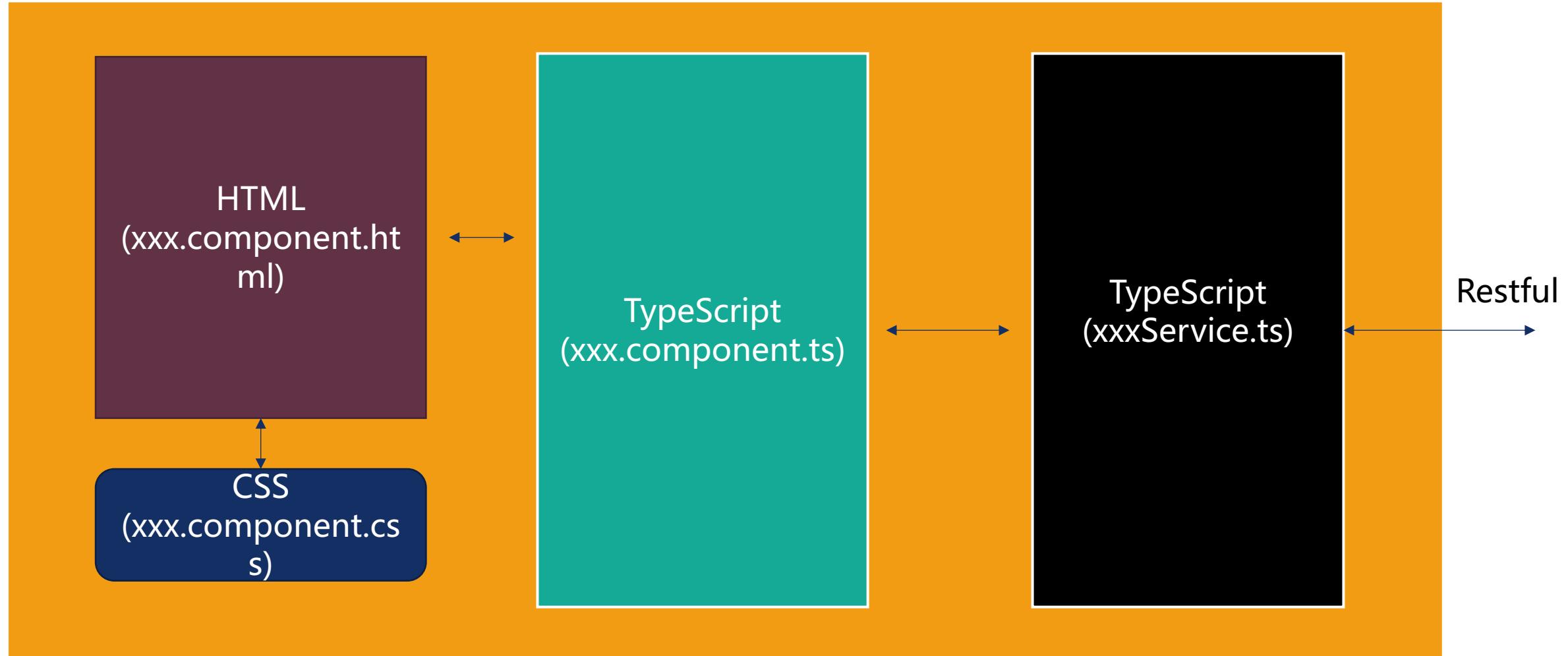
- 生成Spring 项目
 - IntelliJ (如果有兴趣可以问助教要小视频)
- 下载
 - Postman
- 项目GitHub
 - <https://github.com/wang2510/SpringBootTutor>

前后端数据传输画脑图



DTO 的双向传输

Angular 前端画脑图





什么是后端

Spring, Springboot, API

PART I



数据

数据的增删改查以及存储传输是后端解决的核心问题

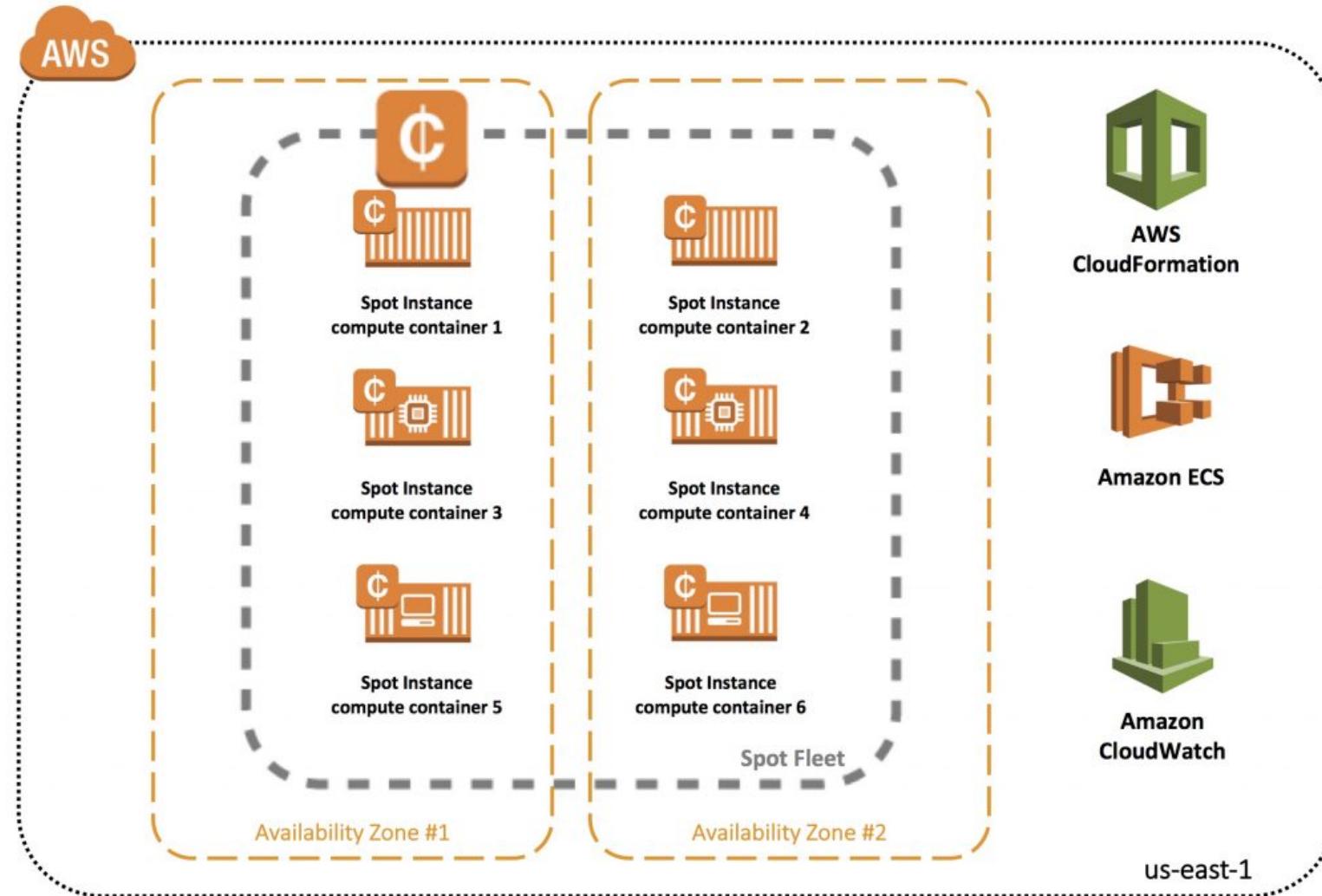
后端有很多不同的框架



所以问一个老问题，什么是框架？

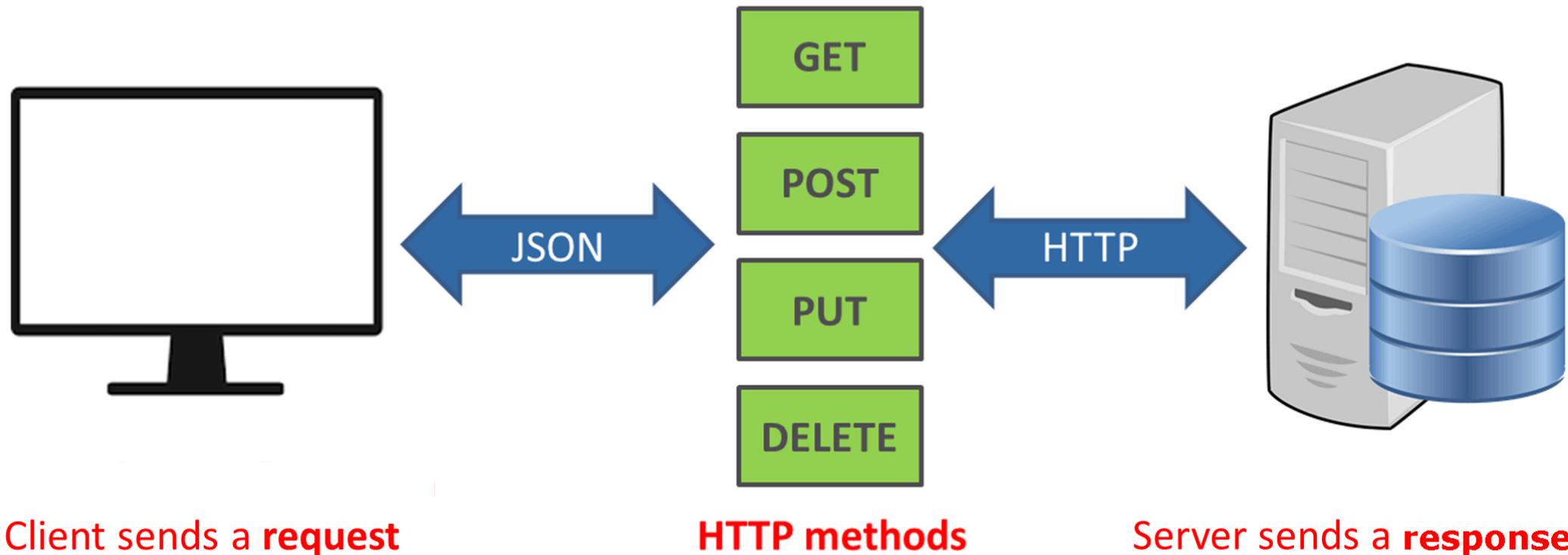
答不出来就去面壁

后端的构成



后端的构成

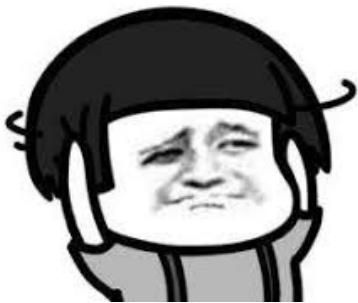




Application programming interface

服务器向外敞开的交流窗口

我不听我不听

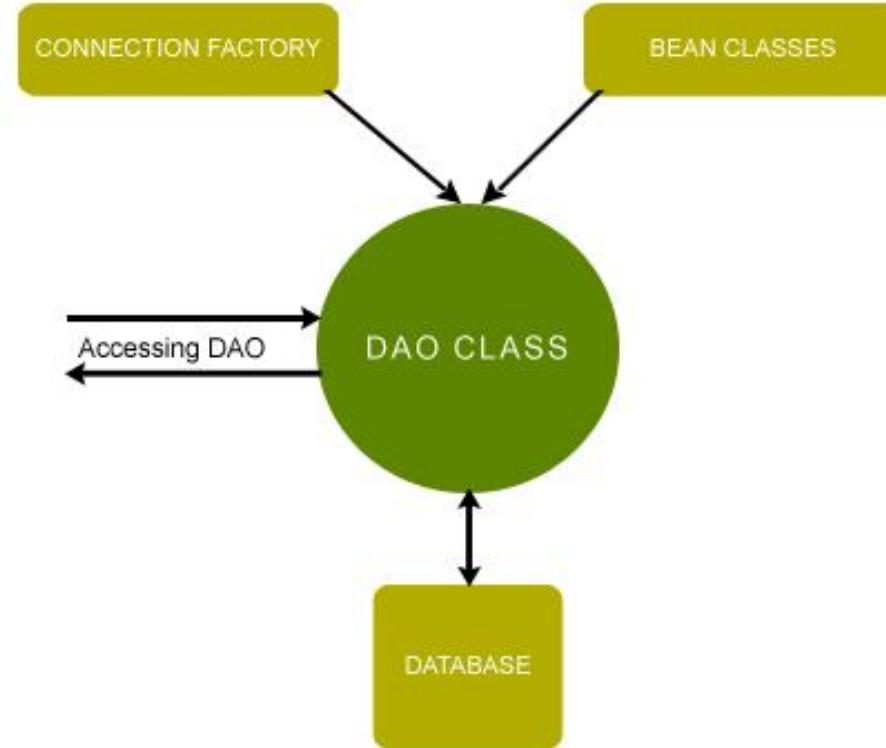


- URL定位资源，用HTTP动词（GET,POST,PUT,DELETE）描述操作。
 - Resource：资源，即数据
 - Representational：某种表现形式，比如用JSON, XML, JPEG等
 - State Transfer：状态变化。通过HTTP动词实现
- 所以RESTful API就是REST风格的API。

- 资源。首先是弄清楚资源的概念。资源就是网络上的一个实体，一段文本，一张图片或者一首歌曲。资源总是要通过一种载体来反应它的内容。文本可以用TXT，也可以用HTML或者XML、图片可以用JPG格式或者PNG格式，JSON是现在最常用的资源表现形式。
- 统一接口。RESTful风格的数据元操CRUD
(create,read,update,delete) 分别对应HTTP方法：GET用来获取资源，POST用来新建资源（也可以用于更新资源），PUT用来更新资源，DELETE用来删除资源，这样就统一了数据操作的接口。

- 可以用一个URI（统一资源定位符）指向资源，即每个URI都对应一个特定的资源。要获取这个资源访问它的URI就可以，因此URI就成了每一个资源的地址或识别符。一般的，每个资源至少有一个URI与之对应，最典型的URI就是URL。

- 无状态。所谓无状态即所有的资源都可以URI定位，而且这个定位与其他资源无关，也不会因为其他资源的变化而变化。有状态和无状态的区别，举个例子说明一下，例如要查询员工工资的步骤为第一步：登录系统。第二步：进入查询工资的页面。第三步：搜索该员工。第四步：点击姓名查看工资。这样的操作流程就是有状态的，查询工资的每一个步骤都依赖于前一个步骤，只要前置操作不成功，后续操作就无法执行。如果输入一个URL就可以得到指定员工的工资，则这种情况就是无状态的，因为获取工资不依赖于其他资源或状态，且这种情况下，员工工资是一个资源，由一个URL与之对应可以通过HTTP中的GET方法得到资源，这就是典型的RESTful风格。



DAO Internal Processing



访问数据库有很多方式

直接访问

DAO层访问

直接访问

[https://www.tutorialspoint.com/jdbc/jdbc-sample-code.htm](https://www.tutorialspoint.com/jdbc/jdbc_sample_code.htm)

```
sql = "SELECT id, first, last, age FROM Employees";
ResultSet rs = stmt.executeQuery(sql);

//STEP 5: Extract data from result set
while(rs.next()){
    //Retrieve by column name
    int id   = rs.getInt("id");
    int age  = rs.getInt("age");
    String first = rs.getString("first");
    String last = rs.getString("last");

    //Display values
    System.out.print("ID: " + id);
    System.out.print(", Age: " + age);
    System.out.print(", First: " + first);
    System.out.println(", Last: " + last);
}
```

Data Access Object

把数据库的每一张表映射成一个class entity

技术关键词



Restful API

Database

Hibernate

Spring/Springboot

Backend

MySQL



后端的实现

Springboot, API, MySQL

PART 2

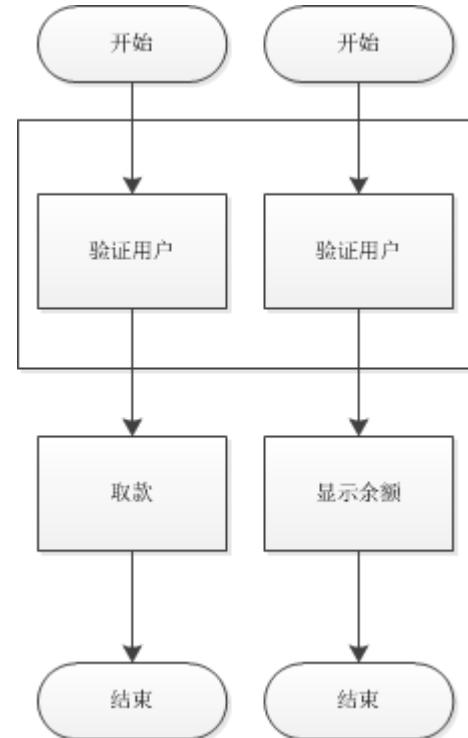
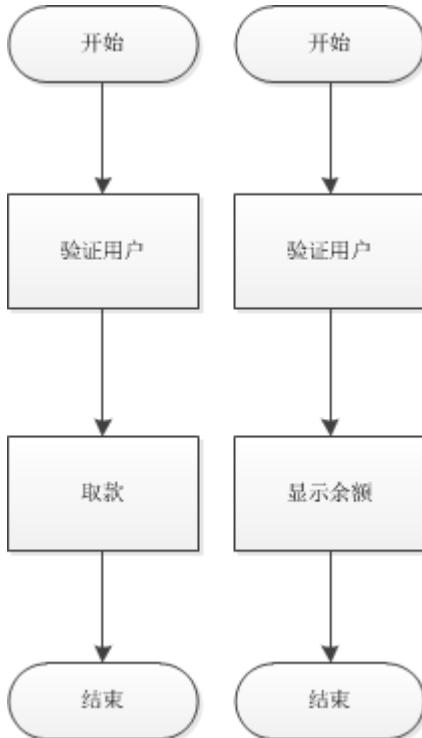
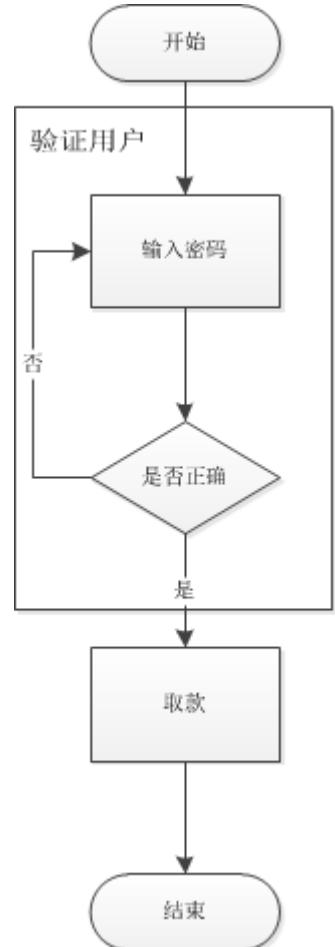


Demo1



Spring/Springboot

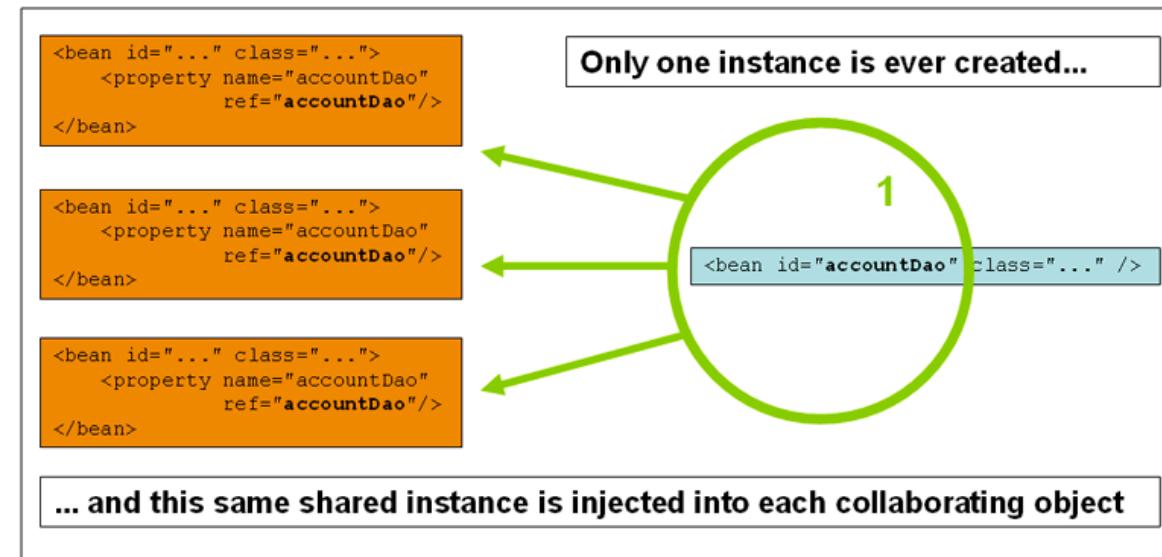
• 面向切面编程



Spring Bean



- **Spring - Bean** Definition. Advertisements. The objects that form the backbone of your application and that are managed by the **Spring** IoC container are called **beans**. A **bean** is an object that is instantiated, assembled, and otherwise managed by a **Spring** IoC container.



Spring Bean



- A. bean是对象，一个或者多个不限定
- B. bean由Spring中一个叫IoC的东西管理
- C. 我们的应用程序由一个个bean构成

Inversion of Control

控制反转通过依赖注入（ DI ）方式实现对象之间的松耦合关系。

程序运行时，依赖对象由【辅助程序】动态生成并注入到被依赖对象中，动态绑定两者的使用关系。

Spring IoC容器就是这样的辅助程序，它负责对象的生成和依赖的注入，让后在交由我们使用。

IoC就是一个对象定义其依赖关系而不创建它们的过程

Spring Bean



- XX objectName = new XX;
- 如果这个Object会被很多别的class新建怎么办？
- 我怎么去管理这个Object保证不会垃圾太多？
- 我怎么统一去管理这个Object的Life Cycle？
- Bean能帮你统一管理！
- 对于普通的Java对象，当new的时候创建对象，当它没有任何引用的时候被垃圾回收机制回收。而由Spring IoC容器托管的对象，它们的生命周期完全由容器控制。

Spring Bean



在Spring中，我们基本不需要 new 一个类，这些都是让 Spring 去做的。

Spring 启动时会把所需的类实例化成对象，如果需要依赖，则先实例化依赖，然后实例化当前类。

因为依赖必须通过构建函数传入，所以实例化时，当前类就会接收并保存所有依赖的对象。

这一步也就是所谓的**依赖注入**。

Spring Bean



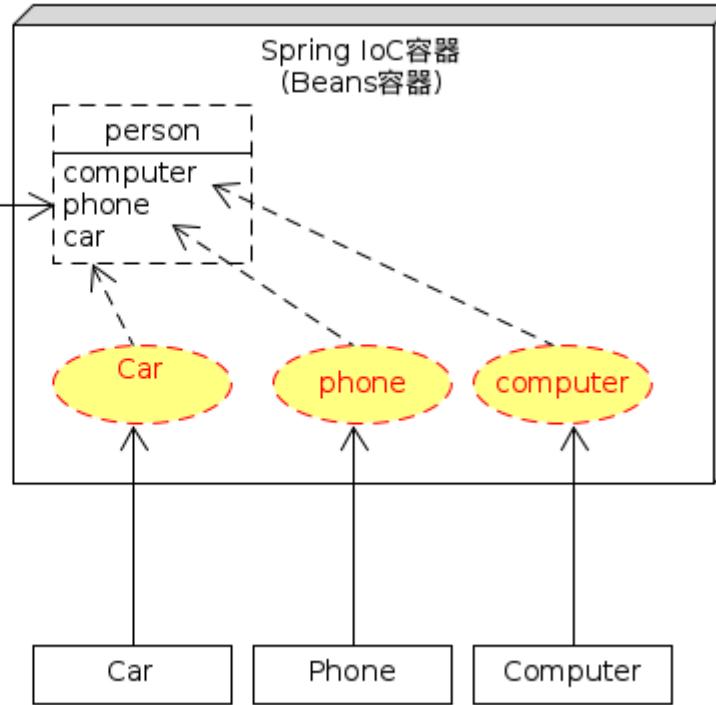
Bean 的重要概念

- A. Bean容器，或称spring ioc容器，主要用来管理对象和依赖，以及依赖的注入。
- B. bean是一个Java对象，根据bean规范编写出来的类，并由bean容器生成的对象就是一个bean。
- C. bean规范。

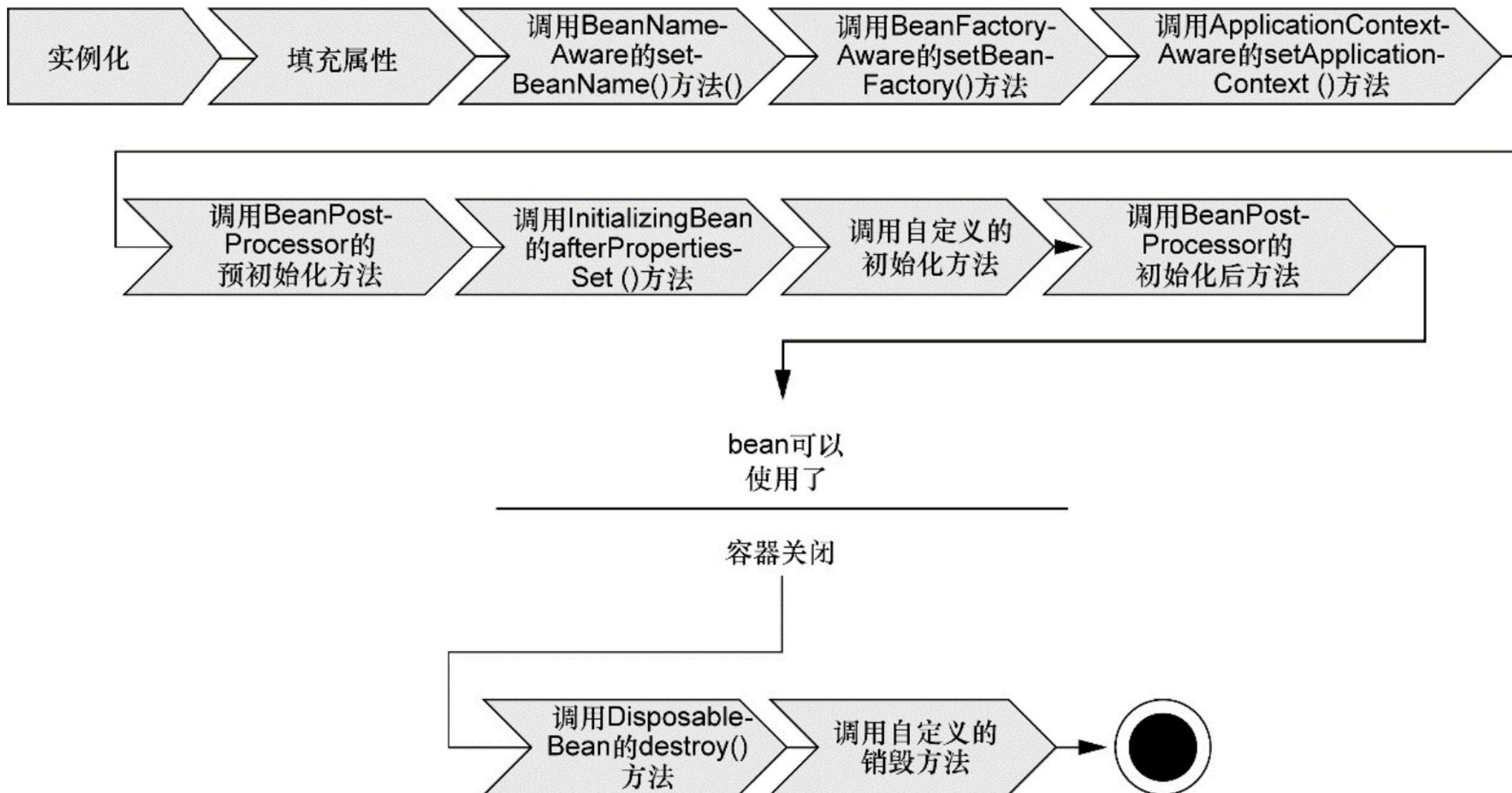
bean规范如下：

- 1.所有属性为private
- 2.提供默认构造方法
- 3.提供getter和setter
- 4.实现serializable接口

Person	
- computer:	Computer
- phone:	Phone
- car:	car
+ Person()	



Spring Bean Life Cycle

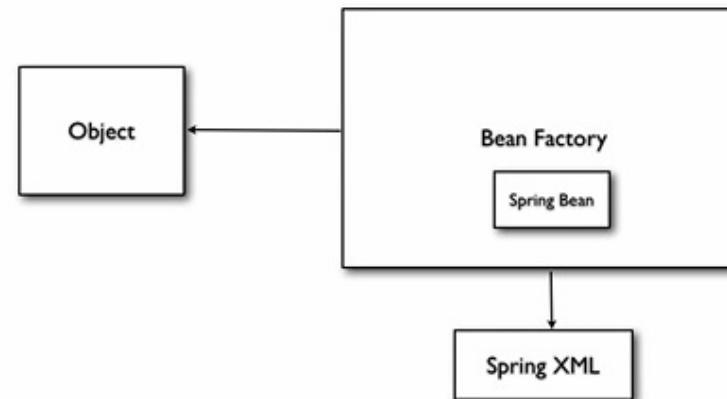


Spring Bean Factory



- **BeanFactory** is the actual container which instantiates, configures, and manages a number of **beans**. These **beans** typically collaborate with one another, and thus have dependencies between themselves. These dependencies are reflected in the configuration data used by the **BeanFactory**.

Spring Bean Factory

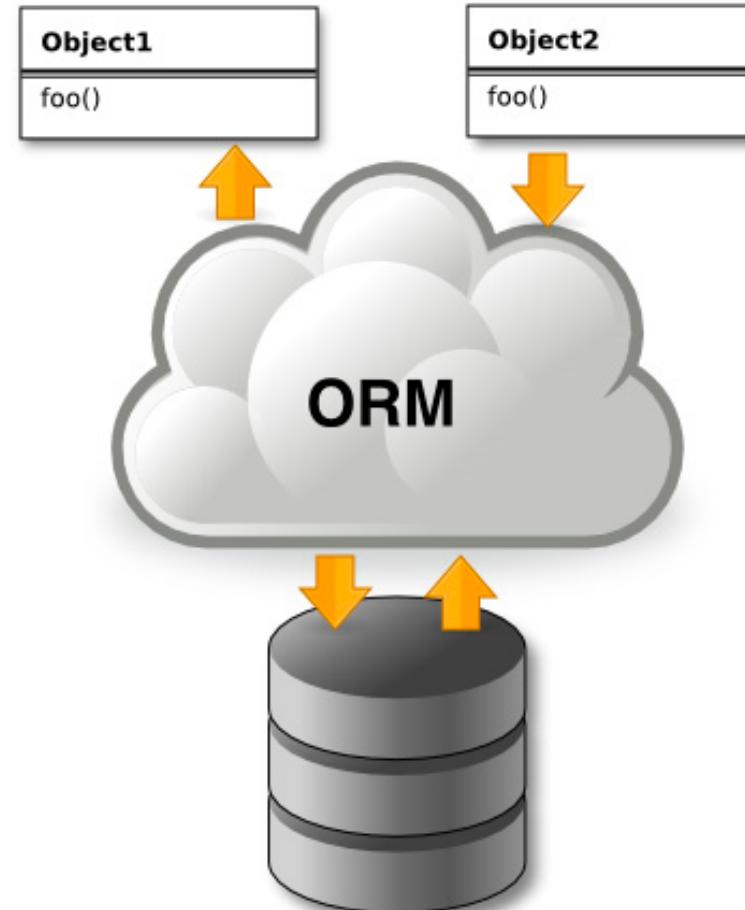


DAO, MYSQL, REPOSITORY

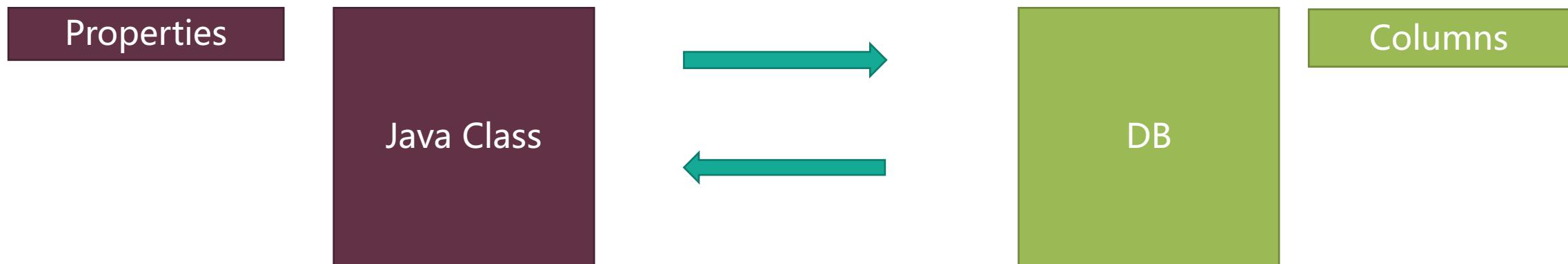
```
• CREATE TABLE course (
    • id bigint(20) NOT NULL,
    • course_name varchar(100) NOT NULL,
    • course_location varchar(30) NOT NULL,
    • course_content varchar(200) NOT NULL,
    • teacher_id bigint(10) NOT NULL,
    • PRIMARY KEY (id),
    • UNIQUE KEY `UK_course_name` (course_name)
  ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT
CHARSET=utf8;
```

- CREATE TABLE user_course (
- id bigint(20) NOT NULL,
- user_id bigint(20) NOT NULL,
- course_id bigint(20) NOT NULL,
- PRIMARY KEY (id),
- CONSTRAINT FK_user_id_user_id FOREIGN KEY (user_id)
REFERENCES jhi_user (id),
- CONSTRAINT FK_course_id_course_id FOREIGN KEY (course_id)
REFERENCES course (id)
-) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT
CHARSET=utf8;

Hibernate



- 那就要说到我们 MVC提到的model了...
 - 宏观来说，每一个在数据库的数据列都是一个model
 - 比如学生（有名字，年纪，年级等等）
 - 比如课程（有名字，有地点，有内容等等）





实现一个简单的课程添加API

- 如何从数据库查询所有课程？
- 回顾什么叫做restful api?
- 如何使用工具Postman

- 实现

- 学生能够查询所有课程
- 学生能够注册课程
- 学生能够查询所有已经注册课程
- 学生能够注销注册课程

- 进阶

- 老师能够查询所有课程
- 老师能够添加课程
- 老师能够修改课程信息
- 老师能够删除课程
- 老师能够查询已经注册某课程的所有学生

Questions?



THANK YOU

