**实习日志**

**第一周：2019.6.3-2019.6.6**

**2019.6.3：第一天**

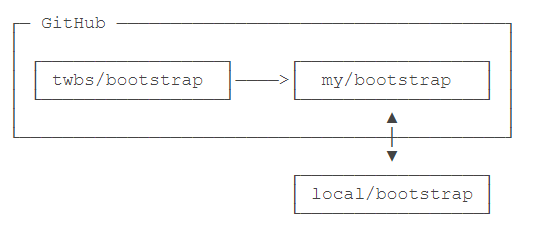
1、下载安装了Java的编译器IntelliJ IDE，安装了JDK，配置了Tomcat，安装了MySQL。

2、了解了一下SSH和SSM。SSH是Struts+Spring+Hibernate的缩写，SSM是SpringMVC+Spring+Mybatis的缩写，大概知道了各个部分是什么，但是具体的还搞不清楚就是了。本来想在IntelliJ IDE上搭建SSH的，但是失败了，看到网上的教程说最好是逐个击破之后，理解之后才好整合三个东西。

**2019.6.4：第二天**

1、确定了要学习的东西，首先是要了解一些Git的一些用法，然后是Maven这个项目管理工具，接着SSH（SpringMVC+Spring+Hibernate）框架逐个击破，先把SpringMVC搞定，然后是Hibernate，最后才是Spring，Spring是管理各层的组件，是一个统筹者的作用，所以要最后去攻破。

2、今天用了一天时间去学习Git，按照廖雪峰的Git教程一步一步操作【[网址](https://www.liaoxuefeng.com/wiki/896043488029600/896067074338496)】。安装了Git，学习了版本管理、远程仓库、分支管理和标签管理，并且整理了一个Word笔记出来，把常用的一些Git语句给整理出来了，以后方便查找。照着整个流程下来，大致了解了分布式版本控制的一些优点，了解了Git和GitHub的区别。Git是分布式版本控制系统，GitHub是免费的远程仓库。



**2019.6.5：第三天**

1、今天又练习了一下Git的一些操作，尤其是分支管理的原理和操作，熟悉了一下一些命令行指令。Git的命令有超多的，记住常用的一些就可以了，一般的可以去查文档就行。然后对于rebase的这个命令的功能不是很懂【[参考网址](https://git-scm.com/book/zh/v2/Git-%E5%88%86%E6%94%AF-%E5%8F%98%E5%9F%BA)】

2、安装了Git的图像界面软件TortoiseGit，随便玩了一下，但是感觉还是命令行比较快，先学习的用命令行的，习惯用命令行了，所以用图形界面不是很习惯。用的时候有一个小问题就是：Tortoise右键目录没有找到git clone和git create repository here，要按住shift+右键，然后就能看到了。

3、在IntelliJ IDEA中配置Git和GitHub，玩了一下怎么把项目整个add到暂存区/commit到本地仓库/push到远程仓库中。【[参考网址](https://blog.csdn.net/qq_27467601/article/details/78720839)】

**2019.6.6：第四天**

1、花了三个小时帮辉哥统计员工平均工作日加班时长，哈哈哈，搞了一两个小时，学会怎么用Excel自定义函数。用VBA写了一个函数，感觉跟MATLAB好像啊。还有知道筛选操作之后不能直接用求和和计数函数，直接用会出现很大的问题，会把表格要用subtotal(funcnum, )这个函数。哈哈哈哈，也是涨知识了，Excel要玩的6也是很牛逼啊！

2、merge（合并）和rebase（变基）的共同点和不同点：

（1）共同点：整合来自不同分支的修改，整合的最终结果所指向的快照始终是一样的，只不过提交历史不同罢了。

（2）不同点：变基是将一系列提交按照原有次序依次应用到另一分支上，而合并是把最终结果合在一起。变基使得提交历史更加整洁，要把变基命令当作是在推送前清理提交使之整洁的工具，并且只在从未推送至共用仓库的提交上执行变基命令，就不会有事。

**总的原则是：**只对尚未推送或分享给别人的本地修改执行变基操作清理历史，从不对已推送至别处的提交执行变基操作。

3、大概了解了一下Maven的作用，Maven的核心功能是合理叙述项目间的依赖关系，通俗点讲，就是通过pom.xml文件的配置获取jar包，而不用手动去添加jar包。网上关于Maven的教程太书面化了，各种概念和名词，表示看不太懂，然后想利用IDEA配置Tomcat并运行一个Web项目来玩玩，毕竟实践出真知。在网上找了一个教程，配置失败，没搞出来，贼难受。到配置Tomcat的时候，出现了no artifact的错误，没搞定这个错误。跟着教程走，但是会遇到和教程不一样的错误，也是很坑啊！

**第二周：2019.6.10-2019.6.14**

**2019.6.10：第五天**

1、CDN原理：

CDN主要是解决互联网中间一公里的速度问题，CDN是通过缓存的方式让访问就近完成，并为终端用户的访问提供最快速、稳定的访问路径。

2、利用Maven配置Tomcat运行一个Web项目（HelloWorld），不是很懂facets和artifacts的具体作用，就只知道要部署这两个东东。

Facets用于配置Web项目结构，用于之后的Artifacts中:

Deployment Descriptors: 配置web.xml文件位置

Web Resource Directories: 配置webapp目录，其中包括css/js/jsp等资源。

Artifacts：

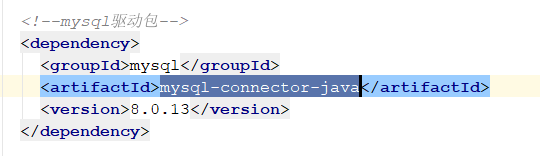
artifact是一个项目资源的组合体。例如，一个已编译的java类的集合，一个已打包的java应用。

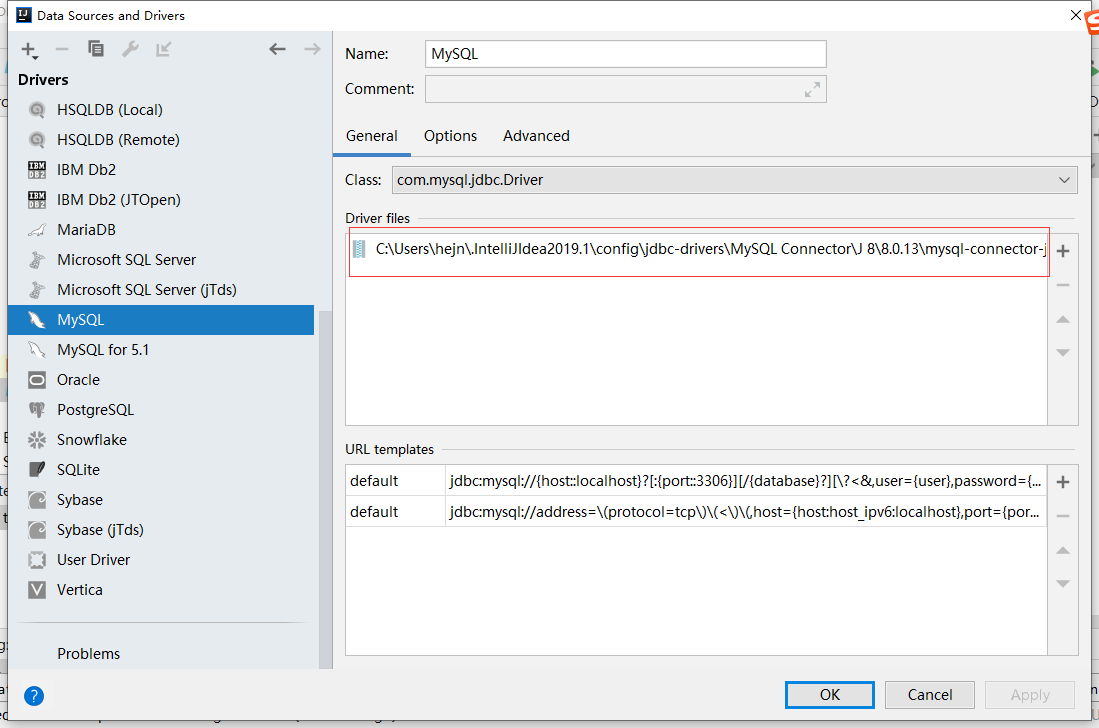
3、在完成第一步的基础上要加SpringMVC框架的过程中，失败了。

**2019.6.11：第六天**

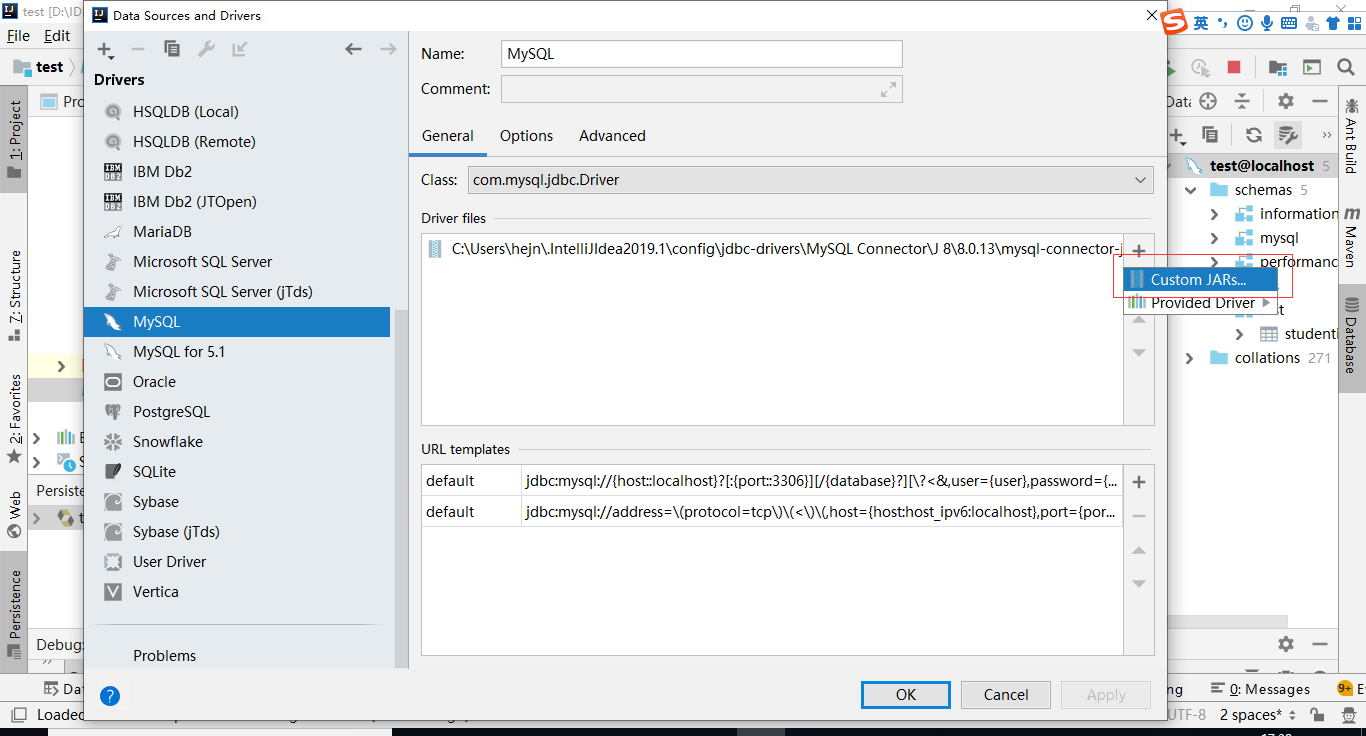
1、要先装MySQL server再装图形界面MySQL workbench，之前直接装的MySQL workbench发现居然没有MySQL服务，我一直以为workbench里会一起装的，以前装过都忘了。

2、MySql的问题，连接驱动版本的问题搞了我半天，pom.xml里和Data Sources and Drivers里的driver files的版本要一致。





而且是只能从Custom JARS中选择本地下载的相应的MySQL连接驱动版本，否则就是容易出错。



3、参考[博客](http://zhangqiheng.cn/2018/06/19/%E5%9C%A8IDEA%E4%B8%8A%E6%90%AD%E5%BB%BASpring+SpringMVC+Hibernate%E9%A1%B9%E7%9B%AE/)， 终于成功的在IDEA上搭建Spring+SpringMVC+Hibernate项目，太难得了！！！！

4、有用的一些资料：

（1）有用的资料：[IntelliJ IDEA 17和Maven构建javaWeb项目](https://www.cnblogs.com/zhangyinhua/p/7768163.html)

（2）[解决IDEA输出乱码问题](https://blog.csdn.net/qq_41264674/article/details/80945140)

（3）jsp页面中添加（重要）<%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8" %>

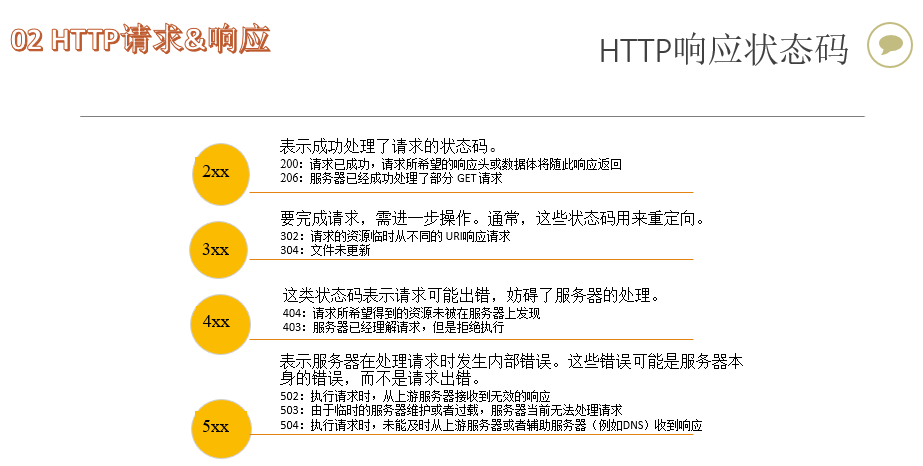
（4）[idea设置类注释和方法注释模板](https://blog.csdn.net/qinqigang/article/details/79071486)

（5）**很好的一篇博客**：[在IDEA上搭建Spring+SpringMVC+Hibernate项目](http://zhangqiheng.cn/2018/06/19/%E5%9C%A8IDEA%E4%B8%8A%E6%90%AD%E5%BB%BASpring+SpringMVC+Hibernate%E9%A1%B9%E7%9B%AE/)

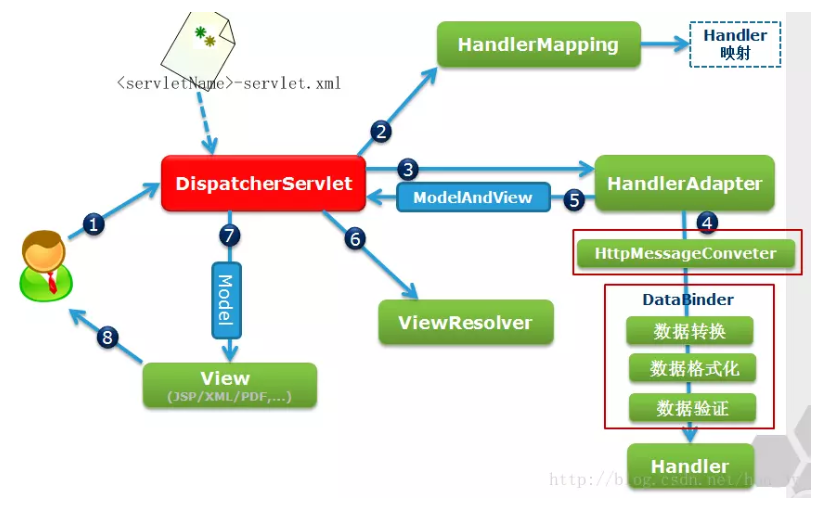
（6）数据库连接那里会出现两个问题一个是jdbc drivers版本的问题，一个是数据库和系统时区差异，第一个解决方法就是把driver的版本弄到最高，第二个解决方法就是使用的数据库是MySQL，驱动是8.0，这是由于数据库和系统时区差异所造成的，在jdbc连接的url后面加上serverTimezone=GMT即可解决问题，如果需要使用gmt+8时区，需要写成GMT%2B8，否则会被解析为空。再一个解决办法就是使用低版本的MySQL jdbc驱动，5.1.28不会存在时区的问题。jdbc:mysql://localhost:3306/mydatabase?serverTimezone=GMT

**2019.6.12：第七天**

1、随便瞄了一下HTTP的东西。



2、SpringMVC工作流程：

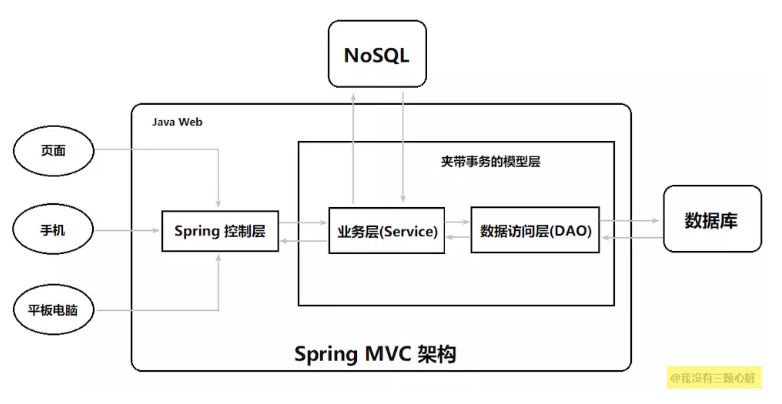


* **用户发送请求**
* **请求交由核心控制器处理**
* **核心控制器找到映射器，映射器看看请求路径是什么**
* **核心控制器再找到适配器，看看有哪些类实现了Controller接口或者对应的bean对象**
* **将带过来的数据进行转换，格式化等等操作**
* **找到我们的控制器Action，处理完业务之后返回一个ModelAndView对象**
* **最后通过视图解析器来对ModelAndView进行解析**
* **跳转到对应的JSP/html页面**

3、大概明白了SpringMVC的 整个工作流程，最主要的是DispatcherServlet起到了一个核心控制的作用。

**2019.6.13：第八天**

1、为解决持久层中一直未处理好的数据库事务的编程，又为了迎合 NoSQL 的强势崛起，Spring MVC给出了方案：



**传统的模型层被拆分为了业务层(Service)和数据访问层（DAO，Data Access Object）。**在 Service下可以通过 Spring的声明式事务操作数据访问层，而在业务层上还允许我们访问NoSQL，这样就能够满足异军突起的NoSQL的使用了，它可以大大提高互联网系统的性能。

2、Controller+Service +Dao三层的功能划分

**Controller是管理业务（Service）调度和管理跳转的。**

**Service是管理具体的功能的。**

**Controller只负责管理，而Service负责实施。**

DAO只完成增删改查，虽然可以1-n，n-n，1-1关联，模糊、动态、子查询都可以。但是无论多么复杂的查询，dao只是封装增删改查。至于增删查改如何去实现一个功能，dao是不管的。

Entity层就是对应的数据库表的实体类。Dao层是使用了Hibernate连接数据库、操作数据库（增删改查）。Service层：引用对应的Dao数据库操作。

Controller层：引用对应的Service层，结合SpringMVC注释，跳转到指定的页面，当然也能接受页面传递的请求数据，也可以做些计算处理。

Service 写api的，Controller写逻辑的业务，model：就是对象模型，Dao跟mapper 一样的处理数据的。

3、今天主要是把service层给剥离出来，让controller层尽量可能的不写跟业务逻辑相关的代码，也就是controller尽可能的使用service实现的API来实现对业务的一个管理，而业务的实现是service的工作，和controller没有关系。其实也就是在MVC三层架构中，给Model和Controller之间加了一座桥梁（Service），让Controller不能直接对应用程序的数据进行一个操作。

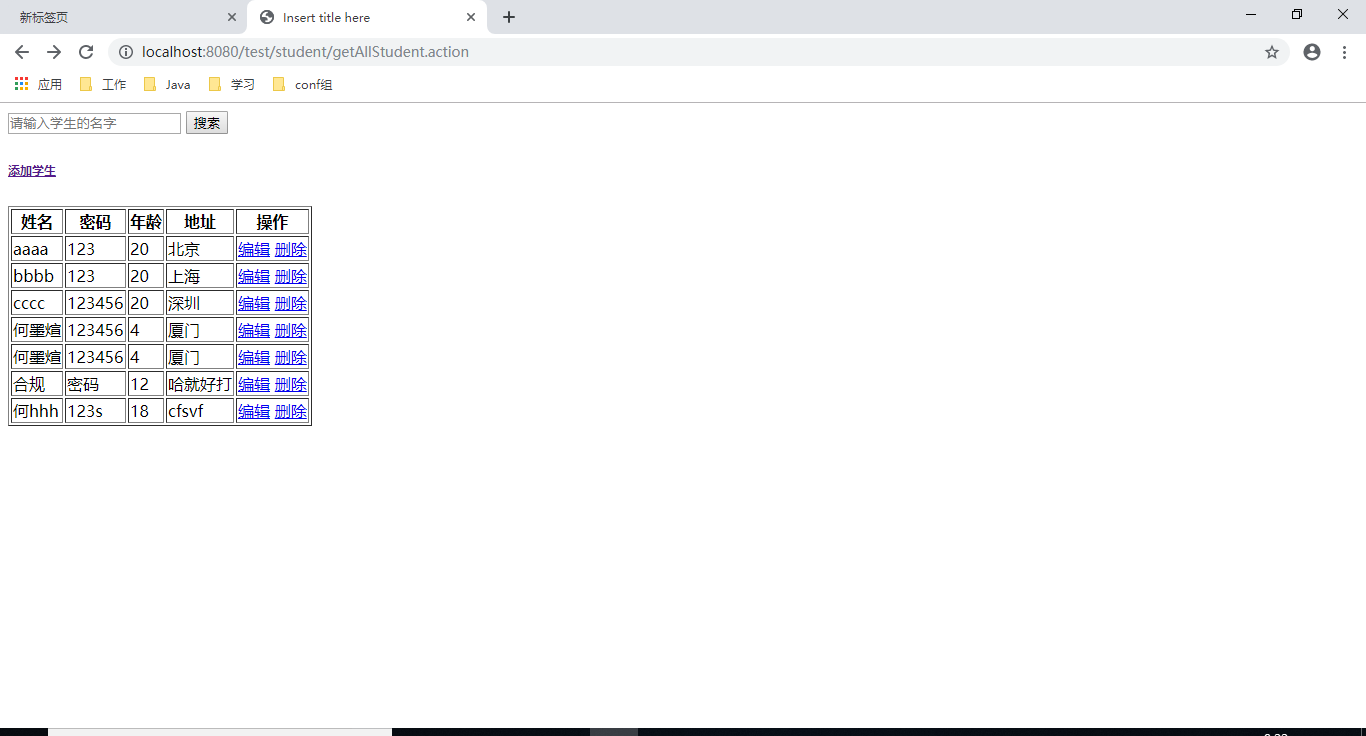
4、今天把从数据库里的所有学生表的数据读取出来并利用ModelAndView的方法将这个数据显现在jsp页面上。接下来想要用Hibernate封装数据库的增删查改方法的时候，出现了问题。想要实现用泛型封装数据库的增删查改方法，让不同的数据对象都可以用，但是出问题了，还没解决。



**2019.6.14：第九天**

1、我放弃用泛型封装数据库的增删查改方法，其实是不行，报错的地方我真的解决不了啊。

2、照着网上的例子，能够实现对于数据库中的学生表可以进行添加编辑和删除。但是搜索的时候出现了问题。在参数传递的时候出现了一点差错好像。



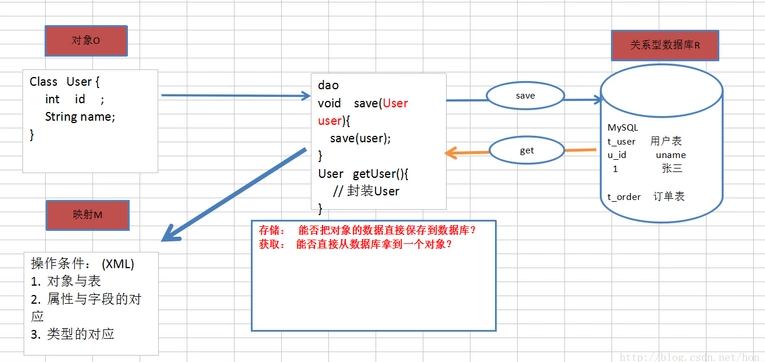
3、Hibernate是一种ORM的框架。那什么是ORM呢？ORM是一种思想

O代表的是Objcet

R代表的是Relative

M代表的是Mapping

ORM->对象关系映射....ORM关注是对象与数据库中的列的关系



Session是Hibernate最重要的对象，Session维护了一个连接（Connection），只要使用Hibernate操作数据库，都需要用到Session对象。

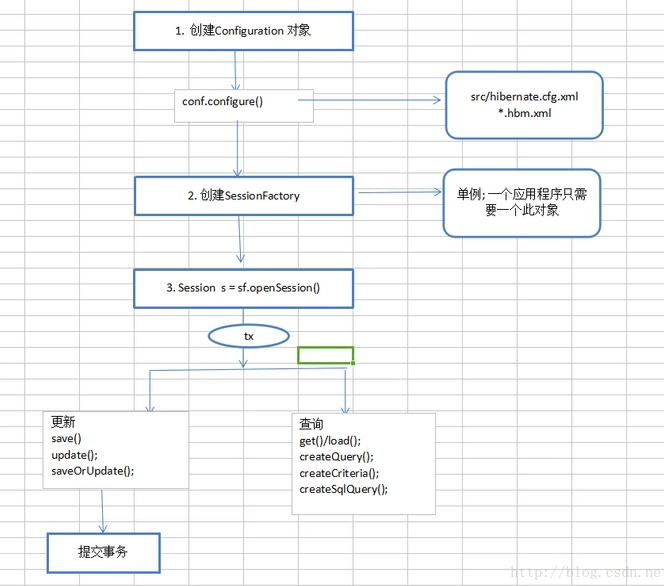
4、把Hibernate的一些操作进行了封装。但是前端的表单传参到后台的时候出现了疑惑，也就是入参的问题。

**第三周：2019.6.17-2019.6.21**

**2019.6.18：第十天**

1、搜索学生的功能完成了，如果name为空，就是搜索所有的学生并显示在表格中；如果name不为空，如果在数据库中存在该学生，那么就显示该学生的信息，如果数据库中不存在该学生，那么就显示空。之前一直以为是传递参数出现了问题，后面发现是传递参数是没有什么问题的，我用的是HttpServletRequest的方法。就是我去数据库中找的时候，返回的数据类型给弄错了，返回的应该是list<StudentinfoEntity>，我给返回的是StudentinfoEntity，所以一直在报错。

2、Hibernate执行流程图：



3、SpringMVC参数传递有一下几种形式：

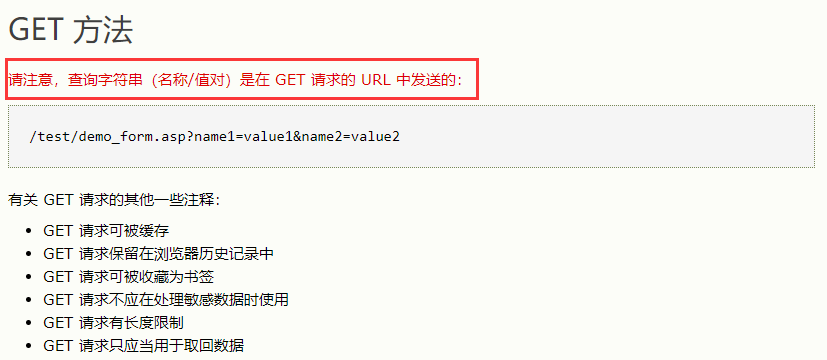
* request、response、session传递。
* 参数绑定
* url动态参数

4、Get和POST

在客户机和服务器之间进行请求-响应时，两种最常被用到的方法是：GET 和 POST。

GET - 从指定的资源请求数据。

POST - 向指定的资源提交要被处理的数据





**2019.6.19：第十一天**

1、前后端开发分离方案：

通常

**前端：负责浏览器端用户交互界面和逻辑等**

**后端：负责数据的处理和存储等**

后端要提供数据接口给前端，这样导致了前后端的开发是一种串行的工作方式，效率低下。

**因此，前端和后端需要依赖API开发来实现并行开发。**

愿景：让前端开发人员和后端开发人员的开发工作相互独立，互不干扰，减少耗时

思路：时间是前端开发人员浪费在等待后端接口上了，能不能不等接口，直接干就完事了

实践：

1. **事先约定好接口的标准**
2. **后端开始开发接口**
3. **前端通过假数据的方式模拟一个后端接口**

这就是—Mock。

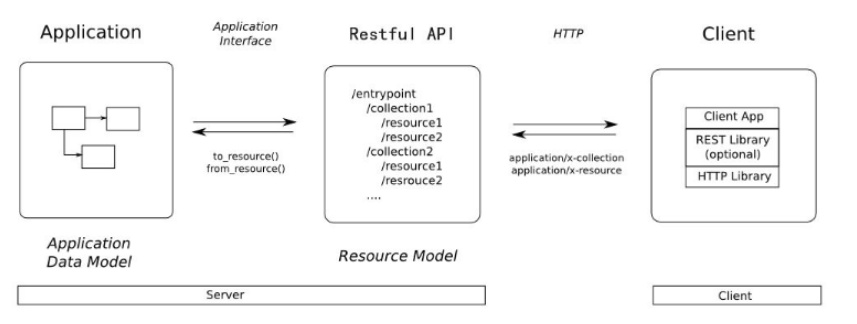
2、RESTful API

REST：Representation State Transfer的缩写【可重新表达的状态迁移—表示层状态迁移】

面向资源（一类数据）编程，（URL）地址即资源。

**对于资源我们的代码中可能的操作只有 增删改查**

**在HTTP协议中对每一个请求URL都会有不同的谓词GET（查）/POST（增）/PUT（改）/PATCH（改）/DELETE（删）**



**RESTful架构，就是目前最流行的一种互联网软件架构。**它结构清晰、符合标准、易于理解、扩展方便，所以正得到越来越多网站的采用。

GET /tickets # 获取ticket列表

GET /tickets/12 # 查看某个具体的ticket

POST /tickets # 新建一个ticket

PUT /tickets/12 # 更新ticket 12.

DELETE /tickets/12 #删除ticekt 12

REST特点如下：

基于HTTP协议

是另一种服务架构

**传递是JSON、POX（Plain Old XML）而不是SOAP格式的数据**

充分利用HTTP谓词（Verb）

**侧重数据的传输，业务逻辑交给客户端自行处理**

3、**REST 是面向资源的，强调描述应用程序的事物和名词。**这样很容易导致的一个结果是我们在设计 RESTful API 时，在 URI 中使用动词。例如：GET /user/getUser/123。正确写法应该是 GET /user/123。

4、我把我之前写的Controller，统统改成基于REST的Controller。URL的形式如果是xx/AA，xx/BB，这样的URL是一样的，如果方法也一样，都是method = RequestMethod.GET，那么这样是不行的，利用URL找controller方法是会出现ambitious（混淆），找不到对应的controller方法的，所以要避免出现这个状况。

我们的REST API :

GET 方式请求 /student 返回所有学生列表

GET 方式请求 / student/2返回id为2的用户

POST 方式请求 /student/ 通过student对象的JSON 参数创建新的student对象

PUT 方式请求 / student/3 更新id为3的发送json格式的用户对象

DELETE 方式请求/student/4删除 ID为4的user对象

**2019.6.20：第十二天**

参考网址：

* 【SpringMVC】—[@RequestParam 和 @PathVariable 注解](https://blog.csdn.net/zt15732625878/article/details/80382965)
* Spring MVC 学习总结（九）——[Spring MVC实现RESTful与JSON（Spring MVC为前端提供服务）](https://www.cnblogs.com/best/p/8006049.html#_label1)

1、@RequestParam 和 @PathVariable 注解是用于从request中接收请求的，两个都可以接收参数，关键点不同的是：

@PathVariable是从一个URL模板里面来取值（/后面?之前）

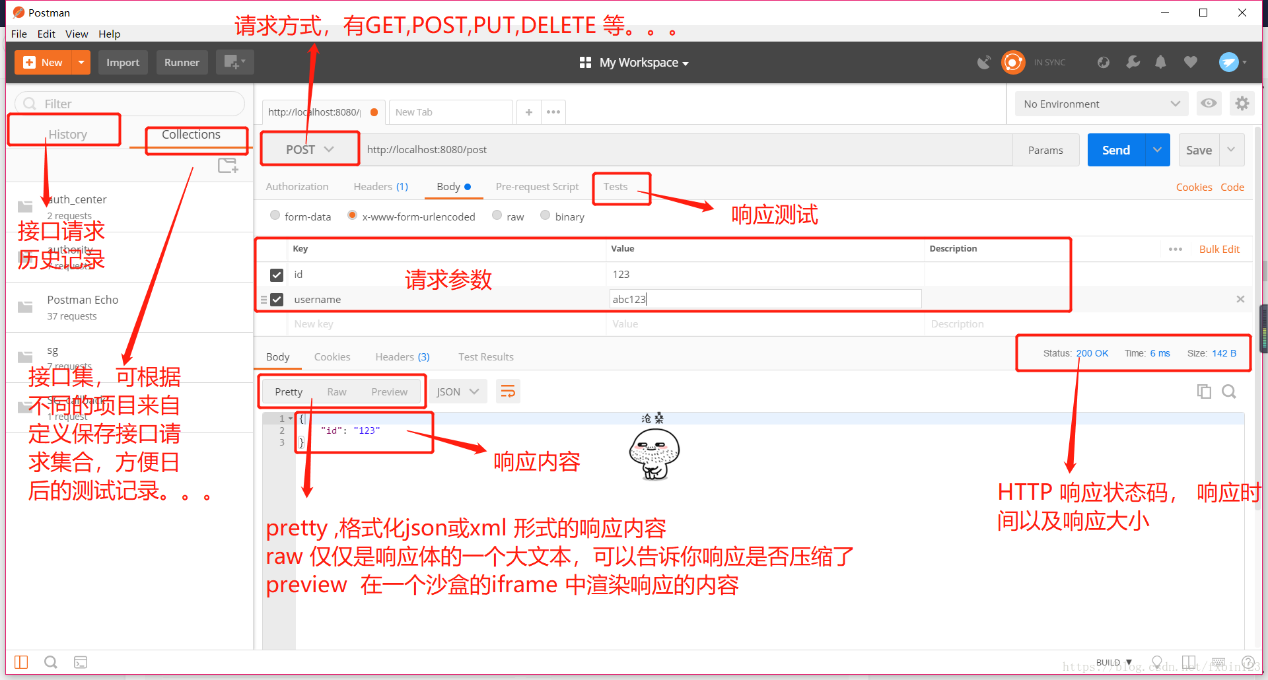
@RequestParam是从request里面取值（？之后）

2、JSON(JavaScript Object Notation, JS 对象标记) 是一种轻量级的数据交换格式。

3、处理http响应，ResponseEntity标识整个http相应：状态码、头部信息以及相应体内容。因此我们可以使用其对http响应实现完整配置。总结：简单粗暴的讲@ResponseBody可以直接返回Json结果， @ResponseEntity不仅可以返回json结果，还可以定义返回的HttpHeaders和HttpStatus。如果这个Controller类标注了@RestController这个注解，那么这个类底下的所有的方法默认会有@ResponseBody这个注解，也就是这个方法可以直接返回JSON数据格式的结果。

4、Postman：Postman 是一个很强大的 API调试、Http请求的工具。用户在开发或者调试网络程序或者是网页B/S模式的程序的时候是需要一些方法来跟踪网页请求的，用户可以使用一些网络的监视工具比如著名的Firebug等网页调试工具。今天给大家介绍的这款网页调试工具不仅可以调试简单的css、html、脚本等简单的网页基本信息，它还可以发送几乎所有类型的HTTP请求！

[参考网址](https://blog.csdn.net/fxbin123/article/details/80428216)



5、AJAX 是一种在无需重新加载整个网页的情况下，能够更新部分网页的技术。AJAX = 异步 JavaScript 和 XML。

AJAX 是一种用于创建快速动态网页的技术。

通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

传统的网页（不使用 AJAX）如果需要更新内容，必需重载整个网页面。

6、没搞出来如何用AJAX接收JSON数据格式的数据并在前端页面中显示出来，这个下周一解决吧。然后就开始看《Spring实战》这本了，看了第一章，对于要理解Spring的一些思想，就必须要理解DI（依赖注入）和AOP（面向切面编程），这两个是最重要的。

**第四周：2019.6.24-2019.6.28**

**2019.6.24：第十三天**

参考网址：

* [手把手教你整合最优雅SSM框架：SpringMVC + Spring + MyBatis](https://blog.csdn.net/qq598535550/article/details/51703190)

1、纵观目前主流的 ORM(对象关系映射)，无论 Hibernate还是Apache OJB，都对数据库结构提供了较为完整的封装，提供了从POJO到数据库表的全套映射机制。程序员往往只需定义好了POJO 到数据库表的映射关系，即可通过 Hibernate或者OJB 提供的方法完成持久层操作。程序员甚至不需要对 SQL 的熟练掌握，Hibernate/OJB 会根据制定的存储逻辑，自动生成对应的 SQL 并调用 JDBC 接口加以执行。

“半自动化”的ibatis，这里的“半自动化”，是相对Hibernate等提供了全面的数据库封装机制的“全自动化”ORM 实现而言，“全自动”ORM 实现了 POJO 和数据库表之间的映射，以及 SQL 的自动生成和执行。而ibatis 的着力点，则在于POJO 与 SQL之间的映射关系。也就是说，ibatis并不会为程序员在运行期自动生成 SQL 执行。具体的 SQL 需要程序员编写，然后通过映射配置文件，将SQL所需的参数，以及返回的结果字段映射到指定 POJO。

2、用Maven去部署Tomcat，这样比较顺利，要不然自己手动部署的话，其实很容易出问题的！在pom.xml里要写上：

<!--这个在部署Tomcat的时候很重要-->

<packaging>war</packaging>

3、载别人的GitHub项目下来用的时候要注意MySQL驱动的版本问题，在pom.xml中导入相应MySQL驱动依赖的时候，注意版本5和版本8的差异，这个差异会导致你的数据库连接成功与否。

<!--Mysql数据库驱动-->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.13</version>

</dependency>

4、跑了一个一个简单的[教务查询系统](https://github.com/JennyHJN/Examination_System)（主要技术SpringMVC + Spring + Mybatis + Shiro + Bootstrap），就是在跑的时候MySQL驱动版本的问题会导致一些错误，我在pom.xml里把5.4版本改成8.0版本的就可以了。还没仔细看Mybatis具体是怎么工作的，就先看了一下效果吧。

5、看了一点《Spring实战》中的装配Bean的内容。我才知道原来在Spring中一个bean就是一个类，类之间的协作依赖关系是用bean来组装依赖的。

装配Bean：创建应用对象之间协作关系的行为。

Spring容器负责闯将应用程序中的Bean并通过DI来协调这些对象之间的关系。Bean就是组件、对象。Spring中的Bean都是单例。

有一种装配是：组件扫描+自动装配

@Component：这个简单的注解表明该类会作为组件类，并告知Spring要为这个类 创建Bean。

@ComponentScan：自动创建无数个bean

**2019.6.25：第十四天**

1、AJAX 是与服务器交换数据的技术，它在不重载全部页面的情况下，实现了对部分网页的更新。

ajax：一种请求数据的方式，不需要刷新整个页面；

ajax的技术核心是 XMLHttpRequest 对象；

ajax请求过程：创建 XMLHttpRequest 对象、连接服务器、发送请求、接收响应数据；

2、@RequestMapping中consumes和produces的区别

HTTP协议Header中的两个东西

ContentType 和Accept

在Request中

ContentType 用来告诉服务器当前发送的数据是什么格式

Accept 用来告诉服务器，客户端能认识哪些格式,最好返回这些格式中的其中一种

consumes 用来限制ContentType

produces 用来限制Accept

3、终于将前后端分离了，后端controller返回的是JSON格式的数据，利用AJAX去抓取到数据，然后渲染到页面上。实现了获取所有的学生信息和搜索指定名字的学生信息。重要的在写AJAX请求的时候URL要写对，还有参数的传递要正确，这样才能获取到数据。在写js函数的时候犯了一个很傻的错误，就是从前端的input传值到js函数里的时候：var stuName = $('#stuName').val();这个语句我写错了，我写成：var stuName = $('stuName').val();忘记在input的id名之前写上”#”号了，一直获取不到input输入框的数据，导致我后面URL拼接的时候一直取不到input输入框的数据，导致AJAX根据URL地址和服务器进行数据请求时失败了。太傻了。

**2019.6.26：第十五天**

1、REST就是一种设计API的模式。最常用的数据格式是JSON。由于JSON能直接被JavaScript读取，所以，以JSON格式编写的REST风格的API具有简单、易读、易用的特点。

2、如何组织URL：**在实际工程中，一个Web应用既有REST，还有MVC，可能还需要集成其他第三方系统。**如何组织URL？一个简单的方法是通过固定的前缀区分。例如，/static/开头的URL是静态资源文件，类似的，/api/开头的URL就是REST API，其他URL是普通的MVC请求。

3、name是对象的名字，id是标识，用以唯一性的区分.就象人既要有名字，又需要身份证号。

可以这么简单理解它们的使用区别：id用于DHTML，即客户端脚本。而name则通常用在服务器端。

4、使用@Controller 注解，在对应的方法上，视图解析器可以解析return 的jsp,html页面，并且跳转到相应页面。若返回json等内容到页面，则需要加@ResponseBody注解。

@RestController注解，相当于@Controller+@ResponseBody两个注解的结合，返回json数据不需要在方法前面加@ResponseBody注解了，但使用@RestController这个注解，就不能返回jsp,html页面，视图解析器无法解析jsp,html页面。

5、在AJAX里：

beforeSend: function(request){

request.setRequestHeader("content-type", "application/json; charset=utf-8");

request.setRequestHeader("Accept", "application/json; charset=utf-8");

}

content-type这个参数写错了写成了content\_type，害的我在后台传参数的时候给我报了一个错是HTTP Status 415 – 不支持的媒体类型。真的是这个错误让我找了好几个小时啊！！！！！我天，真的。

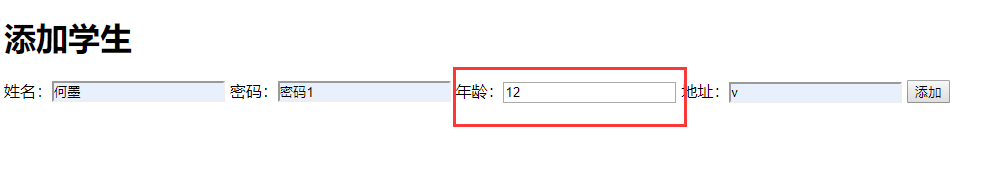
6、有一个很棒的一篇博客，讲的是AJAX和Controller传参的问题：[ajax 访问 Controller 传参—POJO](https://blog.csdn.net/zylwendao/article/details/79074665)

AJAX的data参数的值写法就两种：JSON格式和对象格式。

JSON格式，Controller中的参数要有@RequestBody注解

对象格式，Controller中的参数不需要有@RequestBody注解

7、然后又碰到一个错误，也是在添加学生的AJAX请求的时候，前端页面的表单提交的数据格式和服务器端接收的数据格式不一致导致的，我年龄不小心敲了字符串进去，点击提交按钮的时候，出现了HTTP 400的错误。年龄字段改成数字之后就可以了。我的input的type都是text格式的，后面我给限定了年龄这一个只能输入数字，在年龄对应的input里加入正则表达式就行。



年龄：<input type="text" name="age" id="age"

onKeyUp="this.value=this.value.replace(/\D/g,'')"

onafterpaste="this.value=this.value.replace(/\D/g,'')" >

**2019.6.27：第十六天**

1、浏览器缓存要定时清理，早上发现改了一个js文件的一个名字，但是怎么重启服务器，这个js文件都没有更新。但是 这个对于用户来说体验是很差的，说明开发人员没有考虑这个问题给用户导致了一些不便。

在浏览器中，浏览器会在js和图片等文件解析执行后直接存入内存缓存中，那么当刷新页面时只需直接从内存缓存中读取(from memory cache)；而css文件则会存入硬盘文件中，所以每次渲染页面都需要从硬盘读取缓存(from disk cache)。

缓存是根据url来处理的，只要url不一样就是新的资源。

参考网址：[【常见问题】web开发中浏览器缓存问题](https://blog.csdn.net/u012250875/article/details/78332683)

最好的解决方法是：所有jsp引用相同的taglib，并在公用的jsp中设置一个版本号，并在css或js路径中进行引用，每次发布新版本时候，根据需要来修改版本号。如果资源文件有更新，发布时修改下版本号即可，如果没有更新则不用修改版本号，这样极具可控性。推荐该方法。



2、整理一下自己这个月的学习工作内容，撰写一个月总结出来。

**第五周：2019.7.1-2019.7.5**

**2019.7.1：第十七天**

1、CDN加速原理：

CDN的全称是Content Delivery Network，即内容分发网络。其目的是通过在现有的Internet中增加一层新的CACHE(缓存)层，将网站的内容发布到最接近用户的网络”边缘“的节点，使用户可以就近取得所需的内容，提高用户访问网站的响应速度。从技术上全面解决由于网络带宽小、用户访问量大、网点分布不均等原因，提高用户访问网站的响应速度。

原因：由于互联网的飞速发展，越来越多的人开始使用并依赖网络，那在**膨胀的用户群**和**现行的网络负载能力**就形成了矛盾，人们开始希望网络更加的快速，而不是在无休止的等待中上网。

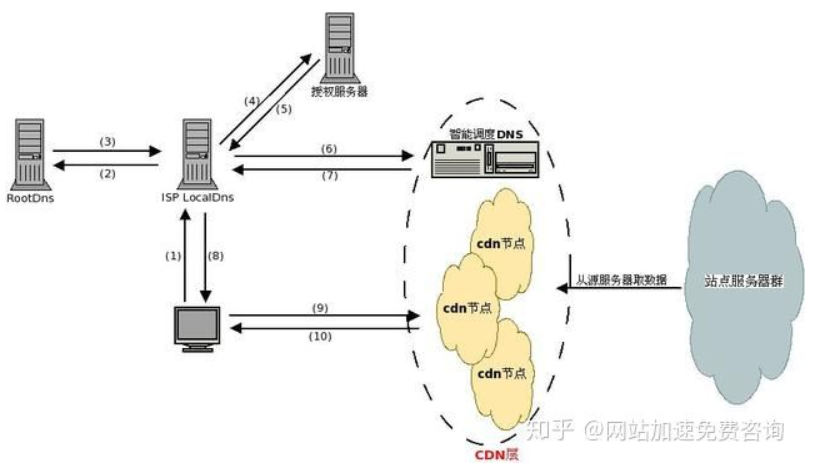
传统的网站访问过程为:

⊙用户在浏览器中输入要访问的域名;

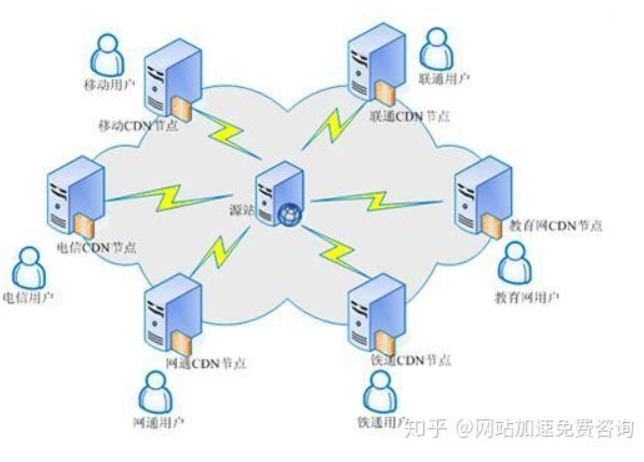
⊙浏览器向域名解析服务器发出解析请求，获得此域名对应的IP 地址;

⊙浏览器利用所得到的IP 地址，向该IP对应的服务器发出访问请求;

⊙服务器对此响应，将数据回传至用户浏览器端显示出来。



与传统访问方式不同，CDN 网络则是在用户和服务器之间增加 Cache 层，将用户的访问请求引导到Cache 节点而不是服务器源站点，要实现这一目的，主要是通过接管DNS 实现。



通过以上分析可以看到，不论是否使用CDN 网络，普通用户客户端设置不需做任何改变，直接使用被加速网站原有域名访问即可。对于要加速的网站，只需修改整个访问过程中的**域名解析**部分，便能实现透明的网络加速服务。

这个CDN（内容分发网络）加速，我的理解是，原本是每个用户都直接向站点服务器提交请求，有了CDN之后，变成是每个用户透明地向离他最近的Cache服务器提交请求，Cache服务器向源站点服务器提交请求并获得数据，接着缓存起来这些数据并给返回数据给用户。这样做的好处是大大缓解了站点服务器的压力，把站点服务器的压力转移到各个Cache服务器上。

2、JDBC（Java DataBase Connectivity,java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用Java语言编写的类和接口组成。【参考网址：[JDBC详细讲解](https://wenku.baidu.com/view/9be1b8d1d05abe23482fb4daa58da0116c171f3b.html?re=view)】

JDBC可做三件事：

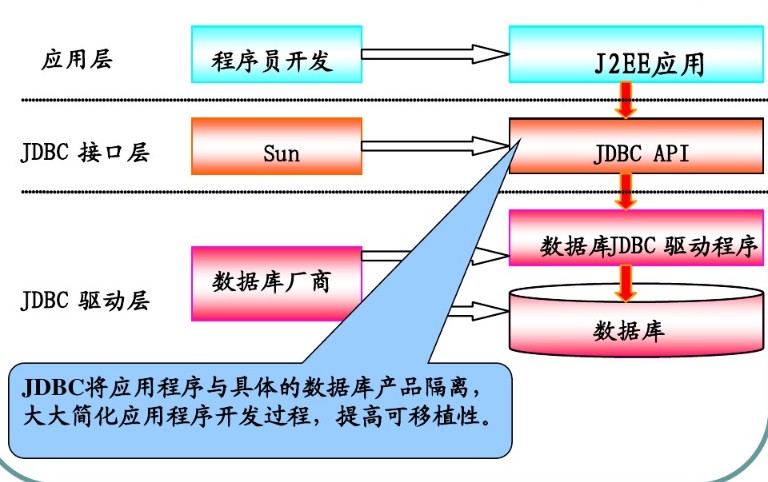
与数据库建立连接（Connection）

发送操作数据库的语句（Statement）

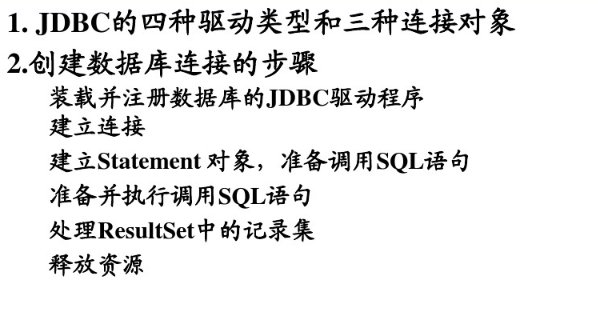
处理结果（ResultSet）

DriverManager 类是 JDBC 的管理层，作用于用户和驱动程序之间。它跟踪可用的驱动程序，并在数据库和相应驱动程序之间建立连接。

尽管JDBC在JAVA语言层面实现了统一，但不同数据库仍旧有许多差异。为了更好地实现跨数据库操作，于是诞生了Hibernate项目，Hibernate是对JDBC的再封装，实现了对数据库操作更宽泛的统一和更好的可移植性。







3、Hibernate与MyBatis的区别：

参考网址：[Java框架总结之MyBatis和Hibernate区别](https://www.jianshu.com/p/7affb2b766ff?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

Hibernate是全自动，而MyBatis是半自动：Hibernate完全可以通过对象关系模型实现对数据库的操作，拥有完整的JavaBean对象与数据库的映射结构来自动生成sql。而MyBatis仅有基本的字段映射，对象数据以及对象实际关系依然需要通过手写sql来实现和管理。

4、POJO、JavaBean、EJB的区别

参考网址：[POJO、JavaBean、EJB的区别](https://blog.csdn.net/nocol123/article/details/76039714)

**POJO**

全称是 Plain Old Java Object,翻译为“纯洁老式的java对象”，一般都使用“简单java对象”来称呼。为什么这么说呢？原因是**POJO类没有从任何类继承，也没有实现任何接口**，更没有被其他框架侵入的java对象。**POJO类只有一些private的参数作为对象的属性，然后针对每一个参数定义get和set方法访问的接口。**

**JavaBean**

JavaBean符合一定规范编写的Java类，不是一种技术，而是一种规范。

JavaBean看作遵从特定命名约定的POJO。

**POJO和JavaBean两者有什么区别？**

POJO其实是比javabean更纯净的简单类或接口。POJO严格地遵守简单对象的概念，而一些JavaBean中往往会封装一些简单逻辑。**POJO的格式是用于数据的临时传递，它只能装载数据， 作为数据存储的载体，而不具有业务逻辑处理的能力。而javabean虽然数据的获取与POJO一样，但是javabean当中可以有其它的方法。**

**EJB**

EJB(Enterprise JavaBean)是一种javabean的组合规范。我们可以认为它是一组"功能"JavaBean的集合。

5、谈谈Spring中的IOC和AOP概念

参考网址：[谈谈Spring中的IOC和AOP概念](https://blog.csdn.net/eson_15/article/details/51090040)

IOC：Inverse of Control，控制反转（依赖倒置）

A需要B，倒置就是A要调用B的话，A并不需要主动获取B，而是由其它人自动将B送上门来。不需要A主动获取B的实例：B b = new B()

Spring就是小卖部，你就是A对象，水就是B对象

第一：在Spring中声明一个类：A

第二：告诉Spring，A需要B

AOP：面向切面编程

面向切面编程的目标就是分离关注点。

从Spring的角度看，AOP最大的用途就在于提供了事务管理的能力。事务管理就是一个关注点，你的正事就是去访问数据库，而你不想管事务（太烦），所以，Spring在你访问数据库之前，自动帮你开启事务，当你访问数据库结束之后，自动帮你提交/回滚事务！

**2019.7.2：第十八天**

1、Spring给容器中注册组件：

（1）包扫描+组件标注注解（@Controller/@Service/@Repository/@Component）[自己写的类]

（2）@Bean[导入的第三方包里面的组件]

（3）@Import[快速给容器中导入一个组件]

1）@Import（要导入到容器中的组件），容器中就会自动注册这个组件，id默认是全类名

2）ImportSelector：返回需要导入的组件的全类名的数组【springboot用的多】

3）ImportBeanDefinitionRegistrar：手动注册bean到 容器中

（4）使用Spring提供的FactoryBean（工厂bean）

1）默认获取到的是工厂bean调用getObject创建的对象

2）要获取工厂bean本身，我们需要给id前面加一个&

&colorFactoryBean

2、Bean的生命周期

Bean的生命周期：Bean创建—初始化—销毁的过程

构造（对象创建）：

**单实例：在容器启动的时候创建对象**

**多实例：在每次获取的时候创建对象**

初始化：

对象创建完成，并赋值好，调用初始化方法

销毁：

单实例：容器关闭的时候

多实例：容器不会管理这个bean；容器不会调用销毁方法

**2019.7.3：第十九天**

1、参考网址：[Spring的Java配置方式—@Configuration和@Bean实现Java配置](https://www.cnblogs.com/linjiqin/p/9655649.html)

@Configuration和@Bean

Spring的Java配置方式是通过 @Configuration 和 @Bean 注解实现的：

a、@Configuration 作用于类上，相当于一个xml配置文件

b、@Bean 作用于方法上，相当于xml配置中的<bean>

2、注解及其含义【参考网址：[@Component, @Repository, @Service的区别](https://blog.csdn.net/fansili/article/details/78740877)】：



@Component, @Service, @Controller, @Repository是spring注解，**注解后可以被spring框架所扫描并注入到spring容器来进行管理**

@Component是通用注解，其他三个注解是这个注解的拓展，并且具有了特定的功能

@Repository注解在持久层中，具有将数据库操作抛出的原生异常翻译转化为spring的持久层异常的功能。

@Controller层是spring-mvc的注解，具有将请求进行转发，重定向的功能。

@Service层是业务逻辑层注解，这个注解只是标注该类处于业务逻辑层。

用这些注解对应用进行分层之后，就能将请求处理，义务逻辑处理，数据库操作处理分离出来，为代码解耦，也方便了以后项目的维护和开发。

3、@Autowired和@Resource的区别【参考网址：[@Autowired和@Resource的区别](https://www.jianshu.com/p/872440c92e8a)】

相同点：

@Resource的作用相当于@Autowired，均可标注在字段或者属性的setter方法上。

不同点：

主要区别就是@Autowired是默认按照类型装配的 @Resource默认是按照名称装配的

4、参考网址：[SpringMVC 使用AOP添加日志](https://blog.csdn.net/liangwenmail/article/details/83540818)

想用AOP给我的小demo加个日志打印，但是出现错误了。还没有解决

5、下午的分享会中知道了Arthas（阿尔萨斯）这个工具，这个工具是阿里巴巴开源的诊断工具，可以帮我们解决一些问题，比如某个类是从哪个jar包加载的？为什么会报各种类相关的Exception，改的代码为什么没有执行到？遇到的问题没办法在线上debug等等，还有其他的一些问题，这个可以好好学习一下，是个比较有用的工具。

接下来的计划：

熟悉配置管理—我的配置中客户定制API模板配置、客户默认API模板配置和粒度模板配置项管理



**第五周：2019.7.8-2019.7.11**

**2019.7.8：第二十天**

1、参考网址：[Spring AOP 的实现方式（以日志管理为例）](https://blog.csdn.net/donggua3694857/article/details/52752503)

失败了，demo也跑不起来了。

2、今天主要是对比了旧模板和新的模板数据库的异同，找出不一样的地方，相同的地方就是直接选择了。

**2019.7.9：第二十一天**

1、参考网址：[SpringAOP拦截Controller,Service实现日志管理(自定义注解的方式)](https://blog.csdn.net/czmchen/article/details/42392985)

2、又是比对的一天

**2019.7.10：第二十二天**

1、XML=可扩展标记语言(eXtensible Markup Language).

可扩展标记语言XML是一种简单的数据存储语言，使用一系列简单的标记描述数据，而这些标记可用

方便的方式建立，虽然XML占用的空间比二进制数据要占用更多的空间，但XML极其简单易于掌握和使用

XML的简单使其易于在任何应用程序中读写数据，这使XML很快成为数据交换的唯一公共语言，XML不是一个依附于特定浏览器的语言

2、XML 被设计用来传输和存储数据。

HTML 被设计用来显示数据。

**2019.7.11：第二十三天**