

今日内容

1. JSP:
 1. 指令
 2. 注释
 3. 内置对象
2. MVC开发模式
3. EL表达式
4. JSTL标签
5. 三层架构

JSP:

1. 指令
 - * 作用：用于配置JSP页面，导入资源文件
 - * 格式：
`<%@ 指令名称 属性名1=属性值1 属性名2=属性值2 ... %>`
 - * 分类：
 1. `page` : 配置JSP页面的
 - * `contentType`: 等同于`response.setContentType()`
 1. 设置响应体的mime类型以及字符集
 2. 设置当前jsp页面的编码（只能是高级的IDE才能生效，如果使用低级工具，则需要设置`pageEncoding`属性设置当前页面的字符集）
 - * `import`: 导包
 - * `errorPage`: 当前页面发生异常后，会自动跳转到指定的错误页面
 - * `isErrorPage`: 标识当前也是是否是错误页面。
 - * `true`: 是，可以使用内置对象`exception`
 - * `false`: 否。默认值。不可以使用内置对象`exception`
 2. `include` : 页面包含的。导入页面的资源文件
 - * `<%@include file="top.jsp"%>`
 3. `taglib` : 导入资源
 - * `<%@ taglib prefix="c"`
`uri="http://java.sun.com/jsp/jstl/core" %>`
 - * `prefix`: 前缀，自定义的
 - 2. 注释：
 1. html注释:
`<!-- -->`: 只能注释html代码片段
 2. jsp注释: 推荐使用

<%-- --%>: 可以注释所有

3. 内置对象

- * 在jsp页面中不需要创建, 直接使用的对象
- * 一共有9个:

作用	变量名	真实类型	
页面共享数据, 还可以获取其他八个内置对象	* <code>pageContext</code>	<code>PageContext</code>	当前
请求访问的多个资源(转发)	* <code>request</code>	<code>HttpServletRequest</code>	一次
会话的多个请求间	* <code>session</code>	<code>HttpSession</code>	一次
用户间共享数据	* <code>application</code>	<code>ServletContext</code>	所有
对象	* <code>response</code>	<code>HttpServletResponse</code>	响应
页面(Servlet)的对象 <code>this</code>	* <code>page</code>	<code>Object</code>	当前
对象, 数据输出到页面上	* <code>out</code>	<code>JspWriter</code>	输出
Servlet的配置对象	* <code>config</code>	<code>ServletConfig</code>	
对象	* <code>exception</code>	<code>Throwable</code>	异常

MVC: 开发模式

1. jsp演变历史

1. 早期只有servlet, 只能使用response输出标签数据, 非常麻烦
2. 后来又jsp, 简化了Servlet的开发, 如果过度使用jsp, 在jsp中即写大量的java代码, 有写html表, 造成难于维护, 难于分工协作
3. 再后来, java的web开发, 借鉴mvc开发模式, 使得程序的设计更加合理性

2. MVC:

1. M: Model, 模型。JavaBean
 - * 完成具体的业务操作, 如: 查询数据库, 封装对象
2. V: View, 视图。JSP
 - * 展示数据
3. C: Controller, 控制器。Servlet
 - * 获取用户的输入
 - * 调用模型

* 将数据交给视图进行展示

* 优缺点:

1. 优点:

1. 耦合性低, 方便维护, 可以利于分工协作
2. 重用性高

2. 缺点:

1. 使得项目架构变得复杂, 对开发人员要求高

EL表达式

1. 概念: Expression Language 表达式语言

2. 作用: 替换和简化jsp页面中java代码的编写

3. 语法: `${表达式}`

4. 注意:

* jsp默认支持el表达式的。如果要忽略el表达式

1. 设置jsp中page指令中: `isELIgnored="true"` 忽略当前jsp页面中所有的el表达式

2. `\${表达式}` : 忽略当前这个el表达式

5. 使用:

1. 运算:

* 运算符:

1. 算数运算符: `+` `-` `*` `/(div)` `%(mod)`
2. 比较运算符: `>` `<` `>=` `<=` `==` `!=`
3. 逻辑运算符: `&&(and)` `||(or)` `!(not)`
4. 空运算符: `empty`

* 功能: 用于判断字符串、集合、数组对象是否为null或者长度是否为0

* `${empty list}`: 判断字符串、集合、数组对象是否为null或者长度为0

* `${not empty str}`: 表示判断字符串、集合、数组对象是否不为null 并且 长度>0

2. 获取值

1. el表达式只能从域对象中获取值

2. 语法:

1. `${域名称.键名}`: 从指定域中获取指定键的值

* 域名称:

1. `pageScope` --> `pageContext`
2. `requestScope` --> `request`

```

3. sessionScope    --> session
4. applicationScope -->
application (ServletContext)
    * 举例：在request域中存储了name=张三
    * 获取：${requestScope.name}

```

2. `${键名}`：表示依次从最小的域中查找是否有该键对应的值，直到找到为止。

3. 获取对象、List集合、Map集合的值

1. 对象：`${域名称.键名.属性名}`
 - * 本质上会去调用对象的getter方法
2. List集合：`${域名称.键名[索引]}`
3. Map集合：
 - * `${域名称.键名.key名称}`
 - * `${域名称.键名["key名称"]}`

3. 隐式对象：

- * `el`表达式中有11个隐式对象
- * `pageContext`:
 - * 获取jsp其他八个内置对象
 - * `${pageContext.request.contextPath}`：动态获取虚拟目录

JSTL

1. 概念：JavaServer Pages Tag Library JSP标准标签库

- * 是由Apache组织提供的开源的免费的jsp标签 <标签>

2. 作用：用于简化和替换jsp页面上的java代码

3. 使用步骤：

1. 导入jstl相关jar包
2. 引入标签库：taglib指令： `<%@ taglib %>`
3. 使用标签

4. 常用的JSTL标签

1. `if`：相当于java代码的if语句
 1. 属性：

- * **test** 必须属性，接受boolean表达式
 - * 如果表达式为true，则显示if标签体内容，如果为false，则不显示标签体内容
 - * 一般情况下，**test**属性值会结合**el**表达式一起使用
- 2. 注意：
 - * **c:if**标签没有**else**情况，想要**else**情况，则可以在定义一个**c:if**标签

2. choose:相当于java代码的switch语句

- | | |
|-----------------------------------|----------------------|
| 1. 使用 choose 标签声明 | 相当于 switch 声明 |
| 2. 使用 when 标签做判断 | 相当于 case |
| 3. 使用 otherwise 标签做其他情况的声明 | 相当于 default |

3. foreach:相当于java代码的for语句

5. 练习:

- * 需求：在request域中有一个存有User对象的List集合。需要使用jstl+el将list集合数据展示到jsp页面的表格table中

三层架构：软件设计架构

1. 界面层(表示层)：用户看得界面。用户可以通过界面上的组件和服务进行交互
2. 业务逻辑层：处理业务逻辑的。
3. 数据访问层：操作数据存储文件。

案例：用户信息列表展示

1. 需求：用户信息的增删改查操作
2. 设计：
 1. 技术选型：Servlet+JSP+MySQL+JDBCTemplate+Duid+BeanUtilS+tomcat
 2. 数据库设计：

```
create database day17; -- 创建数据库
use day17;             -- 使用数据库
create table user(      -- 创建表
  id int primary key auto_increment,
  name varchar(20) not null,
  gender varchar(5),
  age int,
  address varchar(32),
  qq varchar(20),
  email varchar(50)
);
```

3. 开发:

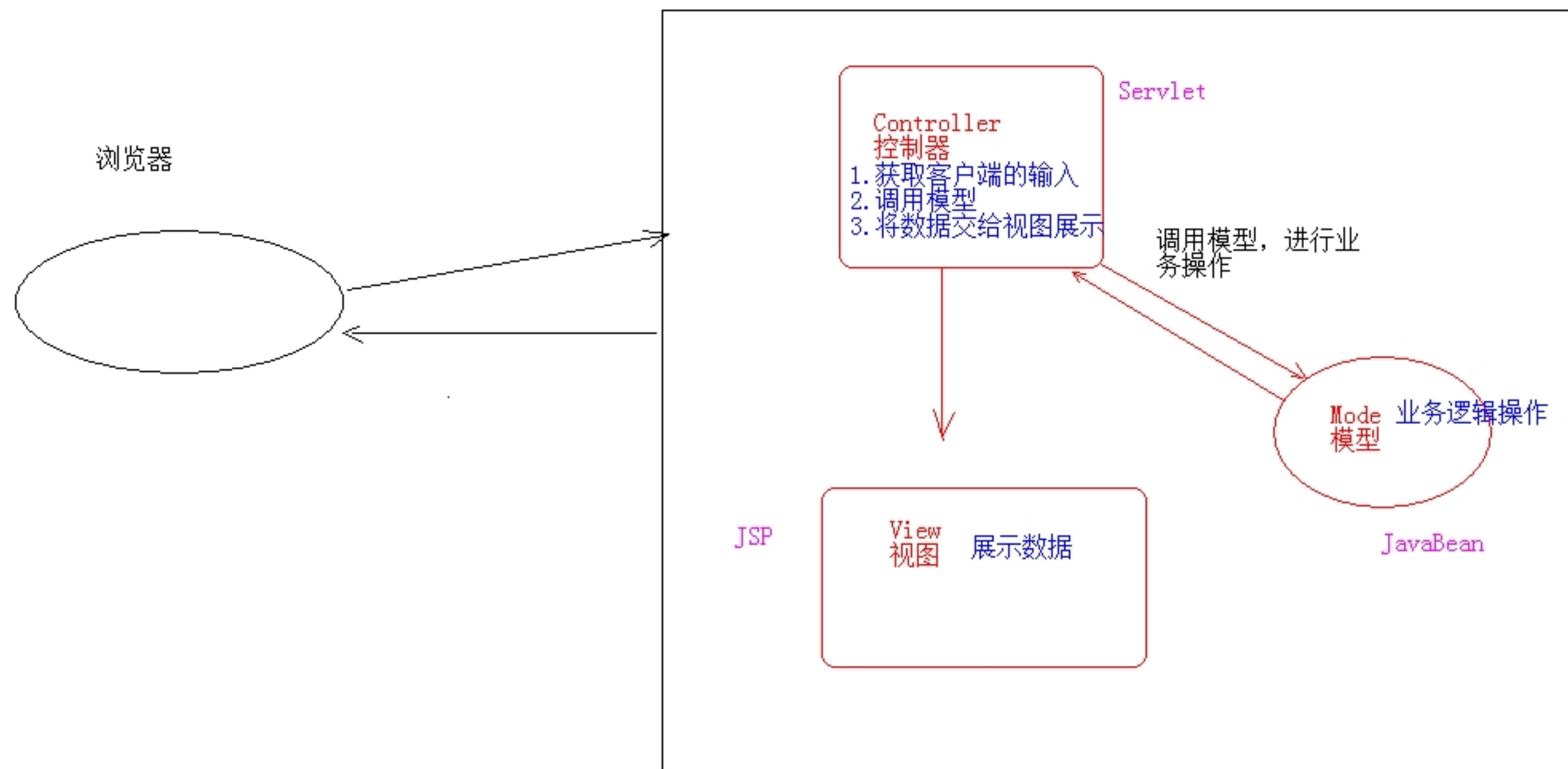
1. 环境搭建

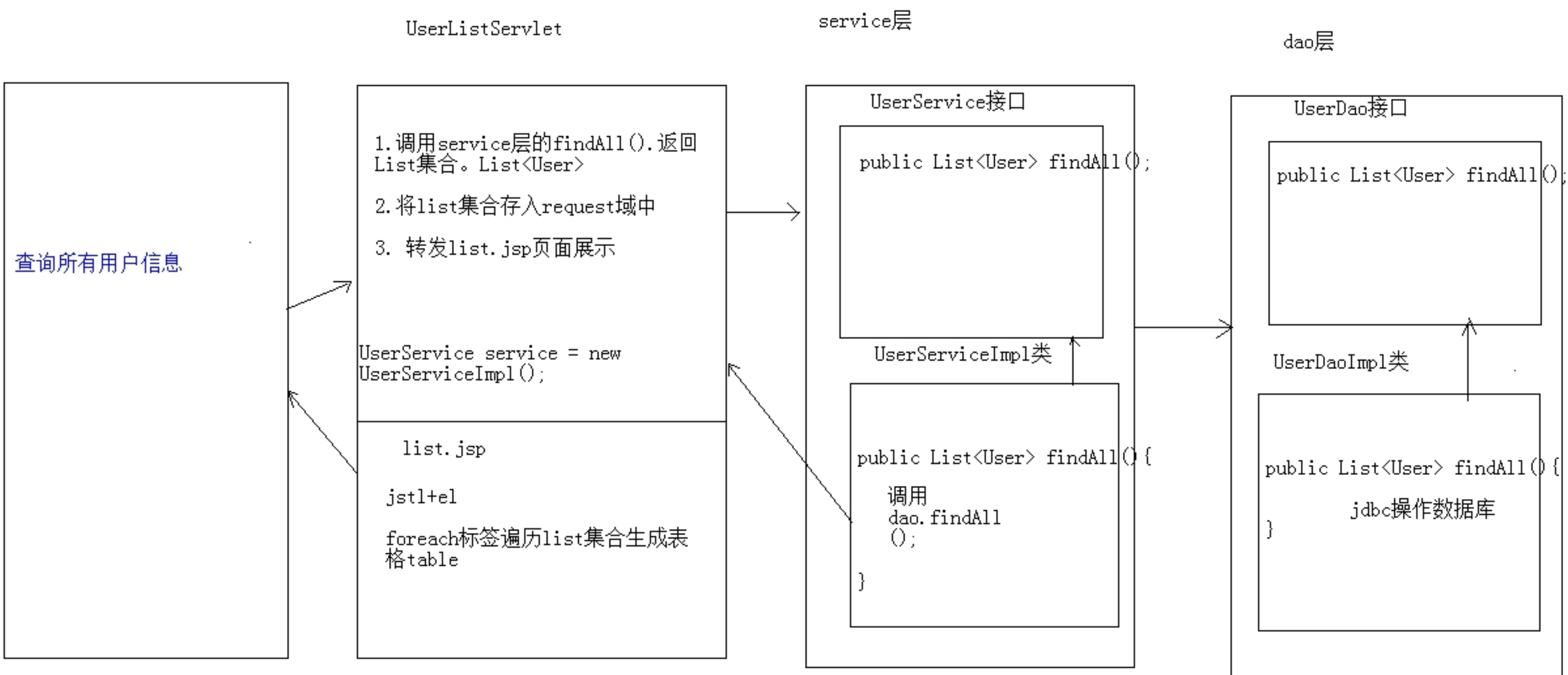
1. 创建数据库环境
2. 创建项目，导入需要的jar包

2. 编码

4. 测试

5. 部署运维





web服务器

界面层(表示层/web层)
SpringMVC 框架

控制器: Servlet

视图: JSP

接受用户参数, 封装数据,
调用业务逻辑层完成处理。
转发jsp页面完成显示

cn.itcast.项目名.web

业务逻辑层(service层)
Spring框架

A功能

find() {}
insert() {}

B功能

find() {}
delete() {}

组合DAO层中的简单方法, 形
成复杂的功能(业务逻辑操作)

cn.itcast.项目名.service

数据访问层(dao层)
MyBatis框架

find() {}
insert() {}
update() {}
delete() {}

定义了对于数据库最基本的
CRUD操

dao: Data Access Object
Catalog Access Object

cn.itcast.项目名.dao

浏览器

DB

