实体类（Entity，Pojo）：

所有的attribute都是private

有一个无参构造（没有参数的constructor） //不代表禁止有参构造

因为所有attribute都是private,所以必须有对应的get/set方法

尽量不在里面写逻辑

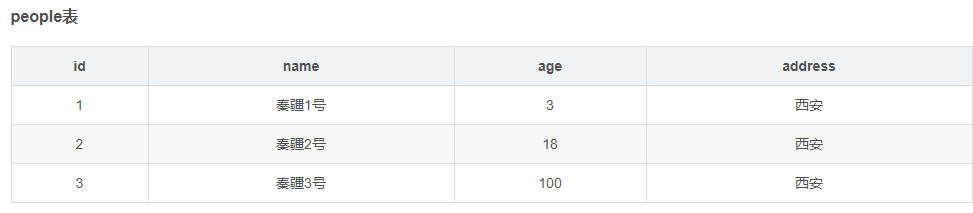
JavaBean有特定的写法： //JAVA Bean就是JAVA实体类

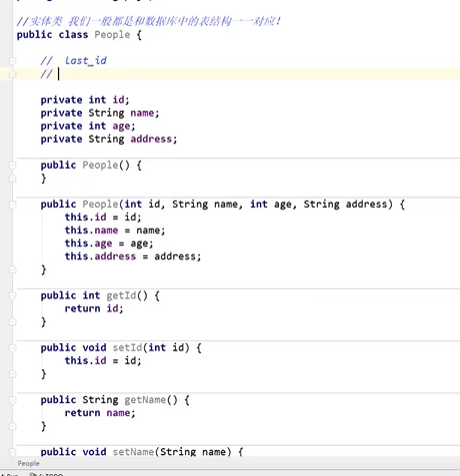
* 必须要有一个无参构造
* 属性必须私有化
* 必须有对应的get/set方法；

一般用来和数据库的字段做映射 ORM； //换句话说为数据库某张表构建一个class，这个class没有特别复杂的操作，只是单纯的getSet

ORM ：对象关系映射

* 表—>类
* 字段–>属性
* 行记录---->对象



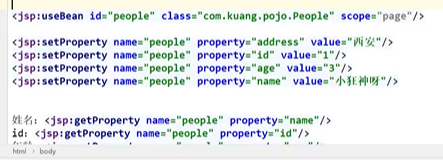


所有的都是private的，

JSP里使用useBean就可以利用无参构造创建对应object

set

get

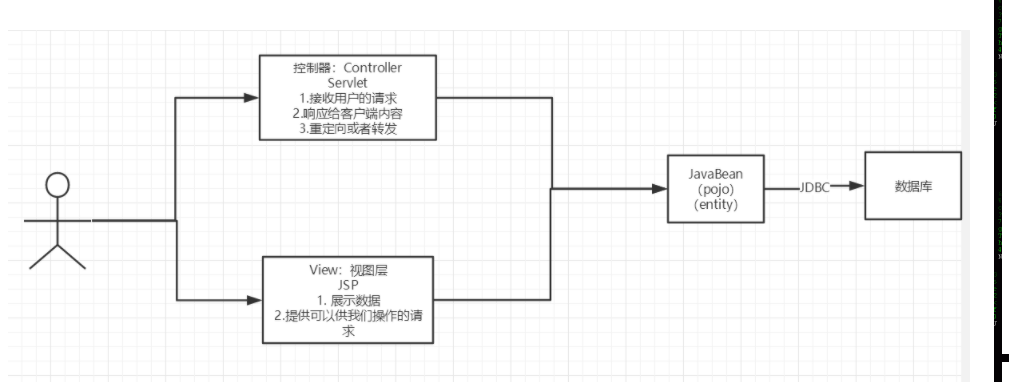


MVC三层架构

Model：模型 //实体类和数据库中对应的字段

View：视图 //JSP页面

Controller: 控制器 //Servlet



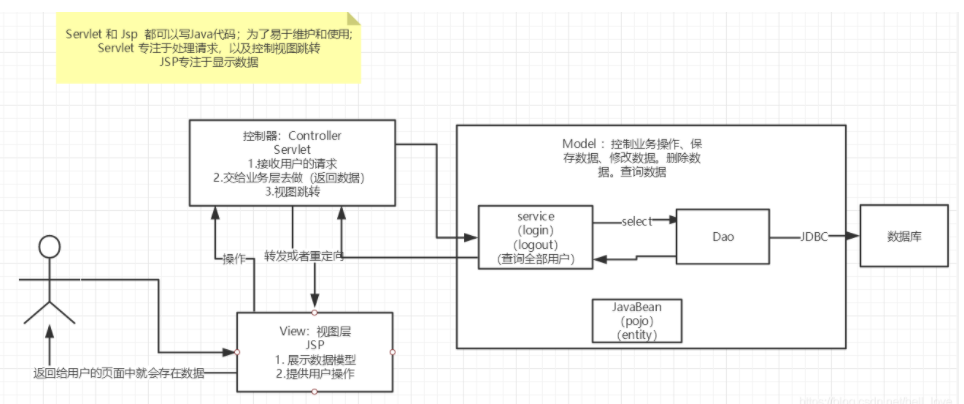
view例如输入密码啥的

JSP也是可以写JAVA代码的，但是为了维护和使用，我们进行分层，Servlet专注于处理请求以及控制VIEW跳转，JSP专注于显示数据

这是早些年的架构，用户直接访问控制层，然后直接通过JDBC操作数据库

弊端：程序把JDBC代码也写在Servlet里，不利于维护，Servlet不仅处理请求 ，响应，视图跳转，还要处理JDBC

MVC三层架构



原来的2.响应客户端内容转给业务层去做

3.重定向保持//视图跳转

现在用户只用看视图，点按钮就行

Service业务层，登陆，登出，注销啥的

Dao层：写JDBC啥的,需要分成一个interface接口，用来供Service使用，还需要一个实际的Dao JAVA CLASS implement这个接口

实体类JAVABEAN， service与Dao都会用到

Model

* 业务处理 ：业务逻辑（Service）
* 数据持久层：CRUD （Dao - 数据持久化对象）

View

* 展示数据
* 提供链接发起Servlet请求 （a，form，img…）

Controller （Servlet）

* 接收用户的请求 ：（req：请求参数、Session信息….）
* 交给业务层处理对应的代码 //不写任何的业务代码
* 控制视图的跳转
* 登录--->接收用户的登录请求--->处理用户的请求（获取用户登录的参数，username，password）---->交给业务层处理登录业务（判断用户名密码是否正确：事务）--->Dao层查询用户名和密码是否正确-->数据库

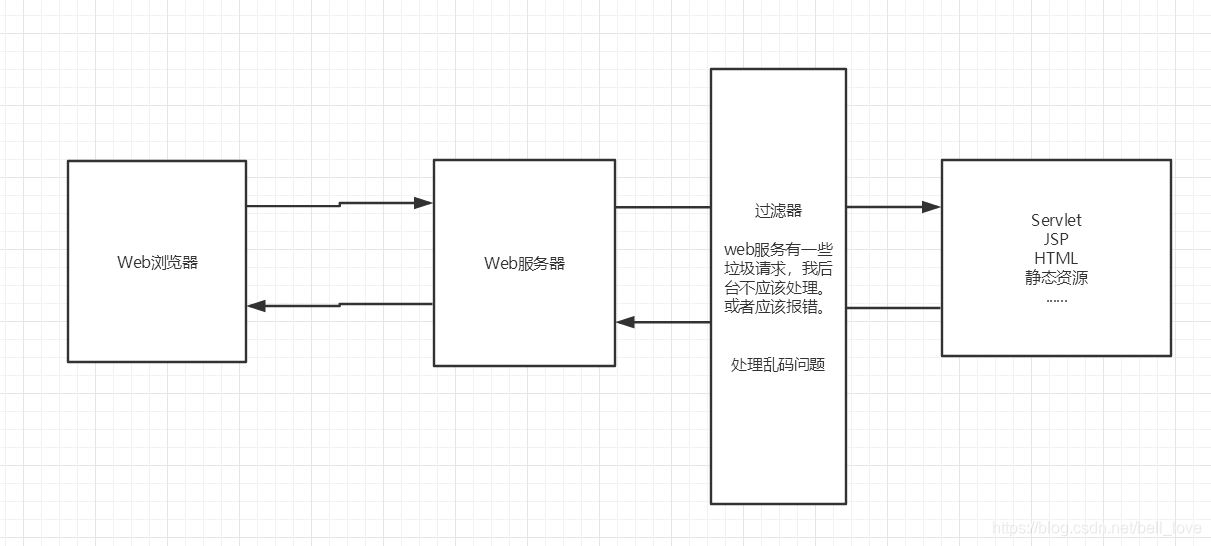
11.Filter

比如 Shiro安全框架技术就是用Filter来实现的

Filter：过滤器 ，用来过滤网站的数据；

* 处理中文乱码
* 登录验证….

（比如用来过滤网上骂人的话，我\*\*\*我自己 0-0）

  
Filter开发步骤：

1. 导包
2. 编写过滤器
   1. 导包不要错 **（注意）**



实现Filter接口，重写对应的方法即可

```java

public class CharacterEncodingFilter implements Filter {

//初始化：web服务器启动，就以及初始化了，随时等待过滤对象出现！

public void init(FilterConfig filterConfig) throws ServletException {

System.out.println("CharacterEncodingFilter初始化");

}

//Chain : 链

/\*

1. 过滤中的所有代码，在过滤特定请求的时候都会执行

2. 必须要让过滤器继续同行

chain.doFilter(request,response);

\*/

public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {

request.setCharacterEncoding("utf-8");

response.setCharacterEncoding("utf-8");

response.setContentType("text/html;charset=UTF-8");

System.out.println("CharacterEncodingFilter执行前....");

chain.doFilter(request,response); //让我们的请求继续走，如果不写，程序到这里就被拦截停止！

System.out.println("CharacterEncodingFilter执行后....");

}

//销毁：web服务器关闭的时候，过滤器会销毁

public void destroy() {

System.out.println("CharacterEncodingFilter销毁");

}

}

1. 在web.xml中配置 Filter

<filter>

<filter-name>CharacterEncodingFilter</filter-name>

<filter-class>com.kuang.filter.CharacterEncodingFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>CharacterEncodingFilter</filter-name>

<!--只要是 /servlet的任何请求，会经过这个过滤器-->

<url-pattern>/servlet/\*</url-pattern>

<!--<url-pattern>/\*</url-pattern>-->

<!-- 别偷懒写个 /\* -->

</filter-mapping>