1 Web

1.1 JavaBean 规范

即如何定义一个 `JavaBean`

- 必须是一个公共类，`public class User`
- 必须有一个空的构造器
- 类的成员变量是私有，如 `private int id`
- 通过 `getter/setter` 访问属性

1.2 分层式结构

1. 数据访问层(DAL)： 面向数据库
2. 业务逻辑层(BLL)： 面向业务逻辑，对数据层的操作，对业务逻辑处理
3. 表示层(UI)： 面向用户

对应关系

- DAL $\Longrightarrow$ Dao
- BLL $\Longrightarrow$ Service
- UI $\Longrightarrow$ HTML、JSP

2 Servlet 的 utl-pattern 匹配规则

2.1 精确匹配

- 只能匹配 `/a/b/MyServlet1`

```
<servlet-mapping>
<servlet-name>MyServlet1</ servlet-name>
<url-pattern>/a/b/MyServlet1</url-pattern>
</servlet-mapping>
```

2.2 路径匹配

- 只要前面的路径匹配即可
- 可匹配 `a/b/hello. do`、`a/b/c`、`a/b`

```
<servlet-mapping>
<servlet-name>MyServlet1</ servlet-name>
<url-pattern>/a/b/*</url-pattern>
</servlet-mapping>
```

2.3 扩展名匹配

- 可匹配 `hello. do`、`a/hello. do`、`a/b/xyz. do`

```
<servlet-mapping>
<servlet-name>MyServlet1</ servlet-name>
<url-pattern>*. do</url-pattern>
</servlet-mapping>
```

2.4 缺省匹配

- 若 `url-pattern` 配成如下任意一 种，则所有 `url` 都可以匹配

```
<url-pattern>/</url-pattern>
<url-pattern>/*</url-pattern>
```

3 配置前端控制器

```
<servlet>
    <servlet-name>dispatcherServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name> <!--固定写法-->
        <param-value>classpath:SpringMVC.xml</param-value><!--配置文件-->
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>dispatcherServlet</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

4 Spring MVC 工作原理

1. 用户发送请求 ⟹ `DispatcherServlet`(前端控制器)
   前端控制器作为统一访问点，进行全局流程控制：收到请求后委托给其他解析器进行处理
2. `DispatcherServlet` ⟹ `HandlerMapping`(处理器映射器)
   处理器映射器把请求映射为 `HandlerExecutionChain` 对象，方便添加各种新的映射策略
   `HandlerExecutionChain` 包含一个 `Handler` 处理器、多个 `HandlerInterceptor` 拦截器

   获取 `Handler` 并将其返回给 `DispatcherServlet`
3. `DispatcherServlet` ⟹ `HandlerAdapter`(处理器适配器)
   处理器适配器使用适配器设计模式，把处理器包装为适配器，支持多种类型的处理器

   根据规则执行不同的 Handler
4. `HandlerAdapter` ⟹ 处理器
   `HandlerAdapter` 根据适配的结果，调用处理器，完成处理功能，返回一个 `ModelAndView` 对象
5. `DispatcherServlet` ⟹ `ViewResolver`(视图解析器)
   视图解析器把逻辑视图名解析为具体的View，通过这种策略可以很方便的更换其他图技术
6. `DispatcherServlet` ⟹ `View`(视图)
   视图把传进来的Model模型数据进行渲染，该Model是一个Map数据结构，因此很容易支持其他视图技术
7. 由 `DispatcherServlet` 返回响应给用户，到此一个流程结束

可以不声明映射器、适配器、解析器，Spring 自动分配

声明则需要将以上全部声明