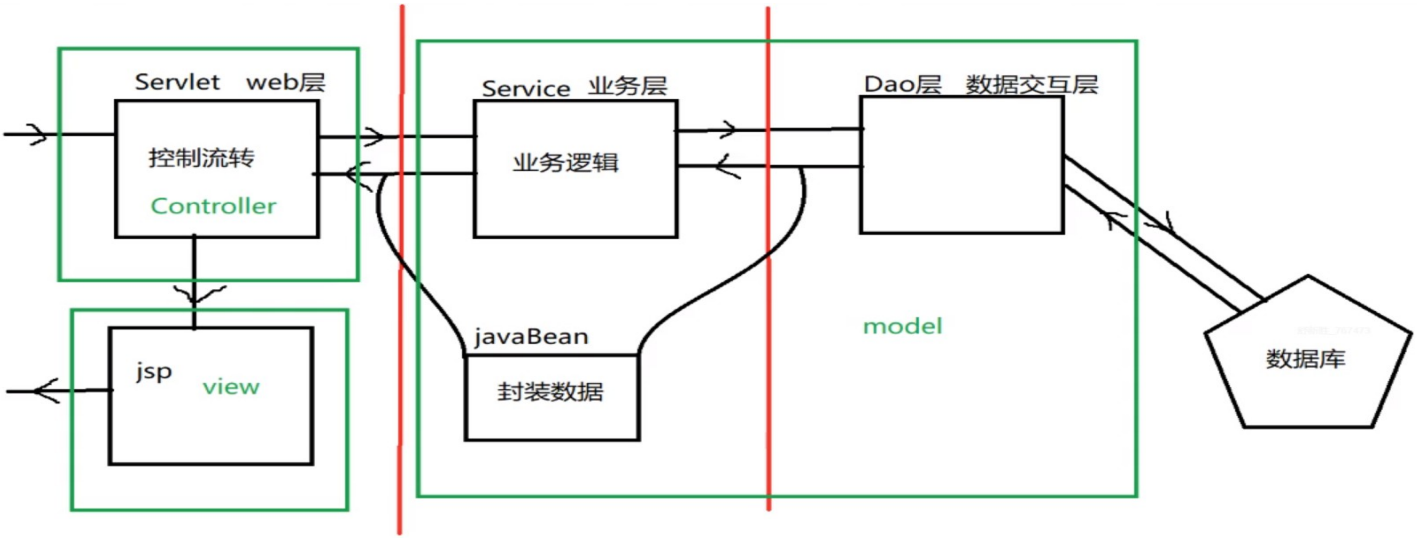
MVC

一 MVC 设计思想

1 Model-View-Controller

模型-视图-控制器，软件编程的通用设计思想，MVC 思想认为任何软件都可以分为：负责封装数据和处理数据的模型、负责展示数据的视图、负责程序控制(程序流转)的控制器。MVC 思想要求这三者尽量独立，互不干扰，每个模块只做自己该做的事，任何一个模块的改变不影响其他模块
优点是使软件结构更加清晰，便于开发维护、分工合作

2 JAVA EE 的经典三层架构



耦合：代码、模块之间彼此依赖，强度高，修改某个模块时，别的模块也要跟着修改
解耦：降低耦合性，多个模块尽量独立，修改任何模块时，别的模块尽量不用任何改动

3 重构项目

分包：
Web
Service
Bean
Utils

二 Filter

1 概念

Filter 称之为过滤器，web 开发时，可以通过 filter 实现对访问的控制
对请求拦截，做一些操作再放行或不放行

2 特点

2.A 可以拦截对资源的访问

2.B 一个过滤器可以拦截多个资源，一个资源可以被多个过滤器拦截

2.C 可以根据访问的 url 地址判断是否拦截

2.D 所谓的拦截就是拦下来代表请求的 request 和代表响应的 response

2.E 拦截后可以控制是否放行，或者在放行之前做一些额外操作

三 开发过滤器

1 写一个类实现 Filter 接口

```
public class MyFilter  implements Filter {
    @Override
    public void init(FilterConfig filterConfig) throws ServletException {

    }

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain filterChain) throws IOException, ServletException {

    }

    @Override
    public void destroy() {

    }
}
```

1.A init()

初始化方法，当 Filter 对象创建时调用此方法

1.B doFilter()

拦截器的核心方法，拦截后都会执行该方法，若不做任何操作，默认不会放行

1.B.1 放行

可通过 传递的参数 FilterChain 放行

```
filterChain.doFilter(servletRequest,servletResponse);
```

1.B.2 拦截规则

匹配某个路径 /path
通配符多有资源：/*

1.B.3 执行流程

1.B.3.1 单个拦截器

访问某个资源->拦截器匹配拦截->放行->资源执行完成->返回拦截器继续执行

1.B.3.2 多个拦截器

访问某个资源->拦截器 1 匹配拦截->放行->拦截器 2 继续拦截->放行->资源执行完成->返回拦截器 2 继续执行->返回拦截器 1 继续执行
注意：当多个拦截器匹配相同资源时，根据 web.xml 中的 mapping 声明的先后顺序来判断哪个先执行，先声明先拦截

1.C destory()

2 在 web.xml 中配置过滤器

```
<!--配置过滤器-->
    <filter>
        <!-- 声明filter 和类所在位置-->
        <filter-name>myFilter</filter-name>
        <filter-class>cn.shu.blog.filter.MyFilter</filter-class>
    </filter>
    <filter-mapping>
        <!--用哪个filter拦-->
        <filter-name>myFilter</filter-name>
        <!--拦截哪个url-->
        <url-pattern>/fs01</url-pattern>
    </filter-mapping>
</web-app>
```

四　Filter 相关对象

1　init()中的 FilterConfig 参数

可以用来获取 ServletContext

2　doFilter()中的 FilterChain

代表过滤器链，提供了 doFilter 方法，用来放行当过滤器

五　生命周期

Web 应用启动时，filter 对象随之创建，创建后执行 init 方法，一旦创建一直存活，知道 web 应用被销毁时 filter 随之销毁，销毁之前调用 destory 方法