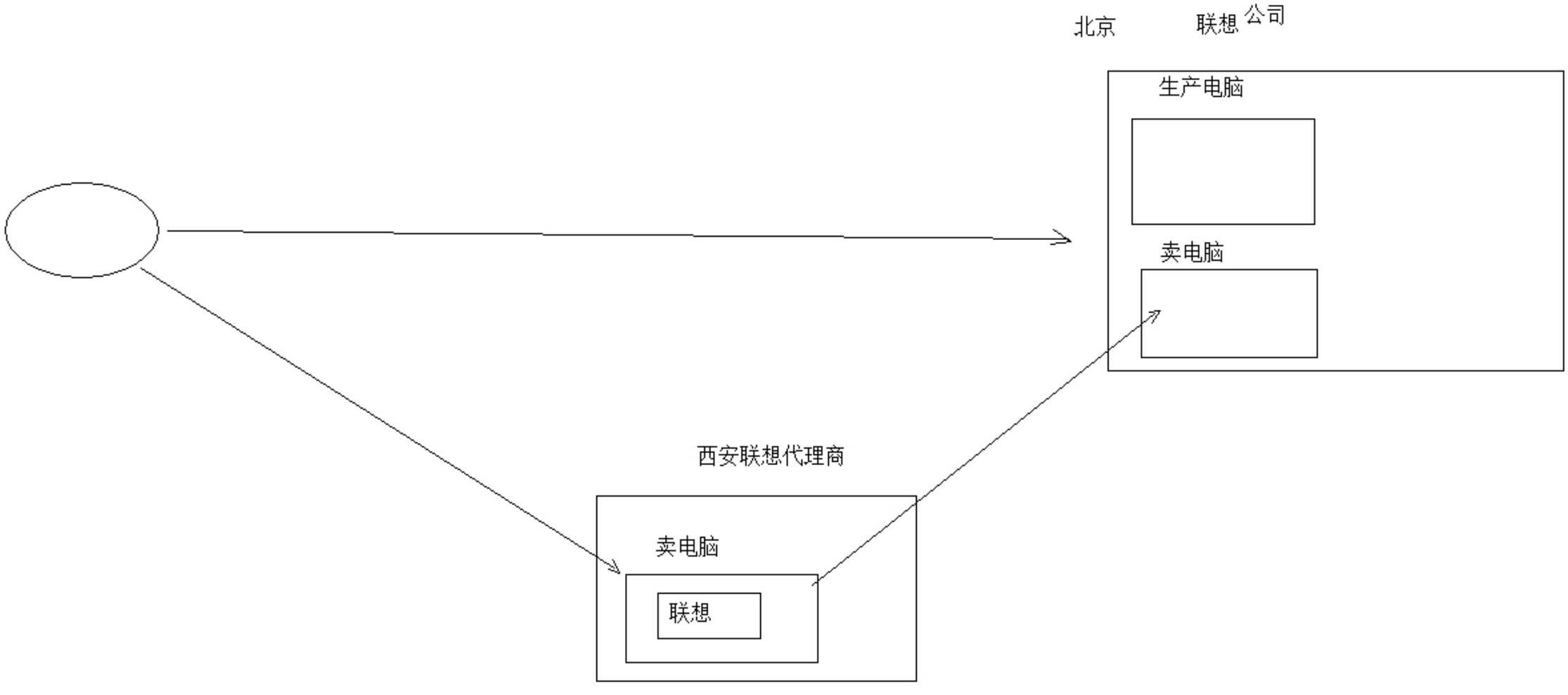
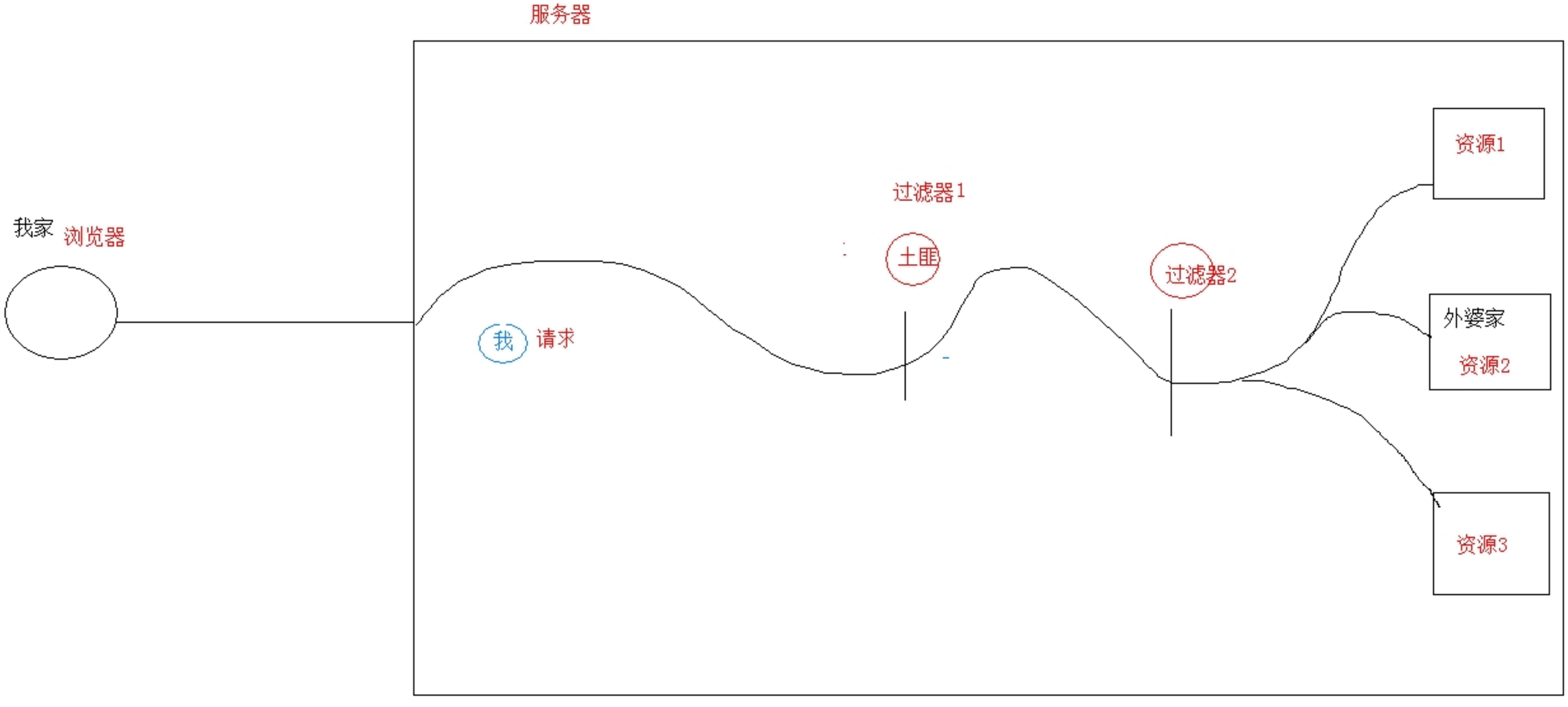
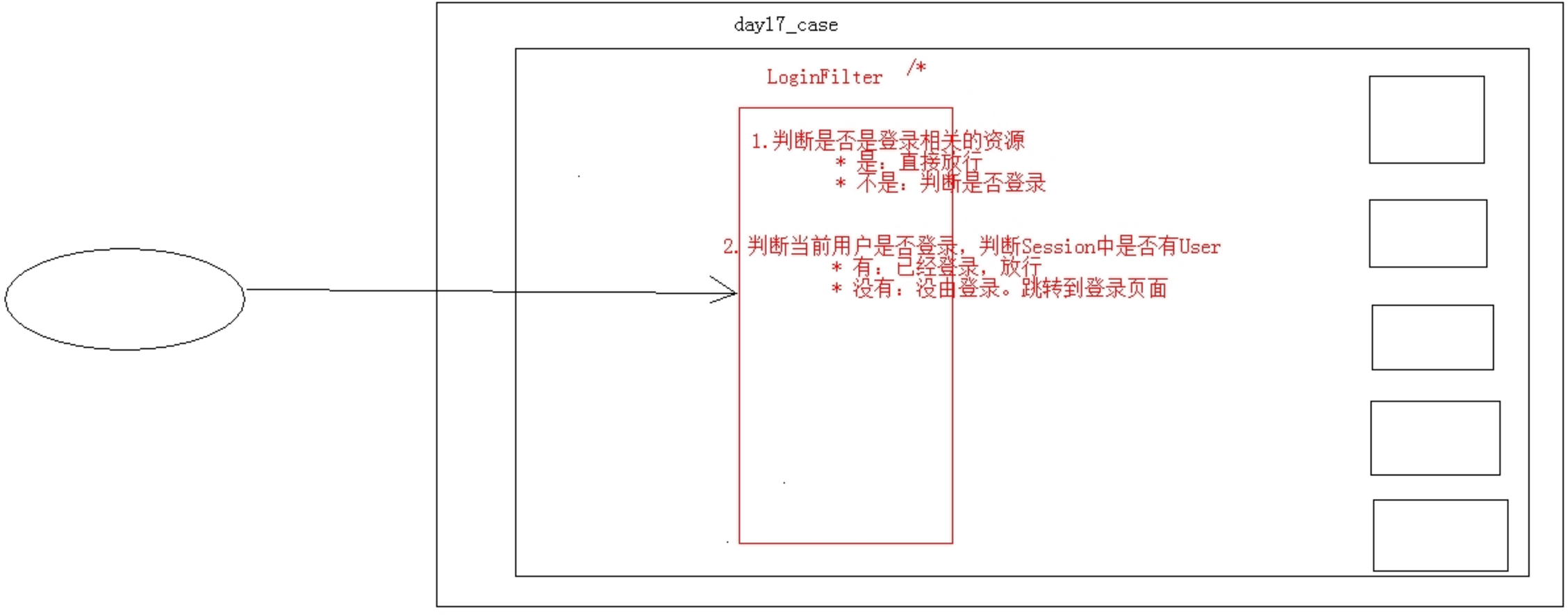
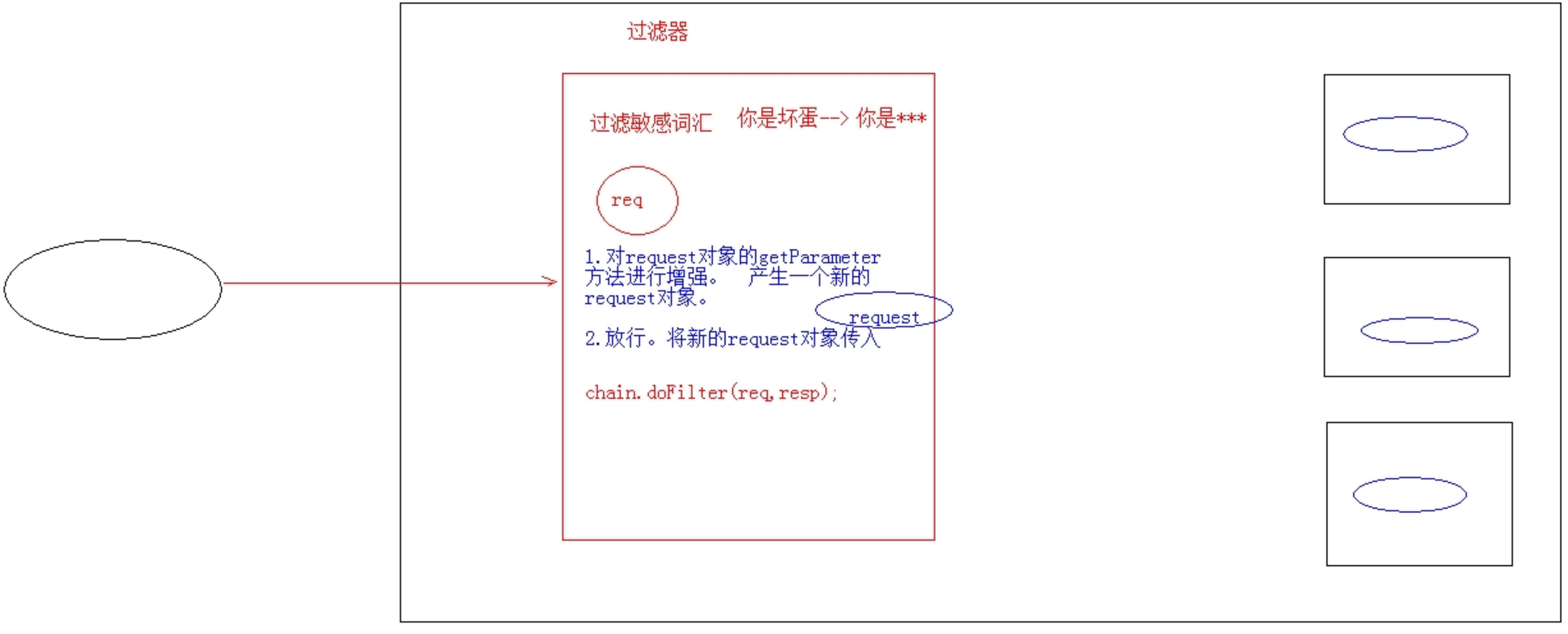

北京
联想 公司
生产电脑
卖电脑
西安联想代理商
卖电脑
联想


服务器
资源1
过滤器1
我家
浏览器
土匪
过滤器2
外婆家
资源2
我
请求
资源3


day17_case
LoginFilter
/*
1.判断是否是登录相关的资源
* 是：直接放行
* 不是：判断是否登录
2.判断当前用户是否登录，判断Session中是否有User
* 有：已经登录，放行
* 没有：没由登录。跳转到登录页面


过滤器
过滤敏感词汇
你是坏蛋--> 你是***
req
1.对request对象的getParameter方法进行增强。 产生一个新的request对象。
request
2.放行。将新的request对象传入
chain.doFilter(req,resp);

今日内容

1. Filter：过滤器
2. Listener：监听器

Filter：过滤器

1. 概念：
    * 生活中的过滤器：净水器,空气净化器，土匪、
    * web中的过滤器：当访问服务器的资源时，过滤器可以将请求拦截下来，完成一些特殊的功能。
    * 过滤器的作用：
        * 一般用于完成通用的操作。如：登录验证、统一编码处理、敏感字符过滤...

2. 快速入门：
    1. 步骤：
        1. 定义一个类，实现接口Filter
        2. 复写方法
        3. 配置拦截路径
            1. web.xml
            2. 注解
    2. 代码：

```
@WebFilter("/*")//访问所有资源之前，都会执行该过滤器
public class FilterDemo1 implements Filter {
    @Override
    public void init(FilterConfig filterConfig) throws ServletException {

    }

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain filterChain) throws IOException, ServletException {
        System.out.println("filterDemo1被执行了....");


        //放行
```

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

            }

            @Override
            public void destroy() {

            }
        }
```

3. 过滤器细节：
    1. web.xml配置
```
        <filter>
            <filter-name>demo1</filter-name>
            <filter-class>cn.itcast.web.filter.FilterDemo1</filter-
class>
        </filter>
        <filter-mapping>
            <filter-name>demo1</filter-name>
            <!-- 拦截路径 -->
            <url-pattern>/*</url-pattern>
        </filter-mapping>
```
    2. 过滤器执行流程
        1. 执行过滤器
        2. 执行放行后的资源
        3. 回来执行过滤器放行代码下边的代码
    3. 过滤器生命周期方法
        1. init:在服务器启动后，会创建Filter对象，然后调用init方法。只执行一次。用于加载资源
        2. doFilter:每一次请求被拦截资源时，会执行。执行多次
        3. destroy:在服务器关闭后，Filter对象被销毁。如果服务器是正常关闭，则会执行destroy方法。只执行一次。用于释放资源
    4. 过滤器配置详解
        * 拦截路径配置：
            1. 具体资源路径： /index.jsp 只有访问index.jsp资源时，过滤器才会被执行
            2. 拦截目录： /user/* 访问/user下的所有资源时，过滤器都会被执行
            3. 后缀名拦截： *.jsp 访问所有后缀名为jsp资源时，过滤器都会被执行
            4. 拦截所有资源：/* 访问所有资源时，过滤器都会被执行
        * 拦截方式配置：资源被访问的方式

* 注解配置：
    * 设置dispatcherTypes属性
        1. REQUEST：默认值。浏览器直接请求资源
        2. FORWARD：转发访问资源
        3. INCLUDE：包含访问资源
        4. ERROR：错误跳转资源
        5. ASYNC：异步访问资源
* web.xml配置
    * 设置<dispatcher></dispatcher>标签即可

5. 过滤器链(配置多个过滤器)
    * 执行顺序：如果有两个过滤器：过滤器1和过滤器2
        1. 过滤器1
        2. 过滤器2
        3. 资源执行
        4. 过滤器2
        5. 过滤器1

    * 过滤器先后顺序问题：
        1. 注解配置：按照类名的字符串比较规则比较，值小的先执行
            * 如： AFilter 和 BFilter，AFilter就先执行了。
        2. web.xml配置： <filter-mapping>谁定义在上边，谁先执行

4. 案例：
    1. 案例1_登录验证
        * 需求：
            1. 访问day17_case案例的资源。验证其是否登录
            2. 如果登录了，则直接放行。
            3. 如果没有登录，则跳转到登录页面，提示"您尚未登录，请先登录"。

    2. 案例2_敏感词汇过滤
        * 需求：
            1. 对day17_case案例录入的数据进行敏感词汇过滤
            2. 敏感词汇参考《敏感词汇.txt》
            3. 如果是敏感词汇，替换为 ***

        * 分析：
            1. 对request对象进行增强。增强获取参数相关方法
            2. 放行。传递代理对象

* 增强对象的功能：
    * 设计模式：一些通用的解决固定问题的方式
    1. 装饰模式
    2. 代理模式
        * 概念：
            1. 真实对象：被代理的对象
            2. 代理对象：
            3. 代理模式：代理对象代理真实对象，达到增强真实对象功能的目的
        * 实现方式：
            1. 静态代理：有一个类文件描述代理模式
            2. 动态代理：在内存中形成代理类
                * 实现步骤：
                    1. 代理对象和真实对象实现相同的接口
                    2. 代理对象 = Proxy.newProxyInstance();
                    3. 使用代理对象调用方法。
                    4. 增强方法

                * 增强方式：
                    1. 增强参数列表
                    2. 增强返回值类型
                    3. 增强方法体执行逻辑

Listener：监听器

* 概念：web的三大组件之一。
    * 事件监听机制
        * 事件　　：一件事情
        * 事件源 ：事件发生的地方
        * 监听器 ：一个对象
        * 注册监听：将事件、事件源、监听器绑定在一起。 当事件源上发生某个事件后，执行监听器代码

* ServletContextListener:监听ServletContext对象的创建和销毁
    * 方法：
        * void contextDestroyed(ServletContextEvent sce) ：ServletContext对象被销毁之前会调用该方法
        * void contextInitialized(ServletContextEvent sce) ：ServletContext对象创建后会调用该方法
    * 步骤：
        1. 定义一个类，实现ServletContextListener接口
        2. 复写方法

3. 配置
    1. web.xml
        <listener>
         <listener-class>cn.itcast.web.listener.ContextLoaderListener</listener-class>
        </listener>

        * 指定初始化参数<context-param>
    2. 注解:
        * @WebListener