

JSP网站开发

第二章 servlet

目录

- 一、理解servlet的工作原理
- 二、掌握servlet的生命周期
- 三、掌握servlet常用的API
- 四、掌握转发和重定向
- 五、理解过滤器和监听器

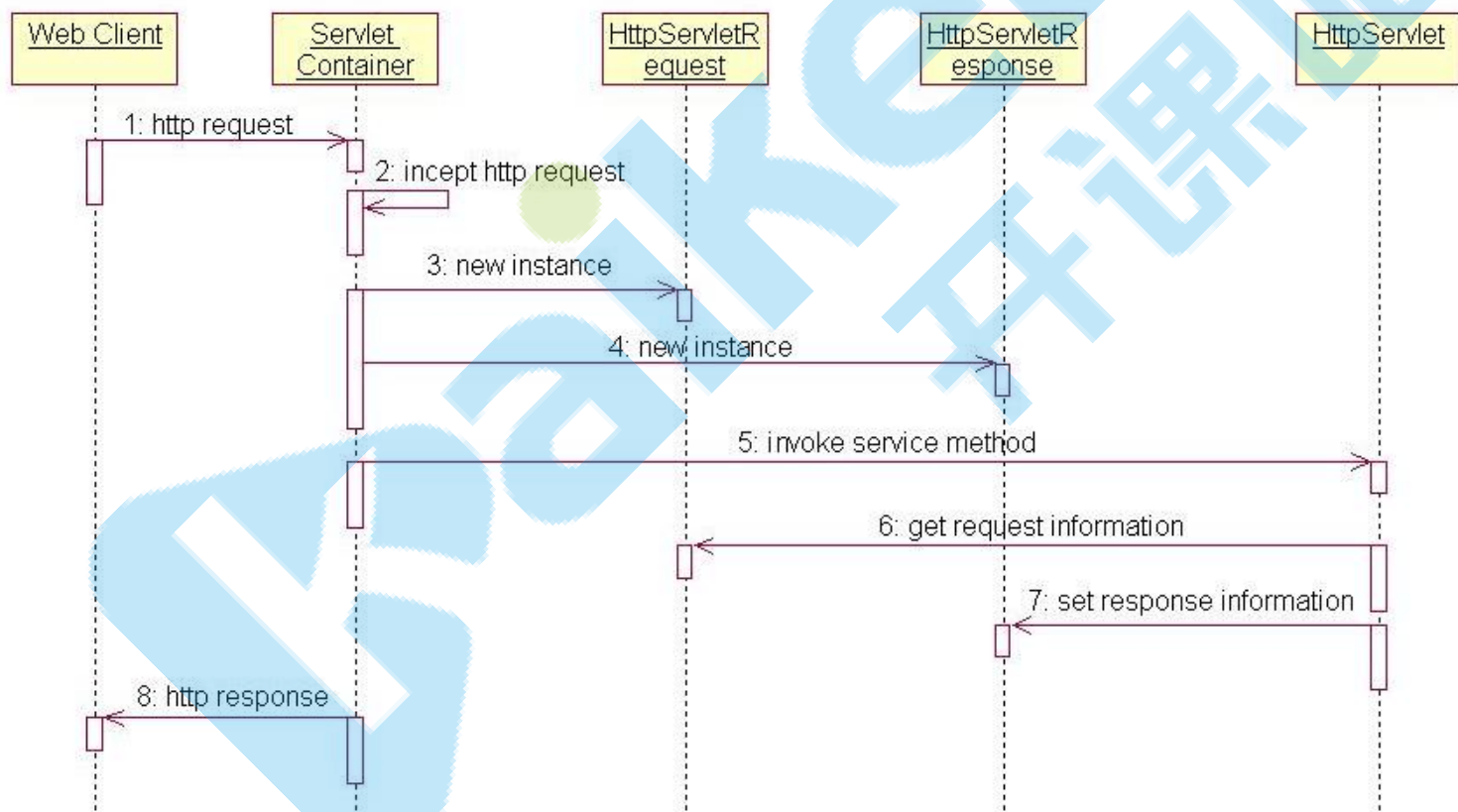
1.Servlet的概念及工作原理

1.1servlet概念:

全称**Java Servlet**，是用Java编写的**服务器端程序**。其主要功能在于交互式地浏览和修改数据，生成动态Web内容。Servlet运行于支持Java应用的服务器中。从原理上讲，Servlet可以响应任何类型的请求，但绝大多数情况下Servlet只用来扩展基于**HTTP协议**的Web服务器。

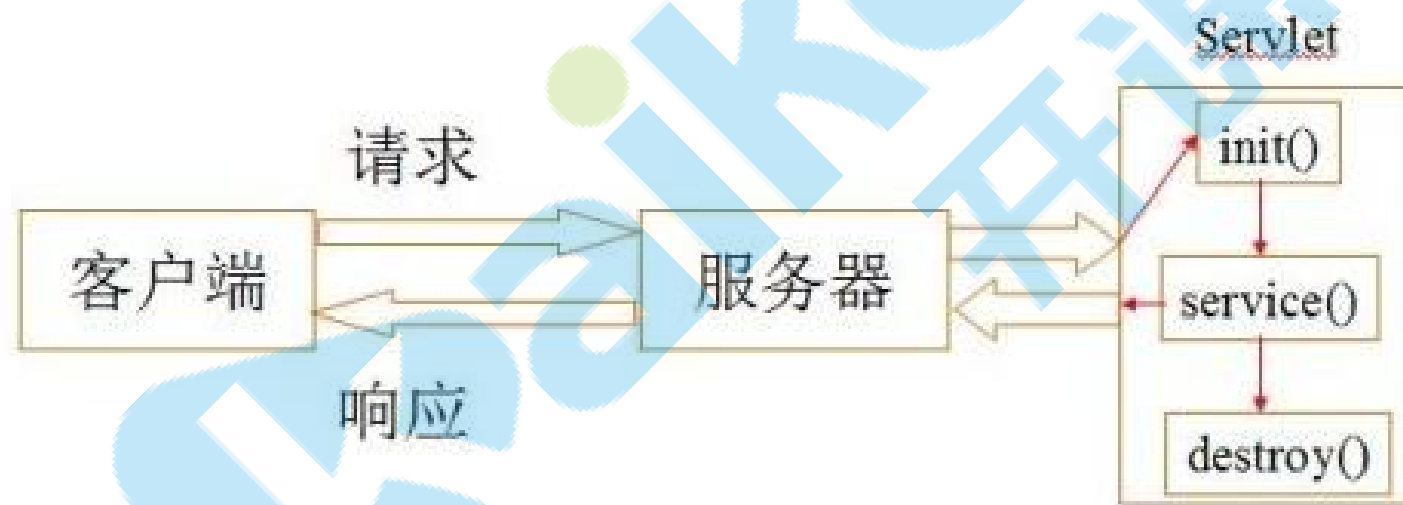
1.Servlet的概念及工作原理

1.2工作原理：



2.Servlet的生命周期

2.1生命周期:



3.第一个servlet程序

3.1servlet编写:

```
//我们第一个servlet类
public class MyFirstServlet extends HttpServlet{

    //重写doPost方法
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        System.out.println("我处理http的POST请求.....");
    }
    //重写doGet方法
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        System.out.println("我处理http的GET请求.....");
    }
}
```

3.第一个servlet程序

3.2servlet配置:

需要在web.xml里配置以下内容:

```
<servlet>
  <servlet-name>myServlet</servlet-name>
  <servlet-class>com.lechenggu.servletDemo.servlet.MyFirstServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>myServlet</servlet-name>
  <url-pattern>/firstServlet</url-pattern>
</servlet-mapping>
```


3.第一个servlet程序

实际上还有一些方法是不常用的。例如：doDelete和doPut

原因：就以我们最常用的Tomcat为例：tomcat的readonly参数默认是true，那也就是说默认情况下，是能接收“获取”的指令；然而delete指令是：通过http请求删除URL上的资源；put指令是：往服务器上传资源；不像get和post是专门用来获取资源的；所以不常用。



了解

3.第一个servlet程序

3.3servlet的入口:

同学们一般会产生疑惑: servlet配置完, 怎么没有main方法呢? 如何调用啊?

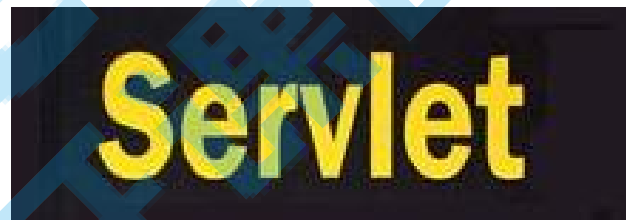
答: Servlet实际上是tomcat容器生成的, 调用init方法可以初始化。他有别于普通java的执行过程, 普通java需要main方法; 但是web项目由于是服务器控制的创建和销毁, 所以servlet的访问也需要tomcat来控制。通过tomcat访问servlet的机制是通过使用http协议的URL来访问, 所以servlet的配置完想要访问, 必须通过URL来访问, 所以没有main方法。

4.Servlet常用的API

4.1 API分类:

Servlet API由两个软件包组成:

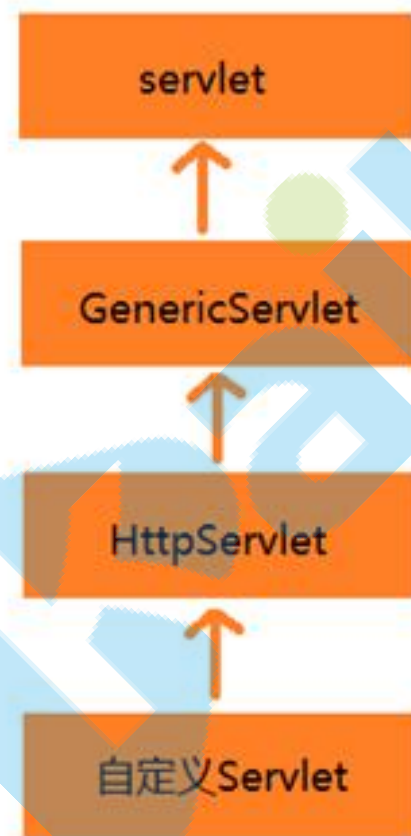
- 1、`javax.servlet`包
- 2、`javax.servlet.http`包



`javax.servlet`包是跟http请求无关的一般性servlet;
`javax.servlet.http`包是跟http协议相关的类。
这两个包都在tomcat的servlet-api.jar包中。

4.Servlet常用的API

4.2ServletAPI:



servlet接口决定了servlet的生命周期和服务的方法

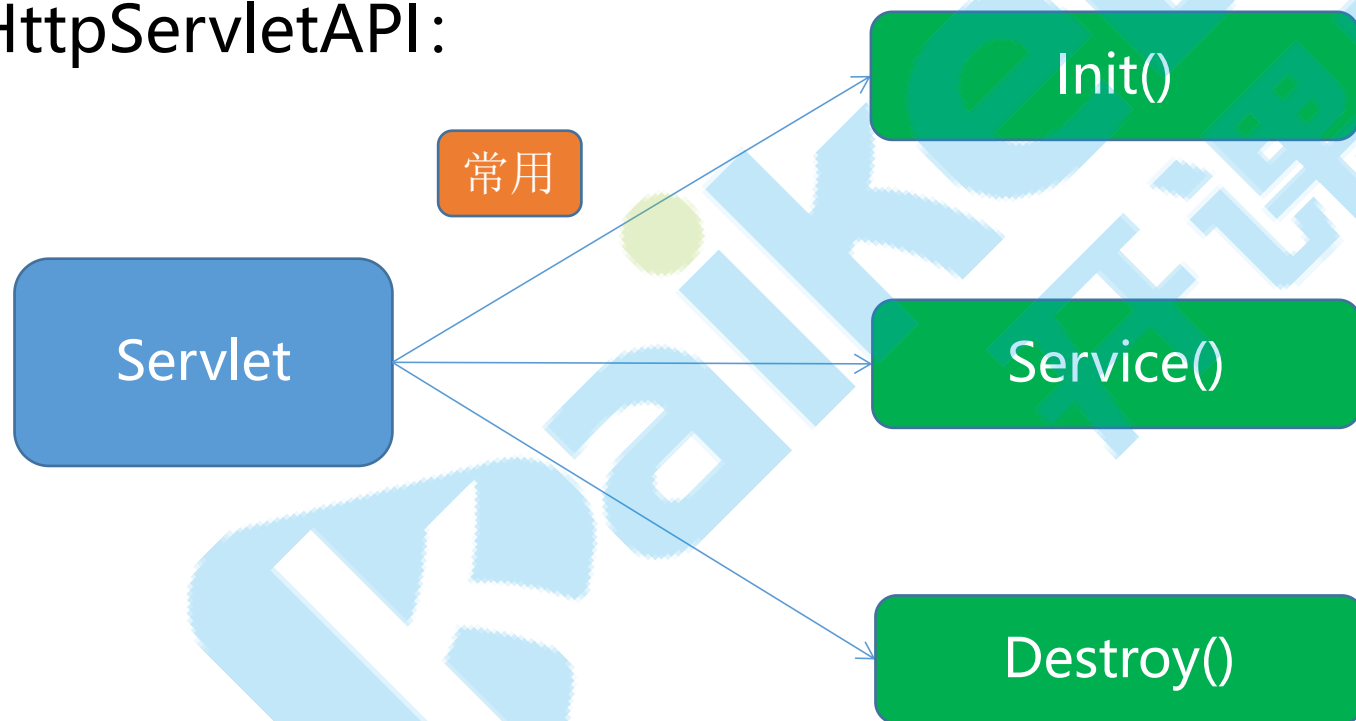
GenericServlet是一个抽象类与协议无关

HttpServlet实现了http协议的servlet

一般需要重写doPost和doGet来实现服务

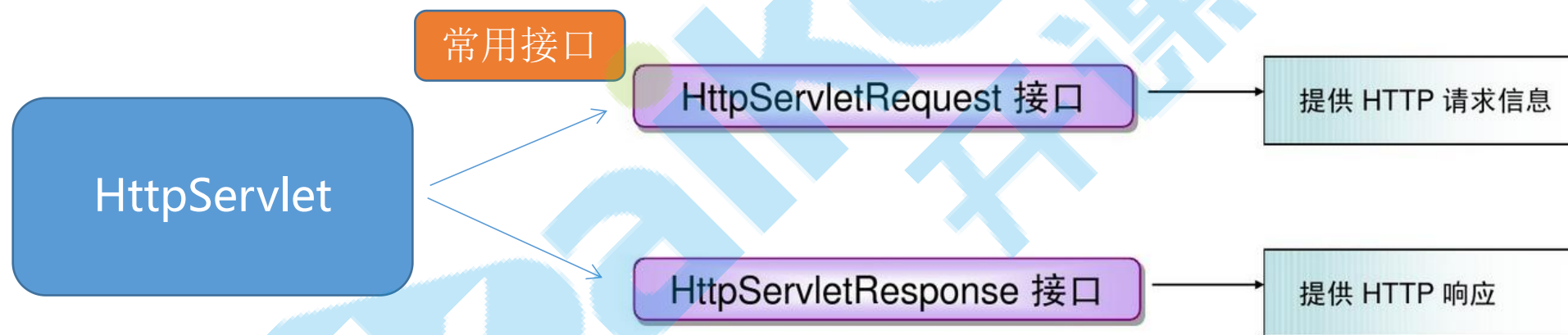
4.Servlet常用的API

4.3HttpServletAPI:



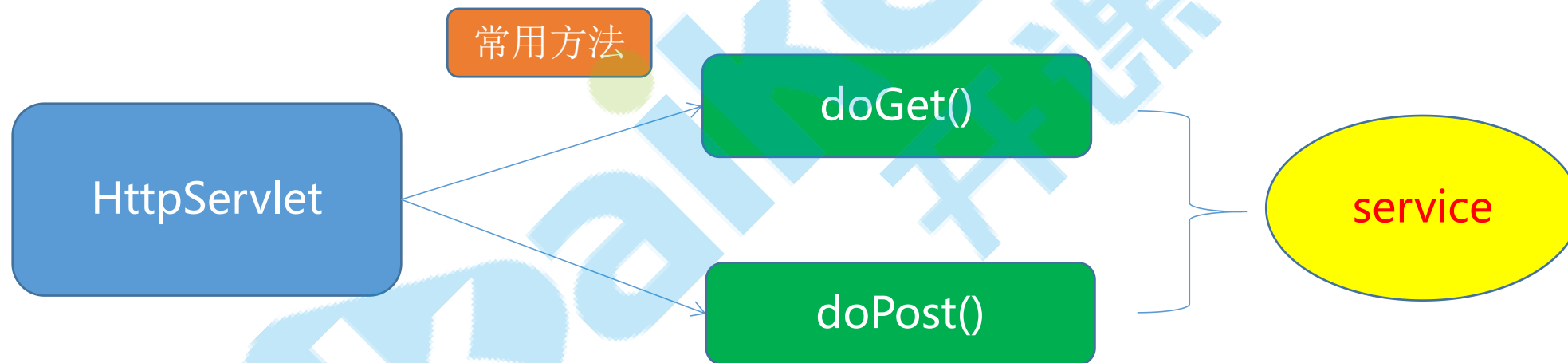
4.Servlet常用的API

4.3HttpServletAPI:



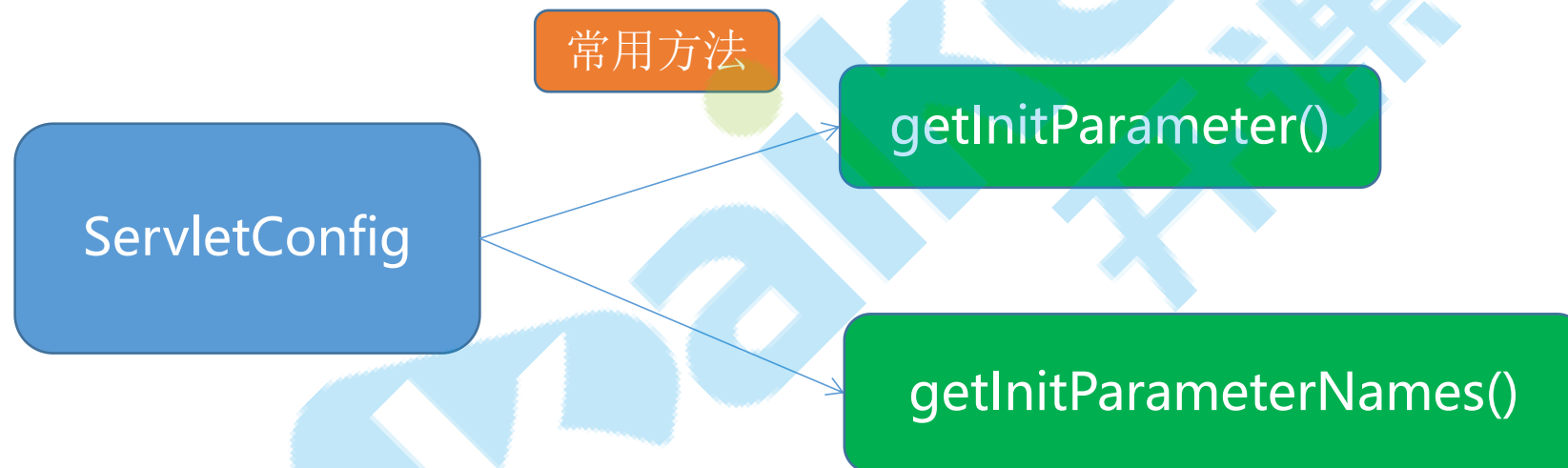
4.Servlet常用的API

4.3HttpServletAPI:



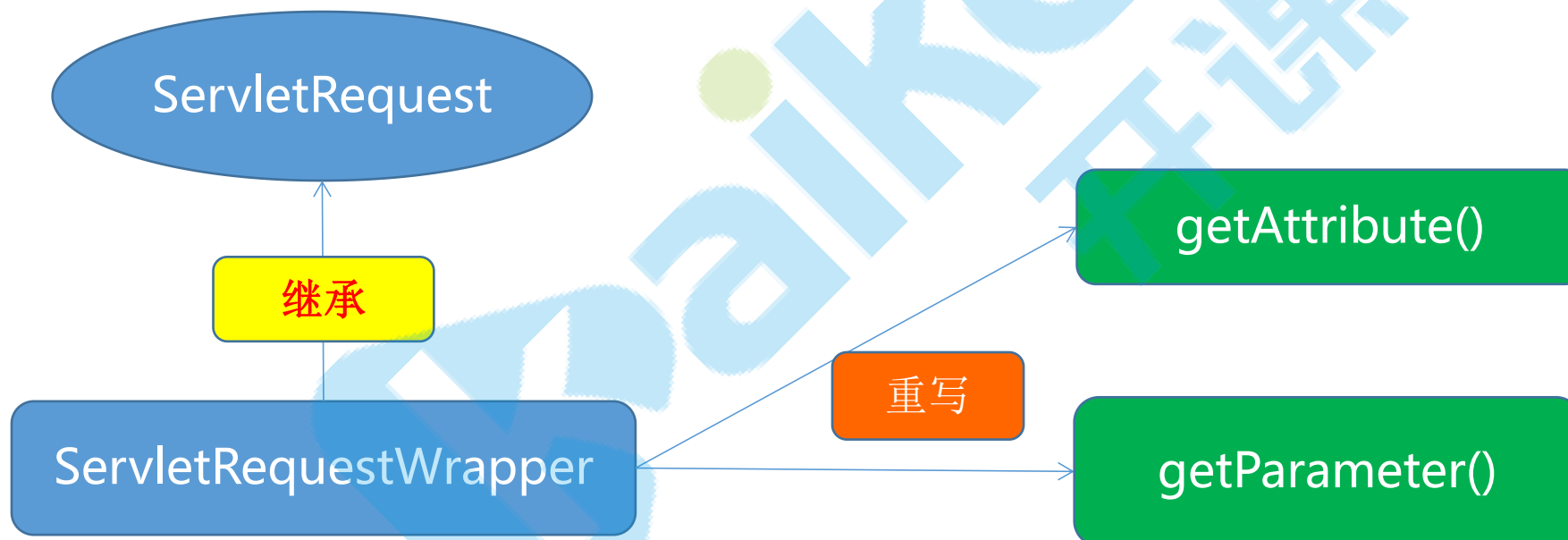
4.Servlet常用的API

4.4HttpServletAPI其他的方法:



4.Servlet常用的API

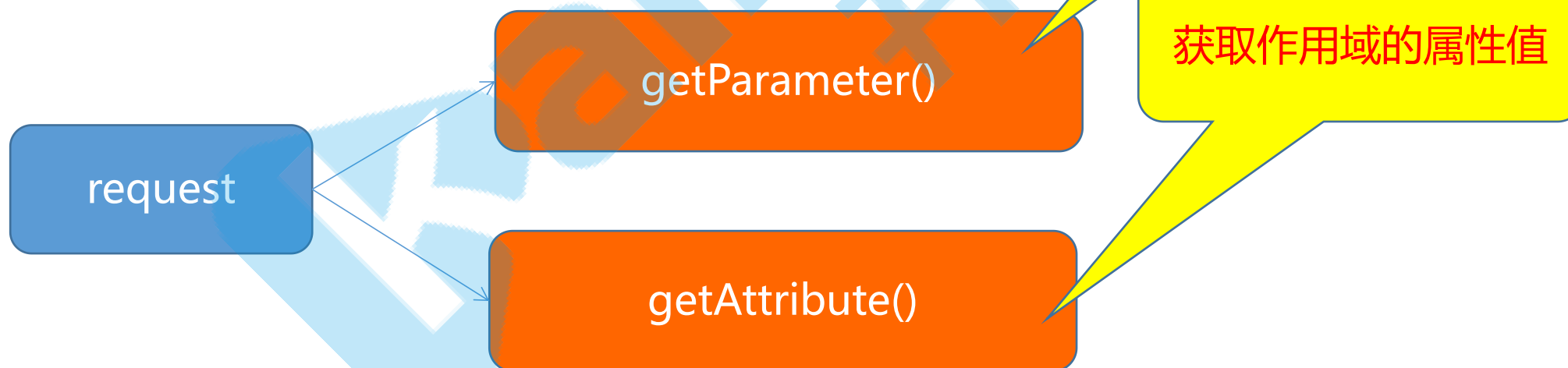
4.4HttpServletAPI其他的方法:



4.Servlet常用的API

4.5获取request对象的参数:

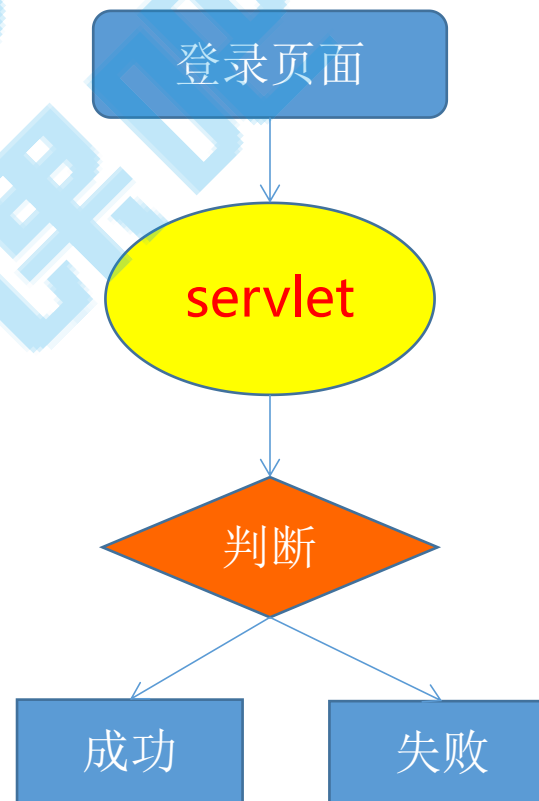
- Request.getParameter()方法
- Request.getAttribute()方法



4.Servlet常用的API

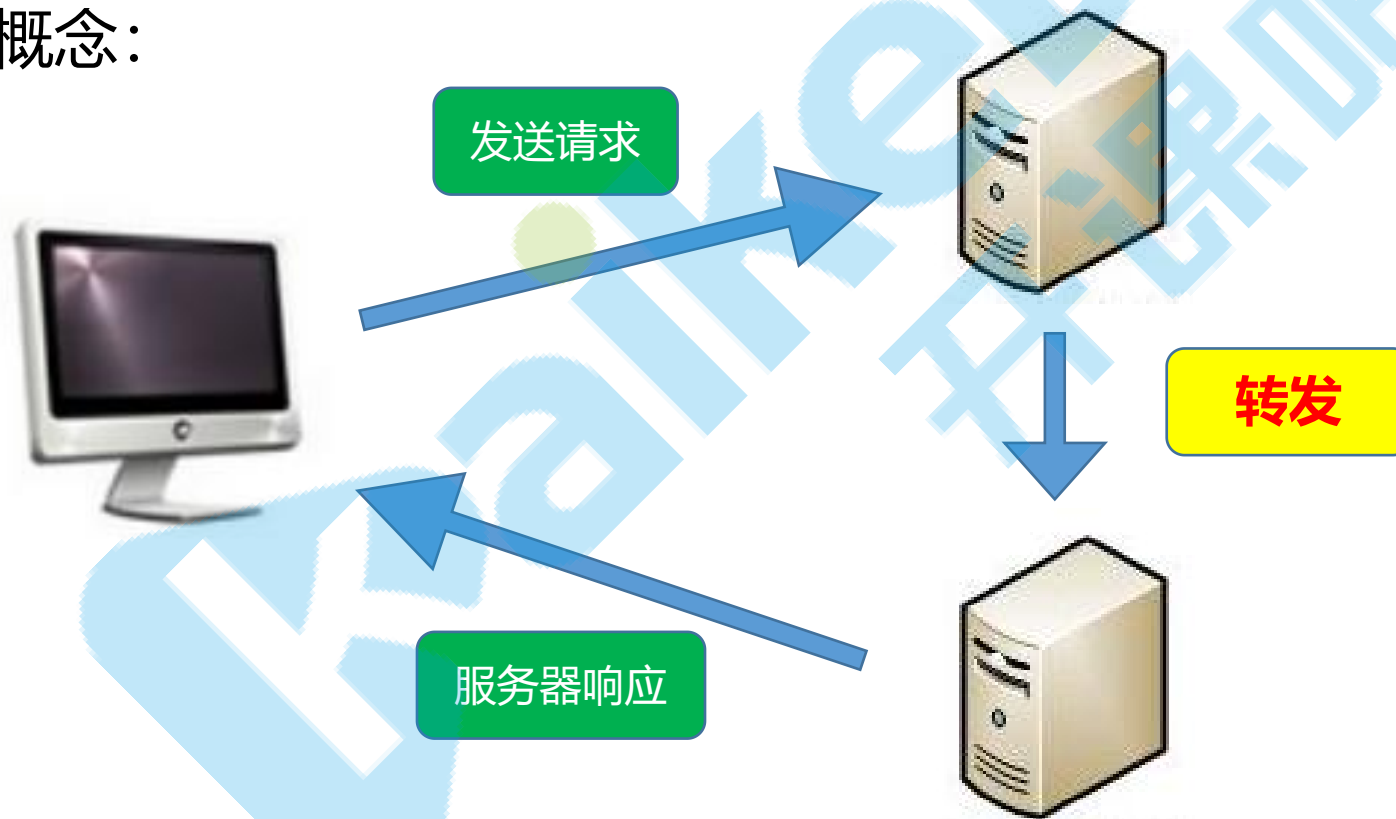
项目分析：使用servlet做用户登录

首先有一个登录页面 (login.html)，然后写一个登录判断的servlet (LoginServlet) 并且在web.xml里配置好，然后在form表单填写提交路径到servlet的url中，并且提交账号、密码；然后在servlet中获取账号、密码，并且判断并显示登陆结果。



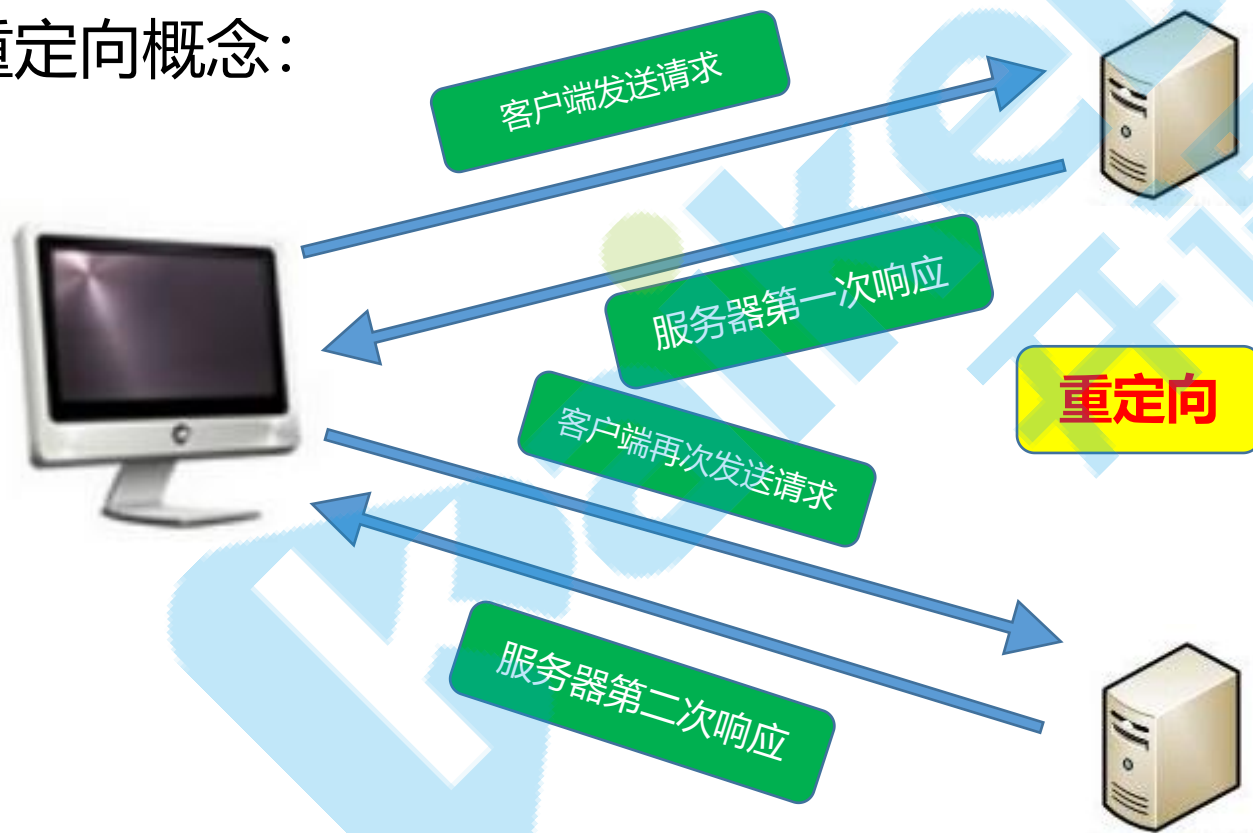
5.转发和重定向

5.1转发概念：



5.转发和重定向

5.2重定向概念：



5.转发和重定向

5.3转发和重定向的区别：

- 转发在服务器端完成的；重定向是在客户端完成的
- 转发的速度快；重定向速度慢
- 转发的是同一次请求；重定向是两次不同请求
- 转发地址栏没有变化；重定向地址栏有变化
- 转发可以携带参数，重定向不能携带参数（重要区别）**

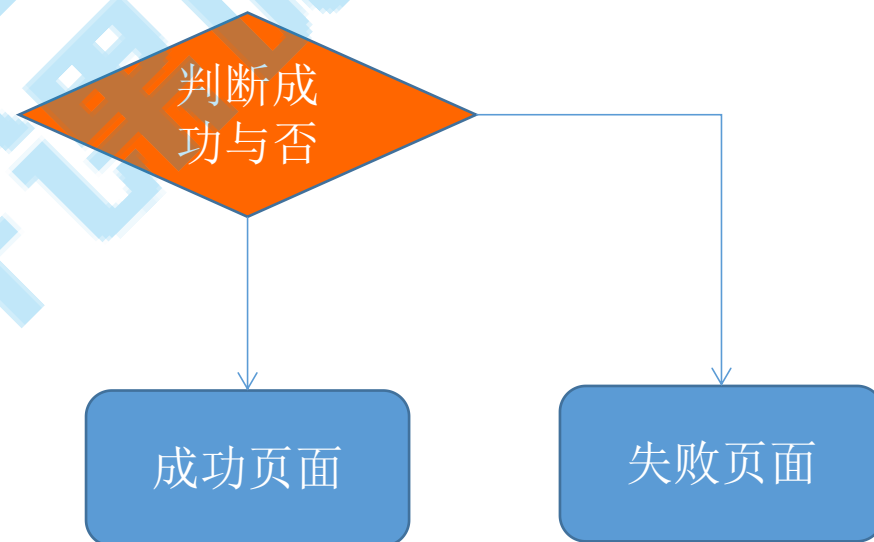
牢记



5.转发和重定向

项目分析：用户登录并且跳转成功/失败页面

根据上个servlet的判断完成之后，用转发或者重定向来跳转到成功页面或者失败页面



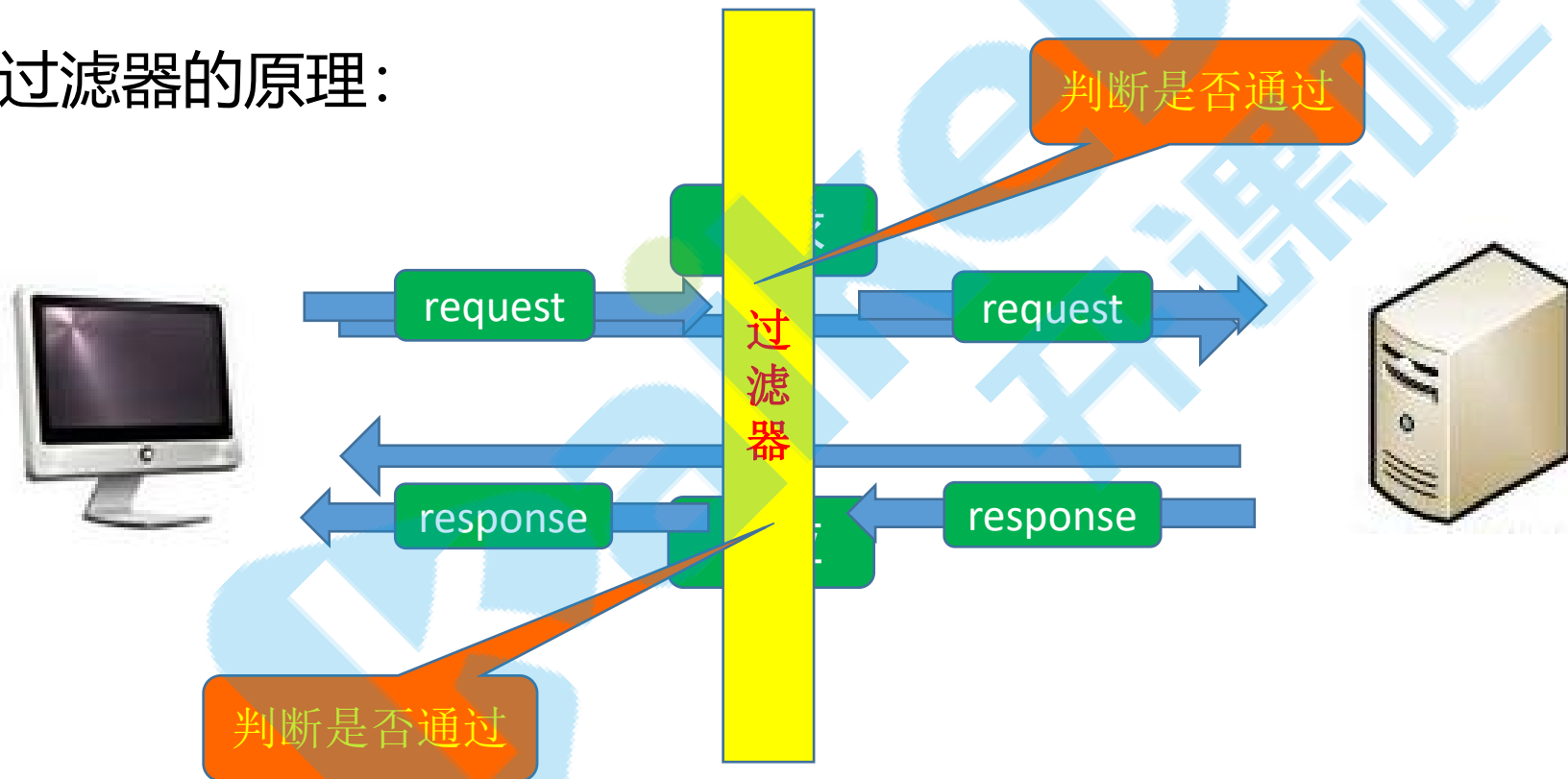
6.过滤器

6.1过滤器的概念：

当客户端发出Web资源的请求时，Web服务器根据应用程序配置文件设置的过滤规则进行检查，若客户请求满足过滤规则，则对客户请求 / 响应进行拦截，对请求头和请求数据进行检查或改动，并依次通过过滤器链，最后把请求 / 响应交给请求的Web资源处理

6.过滤器

6.2过滤器的原理：



6.过滤器

6.3过滤器的重要方法：

- init方法：初始化过滤器
- destory方法：销毁过滤器
- doFilter方法：继续向后执行FilterChain的剩余部分

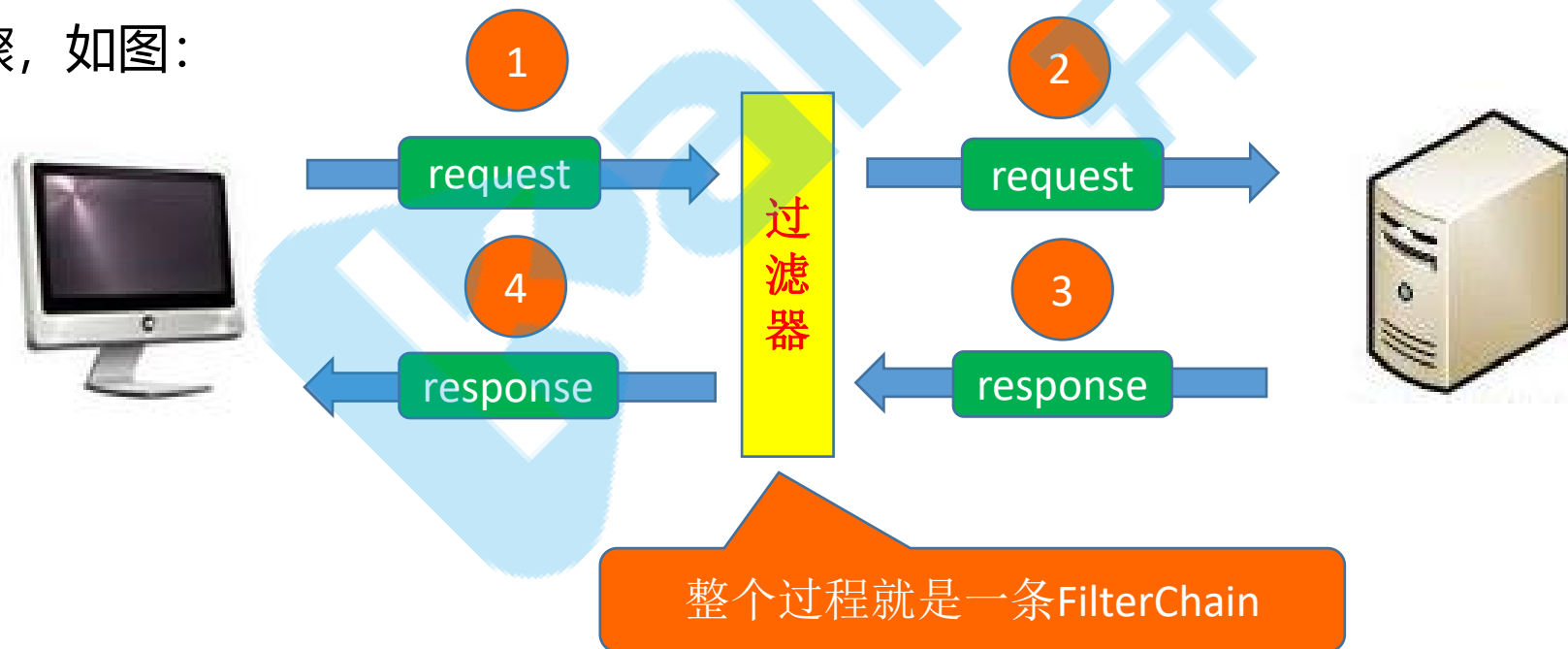
什么是
FilterChain



6.过滤器

6.4过滤器的FilterChain:

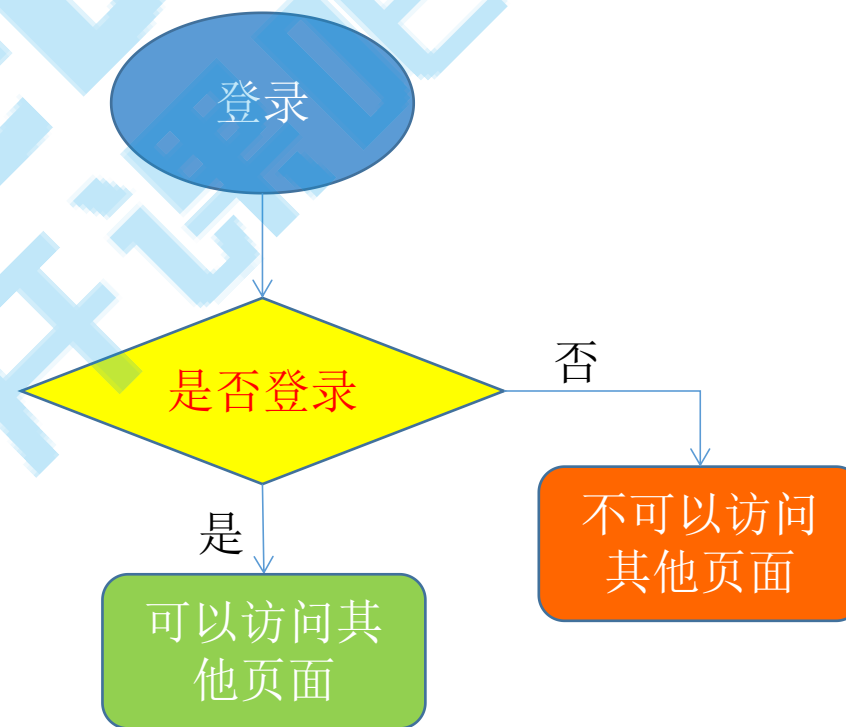
FilterChain是过滤器的链。以上图为例我们可以认为：客户端请求到过滤器再到服务器、服务器响应从服务器到过滤器再到客户端；整个过程实际上是四个步骤，如图：



6.过滤器

项目分析:

我们写用户登陆的时候，实际上需要加上未登录用户的拦截功能。即：URLFilter。来实现：如果用户没有进行正常的登录操作，是坚决不允许用户进行非法跳转的。



7.监听器

7.1监听器的概念：

监听器一般用来监听容器的创建和销毁。当然监听器也可以四大作用域对应的对象的创建和销毁，以及各自范围内参数的变化（包括：增、删、改）。

说白了，监听器很像一个



7.监听器

7.2常见的监听器：

- ServletContextListener
- ServletContextAttributeListener
- HttpSessionAttributeListener
- ServletRequestAttributeListener

7.监听器

7.3 ServletContextListener :

ServletContextListener在日常工作当中比较常用，因为这个监听器在servlet初始化的时候，就能监听到；由于这个特征，我们常常用它来监听初始化状态，并且获取某些初始化的参数，例如：init-Parameter等等.....

7.监听器

7.4属性监听器：

xxxAttributeListener格式的监听器，用来监听四个作用域属性的增、删、改操作何时发生。以此特性，我们可以用它来生成日志：某时某分，某用户在某范围内增/删/改了某数据.....等等；