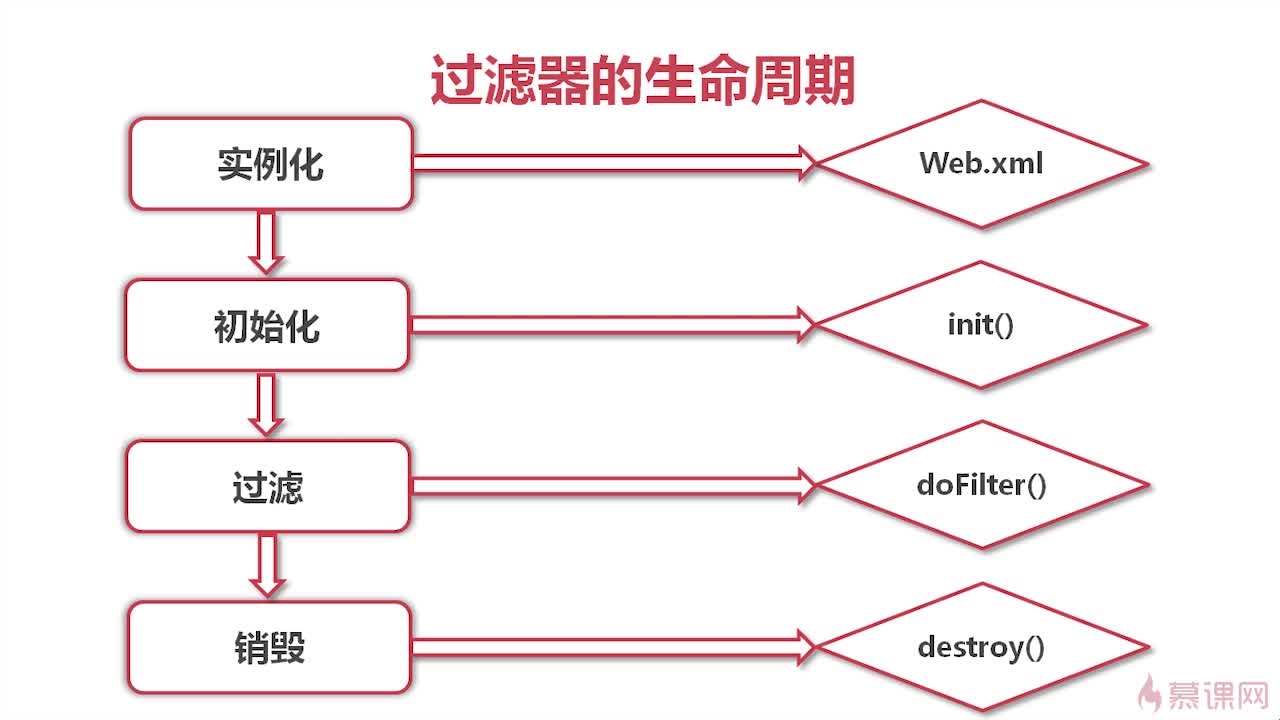
1、过滤器定义：

过滤器是一个服务器端的组件，它可以截取用户端的请求与相应信息，并对这些信息过滤。

2、过滤器工作原理：



3、过滤器生命周期



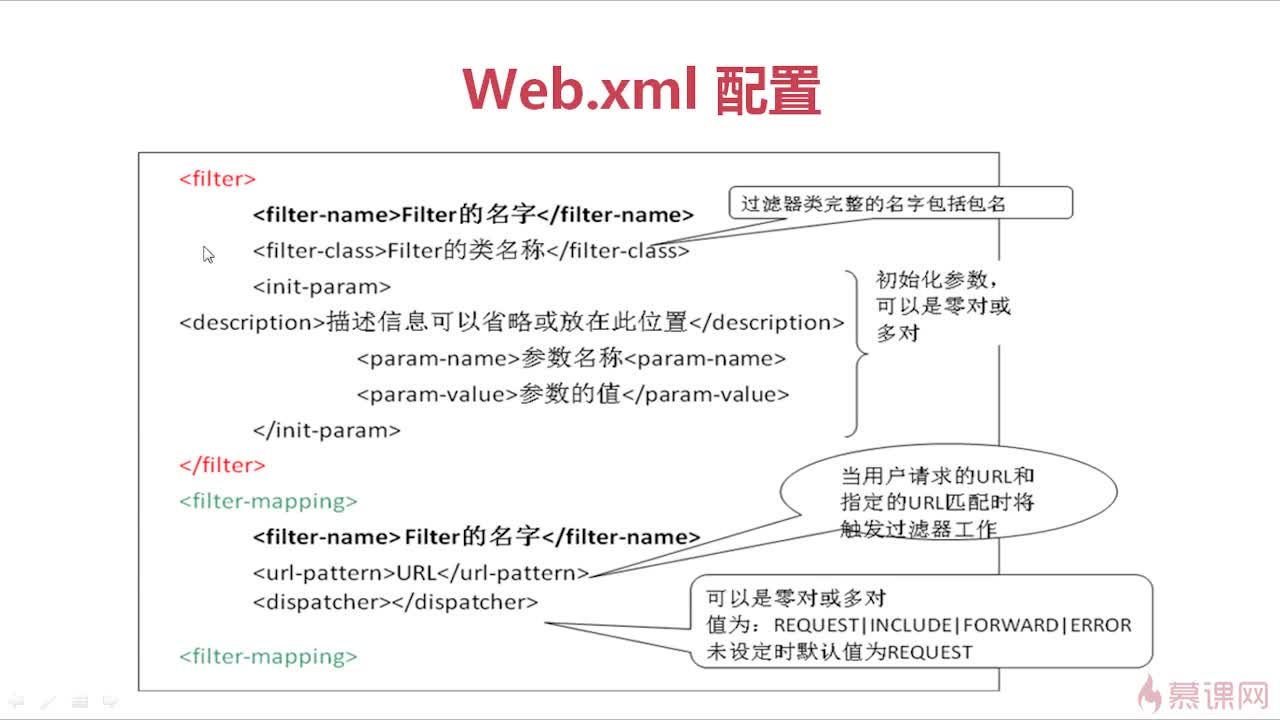
* 过滤器只能实例化一次，等到web容器关闭的时候回执行销毁方法。

4、创建单个过滤器Filter

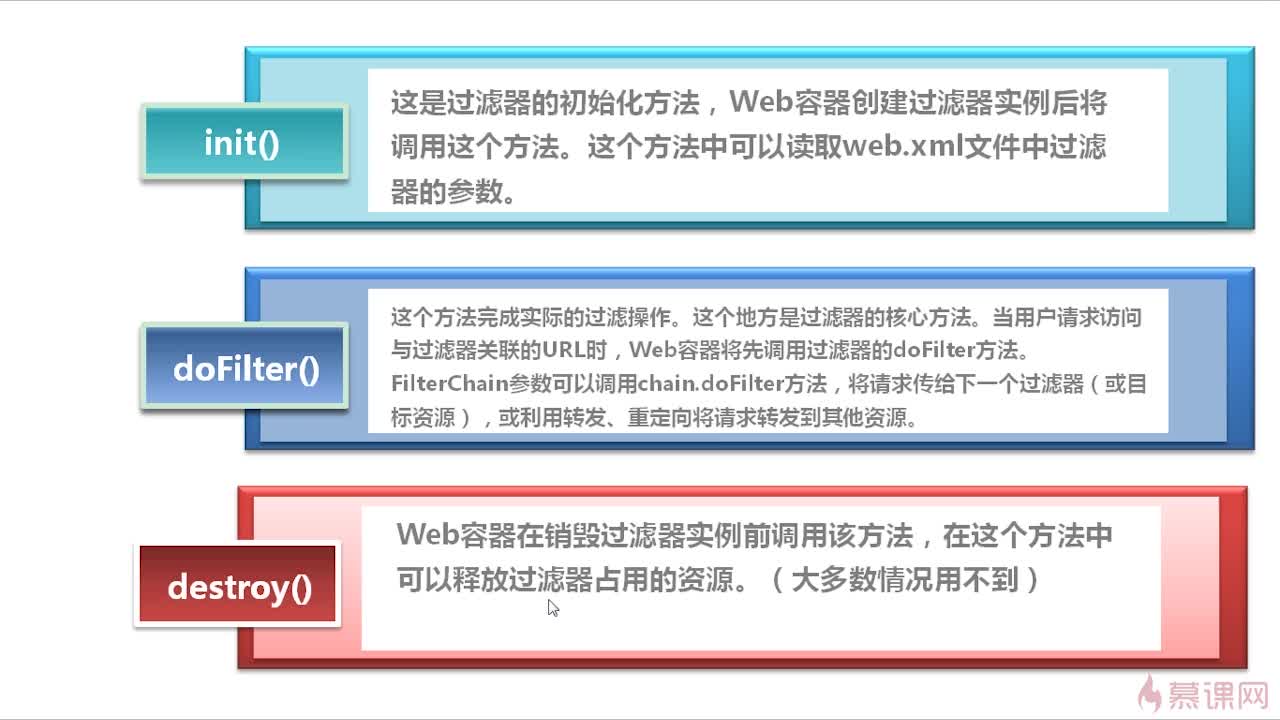
* 1、创建一个java类实现javax.servlet.Fileter接口

2、更改Web.xml配置

* Web.xml配置



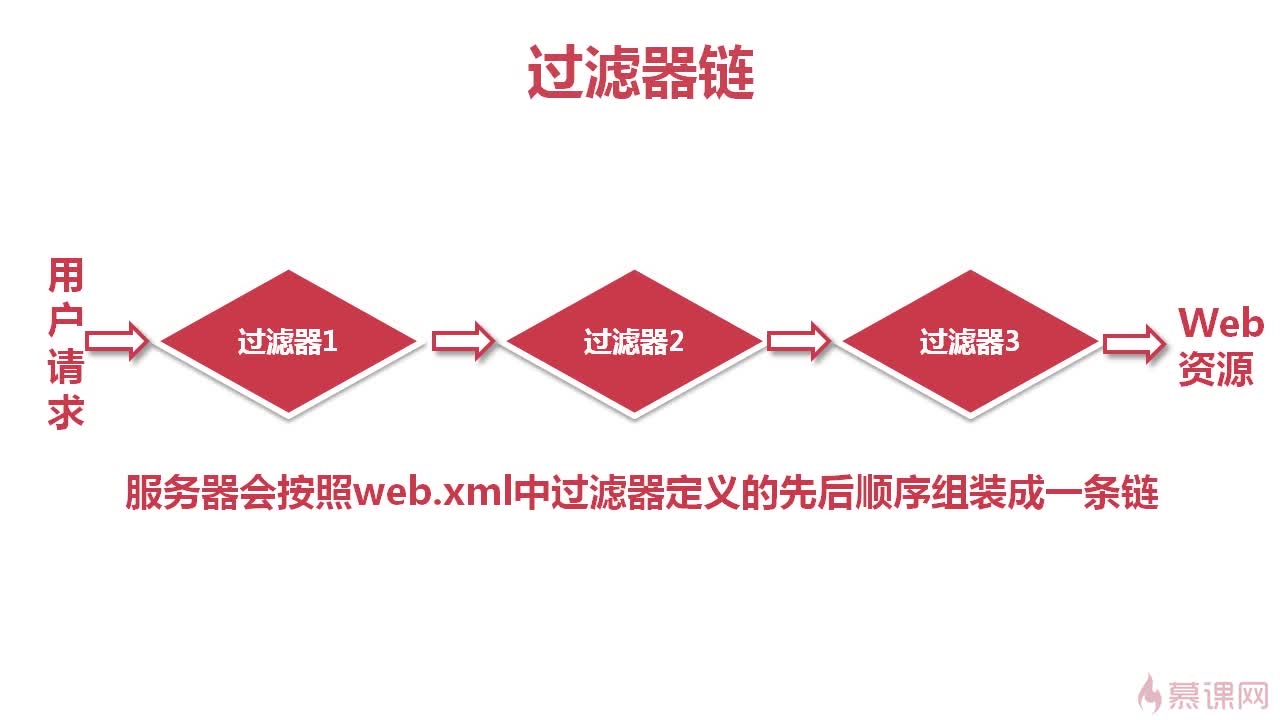
* javax.servlet.Filter方法介绍

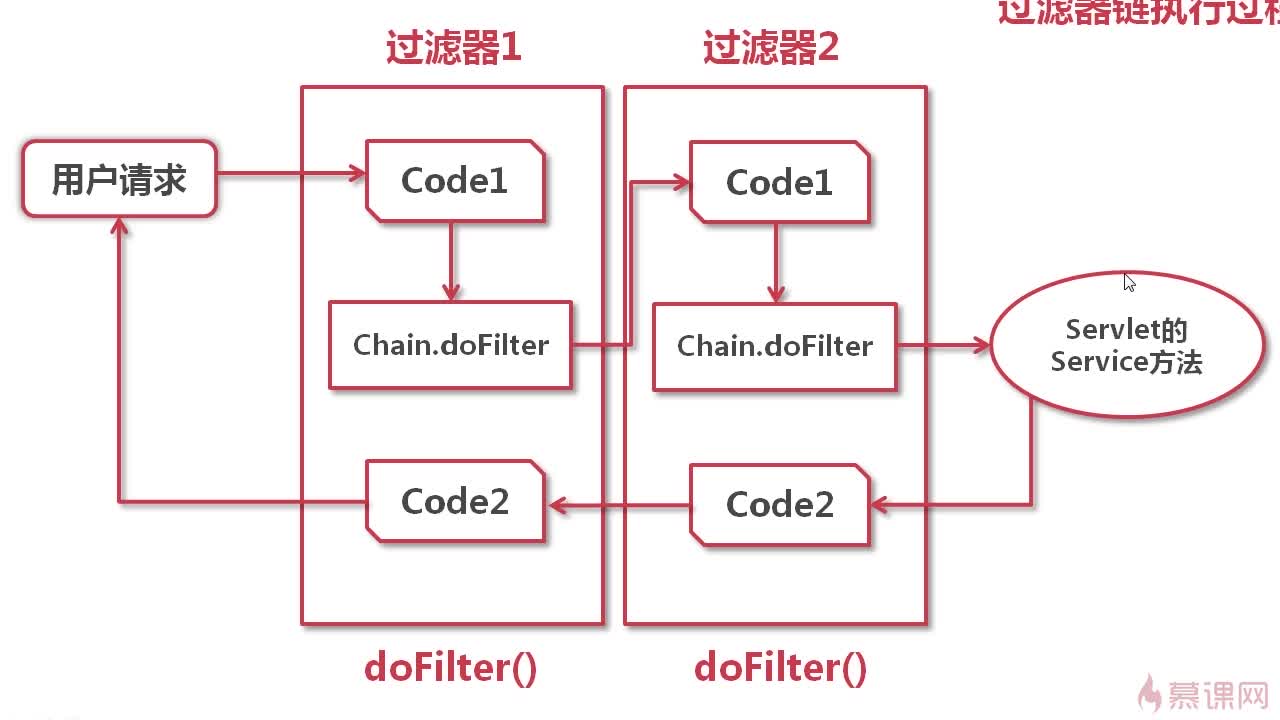


* 过滤器是可以改变请求访问的路径的，比如用户未登录就访问web资源时我们可以让其跳转到登录页面。但是过滤器是无法直接处理请求数据的，因为本身不是一个标准的servlet，本身也不是用来处理请求的。

5、创建多个过滤器—过滤器链

* 过滤器链简介





* 执行顺序

Web容器加载→实例化filter过滤器→用户请求经过过滤器→访问jsp或者servlet→返回用户请求→再次经过filter过滤器→销毁filter过滤器→关闭web容器

* Web.xml中加载顺序

实例化filter过滤器→<filter>中<filter-name>→<filter-mapping>中<filter-name>→<filter-mapping>中<url-pattern>

* Filter过滤器的执行顺序是根据web.xml中的注册顺序执行的。

6、过滤器的分类

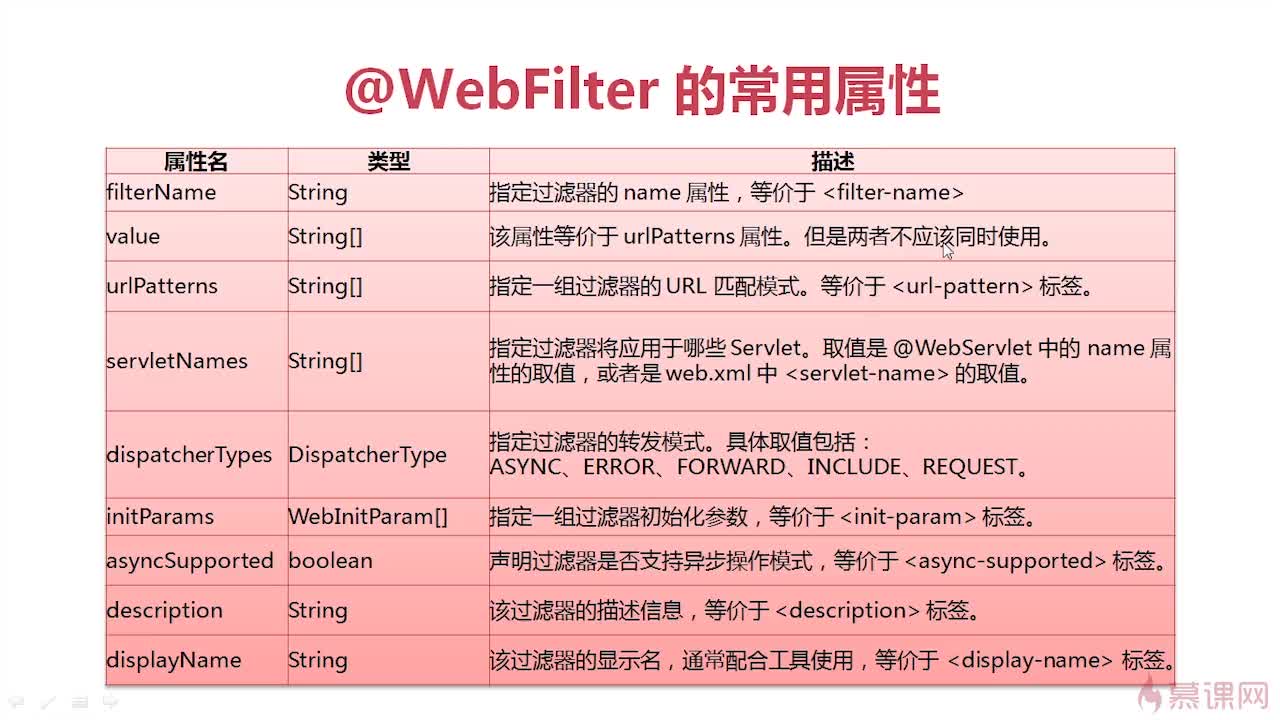
* 过滤器执行的两个条件：访问的URL地址、访问的方式



7、@WebFilter注解—放到代码中的

* Servlet3.0中增加@WebFilter,@WebFilter用于将一个类声明为过滤器，该注解将会在部署时被容器处理，容器将根据具体的属性配置将相应的类部署为过滤器。

不需要手工在web.xml里进行配置



* Jsp出现错误的页面跳转

1、配置web.xml文档

<error-page>

<error-code>404</error-code>//出现的错误页面

<location>/error.jsp</location>//出现错误后跳转的页面

</error-page>

2、访问错误页面会跳转到error.jsp

* error的filter过滤器处理过程

JSP页面出现错误→（通过error-page）跳转到error.jsp页面→检测到error信息→调用ErrorFilter过滤器→过滤器释放错误请求→真正跳转到error.jsp页面

1、利用FilterConfig初始化参数来排除不需要过滤的页面

第一步：在Web.xml中配置初始化参数

Initial Parameters

第二部：在Filter中读取到这个初始化参数，以下是代码：

private FilterConfig config;//在各个方法体之前定义初始化参数

//排除不需要检查session值的其他页面

String noLoginPath= config.getInitParameter("noLoginPath"); //获取初始值中需要排除的页面

if(noLoginPath != null)//判断程序的值，防止出现空值异常

{

String[] strArray = noLoginPath.split(";");//从获取到的值中分割出一个个独自页面

//对每个页面进行独自判断，利用FOR循环

for(int i=0; i<strArray.length;i++){

if(strArray[i] == null || " ".equals(strArray[i]))

continue;//如果值为空或者为空格，跳过这个值

if(request.getRequestURI().indexOf(strArray[i]) != -1){

arg2.doFilter(arg0, arg1);

return;//如果访问页面在初始排除值中，则放行

}

}

config = arg0;

2、在Filter过滤器中修改编码方式：同样利用初始化参数来设置

//访问页面前，设置页面的编码方式

String charset = config.getInitParameter("charset");

if(charset == null){

charset = "UTF-8";

}

request.setCharacterEncoding(charset);