对于CSRF，可能一些朋友比较陌生。我们下面先简单介绍下。

什么是CSRF呢，我们看下Wikipedia的说明：

**Cross-site request forgery**，即跨站请求伪造，也称为 "One Click Attach" 或者"Session Riding"，常缩写成CSRF。是**通过伪装来自受信任用户的请求来利用受信任的网站**。

其中，说起CSRF，经常会举的一个例子，是这样的：

用户A在访问网上银行，在银行网站里进行了一些操作后。

之后点击了一个陌生的链接。而这个链接，是用户B提供的一个恶意网页W，网页W内也包含访问和用户A相同银行信息的链接，可能是转帐，也可能是修改密码。

此时如果A的认证信息还未过期，就会被直接利用，成功帮助W进行了银行网站的对应操作，而这一切，都是在A不知情的情况下进行的。

为了防范CSRF，常见的方式有：

* 请求中包含随机token信息
* Cookie中包含csrf token信息
* 其他的验证请求头Refer等...

在Tomcat中，默认提供了一个防范CSRF的好工具: **CSRF Prevention Filter**。

Tomcat默认提供了各类的Filter，处理不同的场景和需求。像我们前面介绍过的处理编码的[Tomcat自带的设置编码Filter](http://mp.weixin.qq.com/s?__biz=MzI3MTEwODc5Ng==&mid=403224828&idx=1&sn=e93242685ed43aee33b9ee459ab02210&scene=21#wechat_redirect)，还有进行跨域处理的[Tomcat与跨域问题](http://mp.weixin.qq.com/s?__biz=MzI3MTEwODc5Ng==&mid=403375476&idx=1&sn=4ea85119f4597605edad312abdf9eae7&scene=21#wechat_redirect)等等。今天介绍的CSRF Prevention Filter也是其中的一个。

整个Filter的工作流程可以概括成以下内容：

该Filter为Web应用提供了基本的CSRF 保护。它的filter mapping对应到**/\***

并且所有返回到页面上的链接，都通过调用**HttpServletResponse**#encodeRedirectURL(String) 或者 **HttpServletResponse**#encodeURL(String)进行编码。实现机制是**生成一个token并且将其保存到session中，URL的encode也使用同样的token，当请求到达时，会比较请求中的token和session中的token是否一致，只有相同的才允许继续执行。**

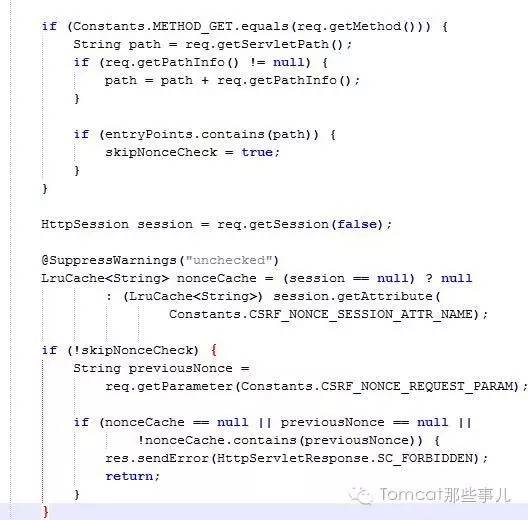
我们通过一个例子，深入源码，来了解下内部的实现细节。

还是使用Tomcat自带的Manager应用来看下。

在其**web.xml**中，有这样的配置：



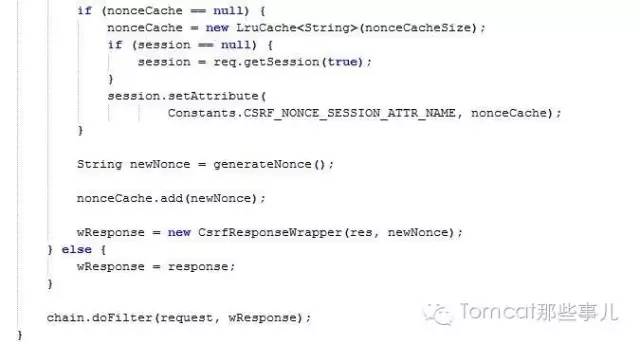
下面的内容是CsrfPreventionFilter的**doFilter**方法，



我们注意到前面的配置里包含一个entryPoints，对照代码，马上就能明白，这项配置用来做类似于exclude的功能，在配置中的映射，可以跳过检查。

而如果没有在entryPoints中，同时在session存在，但不包含对应的Nonce，就会直接返回**403**(SC\_FORBIDDEN)。

如果session不存在，就会在doFilter中走到下面的内容：



做为初次请求，会在session中保存对应的Attribute，同时添加到一个LruCache中。这里的重点在于，使用了HttpServletResponseWrapper的子类CsrfResponseWrapper替换了它做为response传入后续的流程。

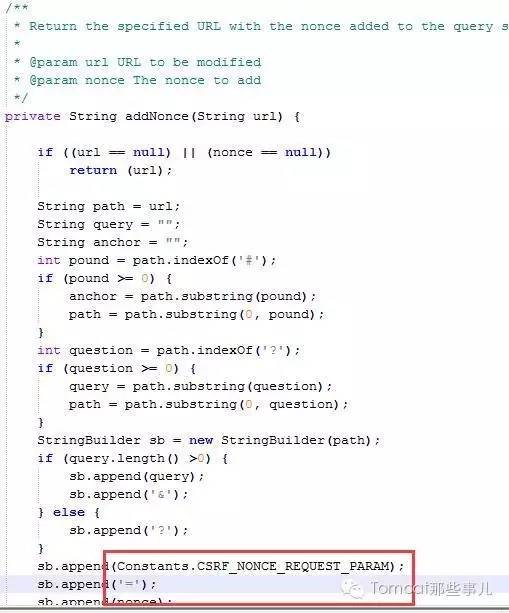
所以，后面所有调用encodeUrl的地方，其实实际调用到的是这个：

public String encodeURL(String url) {

            return **addNonce**(super.encodeURL(url));

}

addNonce对应的，是在传入URL后面增加csrf token或者是nonce的标识，用于后续请求时的识别。



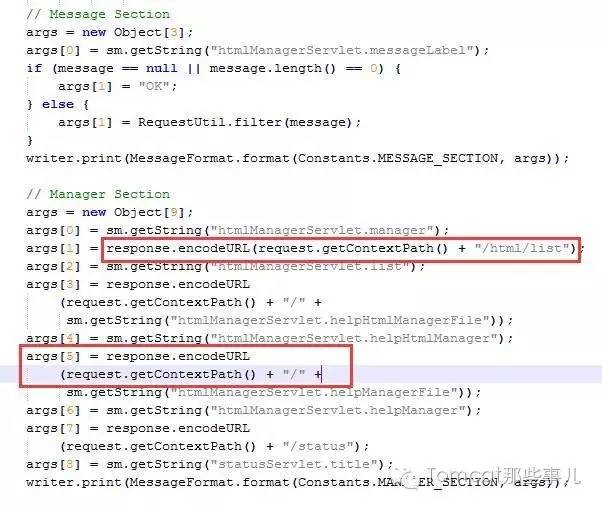
页面具体的编码操作，则是对response的encodeURL的使用，也就是我们上面addNonce的使用：

对应到Manager应用，它的页面是通过Servlet输出的，所以具体的逻辑在Java文件中，我们在页面上的连接观察到，此时获取应用列表的请求URL变成了这样：

http://localhost:8080/manager/html/list?**org.apache.catalina.filters.CSRF\_NONCE**=6BC061DD606D7BA1BDEF7F40657F0C47

每个不在entryPoints中的请求，都会加上org.apache.catalina.filters.CSRF\_NONCE=6BC061DD606D7BA1BDEF7F40657F0C47

这种形式的URL输出，就是在页面上调用encodeURL的结果，对应的Manager中的代码是这个样子：



以上，就是CSRF Prevetion Filter实现的原理和细节。当然，上面返回403的地方，以及生成nonce的地方，都可以通过Filter提供的参数来进行配置，分别对应到denyStatus和randomClass。后者需要提供一个Random的实现。

相关阅读：

[深入Tomcat的Manager应用](http://mp.weixin.qq.com/s?__biz=MzI3MTEwODc5Ng==&mid=401159585&idx=1&sn=7bb6d09acfd174b9756cda4c865d2659&scene=21#wechat_redirect)

[如何避免Manager应用被人利用](http://mp.weixin.qq.com/s?__biz=MzI3MTEwODc5Ng==&mid=2650859202&idx=1&sn=62ef3c1f4aab925d1e1727f10c392978&scene=21#wechat_redirect)

[Tomcat自带的设置编码Filter](http://mp.weixin.qq.com/s?__biz=MzI3MTEwODc5Ng==&mid=403224828&idx=1&sn=e93242685ed43aee33b9ee459ab02210&scene=21#wechat_redirect)

[和Tomcat学设计模式 | 责任链模式及Filter的工作原理](http://mp.weixin.qq.com/s?__biz=MzI3MTEwODc5Ng==&mid=402586209&idx=1&sn=7f8b97fd565c49198a93547d2ac25416&scene=21#wechat_redirect)