使用AJAX，CSRF和CORS

> "仔细观察你自己的网站上的可能的CSRF/XSRF漏洞。它们是最糟糕的漏洞 – 非常容易被攻击者利用。但对于软件开发人员来说，直到你被攻击了你才能有深入理解。"
>
> — **Jeff Atwood**

Javascript客户端

如果你正在构建一个JavaScript客户端来和Web API对接，你将需要考虑客户端是否可以使用网站其他部分使用的身份验证策略，并确定是否需要使用CSRF令牌或CORS标头。

与正在交互的API相同的上下文中发起的AJAX请求通常将使用 `SessionAuthentication` 。 这确保一旦用户登录，任何AJAX请求都可以使用与网站其他部分相同的基于会话的身份验证。

与其通信的API在不同站点上发起的AJAX请求通常需要使用基于非会话的身份验证方案，例如 `TokenAuthentication` 。

CSRF保护

Cross Site Request Forgery防止特定类型的攻击，当用户没有注销登出网站并且具有有效会话时，这种攻击可能发生。在这种情况下，恶意站点可能会在登录会话的上下文中针对目标站点执行攻击操作。

为了防范这些类型的攻击，你需要做两件事情：

1. 确保“安全”的HTTP方法（如： `GET` ， `HEAD` 和 `OPTIONS` ）不用于更改服务器端的任何状态。
2. 确保“不安全”的HTTP方法（如： `POST` ， `PUT` ， `PATCH` 和 `DELETE` ）始终需要有效的CSRF令牌。

如果你使用的是 `SessionAuthentication` ，则需要为任何 `POST` ， `PUT` ， `PATCH` 或 `DELETE` 操作包含有效的CSRF令牌。

为了使用AJAX请求，你需要在HTTP报头中包含CSRF令牌，具体方法参考Django教程（liujiangblog.com）。

CORS

Cross-Origin Resource Sharing，是允许客户端与托管在不同域上的API交互的机制。CORS的工作原理是要求服务器包含一组特定的报头信息，允许浏览器确定是否以及何时允许跨域请求。

在REST框架中处理CORS的最佳方法是在中间件中添加所需的响应头信息。这样可以确保CORS得到透明支持，而无需更改视图中的任何行为。

`Otto Yiu` 维护着能在REST框架下使用的CORS支持工具，也就是 **django-cors-headers** 包。