Fiddler 实用教程

介绍

大家好，今天给大家介绍一款非常强大且实用的网络抓包工具名称叫 Fiddler 。

Fiddler 是最强大最好用的 Web 调试工具之一，它能记录所有客户端和服务器的 http 和 https 请求，允许你监视，设置断点，甚至修改输入输出数据，Fiddler 包含了一个强大的基于事件脚本的子系统，并且能使用.net语言进行扩展。

你对HTTP 协议越了解， 你就能越掌握Fiddler的使用方法。你越使用 Fiddler ，就越能帮助你了解HTTP协议。

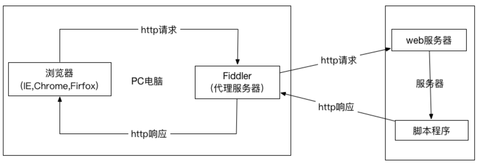
Fiddler无论对开发人员或者测试人员来说，都是非常有用的工具。

官方网址

www.fiddler2.com

工作原理

Fiddler 是以代理web服务器的形式工作的，它使用代理地址: 127.0.0.1 ，端口:8888。当Fiddler退出的时候它会自动注销，这样就不会影响别的程序。不过如果 Fiddler 非正常退出，这时候因为Fiddler没有自动注销，会造成网页无法访问。解决的办法是重新启动下Fiddler。



入门指南

下载安装

点击下载

软件界面介绍

- 菜单条
  - File 数据捕捉开关,会话保存&读取等操作
  - Edit 数据编辑
  - Rules 规则设置，断点控制等
  - Tools 通用配置（https，dns，host等），通用工具
  - View 界面显示布局
  - Help 帮助
- 工具条 包含一些快捷操作
- 会话列表
  - 会话类型分辨

  - 会话字段

- 命令区 Fiddler内置命令
  - select 选择命令

    > select image.

  - `?` text 搜索命令

    > 选择所有 URL 匹配问号后的字符的全部 session

  - `>` size 和 `<` size 命令

    > 请求大小过滤命令

  - `=` status 命令

    > 状态晒选命令

  - `@` host命令

    > 选择包含指定 HOST 的全部 HTTP请求。

  - Bpafter, Bps, bpv, bpm, bpu 批量断点控制命令

    > Bpafter xxx: 中断 URL 包含指定字符的全部 session 响应
    > Bps xxx: 中断 HTTP 响应状态为指定字符的全部 session 响应。
    > Bpv xxx: 中断指定请求方式的全部 session 响应
    > Bpm xxx: 中断指定请求方式的全部 session 响应，等同于bpv xxx
    > Bpu xxx:与bpafter类似。
    > 当这些命令没有加参数时，会清空所有设置了断点的HTTP请求。

  - 更多的其他命令可以参考Fiddler官网手册。
- Statistic 关于HTTP请求的性能和其他数据分析
- Inspectors 请求的详细说明
- AutoResponsor Fiddler 设置拦截某一请求，并重定向
  - 字符匹配

    > example 匹配 http://www.example.com 和 http://example.com.cn

  - 完全匹配

    > EXACT:

  - regex 开头（正则表达式匹配）

    > regex:(?insx).*(css|js|PHP)$ 表示匹配所有以css,js,php结尾的请求url

- Composer 自定义构建请求
- Filters 多维度的过滤规则
- Timeline

高级话题

使用Fiddler进行HTTP断点调试

- 通过设置断点可以

  1. 修改HTTP请求头信息。例如修改请求头的UA, Cookie, Referer 信息，通过"伪造"相应信息达到达到相应的目的（调试，模拟用户真实请求等）。
  2. 构造请求数据，突破表单的限制，随意提交数据。避免页面js和表单限制影响相关调试。
  3. 拦截响应数据，修改响应实体。

为什么以上方法是重要的？假设js前端程序员和服务器程序员是分工合作的，js程序员想要调试Ajax请求的功能，这样便不必等待服务器端程序员开发好所有接口之后再开始开发js端的ajax请求功能，因为通过"模拟"真实的服务器端的响应，便可以保证功能的正确性，而服务器端开发程序员，只要保证最终的响应是符合规定的即可。这大大简化了程序开发的效率，当然也降低了不同业务线程序员联调的难度。

- 设置断点
  Fiddler菜单栏 -> rules -> automatic Breakpoint -> 选择断点方式

  > before response。也就是发送请求之后，但是Fiddler代理中转之前，这时可以修改请求的数据。
  >
  > after response。也就是服务器响应之后，但是在Fiddler将响应中转给客户端之前。这时可以修改响应的结果。