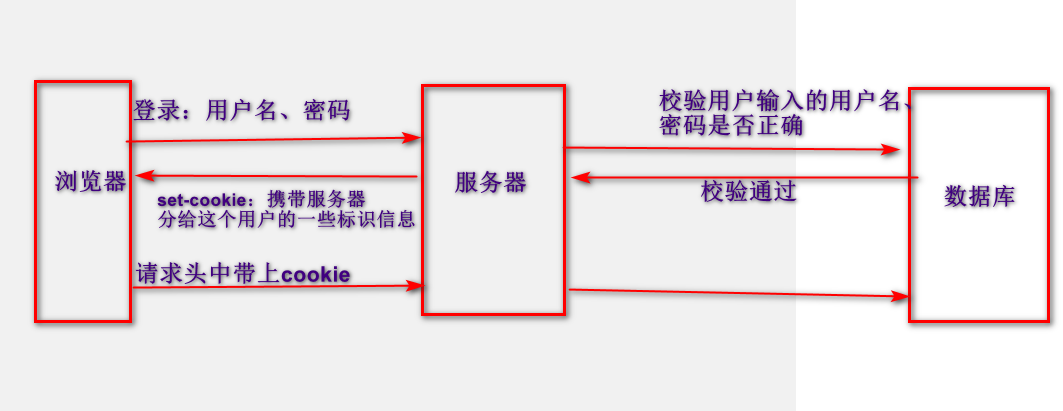
# 接口测试

1. 接口测试安排
   1. 接口测试概念（接口概念、HTTP、Fiddler工具）（1天）
   2. Python-requests用来做接口测试（1天）
   3. Python-pytest框架（1天）
   4. 自动化测试框架，requests+pytest（1.5天）
   5. 持续集成（0.5天）
2. 性能测试安排
   1. Jmeter (2天)
   2. 性能的概念(1天)
   3. 租车系统性能测试(1天)
3. 接口测试概念
   1. 接口按协议分
      1. WebService接口：基于SOAP Simple Object Access Protocol协议，接口之间交互的信息，是XML格式的。测试工具：SOAPUI
      2. Web Socket接口：基于TCP/UDP的，双向的，有客户端和服务端，连接一旦建立，客户端可以发送消息给服务器，服务器也可以发送消息给客户端。长连接的。
      3. HTTP接口：基于HTTP的，单向的，短连接。客户端发起请求，服务器响应。消息：Json、XML、HTML、文本。常用的是GET、POST
   2. 接口的作用和本质
      1. 作用：传输数据。
      2. 本质：数据的输入和输出。
      3. 比如百度的搜索接口，将用户要搜索的内容通过接口传给服务器，服务器查询后将结果返回给浏览器，浏览器解析并展示出来。
   3. 为什么做接口测试
      1. 系统复杂，前端测试效率低。测试人员更早的介入测试，发现的缺陷修复成本比较低。接口测试不等界面开发完成，就可以投入测试。
      2. 接口实现了前端与后端的分离，通过接口，绕过界面直接跟后台交互，发现界面发现不了的一些问题。
      3. 界面经常变化，但是接口一般比较稳定。接口的自动化成本小，收益高。
      4. 安全角度，黑客攻击的时候绕过前端，直接对后台服务器发起攻击，通过接口测试检查后台是否做了参数校验。
         1. 注册：手机号、用户名、密码、邮箱，手机号格式不正确，前端给出提示。
   4. 接口文档
      1. 一般情况下，开发会提供接口文档，接口文档中包括系统有哪些接口，接口的功能是什么，接口的入参有哪些，返回是什么。
      2. 开发进度紧张、开发流程不规范，可能没有接口文档。
         1. 抓包，看界面上操作某个功能时，调用了哪个接口。
   5. 接口测试点
      1. 原理：通过工具（jemter、postman）或者代码（python、java）模拟浏览器向服务器发送请求，服务器响应后，工具/代码对响应的报文进行检查，看是否与预期（接口文档）一致。
      2. 单接口的测试
         1. 接口的功能是否正确
         2. 接口参数
            1. 参数的边界值、等价类划分
            2. 参数的默认值，一些非必填的参数可能有默认值，检查默认值是否正确，是否合理。
            3. 参数异常：参数为空、多参数/少参数、错误的参数
         3. 接口返回值
            1. 正常与异常的返回值，每个都能覆盖到。
      3. 有关联接口的测试
         1. 查看订单的话，先登录，提交订单后，才能查看订单。
         2. 取现的接口
            1. 用户注册
            2. 用户登录
            3. 用户充值1000
            4. 用户取现1001，返回错误码
            5. 用户退出登录
            6. 删除注册用户
      4. 性能测试
      5. 安全测试
         1. 敏感信息（密码）是否加密了
         2. 参数是否做校验了
   6. 测试工具
      1. Postman：免费的工具，Chrome的插件，或者独立部署的版本。
      2. **Jmeter：免费的工具，接口测试/性能测试。**
      3. httprequester：火狐浏览器带的接口测试工具。
      4. SOAPUI ：WebService接口测试工具，基于SOAP协议的。
      5. JAVA+httpclient：java代码实现接口自动化测试。
      6. Robotframework+httplibrary：自动化测试框架，Robotframework是一款测试工具，表格的方式来写测试用例的。
      7. **Python+Requests：python代码实现接口自动化测试。**
4. HTTP协议
   1. HTTP协议（HyperText Transfer Protocol，超文本传输协议）
   2. 特点：
      1. 无连接：短连接，一次连接只处理一个请求，处理完后，连接断开。
      2. 无状态：没有记忆的功能，不记得之前发过的内容。
      3. 媒体独立：可以发送任意类型的数据，文本、视频、音频、图片……
   3. HTTP请求
      1. 请求行：请求方法(get/post) URL(<https://www.baidu.com/s)> HTTP协议版本(HTTP1.1)
      2. 请求头，key:value
         1. Host:[www.baidu.com](http://www.baidu.com/)
         2. **User-Agent**：浏览器的版本号、操作系统版本号/名称
         3. Accpet：接收什么类型的数据
         4. Referer：请求是从哪个URL进来的。比如百度搜索淘宝，点击链接进入淘宝，请求的referer指向百度。
         5. Cache-Control：响应的内容在客户端缓存不缓存，缓存多长时间。
         6. Accept-Encoding：客户端接受什么编码格式。
         7. **Content-Type**: 内容的类型，application/x-www-form-urlencoded表单格式。application/json、text/plain、image/png、image/jpg
      3. 空行：表示请求头结束
      4. 请求数据：post方法使用的
   4. HTTP请求方法
      1. **GET：**查询类的操作，使用GET方法。
      2. **POST：**增删改查，使用POST方法。
      3. CONNECT
      4. HEAD
      5. PUT
      6. DELETE
      7. TRACE
   5. GET与POST的差别
      1. get方法，参数是拼接到URL中的，?参数1=值1&参数2=值2，百度搜索的接口。post方法，参数是放到请求数据中的。
      2. get方法，参数的长度是有限制的（不同的浏览器版本，对长度的限制不同。），post方法，参数的长度没有限制。
      3. get不安全，post安全
         1. get方法，可以收藏为书签，但是post方法不能收藏为书签。
         2. get方法，参数保存在浏览器的历史记录中，但是post的参数不会保存在浏览器历史记录。
         3. get方法，参数拼接到URL中，参数是暴露的。涉及敏感信息交互的，不能使用get方法。
      4. get方法，浏览器的后退按钮不影响，post会重新提交请求。
   6. HTTP响应
      1. 状态行：协议版本、状态码、状态描述。常见：200 OK，404
      2. 响应头部：key：value格式
      3. 空行：表示响应头结束
      4. 响应数据
   7. 响应头
      1. Cache-Control：控制响应内容的缓存。
      2. ETag：资源的版本
      3. Location
      4. **Set-Cookie**：认证使用的。
   8. 认证和授权
   9. Cookie



* + 1. POST <https://www.bagevent.com/user/login> HTTP/1.1 响应中的set-cookie
    2. GET <https://www.bagevent.com/account/dashboard> HTTP/1.1 请求中带了上一步的cookie信息（请求头的cookie）。
    3. Cookie的特点（类似身份证）：
       1. 服务器给浏览器的一个小的文本
       2. Cookie是存储在浏览器端的。
       3. Cookie中包含名字，值，路径/域，过期时间。
  1. Session的特点
     1. 记录在服务器上的（内存中、或者数据库、或者文件）
     2. 相当于在服务器上有一个客户档案表，如果客户访问时，检查档案库里有没有对应的用户。
  2. Session与Cookie的区别
     1. 存放的位置不同，Session存在服务器端，Cookie存在客户端的。
     2. 安全性：Session更安全。
     3. 有效期不同：Cookie可以设置长时间保持，Session一般有效期比较短。
     4. 对服务器的压力：比如2W个用户，Session机制的话，2W个用户信息保存服务器上，占用服务器的资源。
  3. Token 密码加密方式md5
     1. 登录的请求，携带用户名密码



* + 1. 登录时服务器的响应中返回token

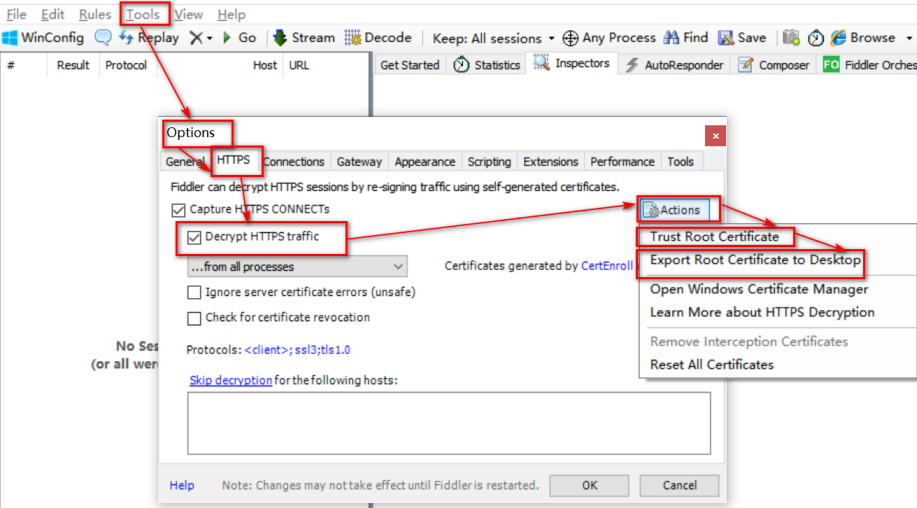


* + 1. 再次发送请求时，携带token（在cookie信息中）



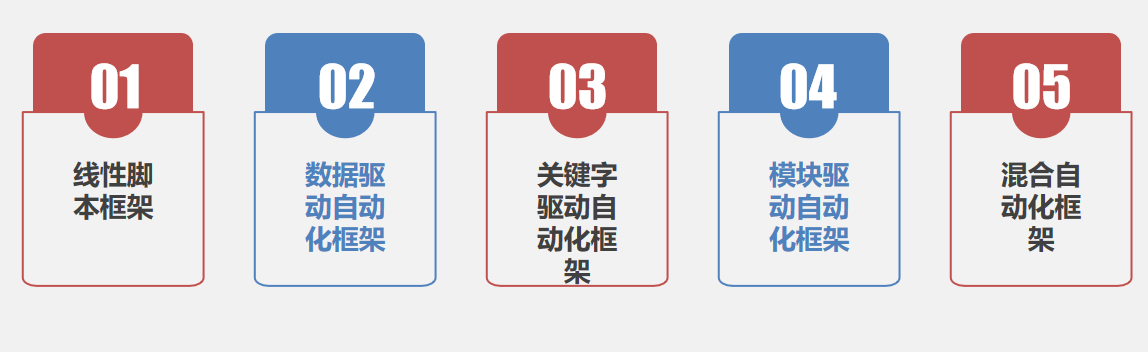
* 1. HTTP与HTTPs的差别
     1. HTTP与HTTPs的差别默认端口号不一样。
     2. HTTP端口号是80，HTTPs端口号是443。
     3. HTTPs的技术门槛较高。
     4. 证书需要年费。
     5. HTTPs加重了服务端的负担，降低了用户的访问速度。
  2. JSON与字典的区别
     1. Json的是数据格式，字典是Python是数据类型，内置函数。
     2. 都是key：value格式的
     3. 空值：字典中空值是None、Json中空值是null
     4. JSON引号只能用双引号，但是字典中单引号、双引号都可以。
     5. Python：
        1. 字典转JSON：json.dumps(dict)
        2. JSON转字典：json.loads(str)

1. Fiddler
   1. Fiddler配置HTTPs



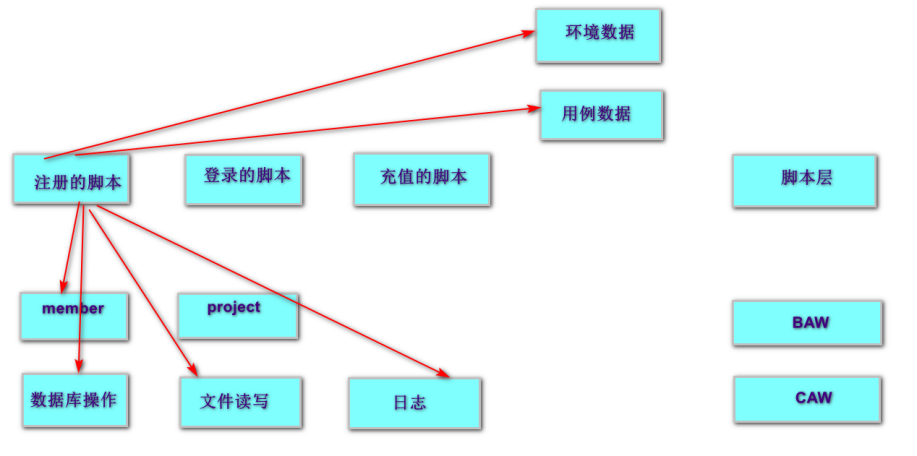
* + 1. 设置失败时cmd命令行，进入fiddler的安装目录，然后
    2. 执行以下命令makecert.exe -r -ss my -n "CN=DO\_NOT\_TRUST\_FiddlerRoot, O=DO\_NOT\_TRUST, OU=Created by [http://www.fiddler2.com](http://www.fiddler2.com/)" -sky signature -eku 1.3.6.1.5.5.7.3.1 -h 1 -cy authority -a sha1 -m 120 -b 09/11/2020
    3. 注意：这个命令最后的日期 09/11/2020 一定要大于现在的日期 否则创建证书是过期的
    4. 仍然设置不成功时，参考<https://www.cnblogs.com/joshua317/p/8670923.html>
  1. Fiddler设置断点
     1. 请求加断点，篡改请求的数据，比如充值金融50W改成100W。
        1. 增加断点：bpu <https://www.baidu.com/s>
        2. 取消断点：bpu
     2. 对Response加断点，篡改响应信息
        1. 增加断点：bpafter <https://www.baidu.com/s>
        2. 取消断点：bpafter

1. PyTest
   1. 安装：pip install pytest -i <http://pypi.douban.com/simple/> --trusted-host [pypi.douban.com](http://pypi.douban.com)
      1. pytest 测试框架
      2. pytest-html 测试报告
      3. pytest-xdist 多线程运行测试用例
      4. pytest-rerunfailures 执行失败时重试
      5. pytest-check 复合断言插件
      6. pymysql 连接数据库
      7. pyyaml 操作yaml文件，常用yaml来描述测试数据
2. 自动化测试框架



* 1. 线性脚本（第一版的注册、登录的代码）
     1. 录制-回放模式
     2. 没有任何封装
     3. 缺点
        1. 很多重复的代码
        2. 不方便维护与修改
     4. 优点
        1. 写起来简单
  2. 数据驱动
     1. 测试数据与测试逻辑分离
     2. 测试数据可以通过外部来输入，比如csv文件、yaml文件、xml文件等
     3. 优点
        1. 方便维护
        2. 灵活
     4. 缺点
        1. 复杂
  3. 关键字驱动
     1. 典型框架：Selenium
     2. 把每个操作封装成函数，开发脚本时，直接调用这些封装好的函数。
        1. 打开浏览器
        2. 最大化浏览器
        3. 点击链接
        4. 输入参数
        5. 关闭浏览器
     3. 缺点
        1. 成本高
        2. 难度大
     4. 一般是数据驱动和关键字驱动结合起来使用
  4. 模块化自动化
     1. 按系统的功能模块划分
  5. 混合自动化
     1. 集成了其他框架的优点。

1. 框架的好处
   1. 可扩展性比较高
   2. 代码容易维护和修改
   3. 提高脚本的通过率、稳定性（减少环境对脚本的影响）
   4. 提高脚本的执行效率
   5. 提供测试报告
   6. 发送邮件
2. 5年以内，在别人搭建好的框架基础上增量写代码的能力。
3. 5年以上，具备搭建框架的能力。
4. 数据分层
   1. 测试环境数据：http://jy001:8080，经常变化的。（ini配置文件）
   2. 常量：接口地址，接口被设计出来后，一般不会变化的数据。/futureloan/mvc/api/member/register （代码）
   3. 测试数据：测试用例，输入的参数以及预期结果。（yaml）
5. 代码分层
   1. 一个项目实现自动化，自动化可能包含哪些内容？
      1. 发送邮件、打印日志、界面通用操作、数据库（查、删）
      2. 业务模块、文件操作（csv、excel、yaml文件读写）
   2. 金融项目、租车系统，两个项目通用的有什么？
      1. /futureloan/mvc/api/member/login mobilephone\password
      2. /carrental/car/login.action username\password
      3. 发送邮件、打印日志、数据库操作、文件操作
   3. 金融项目、租车系统，两个项目不同的地方
      1. 业务模块
   4. 代码分层



* + 1. CAW Common Action Word：通用AW层，项目之间可以共享。
    2. BAW Business Action Word：业务AW层，业务特有的，按模块管理方法。
    3. 脚本层：使用CAW、BAW中封装的方法构造测试脚本。
    4. 数据层：只有脚本访问数据，其他层数据无关。
  1. 实现思路
     1. 建好目录
     2. 写好环境文件，封装一个方法读取环境文件
     3. 写好测试用例数据文件，封装一个方法读取数据文件
     4. 封装业务层的方法
     5. ……

1. 注册的测试逻辑/测试步骤
   1. 注册失败的测试步骤
      1. 准备测试数据
      2. 下发请求
      3. 检查请求的返回结果与预期的结果是否一致。
   2. 注册成功的测试步骤
      1. 准备测试数据
      2. 下发请求
      3. 检查结果
         1. 检查请求的返回结果与预期的结果是否一致。
         2. 检查用户实际有没有注册成功
            1. 数据库查询有对应的记录
            2. 调用查询的接口，返回的结果中有注册的用户
            3. 登录成功
      4. 清理环境，删除注册的用户
         1. 调删除的接口
         2. 数据库中删除对应的记录
   3. 重复注册的测试步骤
      1. 准备测试数据
      2. 下发注册请求
      3. 检查注册成功
      4. 再次下发注册的请求
      5. 检查注册失败
      6. 清理环境，删除注册的用户
2. 自动化原则
   1. 执行脚本前，系统是什么样的环境，执行完脚本后，系统还恢复成什么样的环境。
   2. 脚本执行过程中在系统中增加的数据，执行完脚本要清理掉。