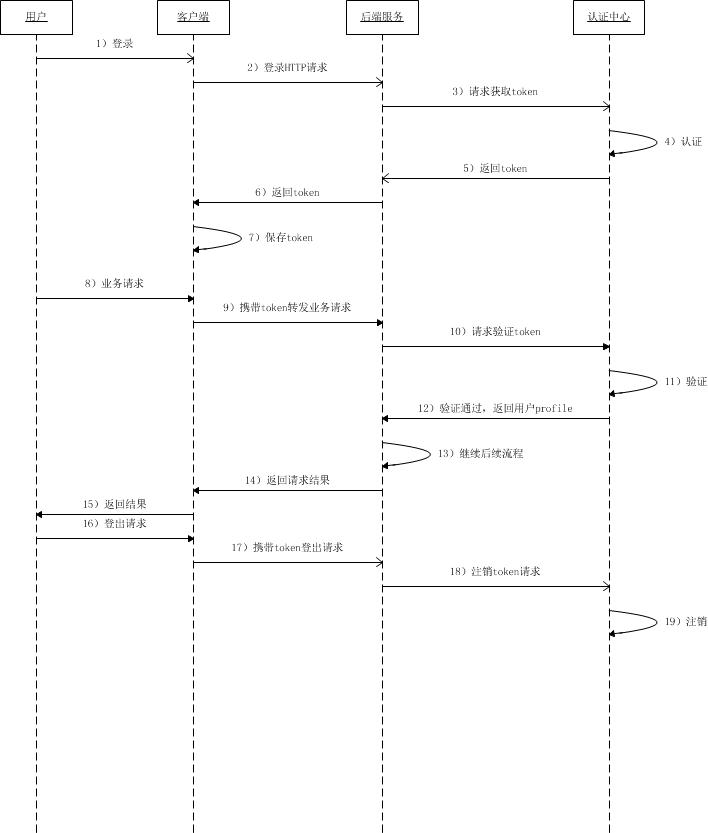
Token认证机制

# Token认证的好处

* **支持跨域访问**: Cookie是不允许垮域访问的，这一点对Token机制是不存在的，前提是传输的用户认证信息通过HTTP头传输.
* **无状态**:Token机制在服务端不需要存储session信息，因为Token 自身包含了所有登录用户的信息，只需要在客户端的cookie或本地介质存储状态信息.
* **更适用CDN**: 可以通过内容分发网络请求你服务端的所有资料（如：javascript，HTML,图片等），而你的服务端只要提供API即可.
* **去耦**: 不需要绑定到一个特定的身份验证方案。Token可以在任何地方生成，只要在你的API被调用的时候，你可以进行Token生成调用即可.
* **更适用于移动应用**: 当你的客户端是一个原生平台（iOS, Android，Windows 8等）时，Cookie是不被支持的（你需要通过Cookie容器进行处理），这时采用Token认证机制就会简单得多。
* **CSRF**:因为不再依赖于Cookie，所以你就不需要考虑对CSRF（跨站请求伪造）的防范。
* **不需要为登录页面做特殊处理**: 如果你使用Protractor 做功能测试的时候，不再需要为登录页面做特殊处理.

# Token机制的处理过程概要描述

1. 用户在客户端输入用户名/密码；
2. 客户端将请求（HTTP）转发给后端服务；
3. 后端服务接收到请求后，解析客户端的IP地址，连同用户名/密码发送给认证中心进行认证；
4. 认证中心负责对用户名/密码进行认证，认证通过后，生成token，并将该token与客户端的IP地址绑定；
5. 认证中心返回token给后端服务；
6. 后端服务将认证中心生成的token返回给客户端；
7. 客户端保存token，以作为后续请求的认证凭据；
8. 客户端发起新的业务请求，同时在请求头部带上token；
9. 后端服务接收业务请求时，首先获取请求头部的token，解析请求的IP地址，向认证中心确认该token是否有效。
10. 如果认证中心验证该token有效，则返回用户profile给后端服务；
11. 后端服务获取认证中心的验证结果，继续执行该业务请求的后续动作。如果验证不通过，则跳转到登录界面，请求新的token。



# Token的内容

{用户名|客户端IP地址|时间戳}

# Token加密算法

对称加密 默认采用DES

Base64

示例：/NQ5ultMnf9deFRuHmyGk68L/nv9fTwa4g+4lpY0pTYOa/q1I5HTxw==

# Token验证规则

1）认证中心接收token后，首先检查是否以前有生成过该token；

2）验证该token是否过期（有效时间可以为30分钟，系统可配）；

3）解密该token，检查其中的IP地址是否与客户端请求的IP地址相匹配。

# 认证中心拓扑图

