## 九、web

### 1、servlet的实现以及其中的方法；

实现包Servlet的方式：

1.实现Servlet接口

2.继承GenericServlet类

\*优化方案：

1.对Servlet接口提供适配器的操作

2.将getServletConfig方法实现了，复写init()方法时复写无参的

3. 实现了ServletConfig方法接口，获取ServletContext对象

//获取getServletContext();

ServletContext se = this.getServletConfig().getServletContext();

//优化获取

ServletContext sc = this.getServletContext();

3.继承HttpServlet类

对Http协议的一种封装

### 2、token生成的原理；

**一：token的优势**

**1.无状态、可扩展**

在客户端存储的Tokens是无状态的，并且能够被扩展。基于这种无状态和不存储Session信息，负载负载均衡器能够将用户信息从一个服务传到其他服务器上。

如果我们将已验证的用户的信息保存在Session中，则每次请求都需要用户向已验证的服务器发送验证信息(称为Session亲和性)。用户量大时，可能会造成一些拥堵。但是不要着急。使用tokens之后这些问题都迎刃而解，因为tokens自己hold住了用户的验证信息。

**2.安全性**

请求中发送token而不再是发送cookie能够防止CSRF(跨站请求伪造)。即使在客户端使用cookie存储token，cookie也仅仅是一个存储机制而不是用于认证。不将信息存储在Session中，让我们少了对session操作。

token是有时效的，一段时间之后用户需要重新验证。我们也不一定需要等到token自动失效，token有撤回的操作，通过token revocataion可以使一个特定的token或是一组有相同认证的token无效。

**3.可扩展性**

Tokens能够创建与其它程序共享权限的程序。例如，能将一个随便的社交帐号和自己的大号(Fackbook或是Twitter)联系起来。当通过服务登录Twitter(我们将这个过程Buffer)时，我们可以将这些Buffer附到Twitter的数据流上(we are allowing Buffer to post to our Twitter stream)。

使用tokens时，可以提供可选的权限给第三方应用程序。当用户想让另一个应用程序访问它们的数据，我们可以通过建立自己的API，得出特殊权限的tokens。

**4.多平台跨域**

我们提前先来谈论一下CORS(跨域资源共享)，对应用程序和服务进行扩展的时候，需要介入各种各种的设备和应用程序。

Having our API just serve data, we can also make the design choice to serve assets from a CDN. This eliminates the issues that CORS brings up after we set a quick header configuration for our application.

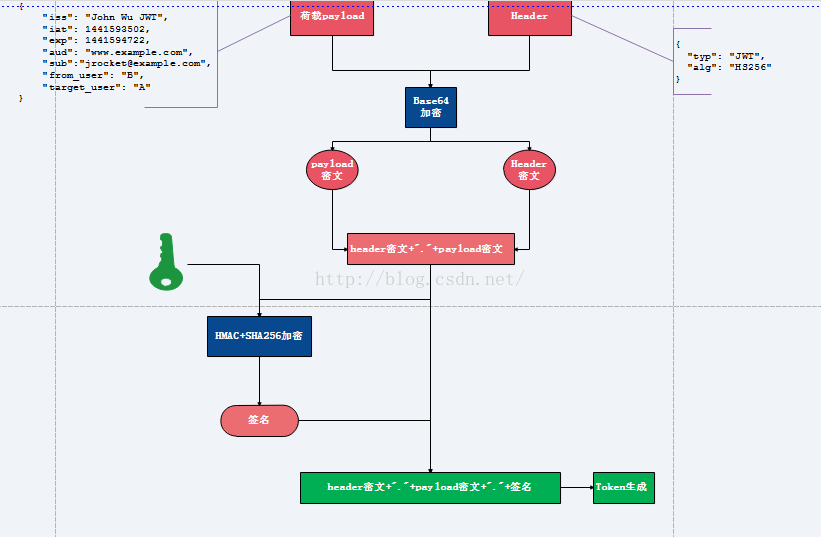
只要用户有一个通过了验证的token，数据和资源就能够在任何域上被请求到。

Access-Control-Allow-Origin: \*

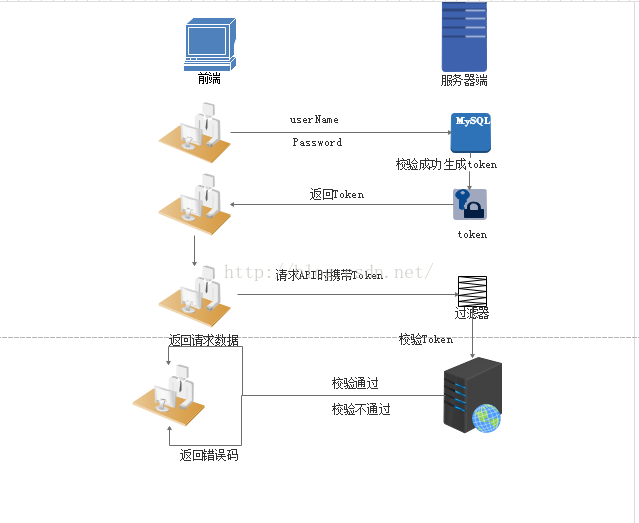
**5.基于标准**

创建token的时候，你可以设定一些选项。我们在后续的文章中会进行更加详尽的描述，但是标准的用法会在JSON Web Tokens体现。

二.Token的原理



三，实现思路



## 十、关于前端

### 1、闭包的理解；

<https://blog.csdn.net/qq_21132509/article/details/80694517>

### 2、js函数的理解；

<https://blog.csdn.net/lzpllli/article/details/79696553>

### 3、如何优化前端？

<https://blog.csdn.net/asdf8968/article/details/82952709>