**JWT(JSON Web Token)實作分享**

* **JWT：IETF - RFC7519 (加強安全性JWS、JWE)**
* **常見JWT Library 有jjwt、[jose4j](https://bitbucket.org/b_c/jose4j/wiki/browse/)、soteria、[java-jwt](https://github.com/auth0/java-jwt)、nimbus.**
* **應用Java EE Security 1.0 整合JWT (Payara Server 4.1.2.174)**
* **應用JAX-RS 的NameBinding、ContainerRequestFilter整合JWT**

## Why the need for Web Tokens?

|  |  |
| --- | --- |
| Server-Based Authentication  Server-Based Authentication | Token-Based Authentication  Token-Based Authentication |

### Drawbacks of Server-Based Authentication

* **Hard to scale -** distributed system
* **Cross-origin request sharing (CORS) -** by default, HTTP requests don’t include cookies on cross-origin requests.
* **Coupling with the web framework**

### Advantages of Token-Based Authentication

* **Stateless, easier to scale**
* **Reusability**
* **Security:** Since we are not using cookies, we don’t have to protect against [cross-site request forgery](https://en.wikipedia.org/wiki/Cross-site_request_forgery) (CSRF) attacks. We should still encrypt our tokens using **JWE** if we have to put any sensitive information in them, and transmit our tokens over **HTTPS** to prevent man-in-the-middle attacks.
* **Performance**
* It is important to understand that the purpose of using JWT is **NOT** to hide or obscure data in any way. The reason why JWT are used is to prove that the sent data was actually created by an authentic source.

很重要的一點，使用JWT的目的不是以任何方式隱藏或模糊數據。 使用JWT的原因是為了證明發送的數據實際上是由真實來源創建的。

* *Since JWT are signed and encoded only, and since JWT are not encrypted, JWT do not guarantee any security for sensitive data.*

## Structure of a JSON Web Token

A JWT is represented as a sequence of [base64url](https://en.wikipedia.org/wiki/Base64) encoded values that are separated by period characters.



JSON Web Token example:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ0b3B0YWwuY29tIiwiZXhwIjoxNDI2NDIwODAwLCJodHRwOi8vdG9wdGFsLmNvbS9qd3RfY2xhaW1zL2lzX2FkbWluIjp0cnVlLCJjb21wYW55IjoiVG9wdGFsIiwiYXdlc29tZSI6dHJ1ZX0.yRQYnWzskCZUxPwaQupWkiUzKELZ49eM7oWxAQK\_ZXw

### Header

This JWT Header declares that the encoded object is a JSON Web Token, and that it is signed using the **HMAC SHA-256** algorithm.

### Payload (Claims)

The claim contains the information we want to transmit, and that the server can use to properly handle authentication. There are multiple claims we can provide; these include registered claim names, public claim names and private claim names.

* **Registered Claims**
* **iss**: The issuer of the token
* **sub**: The subject of the token
* **aud**: The audience of the token
* **exp**: Token expiration time defined in Unix time
* **nbf**: “Not before” time that identifies the time before which the JWT must not be accepted for processing
* **iat**: “Issued at” time, in Unix time, at which the token was issued
* **jti**: JWT ID claim provides a unique identifier for the JWT
* **Public Claims**

Public claims need to have collision-resistant names. By making the name a [URI](https://en.wikipedia.org/wiki/Uniform_resource_identifier) or [URN](https://en.wikipedia.org/wiki/Uniform_resource_name) naming collisions are avoided for JWTs where the sender and receiver are not part of a closed network.

An example of a public claim name could be: https://www.toptal.com/jwt\_claims/is\_admin, and the best practice is to place a file at that location describing the claim so that it can be dereferenced for documentation.

* **Private Claims**

Private claim-names may be used in places where JWTs are only exchanged in a closed environment between known systems, such as inside an enterprise. These are claims that we can define ourselves, like user IDs, user roles, or any other information.

It is important to note that we want to keep a web token as small as possible, so use only necessary data inside public and private claims.

**Example Payload**

{

“iss”: “toptal.com”,

“exp”: 1426420800,

“https://www.toptal.com/jwt\_claims/is\_admin”: true,

“company”: “Toptal”,

“awesome”: true

}

This example payload has **two registered claims**, **one public claim** and **two private claims**. Once it is base64 encoded, we have the second part of our JWT.

## Signature

The JWT standard follows the JSON Web Signature (**JWS**) specification to generate the final signed token. It is generated by combining the encoded JWT Header and the encoded JWT Payload, and signing it using a strong encryption algorithm, such as HMAC SHA-256. The signature’s secret key is held by the server so it will be able to verify existing tokens and sign new ones.

$encodedContent = base64UrlEncode(header) + “.” + base64UrlEncode(payload);

$signature = hashHmacSHA256($encodedContent);

## Security and Encryption with JWT

It is critical to use TLS/SSL in conjunction with JWT, to prevent man-in-the-middle attacks. In most cases, this will be sufficient to encrypt the JWT payload if it contains sensitive information. However, if we want to add an additional layer of protection, we can encrypt the JWT payload itself using the [JSON Web Encryption](http://tools.ietf.org/html/draft-ietf-jose-json-web-encryption-40) (**JWE**) specification.

* **應用Java EE Security 1.0 整合JWT**
* **提供兩種模式：**
* 一般模式 - header傳遞JWT、@RequestScoped的驗證程式
* RememberMe 模式 - cookie傳遞JWT、@ApplicationScoped的驗證程式、Token有效期限大多設定比[一般模式]長很多。
* **相關重要 Class：**
* javax.security.enterprise.authentication.mechanism.http.HttpAuthenticationMechanism

🡺 建立Class繼承此介面，檢查各Request的認證 (@RequestScoped、@RememberMe)

* javax.security.enterprise.authentication.mechanism.http.RememberMe

🡺 標註上述 Class 是否支援 RememberMe 機制，及設定相關參數(Token期限、判斷啟用此機制的Method)

* javax.security.enterprise.identitystore.IdentityStore

🡺 建立Class繼承此介面，實作登入帳密驗證(與授權)的商業邏輯部分(如串接DB、AD或CAS、OAuth…)。

* javax.security.enterprise.identitystore.IdentityStoreHandler

🡺 透過 Inject 此介面，可使用**所有繼承IdentityStore的Class**的Method

* javax.security.enterprise.identitystore.RememberMeIdentityStore

🡺 建立Class繼承此介面，實作 RememberMe Token驗證 (@ApplicationScoped)

* javax.security.enterprise.credential.Credential

🡺 建立Class繼承此介面，存放認證(與授權)成功後的資訊。

* **範例程式說明**
* **應用JAX-RS 的NameBinding、ContainerRequestFilter整合JWT**
* **範例程式說明**