# Guida ai diversi tipi di autenticazione

## Autenticazione con Utente Anonimo

Per accedere tramite utente anonimo è necessario impostare, lato webclient, determinate porzioni di codice che attivino il riconoscimento di tale utente da parte del sistema di autenticazione di Geoweb.

#### Procedure di configurazione

In Geoweb, il sistema di riconoscimento di un utente anonimo è già configurato, sebbene sia disabilitato di default. Perciò, per abilitare questo tipo di autenticazione, è necessario semplicemente commentare e decommentare alcune porzioni di codice all'interno di uno specifico file.

I file che contengono codice relativo agli utenti anonimi sono i seguenti:

- spring-security.xml
- configuration.properties
- GwAnonymousAccessService.java

#### Commentare e decommentare il codice

Il primo passo è il più importante: *spring-security.xml* è il file in cui viene configurata la gestione degli accessi e delle autenticazioni. Al suo interno basterà cercare il tag `@@@ANONYMOUS_USER@@@` e seguire quanto riportato nei commenti a seguito di esso, poiché questi contengono tutte le informazioni necessarie su quali parti di codice è necessario commentare e quali decommentare.

**NOTA:** Il suddetto tag può ricorrere più volte all'interno del documento. Si consiglia di assicurarsi di aver controllato ed eseguito tutte le procedure riportate nei commenti sotto ognuno di questi tag.

```
Come sarà possibile notare nei suddetti commenti,
l'abilitazione dell'utente anonimo richiede che venga fornito anche il nome
di un gruppo.
Tale nome è impostato di default come 'ROLE_ANONYMOUS' ma può essere
sostituito con il nome di un altro gruppo,
scelto tra quelli già presenti o semplicemente creato ad-hoc per l'utente
anonimo (scelta consigliata).
```

#### Abilitare o disabilitare servizi di interoperabilità per autenticazione anonima

All'interno del file *configuration.properties*, sotto la sezione *ANONYMOUS USER*, è presente una proprietà chiamata `anonymousUser.interoperabilityServicesAllowed`. Tramite questa proprietà è possibile abilitare o disabilitare l'accesso ai servizi di interoperabilità da parte di un utente anonimo. Ciò avviene attraverso un controllo effettuato dalla funzione `getInteroperabilityServicesAllowed()` fornito dal servizio *GwAnonymousAccessService.java*.

**Versioni di Geoweb precedenti alla 4.4.0**

Nelle versioni precedenti di Geoweb, l'abilitazione dell'utente anonimo può risultare più complessa a causa dell'assenza del suddetto tag. Nonostante ciò, le informazioni necessarie sono le stesse: di seguito saranno elencate le porzioni di codice che devono essere decommentate o (se assenti) inserite all'interno del file *spring-security.xml* per abilitare l'autenticazione anonima.

```
<intercept-url pattern="/**" access="isAuthenticated() or isAnonymous()" />
```

**NOTA:** Questo tag è già presente, scritto e decommentato, all'interno del codice, ma è privo della parte `or isAnonymous()`. È indispensabile assicurarsi che quella proprietà sia presente e che, in caso si intenda disabilitare questo tipo di autenticazione, venga commentata.

```
<anonymous username="" granted-authority="ROLE_ANONYMOUS" />
```

**NOTA:** È qui che viene definito il nome assegnato all'utente anonimo (di default viene assegnata una stringa vuota, visualizzata poi nel framework con il nome `anonymousUser`) ed il nome del gruppo a cui tale utente avrà accesso (vedi sopra).

# Autenticazione SAML

## CREAZIONE KEYSTORE SAML

Passaggi

1. Sono necessari i file:
   - **saml_gw_sp.xml** (certificato prodotto da *Geoweb*)
   - **saml_idp.xml** (certificato prodotto da *identity provider*).
2. Da questi file si possono isolare i certificati (da salvare in file .crt) prendendo ciò che si trova dentro al tag *<ds:X509Certificate>*
3. Posizionarsi col prompt dei comandi dentro alla cartella dove si trovano questi file
4. ```
   keytool -genkey -keyalg RSA -sigalg SHA256withRSA -alias bu-test -
   keystore gw-key.jks -validity 36500 -keysize 2048
   ```

   (L'alias deve corrispondere a quanto definito nel configuration) La procedura richiederà anche l'inserimento della password del keystore (anche questo sta sul configuration) e l'inserimento del nome dell'organizzazione e località (*Geoweb Italia S.r.L.* e *IT*)

5. Dopo aver generato il file .jks con il comando precedente si deve eseguire il comando

   ```
   keytool -export -keystore gw-key.jks -rfc -alias bu-test -file cert.txt
   ```

   per esportare il certificato cert.txt che va salvato in un certificato pubblico (su fiera ho gw-key-cert.public)
   ```

# Autenticazione con LDAP

Per accedere tramite LDAP è necessario, similmente all'accesso tramite utente anonimo, impostare correttamente determinate porzioni di codice all'interno del file *spring-security.xml*, principalmente decommentandole.

**NOTA:** È fondamentale sapere che Geoweb funziona solo con la configurazione LDAP Active Directory!

### Ottenere le informazioni necessarie

La prima cosa da fare per configurare l'autenticazione tramite LDAP è ottenere i dati necessari allo stesso. Questi devono essere forniti dal cliente e comprendono un indirizzo LDAP ed eventuali numeri di porta configurati dal cliente, nel caso in cui egli non stia utilizzando le porte standard.

### Inserire le informazioni nella configurazione

Una volta ottenute le informazioni, queste andranno inserite nell'apposita porzione di codice all'interno del file *spring-security.xml* di cui viene accennato sopra.
Il codice (da decommentare) avrà la seguente struttura:

```
<beans:bean id="adAuthenticationProvider"
class="org.springframework.security.ldap.authentication.ad.ActiveDirectoryLd
apAuthenticationProvider">
      //di seguito va inserito il dato relativo all'ambiente dove LDAP è
stato impostato (esempio: "geoweb.local")
    <beans:constructor-arg value="{ambiente LDAP}" />
      //di seguito va inserito l'indirizzo IP relativo all'ambiente di cui
sopra
    <beans:constructor-arg value="ldap://{indirizzo IP}" />
    <beans:property name="convertSubErrorCodesToExceptions" value="true" />
</beans:bean>
```

**NOTA:** Nel caso in cui l'utente non stia utilizzando le porte standard, questo è il punto in cui vanno inseriti gli eventuali numeri di porta da lui forniti.

### Configurare riferimento per authentication manager

Una volta inseriti i dati, un'altra porzione di codice di cui occuparsi è quella relativa all'*authentication manager*, tramite il quale il servizio di autenticazione troverà il riferimento al LDAP che è stato appena impostato.
Il codice (da decommentare) avrà la seguente struttura:

```
<authentication-manager alias="authenticationManager">
  <authentication-provider ref="adAuthenticationProvider"></authentication-
provider>
```

```
</authentication-manager>
```

**NOTA:** La stringa che segue il parametro “ref”, in questo caso 'adAuthenticationProvider', fa riferimento al parametro ID del bean nel codice precedente.

# Autenticazione con OAuth2

L'accesso tramite OAuth2 riguarda chiamate provenienti dall'esterno, messe a disposizione da Geoweb (come, ad esempio, i servizi di interoperabilità). Gli sviluppatori dovranno solo fornire all'utente determinati dati necessari ad effettuare le chiamate.

```
È possibile trovare una spiegazione approfondita del funzionamento di tali
chiamate
all'interno della sezione di questa Wiki dedicata agli utenti.
```

### Recuperare dati di accesso

È importante sapere che, in Geoweb, le chiamate tramite OAuth2 funzionano tramite un sistema di token: utilizzando determinati parametri, l'utente può effettuare una richiesta per un token, il quale verrà utilizzato per aggiornare un ticket di accesso, necessario per ottenere l'autorizzazione ai servizi esterni di Geoweb.

I parametri necessari per richiedere il token sono:

- grant_type (fisso, il cui valore è 'password')
- client_id & client_secret (di questi valori parleremo successivamente in questo paragrafo)
- username & password (relativi ad un account specifico fornito da Geoweb)

**NOTA:** Per questioni di sicurezza, è sempre consigliabile fornire credenziali (username e password) relative ad un account di Geoweb espressamente creato per questo scopo.

### client_id e client_secret

*client_id* e *client_secret* sono due valori specifici (paragonabili ad una coppia *username/password*) definiti all'interno del file *spring-security.xml*. All'interno di questo file bisognerà cercare il seguente tag:

```
<oauth:client-details-service id="clientDetails">
```

All'interno di esso si troverà una o più strutture che definiscono questi specifici valori:

```
<oauth:client client-id="restappgeoweb"
  authorized-grant-
types="password,authorization_code,refresh_token,implicit"
  secret="restappgeoweb" authorities="ROLE_USER" />
```

**NOTA**: I campi client-id e secret rappresentano rispettivamente i valori client_id e client_secret necessari all'utente per compilare la richiesta del token.

Ad ogni modo, l'utente dovrebbe essere in grado di ottenere questi dati tramite le informazioni fornite dalla sezione per gli utenti di questa Wiki.

# Autenticazione con Header SM_USER

(AKA EXTERNAL PRE AUTHENTICATION SERVICE) L'autenticazione con l'header SM_USER avviene sfruttando come credenziali di accesso dei dati preimpostati e specificatamente codificati per questo scopo. Viene tipicamente utilizzata per autorizzare chiamate rivolte a Web Service, che possono di volta in volta essere integrati nei vari progetti di commessa. Ma funziona per tutte le chiamate. L'utilizzo ne è scoraggiato, in favore di altri tipi di autenticazine come OAuth2. Viene mantenuta in quei casi dove o ci siano difficoltà nell'implementare altri tipi di autenticazione, o in fase di develop, per poter testare velocemente i web services, bypassando possibili problemi derivanti dall'autenticazione, in ottica di affrontarli piu avanti.

### Creazione credenziali

Il primo passo è quello di creare le credenziali relative all'utente SM_USER. Esse devono essere dichiarate all'interno del file *configuration.properties* e sono le proprità `preAuthUser` e `preAuthKey`. Per convenzione sotto la sezione `PRE AUTHENTICATION CREDENTIALS (optional)` del macro blocco `A U T H E N T I C A T I O N`

```
#####################################################################
#   A U T H E N T I C A T I O N
#####################################################################

#PRE AUTHENTICATION CREDENTIALS (optional)
#Next two parameters, both optional, work in conjunction and maybe used to
create a simplified authentication mechanism (if oauth2 is not possible, or
not jet implemented, for example)
#Every call to server will be considered allowed if contains in its headers
section a parameter called SM_USER properly populated.
#This parameter should be computed making the sha512 of the string resulting
from the concatenation of preAuthUser and preAuthKey parameter, with a |
character in the middle.
#Here the pseudo code:
#
#   var SM_USER = SHA512(preAuthUser+"|"+preAuthKey)
#
#   Ex:
#
#       SM_USER:
2ca0e5a3633f7c8306505d3c7edcdaac29c93ae689e0b1182c3da4bfdc763758745e76849260
42b840d6beb193ffd4e11fa9d1d73d0bec43f42348cca4f2aedd
#
preAuthUser=ws_test_user
```

```
preAuthKey=ws_test_key
```

### Codificare le credenziali

Una volta ottenute queste credenziali, sarà necessario codificare una concatenazione delle due stringhe, preAuthUser e preAuthKey, come nella formula seguente:

```
    SM_USER = SHA512(preAuthUser+"|"+preAuthKey)
```

Nel nostro esempio, il risultato sarebbe il seguente:

```
    SM_USER = SHA512('ws_test_user|ws_test_key')
```

```
    SM_USER =
'8f8ef642349a8b961cb588b0a92901676970415f60a9d6e6961346c0969390c4d0afc50bd14
4207106df2e57707dddbe81f3684b267abd706034495f06398495'
```

Per codificare una stringa con SHA512, si può utilizzare uno dei tanti tool online free.

### Utilizzare il token

Una volta ottenuta la stringa bisogna utlizzarla come valore dell'header con nome SM_USER per ogni chiamata che necessita di essere autorizzata.

| Header name | Header value |
|---|---|
| SM_USER | 8f8ef642349a8b961cb588b0a92901676970415f60a9d6e6961346c0969390c4d0afc50bd144207106df2e57707dddbe81f3684b267abd706034495f06398495 |

### spring-security

```
<!-- GW_PRE_AUTH
        pre-authentication mechanism that uses 'preAuthUser' and
'preAuthKey' parameters inside configuration.properties
        @Discouraged. Avoid to use in production. Enable only for a quick
way to test code.
        -->
        <!---->
        <custom-filter after="PRE_AUTH_FILTER" ref="preAuthFilter"/>
        ...
<beans:bean id="preAuthFilter"
class="org.springframework.security.web.authentication.preauth.RequestHeader
AuthenticationFilter">
        <beans:property name="authenticationManager"
ref="authenticationManager" />
        <beans:property name="exceptionIfHeaderMissing" value="false" />
        <beans:property name="principalRequestHeader" value="SM_USER" />
    </beans:bean>
```

```
    <beans:bean id="preAuthAuthenticationProvider"
class="org.springframework.security.web.authentication.preauth.PreAuthentica
tedAuthenticationProvider">
        <beans:property name="preAuthenticatedUserDetailsService">
            <beans:bean id="userDetailsServiceWrapper"
class="org.springframework.security.core.userdetails.UserDetailsByNameServic
eWrapper">
                <beans:property name="userDetailsService"
ref="gwPreAuthUserDetailsService" />
            </beans:bean>
        </beans:property>
    </beans:bean>
    <beans:bean id="gwPreAuthUserDetailsService"
class="com.geowebframework.webclient.security.GwPreAuthUserDetailsService">
<!--GwPreAuthUserDetailsService implements UserDetailsService -->
    </beans:bean>
    ...
   <authentication-manager alias="authenticationManager">
      <authentication-provider
ref="preAuthAuthenticationProvider"></authentication-provider>
        ...
```

# Servizi di interoperabilità

Questa sezione riguarda l'accesso ai servizi di interoperabilità messi a disposizione da Geoweb. Il codice relativo a tali servizi è reperibile nel file *GwInteroperabilityService.java* sotto le voci dei sei principali servizi trattati nella sezione di questa Wiki dedicata agli utenti:

- insertGwRecord
- insertGwRecord2
- insertGwRecord3
- updateGwRecord
- updateGwRecord2
- updateGwRecord3

**NOTA:** Su Utente Anonimo: i servizi di interoperabilità sono stati forniti di un controllo specifico per gli utenti anonimi. Tali servizi non sono stati pensati per essere usati da utenti anonimi, ma è tuttavia possibile autorizzarne l'utilizzo.

Questo avviene tramite un controllo effettuato sulla proprietà `anonymousUser.interoperabilityServicesAllowed`, di cui si parla nel paragrafo relativo all'autenticazione tramite utente anonimo. Abilitare questa proprietà fa sì che questi servizi riconoscano l'utente anonimo come autorizzato.

**NOTA su LDAP:** La porzione di codice relativa al riconoscimento dell'utente, nei servizi di interoperabilità, è stata configurata in modo tale da riconoscere anche un utente autenticato tramite LDAP come autorizzato.

**NOTA sui servizi insertGwRecord e updateGwRecord:** Dall'aggiornamento di Spring 5 in poi, le entità Multipart sono state modificate. Questo comporta differenze nel ricevere dati sotto forma di file,

che dovranno essere ora di tipo 'multipart/form-data'.

A tal proposito, una riga di codice all'interno di tali servizi è stata commentata, poiché avrebbe costretto il servizio ad accettare esclusivamente dati di tipo *application/json*. La riga in questione è la seguente:

```
@RequestBody(required = false) HashMap<String,Object> parameters

(ciò significa che, fintanto che "parameters" viene richiesto dal servizio, non sarà possibile ricevere dati sotto forma di file)
```

From:
https://wiki.geowebframework.com/ - **GeowebFramework**

Permanent link:
**https://wiki.geowebframework.com/doku.php?id=custom:development:authentications**

Last update: **2025/09/10 16:31**