

Guida all'autenticazione tramite OAuth2

Richiesta del token

In questo primo passo si punta a recuperare uno specifico **token** univoco necessario per l'autenticazione dell'utente.

URL

```
GET - http://[indirizzo]/[contesto]/oauth/token
```

```
(esempio) http://localhost:8080/webclient/oauth/token
```

Parametri

I parametri da inserire in questa prima richiesta, necessari per ottenere il token.

```
grant_type=fisso, il suo valore è 'password'  
client_id= fisso, il suo valore è 'restappgeoweb'  
client_secret= fisso, il suo valore è 'restappgeoweb'  
username= relative ad uno specifico utente Geoweb  
password= relative ad uno specifico utente Geoweb
```

```
(esempio query params)  
grant_type:password  
client_id:restappgeoweb  
client_secret:restappgeoweb  
username:user  
password:password
```

```
(esempio)  
"http://localhost:8080/webclient/oauth/token?grant_type=password&client_id=r  
estappgeoweb&client_secret=restappgeoweb&username=user&password=password"
```

Se la richiesta è stata impostata correttamente, la risposta ricevuta sarà un oggetto il cui valore è una stringa alfanumerica con questo aspetto:

```
"c6bee265-df6d-4e2f-81e7-f58w3547ff65"
```

NOTA: a seconda del metodo e del software utilizzato, l'oggetto ricevuto in risposta potrebbe contenere più di un valore. Sfortunatamente, il valore corretto dipende dal software in uso: ad esempio, Postman restituisce un oggetto XML che contiene due tag <value>, e quello corretto è il più esterno dei due (il primo che si incontra leggendo).

Richiesta con autenticazione

Una volta ottenuto il **token**, l'utente sarà in grado di inviare richieste seguendo lo stesso indirizzo per il quale ha ottenuto l'autorizzazione (servizi di interoperabilità o altre chiamate esterne, come annunciato all'inizio di questo documento).

```
(esempio)
http://localhost:8080/webclient/services/insertGwRecord/[gwClassName]/json
```

Header

Questi sono i valori che dovranno essere impostati nella sezione "headers" di questa richiesta.

```
Content-Type: 'application/json'
Authorization: 'Bearer '+il token recuperato nella richiesta precedente

(esempio) 'Bearer c6bee265-df6d-4e2f-81e7-f58w3547ff65'
```

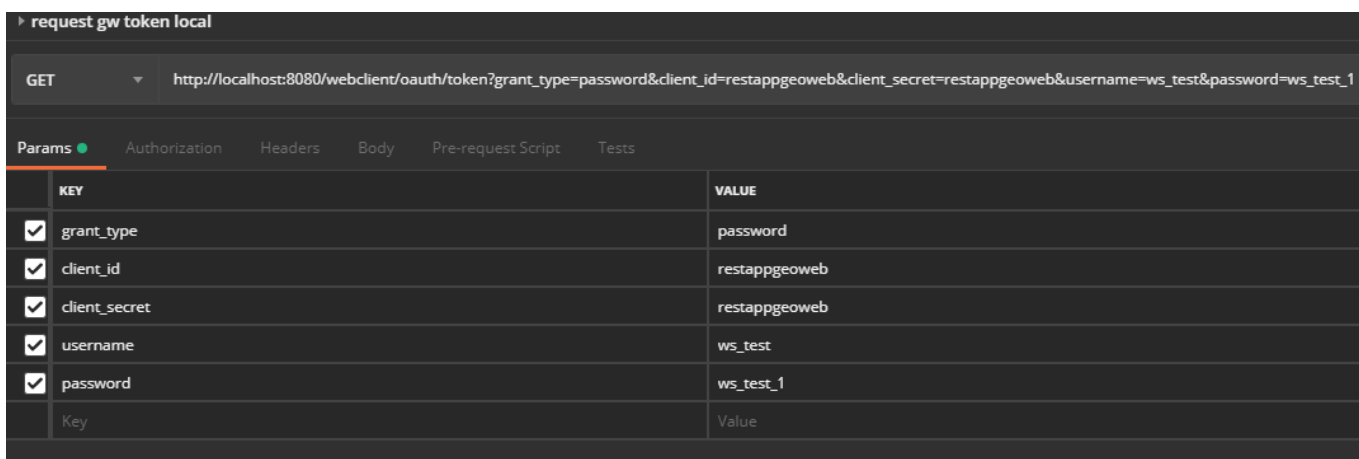
Esempio di autenticazione

Di seguito, un esempio di richiesta di autenticazione tramite OAuth 2, realizzata utilizzando Postman.

Richiesta token

Richiesta:

```
GET –
http://localhost:8080/webclient/oauth/token?grant_type=password&client_id=restappgeoweb&client_secret=restappgeoweb&username=ws_test&password=ws_test_1
```



The screenshot shows a Postman request configuration for a GET request to the endpoint `http://localhost:8080/webclient/oauth/token?grant_type=password&client_id=restappgeoweb&client_secret=restappgeoweb&username=ws_test&password=ws_test_1`. The 'Params' tab is selected, displaying a table of query parameters.

KEY	VALUE
<input checked="" type="checkbox"/> grant_type	password
<input checked="" type="checkbox"/> client_id	restappgeoweb
<input checked="" type="checkbox"/> client_secret	restappgeoweb
<input checked="" type="checkbox"/> username	ws_test
<input checked="" type="checkbox"/> password	ws_test_1
Key	Value

Risposta:

```
<org.springframework.security.oauth2.common.DefaultOAuth2AccessToken>
```

```

<value>613c5d4a-8a9b-4e6f-9f90-60696f760a6b</value>
<expiration>2018-12-10 15:28:51.565 UTC</expiration>
<tokenType>bearer</tokenType>
<refreshToken
class="org.springframework.security.oauth2.common.DefaultExpiringOAuth2RefreshToken">
  <value>8aaab46c-c328-4646-93ab-4f017118c9d3</value>
  <expiration>2018-12-10 15:58:51.562 UTC</expiration>
</refreshToken>
<scope class="java.util.Collections$UnmodifiableSet">
  <c class="linked-hash-set"/>
</scope>
<additionalInformation class="empty-map"/>
</org.springframework.security.oauth2.common.DefaultOAuth2AccessToken>

```

```

Body Cookies (1) Headers (5) Test Results
Pretty Raw Preview XML
1 <org.springframework.security.oauth2.common.DefaultOAuth2AccessToken>
2   <value>613c5d4a-8a9b-4e6f-9f90-60696f760a6b</value>
3   <expiration>2018-12-11 08:51:03.678 UTC</expiration>
4   <tokenType>bearer</tokenType>
5   <refreshToken class="org.springframework.security.oauth2.common.DefaultExpiringOAuth2RefreshToken">
6     <value>5ed9236c-66f3-4d97-b2ee-5e90f0b96318</value>
7     <expiration>2018-12-11 09:21:03.676 UTC</expiration>
8   </refreshToken>
9   <scope class="java.util.Collections$UnmodifiableSet">
10     <c class="linked-hash-set"/>
11   </scope>
12   <additionalInformation class="empty-map"/>
13 </org.springframework.security.oauth2.common.DefaultOAuth2AccessToken>

```

Richiesta servizio 'updateGwRecord'

Richiesta:

```

POST - http://localhost:8080/webclient/rest/updateGwRecord
Header - Authorization: Bearer 613c5d4a-8a9b-4e6f-9f90-60696f760a6b
(Aggiunta di dati a seconda delle necessità dell'utente)

```

► updateGwRecord

POST http://localhost:8080/webclient/rest/updateGwRecord/_test_new_permissions_check

Params Authorization Headers **Body** Pre-request Script Tests

none form-data x-www-form-urlencoded raw binary

KEY	VALUE
<input checked="" type="checkbox"/> id_resource	2199453
<input checked="" type="checkbox"/> fax	zzz
<input checked="" type="checkbox"/> file	Scegli file Nessun file selezionato
Key	Value

Risposta:

```
{
  "itemId": null,
  "success": true,
  "description": "Operazione eseguita con successo"
}
```

Body Cookies (1) Headers (3) Test Results

Pretty Raw Preview JSON ↕

```
1 {
2   "itemId": null,
3   "success": true,
4   "description": "Operazione eseguita con successo"
5 }
```

From: <https://wiki.geowebframework.com/> - GeowebFramework

Permanent link: <https://wiki.geowebframework.com/doku.php?id=custom:interoperability:oauth2&rev=1573029751>

Last update: 2019/11/06 09:42

