Triploa services

(in working...)

Indice generale

1 Overview

The Triploa services consists of a set of callable methods, and some endpoints.

To perform an action using the Triploa services, you need to select a calling convention, send a request to its endpoint specifying a method and some arguments, and will receive a formatted response.

Here a list of parameter:

- The required parameter *method* is used to specify the calling method;
- the required parameter *api_key* is used to specify the application *api_key;*
- the optional parameter *format* is used to specify a response format.

2 User authentication

Some request need a user authentication, every user should be authenticated using an *auth_token* given as result after the login request and valid for all the session. This *auth_token* is a required parameter for all the operation of inserting and updating data.

3 Request formats

3.1 SOAP

SOAP requests are "envelopes" of specially formatted XML data posted to a URL. You can find out more about SOAP at www.w3.org/TR/soap/.

The SOAP Server Endpoint URL is: http://[server_url]/service/soap/

If you use the SOAP request format you can have only SOAP response format so the optional parameter *format* is ignored.

Here a sample envelope:

```
POST /service/soap/travel.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://loa1.cli.di.unipi.it/getTravel"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/
envelope/">
  <soap:Body>
    <getTravel xmlns="http://loa1.cli.di.unipi.it/">
      <api_key>[api_key]</api_key>
      <id>[id]</id>
      <format>[format]</format>
    </getTravel>
  </soap:Body>
</soap:Envelope>
```

3.2 REST

REST is the simplest request format to use - it's a simple HTTP GET or POST action.

The REST Endpoint URL is http://[server_url]/service/rest/

If you use this kind of request and set the *format* parameter you can have in response both rest or json response. By default, REST requests will send a REST response.

4 Response format

4.1 SOAP

The SOAP responses are “envelopes” of specially formatted XML data containing the data requested.

A method call return this:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/
envelope/">
  <soap:Body>
    <getTravelResponse xmlns="http://loa1.cli.di.unipi.it/">
      <getTravelResult>[xml containing specially formatted data representing the object
requested]</getTravelResult>
    </getTravelResponse>
  </soap:Body>
</soap:Envelope>
```

4.2 REST

The REST response is a simple XML block. Also object is serialized in XML format and is tagged by is class name.

A method call returns this:

```
<?xml version="1.0" encoding="utf-8" ?>
<rsp stat="ok">
        <travel>
                <id>1024</id>
                <from>milano</from>
                <to>roma</to>
        </travel>
</rsp>
```

If an error occurs, the following is returned:

```
<?xml version="1.0" encoding="utf-8" ?>
<rsp stat="fail">
     <err code="[error-code]" msg="[error-message]" />
</rsp>
```

4.3 JSON

JSON, or JavaScript Object Notation, is a simple machine-readable data-interchange format, which makes constructing API applications in JavaScript easy. For more information about JSON, visit json.org.

To return an API response in JSON format, send a parameter *format* in the request with a value of *json.*

For examle travel is serialized as:

```
{
        “travel” : {
                “id”:1024,
                “from”:”milano,
                “to”:”roma”
        }
}
```

A failure response:

```
{
        “err”:{
                “code”:[error-code],
                “msg”:[error-message]
        }
}
```

For the json serialization is used the JsonExSerializer library. You can find it at:

http://code.google.com/p/jsonexserializer/

5 Method list

5.1 travel

5.1.1 getTravel

Arguments:

- *api_key,* your api application key;
- *id,* the id of the travel to return.

Example response:

```
<travel>
        <id>1024</id>
        <from>milano</from>
        <to>roma</to>
</travel>
```

6 Error list

7 Example

8 Working comments

In italiano!

Attualmente funziona la richiesta/risposta SOAP con un unico metodo chiamabile

http://[server_url]/service/soap/travel

e chiedendo il getTravel. Non controlla il parametro *api_key* mancando il collegamento con security che fornirà sia questo che la generazione e controlli su *auth_token.*

In risposta viene restituito un semplice envelope con milano-roma!