Sumeet Haldipur
TE COMPS
2019130018
Batch A

EXPERIMENT 4

Aim:

Create a Restful webservice to demonstrate different HTTP methods

Requirements:

Django, Web Browser(Chrome)

Problem Statement:

Olympic management system is an app which targets to provide various sports event organizers a platform to promote their events and also to give the general users/participants information about the sports events in which they're interested

Theory:

1) Available HTTP Methods:

The primary or most commonly-used HTTP methods are POST, GET, PUT, PATCH, and DELETE. These methods correspond to create, read, update, and delete (or CRUD) operations, respectively. There are a number of other methods, too, but they are utilized less frequently. Below is a table summarizing HTTP methods available in Oro API and their return values in combination with the resource URIs:

HTTP Method	CRUD operation	Entire Collection (e.g. /users)	Specific Item (e.g. /users/{id})
GET	Read	200 (OK), list of entities. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single entity. 404 (Not Found), if ID not found or invalid.
POST	Create	201 (Created), Response contains response similar to **GET** /user/{id} containing new ID.	**not applicable**
PATCH	Update	**not applicable**	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
DELETE	Delete	200(OK) or 403(Forbidden) or 400(Bad Request) if no filter is specified.	200 (OK). 404 (Not Found), if ID not found or invalid.

PUT	Update/Replace	**not implemented**	**not implemented**

GET

The HTTP GET method is used to *read* (or retrieve) a representation of a resource. In case of success (or non-error), GET returns a representation in JSON and an HTTP response status code of 200 (OK). In an error case, it most often returns a 404 (NOT FOUND) or 400 (BAD REQUEST).

POST

The POST method is most often utilized to *create* new resources. In particular, it is used to create subordinate resources. That is subordinate to some other (e.g. parent) resource. In other words, when creating a new resource, POST to the parent and the service takes care of associating the new resource with the parent, assigning an ID (new resource URI), etc. On successful creation, HTTP response code 201 is returned.

PATCH

PATCH is used to *modify* resources. The PATCH request only needs to contain the changes to the resource, not the complete resource. In other words, the body should contain a set of instructions describing how a resource currently residing on the server should be modified to produce a new version.

DELETE

DELETE is quite easy to understand. It is used to *delete* a resource identified by filters or ID. On successful deletion, the HTTP response status code 204 (No Content) returns with no response body. Also, the HTTP methods can be classified by the *idempotent* and *safe* properties. The *safe* methods are the HTTP methods that do not modify resources. For instance, using GET or HEAD on a resource URL, should NEVER change the resource.

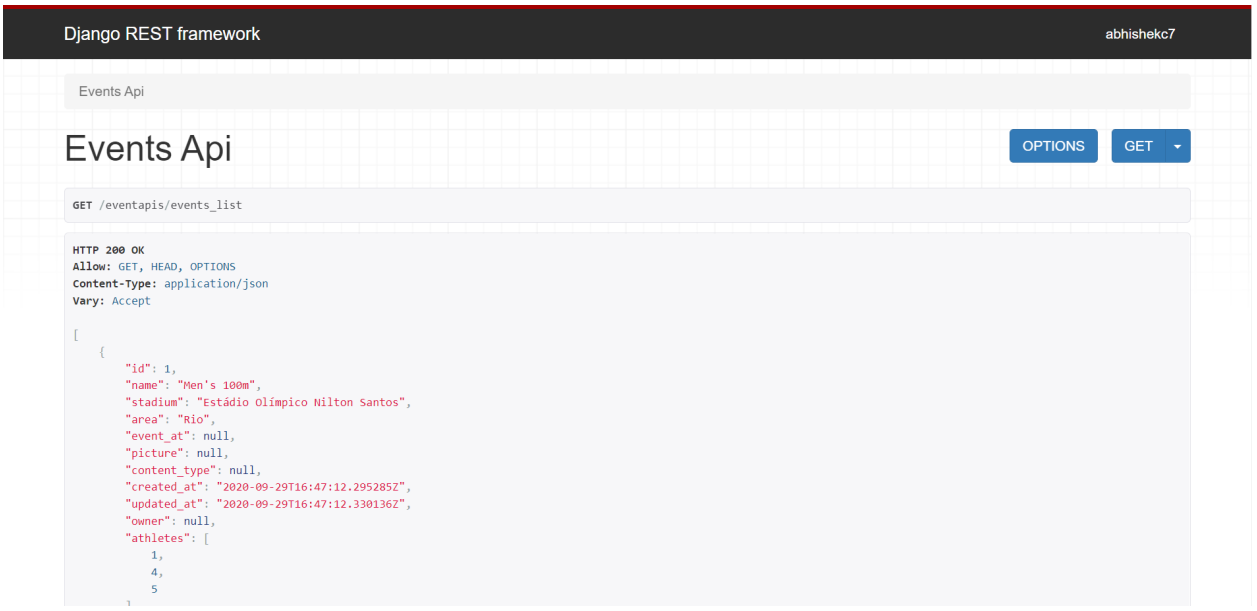
HTTP Method	**Idempotent**	**Safe**
OPTIONS	yes	yes
GET	yes	yes
HEAD	yes	yes
PUT	yes	no
POST	no	no
DELETE	yes	no
PATCH	no	no

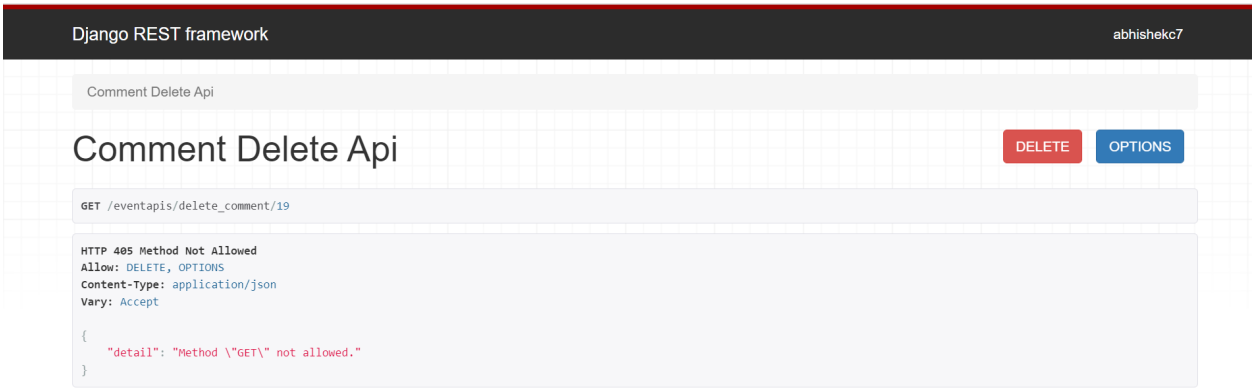
2) REST API

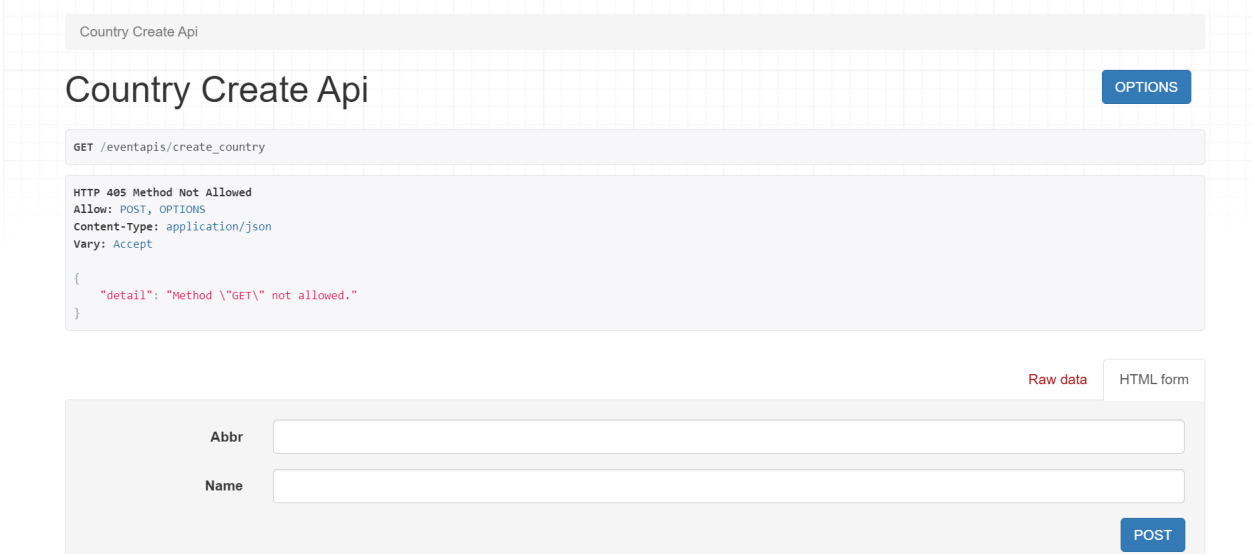
REST is a set of architectural constraints, not a protocol or a standard. API developers can implement REST in a variety of ways. When a client request is made via a RESTful API, it transfers a representation of the state of the resource to the requester or endpoint. This information, or representation, is delivered in one of several formats via HTTP: JSON (Javascript Object Notation), HTML, XLT, Python, PHP, or plain text. JSON is the most generally popular file format to use because, despite its name, it’s language-agnostic, as well as readable by both humans and machines. Something else to keep in mind: Headers and parameters are also important in the HTTP methods of a RESTful API HTTP request, as they contain important identifier information as to the request's metadata, authorization, uniform

resource identifier (URI), caching, cookies, and more. There are request headers and response headers, each with their own HTTP connection information and status codes.

Implementation:







Conclusion:

- Learned about REST APIs and how it differs from other APIs.
- Learned to use Django Rest Framework to create a Restful Web service.
- Learned to generate access & refresh token to implement a OAuth 2.0 type authentication.

References:

1. https://www.redhat.com/en/topics/api/what-is-a-rest-api
2. https://doc.oroinc.com/api/http-methods/
3. https://www.swift.org/documentation/api-design-guidelines/