1 Geo Object

Geometry Objects extend a geometry with additional information to put it into historical context

Structure

Parameter	Detail	Example
geom	Object Geometry	POINT(1.0 2.3)
geom_id	Geometry ID	12
name	name of the Object	Schoenefeld AP
description	Additional Information	Flughafen in ...
valid_from	start of Object	'01/01/1946'
valid_until	end of Object	'01/01/2020'
source	additional context	wikipedia.de/xyz

API calls

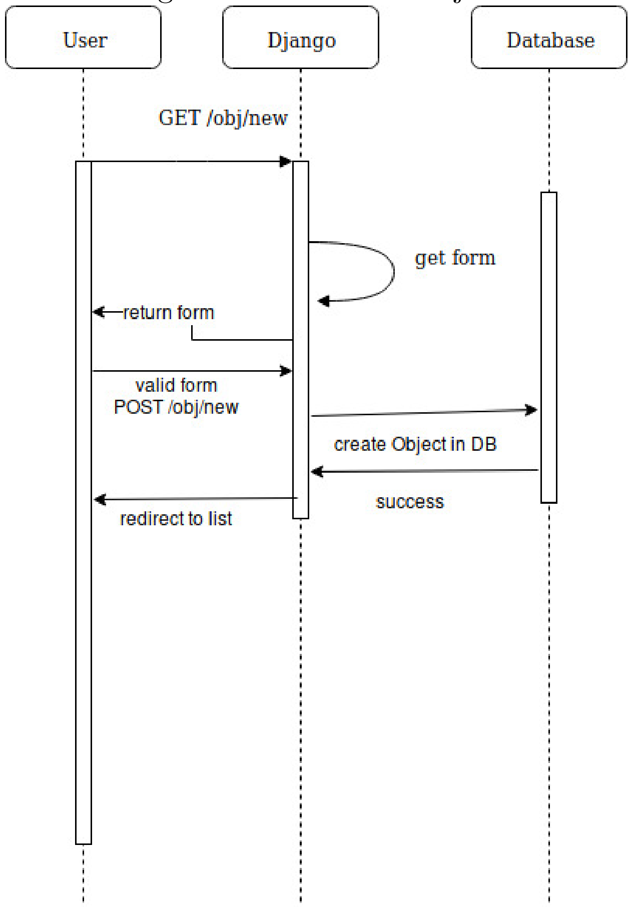
Create

geo-objects can be created with a *POST* Request to the $< root >$*/obj/new/* URL. The following parameters need to be passed inside the request.

Parameter	Expected Value	Example	Function
geom	WKT	POINT(1.0 2.3)	Object Geometry
geom_id	positive int	12	Geometry ID
name	String($<= 200 symbols$)	Schoenefeld AP	name of the Object
description	String($<= 200 symbols$)	Flughafen in ...	Additional Information
valid_from	date	'01/01/1946'	start of Object
valid_until	date	'01/01/2020'	end of Object
source	url	wikipedia.de/xyz	additional context

to prevent *Cross Site Request Forgery (csrf)* Django implements a middleware to handle data-permissions. Authenticated applications generate these tokens and send it as part of the request. The token can be found as *csrfmiddlewaretoken* in valid requests.

Figure 1: UML new Object



Backend + Database: If the data is correct, a new Geometry-Object is created in the database. Then the fields are populated with the corresponding data from the request. A *GET* Request to this URL generates a form for the parameters and a map to draw onto. Sending this form will trigger the *POST* Request mentioned above.

Edit

geo-objects can be edited with a *POST* Request to the
$< root > /obj/ < ID > /edit$ URL. The following parameters need to be passed inside the request.

Figure 2: UML edit Object

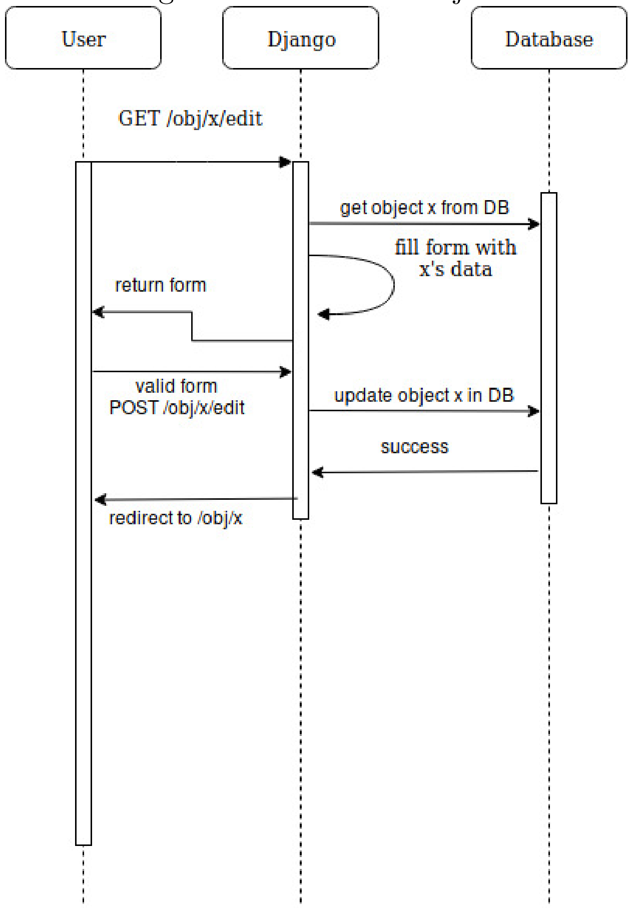
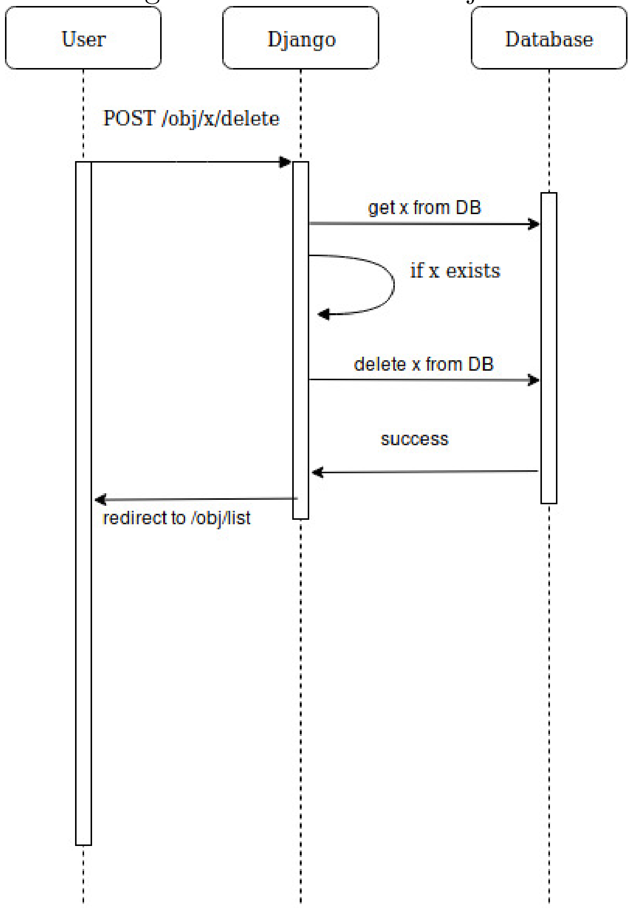


Figure 3: UML delete Object



Parameter	Expected Value	Example	Function
geom	WKT	POINT(1.0 2.3)	Object Geometry
geom_id	positive int	12	Geometry ID
name	String($<= 200 symbols$)	Schoenefeld AP	name of the Object
description	String($<= 200 symbols$)	Flughafen in ...	Additional Information
valid_from	date	'01/01/1946'	start of Object
valid_until	date	'01/01/2020'	end of Object
source	url	wikipedia.de/xyz	additional context

to prevent *Cross Site Request Forgery (csrf)* Django implements a middleware to handle data-permissions. Authenticated applications generate these tokens and send it as part of the request. The token can be found as *csrfmiddlewaretoken* in valid requests.

Backend + Database: If the data is correct, the Geometry-Object will be updated with the corresponding data from the request. When a *GET* request is sent to this URL, the server responds with a form containing the current data from the database. Sending the form will trigger the *POST* Request explained above.

View

Geometry Objects can be viewed with a *GET* Request to the $< root >$ */obj/* $< ID >$ / URL.

Backend + Database: When receiving the request, the Geometry-Object is loaded from the database. The given ID determines the queried object. If the query was successful, the data will be inserted into a template and be shown to the user.

Delete

geo-objects can be deleted with a *POST* Request to the $< root >$ */obj/* $< ID >$ */delete/* URL.

Backend + Database: When the request is received, the Geometry-Object is deleted from the database. The key indicates the desired object to delete.

2 Geometries

A geometry is the smallest data-structure and takes the following parameters to describe a simple Geometry. It can hold Points, Lines and Polygons.

Structure

Parameter	Detail	Example
geometry	Well-Known Text	POINT (14.1 13.2)
ID	Identifier	0, ... ,9999
geom_id	Geometry Identifier	0, ... , 9999

API calls

Create

Requests to the $< root >$ */geom/new* allow to create a new Geometry in the database.

GET A *GET* Requests to this URL will return a form corresponding to the structure 2. The geometry will be generated by drawing on the map or can be represented as Well-known Textformat. Sending this form will call the *POST* method

POST Requests of the type *POST* instruct the backend to generate a new geometry. This only happens if request holds values corresponding to the structure 2 of the geometry. To provide security against cross-site request-forgery the form from the *GET* request generates a token to authenticate the transaction.

Backend + Database If the contents of the *POST* are correct, a new database-element is created and populated with the corresponding data from the request.

View

Requests to the $< root > /geom/ < ID >$ allow to inspect the geometry with the choosen *ID*

GET This request triggers a database-query for a geometry with the specified ID. If this succeeds, a template - populated with the geometry-data is returned.

Edit

Requests to the $< root > /geom/ < ID > /edit/$ allow to edit the geometry with the choosen ID

GET If a *GET* request is received, the server queries the database for the object. The database-contents will filled into a form and returned to the user for changes. Sending this form will trigger the *POST* and generate a *csrf-token*.

POST If the request holds all mandatory fields of the structure 2 and a *csrf-token*, the values from the request will be inserted into the database.

Delete

Requests to the $< root > /geom/ < ID > /delete/$ allow to delete the geometry with the choosen ID

POST A simple *POST* request containing the ID of the expandable geometry.

Backend + Database If the requests ID is valid, the resource will be deleted from the database and the user will be redirected (Overview of all geometries in the database). Invalid IDs will return a 404 code.

Paths

A path simulates a progression through time. It consists of multiple Geo-Objects. Those objects themselves have a have fields holding their start-

and end-values. The start- and end-values of the path allow to define a reference for the values of the contained geo-objects. Similar to the update from *Geometry* to *Geo-Object*, this new data-type holds his own name and description in addition to the date-fields. *Paths* and *Geo Objects* behave in a Many-to-Many Relation, which allows a single object to be used in multiple paths.

Structure

Parameter	Expected Value	Example	Function
name	String($<= 200 symbols$)	Battle of ...	name of the Path
description	String($<= 200 symbols$)	A historic event ...	Additional Information
valid_from	date	'02/13/1312'	start of Path
valid_until	date	'06/02/1324'	end of Path
spots	list of IDS	['1','3','55']	IDs of the Geometry Objects

API calls

Create

Requests to the $< root >$ */path/new/* allow to create a new path

GET Returns a form based on the structure 2 for user-friendly path-creation. Sending this form triggers the POST.

POST If the structure 2 of the request is valid, this will create a new path in the database. This requires a *csrf-token*.

View

Requests to the $< root >$ */path/* $< ID >$ / allow to inspect the path with the choosen ID

GET This request triggers a database-query for a path with the specified ID. If this succeeds, a template - populated with the database values is returned.

Edit

Requests to the $< root >$ */path/* $< ID >$ */edit/* allow to edit the path with the choosen ID

GET If a *GET* request is received, the server queries the database for the path. The database-contents will filled into a form and returned to the user for changes. Sending this form will trigger the *POST* and generate a *csrf-token*.

POST If the request holds all mandatory fields of the structure 2 and a *csrf-token*, the values from the request will be persistent in the database.

Delete

Requests to the $< root >$ */path/* $< ID >$ */delete/* allow to delete the path with the choosen ID

POST A simple *POST* request containing the ID of the expandable path. The path will be deleted, any Geo-Objects attached through the Many-to-Many Relation will not be removed.

Backend + Database If the requests ID is valid, the resource will be deleted from the database and the user will be redirected (Overview of all paths in the database). Invalid IDs will return a 404 code.

Remove Geo-Object from Path

Requests to the $< root >$ */path/* $< ID >$ */remove/* $< objectID >$ allow to delete a specified object from the paths *Many2Many* relation.

POST This request removes the desired Geo-Object $< objectID >$ from the selected path $< ID >$. This request needs no further parameters