Request & Response Life Cycle in Django

## **Understanding Django life cycle.**

## Image for postRequest-Response life cycle in Django

While setting up the django application in the server we need a webserver and a wsgi server. Webserver helps us in serving static files/content. If we did not use the webserver to serve static files then it has to served by WSGI server which results in more number of requests to the server. So, gradually it slow down the application performance. Web server balances the requests load on the server. We can divide request resources in two types.

**1. static resource**

**2. dynamic resource**(It has to process the data based on request to provide resource)

If browser request for the static resources like images/css/javascript/etc, then NGINX serves the request without sending it to the uWSGI server.If browser requests for a dynamic request then NGINX passes it to the uWSGI to process the request.At this stage NGINX acts like a reverse proxy server. A reverse proxy server is a type of proxy server that typically sits behind the firewall in a private network and directs browser/client requests to the appropriate back-end server(uWSGI).Advantages of reverse proxy server are Load balancing, Web acceleration, Security and anonymity.

WSGI is a tool created to solve a basic problem: connecting a web server to a web framework. WSGI has two sides: the ‘server’ side and the ‘application’ side. To handle a WSGI response, the server executes the application and provides a callback function to the application side. The application processes the request and returns the response to the server using the provided callback. Essentially, the WSGI handler acts as the gatekeeper between your web server (Apache, NGINX, etc) and your Django project.

While deploying the django application on the server “Nginx, gunicorn” combination widely used.When a client sends a request to the server it first passed to the webserver(NGINX), It contains the configuration rules to dispatch the request to the WSGI(gunicorn) server or served by itself(serving static files). WSGI server pass the request to the Django application. Django has the following layers in dealing with request-response lifecycle of a Django application.

## Layers of Django Application

1. Request Middlewares
2. URL Router(URL Dispatcher)
3. Views
4. Context Processors
5. Template Renderers
6. Response Middlewares