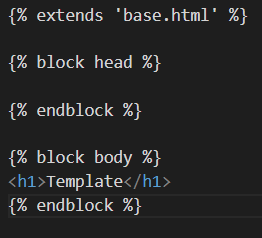
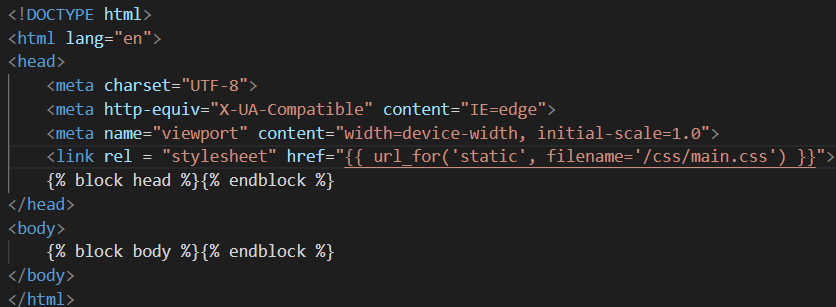


Jinja 2 syntax, Flask’s template engine, insert all the code that the pages inheriting this template will use.

* {%%} for if else statements/loops, {{}} for things you want to be printed in strings



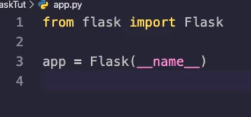
How to extend an HTML



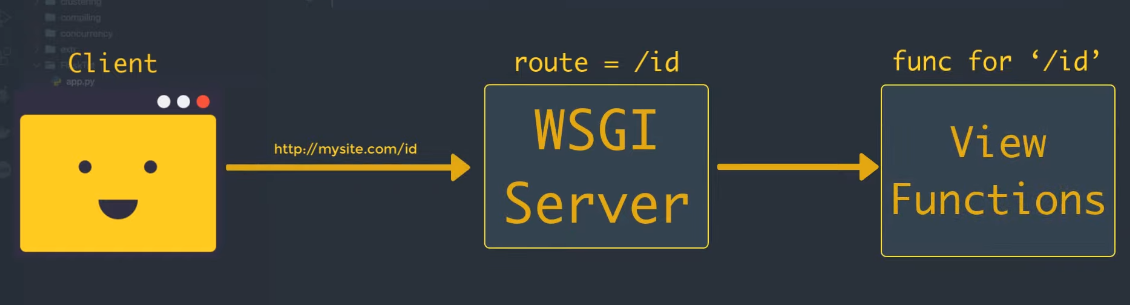
Linking a stylesheet



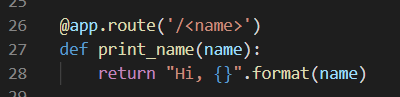
Connect through Flask

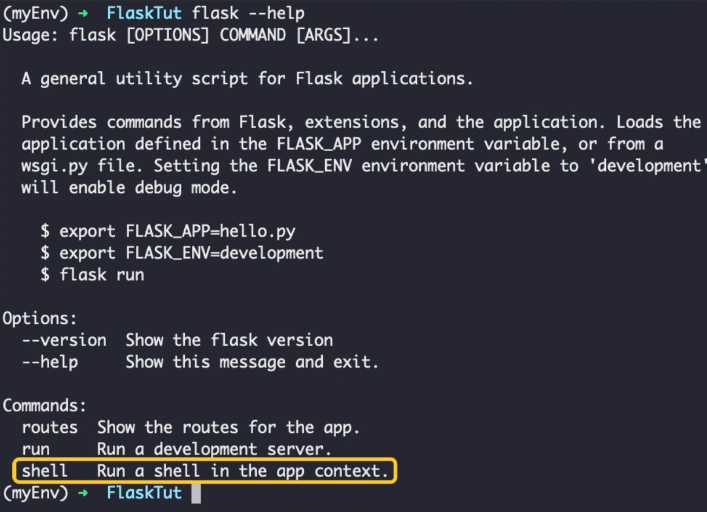


Create instance of application, passes on requests from clients using web server gateway interface



**Dynamic Routing**



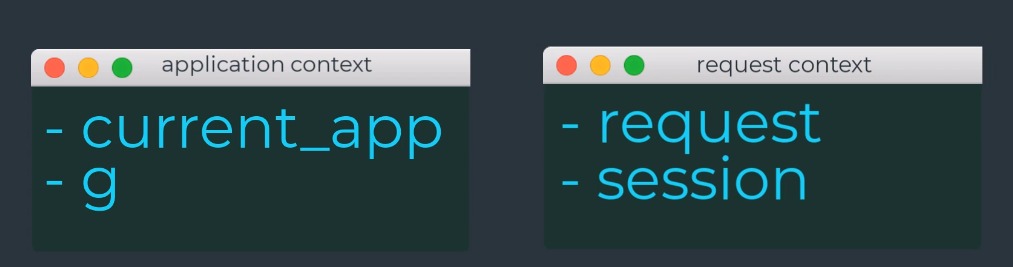


**Application & Request Context**

Flask uses Context, view functions can be written like a global variable (though it cannot be as multithreaded apps have each thread needed a diff request object.)

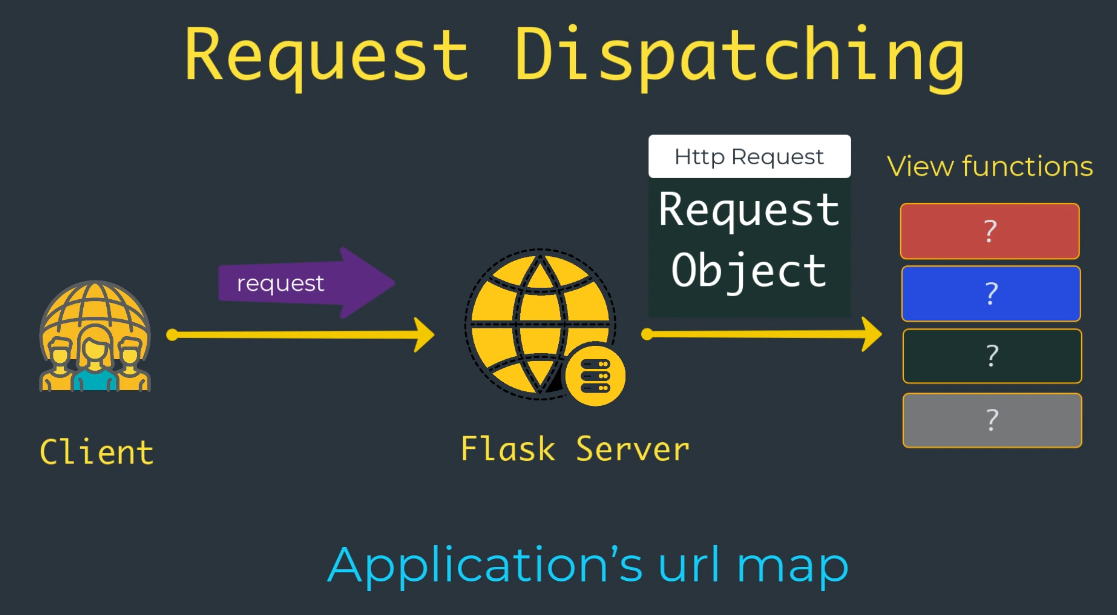
Context achieves this by making several objects global w/n a thread w/o interfering each other

**Application Context and Request Context**

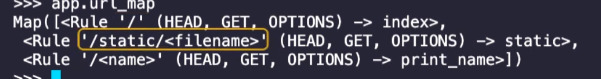


* **Current\_app**: An instance of the active application
* **G**: an object the app can use for temp storage when handling request, reset on each request
* **Request**: the actual request context, has the contents of an HTTP request sent by the client
* **Session**: Dictionary that the app can use to store values that are remembered between requests.
* Flask activates the application and request contexts before dispatching a request to the app, and removes them after the request is handled.
* When the app context is pushed (app\_context.push()), current\_app and g become available to thread.
* When request is pushed (request\_context.push()), request and session variables become available.
* If any are accessed w/o an active app or request context, error generated

**Request Dispatching**

****

* URL Map built using data provided in app.route decorator



* Special flask route giving access to static files, HEAD/GET/OPTIONS shown in the URL map are the request methods handled by the route

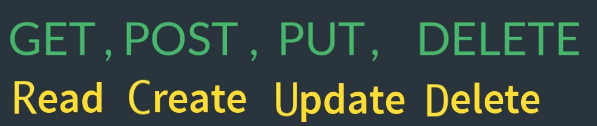
**Request Object**

* A context variable named request, contains all the info the client included in the HTTP request.
* Some methods included
* **Methods**
  + **Get\_data**: Return buffered data from the request body
  + **Get\_json**: Return a python dictionary with a parsed json included in the body of the request
  + **Is\_secure**: returns true if the request came from a secure HTTPS connection
* **Variables**
  + **Endpoint**: Name of the flask endpoint that’s handled by the request. Flask uses the name of the view function as the endpoint name for route
  + **Method**: Gives the HTTP request method, like GET or POST
  + **Host**: The host defined in the request, including port # if given by the client.
  + **URL**: The complete URL requested by the client
    - **Base\_url:** Same as URL w/o the query string component
  + **Environ**: Raw WISGY environment dictionary for the request
* **Request Hooks**
  + Implemented as decorators
  + Can be used to execute code before or after each request is proceded. (ex. at start, may need to create DB connection or authenticate user making the rest)
  + FLASK can reduce common functions to be invoked before or after a request dispatch
  + **Before\_request**: Registers a function to run before each requests
  + **Before\_first\_request**: Registers a function to run before the first handled requests (can add server init tasks)
  + **After\_request**: Registers a function to run after each request (only if no unhandled exceptions occurred).
  + **Teardown\_request**: Registers a function to run after each request, even if unhanded exceptions occurred
  + **Pattern**:g context global, as storage, ex. before\_request handler loads the login user from database and store log in user, then when view function is invoked it can retrieve the user from there.

**Response**

* When flask invokes a view function **it expects its return value to be the response to the request.**
* Mostly it's a simple string in the form of an HTML page but the HTTP protocol requires more than a string as a response to a request
* **Status Code**: Important part of HTTP response, Flask by default sets to 200, indicating the request was carried out successfully.
  + When view function needs to respond w/ a different status code, it can add the numeric code as a 2nd return value, after the response text.
* **Header**: Third argument, dictionary added to HTTP response.
* Instead of returning 1-3 values as a tuple, Flask view functions can return our **response object**.
  + To make this query response function takes 1-3 arguments, the same values that can be returned from a view function, and returns an equivalent response object.
  + Sometimes useful to generate the response object inside view function then use its method to fully configure the response.
  + **Methods:**
    - **Set\_cookie**: Adds a cookie to the response
    - **Delete\_cookie**: Removes a cookie from the response
    - **Set\_data**: Sets response body as a string or bytes value
    - **Get\_data**: Gets the response body
  + **Variables:**
    - **Status\_code**: Return the numeric HTTP status code
    - **Headers**: A dictionary-like object with all headers that’ll be sent to the response
    - **Content\_length**: Provides the length of response body
    - **Content\_type**: The media type of response body we return back.
* **Redirect**
  + Doesn’t return a page or string, but just gives the browser a new URL to navigate to.

**Part 4**

* **Re**presentational **S**tate **T**ransfer
  + A way for 2 computer systems to communicate over http like browsers and web servers
  + More of a set of guidelines
  + **Client-Server**: System A makes HTTP request to a URL hosted by B, which returns a response
  + **Stateless**: Client request should contain all info necessary to respond to a request.
  + **Cacheable**: Response should be defined as cacheable or not
  + **Layered**: Requesting client need not know whether it’s communicating with an actual server, proxy, or intermediary
* **Important Terminologies**
  + **Endpoint/Resource:** URL + Domain + Port + Path + Query string
  + **HTTP Methods:** GET (get data as response), POST (create a new record), PUT (update record), DELETE (remove a record).
  + **HTTP Headers:** Auth token, cookies, can be contained in these headers
* **J**ava**S**cript **O**bject **N**otation
  + Language Independent Data Interchange Format
  + Can have nested objects

