Open-Source Technology Use Report

Proof of knowing your stuff in CSE312

Flask

General Information & Licensing

Code Repository	https://github.com/pallets/flask
License Type	BSD-3
License Description	● Any source code must contain the license and copyright statement ● Documentation for binaries must contain the license and copyright statement ● Redistribution for commercial purposes is fine as long as Flask is clearly listed as the copyright holder
License Restrictions	● We are not allowed to use the name of the copyright holder or any contributors for endorsing products made using Flask without written permission
Who worked with this?	

Use as many of the sections below as needed, or create more, to explain every function, method, class, or object type you used from this library/framework.

Purpose

Replace this text with some that answers the following questions for the above tech:

- This tech provides us a backbone for serving content to the users and receiving data back from them via HTTP/TCP connections. It also has good templating features for dynamically generating content given some sort of input.
- The majority of our code uses flask in some way, especially if it relates directly to content seen by the user.

Magic

Dispel the magic of this technology. Replace this text with some that answers the following questions for the above tech:

- How does this technology do what it does for you in the **Purpose** section of this report? Please explain this in detail, starting from after the TCP socket is created. Remember, to be allowed to use a technology in your project, you must be able to know how it works.
- Where is the specific code that does what you use the tech for? You ***must*** provide a link to the specific file in the repository for your tech with a line number or number range.
    - If there is more than one step in the chain of calls *(hint: there will be)*, you must provide links for the entire chain of calls from your code, to the library code that actually accomplishes the task for you.
    - Example: If you use an object of type HttpRequest in your code which contains the headers of the request, you must show exactly how that object parsed the original headers from the TCP socket. This will often involve tracing through multiple libraries and you must show the entire trace through all these libraries with links to all the involved code.

*This section may grow beyond the page for many features.

We are using the flask framework since it contains tools and libraries used to develop a web application efficiently. Objects such as requests we implement is based on Werkzeug. This WSGi toolkit enables us to build a web frame on it. Jinjia2 template allows us to pass Python variable into HTML template easier, and also enable to render a dynamic web page with a specific data source when combines a template.

`render_template('home.html')`:

- When we implement ***`render_template`***, https://github.com/pallets/flask/blob/cc66213e579d6b35d9951c21b685d0078f373c44/src/flask/templating.py#L135 we only need to provide the name of the template and the variables we want to pass to the template engine as keyword arguments. Then Flask will look for templates in the `templates` folder by ***`jinja_env.get_or_select_template.`*** https://github.com/pallets/jinja/blob/ae53ea5350c5dd622c7e4b85f8aca5dedc5af7a7/src/jinja2/environment.py#L1067
  If a iterable pf template names is given, ***`select_template()`*** will be called, if one name is given***`,get_template()`*** ***`wi`***ll be called
  https://github.com/pallets/jinja/blob/ae53ea5350c5dd622c7e4b85f8aca5dedc5af7a7/src/jinja2/environment.py#L1081
  https://github.com/pallets/jinja/blob/ae53ea5350c5dd622c7e4b85f8aca5dedc5af7a7/src/jinja2/environment.py#L1084
    - ***`select_template()`*** and ***`get_template()`*** will load template by name

  and return a template that is found, if it does not exist, **`TemplateNotFound`** exception is raised.
- ***`render_template`*** invokes _render
    - ***`_render`*** https://github.com/pallets/flask/blob/cc66213e579d6b35d9951c21b685d0078f373c44/src/flask/templating.py#L127
    - Inside _render, ***`template.render(context)`*** https://github.com/pallets/jinja/blob/ae53ea5350c5dd622c7e4b85f8aca5dedc5af7a7/src/jinja2/environment.py#L1269
      will convert template into string in case that we can find the element of the template fastly by just adding one parameter that we want to replace.

run(debug=True, host='0.0.0.0')

- We implement run() to runs the application on a local development server.
- It invokes ***run_simple()***
  https://github.com/pallets/flask/blob/cc66213e579d6b35d9951c21b685d0078f373c44/src/flask/app.py#L1191
  to help us connect the local and free to reload when a f. then a
  is_running_from_reloader()
  https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L898
  is called to check if the server is running as a subprocess within the Werkzeug reloader.
  - Inside ***is_running_from_reloader()*** , ***prepare_socket***
  (Address remaining ResourceWarning *related to the socket used by* run_simple*. Remove* prepare_socket*, which now happens when creating the server. #2421)*
  is implemented in order to prepare a socket for use by the WSGI server and reloader
  - .***prepare_socket*** calls a ***socket.socket()*** to create sockets.

route()

- route helps us decide which request methods the server receives and which path is to. Inside ***route(),*** ***add_url_rule()*** is called.
  https://github.com/pallets/flask/blob/cc66213e579d6b35d9951c21b685d0078f373c44/src/flask/scaffold.py#L455
  It deal with url with following rules:
  - If a rule ends with a slash and is requested without a slash by the user, the user is automatically redirected to the same page with a trailing slash attached.
  - If a rule does not end with a trailing slash and the user requests the page with a trailing slash, a 404 not found is raised.
  - If a URL contains a default value, it will be redirected to its simpler form with a 301 redirect

flask.redirect()

- We use ***flask.redirect()***
  https://github.com/pallets/flask/blob/cc66213e579d6b35d9951c21b685d0078f373c44/src/flask/helpers.py#L266
  to redirect to another location. The first parameter is where the URL to direct to. If we are dealing with the application handling the current activity, it will use its ***redirect()*** method, if not it will use ***werkzeug.utils.redirect()***.
  https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/utils.py#L244
  Then the code passe to the second parameter can be any one of below:
  - *301, 302, 303, 305, 307, and 308.*