Custom Error Pages

Flask comes with a handy **`abort()`** function that aborts a request with an HTTP error code early. It will also provide a plain black and white error page for you with a basic description, but nothing fancy.

Depending on the error code it is less or more likely for the user to actually see such an error.

Common Error Codes

The following error codes are some that are often displayed to the user, even if the application behaves correctly:

404 Not Found
: The good old "chap, you made a mistake typing that URL" message. So common that even novices to the internet know that 404 means: damn, the thing I was looking for is not there. It's a very good idea to make sure there is actually something useful on a 404 page, at least a link back to the index.

403 Forbidden
: If you have some kind of access control on your website, you will have to send a 403 code for disallowed resources. So make sure the user is not lost when they try to access a forbidden resource.

410 Gone
: Did you know that there the "404 Not Found" has a brother named "410 Gone"? Few people actually implement that, but the idea is that resources that previously existed and got deleted answer with 410 instead of 404. If you are not deleting documents permanently from the database but just mark them as deleted, do the user a favour and use the 410 code instead and display a message that what they were looking for was deleted for all eternity.

500 Internal Server Error
: Usually happens on programming errors or if the server is overloaded. A terribly good idea is to have a nice page there, because your application *will* fail sooner or later (see also: Application Errors).

Error Handlers

An error handler is a function that returns a response when a type of error is raised, similar to how a view is a function that returns a response when a request URL is matched. It is

passed the instance of the error being handled, which is most likely a **`HTTPException`**. An error handler for "500 Internal Server Error" will be passed uncaught exceptions in addition to explicit 500 errors.

An error handler is registered with the **`errorhandler()`** decorator or the **`register_error_handler()`** method. A handler can be registered for a status code, like 404, or for an exception class.

The status code of the response will not be set to the handler's code. Make sure to provide the appropriate HTTP status code when returning a response from a handler.

A handler for "500 Internal Server Error" will not be used when running in debug mode. Instead, the interactive debugger will be shown.

Here is an example implementation for a "404 Page Not Found" exception:

```
from flask import render_template

@app.errorhandler(404)
def page_not_found(e):
    # note that we set the 404 status explicitly
    return render_template('404.html'), 404
```

When using the application factory pattern:

```
from flask import Flask, render_template

def page_not_found(e):
  return render_template('404.html'), 404

def create_app(config_filename):
    app = Flask(__name__)
    app.register_error_handler(404, page_not_found)
    return app
```

An example template might be this:

```
{% extends "layout.html" %}
{% block title %}Page Not Found{% endblock %}
{% block body %}
  <h1>Page Not Found</h1>
  <p>What you were looking for is just not there.
  <p><a href="{{ url_for('index') }}">go somewhere nice</a>
{% endblock %}
```

Returning API errors as JSON

When using Flask for web APIs, you can use the same techniques as above to return JSON responses to API errors. **`abort()`** is called with a `description` parameter. The **`errorhandler()`** will use that as the JSON error message, and set the status code to 404.

```
from flask import abort, jsonify

@app.errorhandler(404)
def resource_not_found(e):
    return jsonify(error=str(e)), 404

@app.route("/cheese")
def get_one_cheese():
    resource = get_resource()

    if resource is None:
        abort(404, description="Resource not found")

    return jsonify(resource)
```