Felipe Cadar Chamone - 2016006417

Lab 3 - Qualidade de Código

1. Qual o projeto analisado?

https://github.com/pallets/flask

2. Quais os 5 arquivos mais complexos do projeto?

1. src/flask/cli.py: AvgCCN: 4.0 :
2. src/flask/debughelpers.py: AvgCCN: 4.0 :
3. src/flask/json/**init**.py: AvgCCN: 3.8
4. src/flask/logging.py: AvgCCN: 3.7 :
5. src/flask/config.py: AvgCCN: 3.2 :

3. Quais os 5 métodos/funções mais complexos do projeto?

1. CCN: 15 Lenght:116 make_response@1604-1719@src/flask/app.py
2. CCN: 13 Lenght:114 run@780-893@src/flask/app.py
3. CCN: 12 Lenght:149 url_for@195-343@src/flask/helpers.py
4. CCN: 13 Lenght:73 find_app_by_string@147-219@src/flask/cli.py
5. CCN: 12 Lenght:31 routes_command@903-933@src/flask/cli.py

4. Selecione um dos 5 métodos/funções mais complexas e:

O método converte um valor de retorno, que pode ser de vários tipos de objetos e o converte para um objeto padrão. Uma refatoração seria padronizar o retorno de todos esses metodos, para que essa função não precise ter a complexidade crescente sempre que um novo método com um novo retorno for adicionado.

Código:

```
def make_response(self, rv):
        """Convert the return value from a view function to an instance of
        :attr:`response_class`.
        :param rv: the return value from the view function. The view
function
            must return a response. Returning ``None``, or the view ending
            without returning, is not allowed. The following types are
allowed
            for ``view_rv``:
            ``str``
                A response object is created with the string encoded to
UTF-8
                as the body.
            ``bytes``
                A response object is created with the bytes as the body.
            ``dict``
```

```
                A dictionary that will be jsonify'd before being returned.
            ``tuple``
                Either ``(body, status, headers)``, ``(body, status)``, or
                ``(body, headers)``, where ``body`` is any of the other
types
                allowed here, ``status`` is a string or an integer, and
                ``headers`` is a dictionary or a list of ``(key, value)``
                tuples. If ``body`` is a :attr:`response_class` instance,
                ``status`` overwrites the exiting value and ``headers`` are
                extended.
            :attr:`response_class`
                The object is returned unchanged.
            other :class:`~werkzeug.wrappers.Response` class
                The object is coerced to :attr:`response_class`.
            :func:`callable`
                The function is called as a WSGI application. The result is
                used to create a response object.
        .. versionchanged:: 0.9
           Previously a tuple was interpreted as the arguments for the
           response object.
        """

        status = headers = None

        # unpack tuple returns
        if isinstance(rv, tuple):
            len_rv = len(rv)

            # a 3-tuple is unpacked directly
            if len_rv == 3:
                rv, status, headers = rv
            # decide if a 2-tuple has status or headers
            elif len_rv == 2:
                if isinstance(rv[1], (Headers, dict, tuple, list)):
                    rv, headers = rv
                else:
                    rv, status = rv
            # other sized tuples are not allowed
            else:
                raise TypeError(
                    "The view function did not return a valid response
tuple."
                    " The tuple must have the form (body, status,
headers),"
                    " (body, status), or (body, headers)."
                )

        # the body must not be None
        if rv is None:
            raise TypeError(
                f"The view function for {request.endpoint!r} did not"
                " return a valid response. The function either returned"
                " None or ended without a return statement."
            )
```

```
        # make sure the body is an instance of the response class
        if not isinstance(rv, self.response_class):
            if isinstance(rv, (str, bytes, bytearray)):
                # let the response class set the status and headers instead
of
                # waiting to do it manually, so that the class can handle
any
                # special logic
                rv = self.response_class(rv, status=status,
headers=headers)
                status = headers = None
            elif isinstance(rv, dict):
                rv = jsonify(rv)
            elif isinstance(rv, BaseResponse) or callable(rv):
                # evaluate a WSGI callable, or coerce a different response
                # class to the correct type
                try:
                    rv = self.response_class.force_type(rv,
request.environ)
                except TypeError as e:
                    raise TypeError(
                        f"{e}\nThe view function did not return a valid"
                        " response. The return type must be a string,"
                        " dict, tuple, Response instance, or WSGI"
                        f" callable, but it was a {type(rv).__name__}."
                    ).with_traceback(sys.exc_info()[2])
            else:
                raise TypeError(
                    "The view function did not return a valid"
                    " response. The return type must be a string,"
                    " dict, tuple, Response instance, or WSGI"
                    f" callable, but it was a {type(rv).__name__}."
                )

        # prefer the status if it was provided
        if status is not None:
            if isinstance(status, (str, bytes, bytearray)):
                rv.status = status
            else:
                rv.status_code = status

        # extend existing headers with provided headers
        if headers:
            rv.headers.update(headers)

        return rv
```