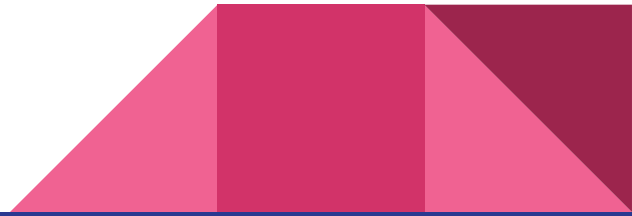
Runtime-Terror API

By: Kia Afzali, Victoria Delk, Rohit Ghosh, Yanyu Wang, Charles Zhang

Request Handler

- **What are the important methods and what do they do?**
  - `SetRequest()` passes the data of a request to a request handler
  - `GenerateResponse()` prompts the handler to generate a response, depending on the handler type
- **When are these methods called and who calls them?**
  - `SetRequest()` is called by a `Dispatcher` when it takes in a request
  - `GenerateResponse()` is called by the same `Dispatcher` after `SetRequest()` has passed the request to the correct request handler
- **How could one test the request handlers?**
  - Request handlers are tested through unit tests and integration tests on the derived classes of class `RequestHandler`:
    - `StaticHandler`
    - `EchoHandler`
    - `BadHandler`
  - and class `Dispatcher`



Config File Format

- **How are server-level parameters (like the port number) specified?**
  - The config file is parsed and read by `ConfigParser` and `ConfigReader` classes
  - Then the port that the server listens on will be specified in the main function and can be accessed through `ConfigReader`'s `GetPort()` function

- **How are URLs associated with request handlers?**
  - URLs are associated with request handlers using a `std::unordered_map` in `HandlerManager`
  - Handlers are stored in `handler_map_` using `AssignHandler()` when a `HandlerManager` object is constructed and mapped to URLs in the config file
  - The URL is used as a key in `handler_map_` to access the corresponding handler
  - If a URL->handler mapping is not found, `BadHandler` is returned

- **How are request handlers configured?**
  - `RequestHandler` objects are configured using the HTTP request when it comes in
  - For static requests, the `RequestHandler` object checks that the file path is valid, retrieves the file, and generates a response by adding header information
  - For echo requests, the `RequestHandler` adds header information to the HTTP request body and returns it

Example Config

```
server {
    listen 80;

    location /echo {
        handler echo;
    }

    location /static {
        handler static;
        root /static;
    }
}
```

Dispatcher Mechanism

- **How is the config file used to construct request handler objects?**
  - The parsed config file is handed to a `ConfigReader`, which extracts any defined `Locations` and returns them in a vector
  - This vector of `Locations` is used to construct a `HandlerManager`, which constructs a `RequestHandler` using `handler` and `root` and maps it to the corresponding `url`

- **What is the lifetime of request handler objects?**
  - Each `RequestHandler` object exists until the webserver is shut down

- **How are HTTP requests parsed and dispatched to the correct request handler?**
  - When the webserver receives a request, it is used to construct a `Request` object
  - This `Request` object has a member function `GetRequestURL()` that extracts the request's URL
  - The `Request` object is passed into a `Dispatcher` object, which contains a `HandlerManager` that maps the extracted URL to the correct handler