What is the Interface Oasis?

The name Interface Oasis roots from the book “Ready Player One”. In the book the Oasis is a virtual reality which connects people around the globe. Just like the Oasis in the book connects people this client represents a universal interface for Web APIs. The can send and receive http requests and responses and process them according to a custom procedure. The connector of everything in the Interface Oasis is the Trigger. A Trigger is the object which will make the client take the initiative and send a request to a custom defined server. Data and the URL to the server itself can all be dynamically defined by previously received data.

This is where the Parser starts working. A parser is confronted with a parsed http header (the http header is represented by a map with the header argument names as key and their values as value) and an unparsed body. While the program can automatically parse GET URLs and create a map from which the GET query attributes can be accessed, the individuality stands out with a custom parser. The program offers different rules to parse the unparsed body into a list (JSON and XML is supported). This list is then forwarded to an IndexAssigner. This object will first scan the list for defined regexes and assigner custom defined keys to the values of found matches. Afterward, custom indexes of the list are also assigned to custom defined keys. When the Map is complete it is passed to a Responder.

A responder consists of the responder itself as well as a header and a body. The body and the header can both be dynamically generated from the parsed map. This is implemented with the Constant object, the original source of the program’s dynamics. A Constant is made up from multiple Values. Those Values are either a constant string or they use the value of specified key of the map. This dynamic construct ultimately returns one dynamically generated string which will be used as a value in the responder. Constants allow the user to generate a dynamic URL and much more header arguments (even custom ones). In addition, the client can send POST requests in which constants are used to represent the content. This dynamic construct can then be sent by the responder.

However, the connector piece, the Trigger, can either be triggered by a Listener or a Responder (a Listener receives a request and sends a response, a responder sends a request and receives a response). A listener listens on a local port and reports to the Trigger it receives a valid request. The responder responds to the Trigger whenever it receives a valid response to a request. Both will pass their received data to the trigger so that the trigger can use different parsers to generate maps for its responders who will be invoked afterward.

All in all there are two most common use-cases for the Interface Oasis. Either the client is running on a web server and it receives requests to forward them to other services with the suitable syntax or the client is constantly making requests to servers and the server responses themselves are then again used to pass data to a third party. In either case a completely different syntax can be used for every single party.

If the user requires any more Rules or Parsers it is always possible to implement them yourself. The Rule objects are generated via reflection and the ParserHandler class has the right methods to acquire data types from strings and report errors if the rule creation failed. Both, the Parser and the Rule, have an interface which is sufficiently commented to understand what which method is supposed to do.

Because the Listener and Responder can log nearly everything they do, and the responder can send manual requests to a server, the program is also suitable for simple web debugging.