**Server:**

Responsible for:

* Listening for requests to server at specified port
* Verifying support for request’s HTTP method (e.g. HEAD and GET)
* Setting header fields:
  + Cookies
  + Response code – if error returned from router
* Completing request if error returned from router
* Passing request & response to router

Not responsible for:

* Setting header fields:
  + Cache control (HTTP/1.1 and HTTP/1.0)
  + Response code – if no error returned from router
  + Content type
  + Content length
  + Transfer encoding (chunked for HTTP/1.1, content length for HTTP/1.0)

**Router:** One piece which has not been covered is that the router should be configurable. I think the handleRequest function on the controller should take the function that the router found. This way the router is capable of picking a function or controller that has been configured. If the router has not been configured, then the router will use the literal url values as described below.

The router is a function module and it has a function to search for controllers and actions (functions) of a particular controller. One way of looking for functions of names on a controller would be by using the hasProperty function provided by JavaScript.

The router can expect that the url format will roughly follow - "{controller}/{action}/?{param1=value}&{param2=value}”

Example: /Account/logonwithparam?fname=kevin&lname=cain

If the router finds a controller with the specified function, it will route the request and response to the controller’s “handleRequest(req, res, action, sid)” function. It is still undetermined what, if anything, the function will return.

Router error scenarios: It does not find a controller, or it finds a controller but the specified action (function) for that controller does not exist. In both cases the router lets the server know that an error has occurred and the server knows how to handle the response.

**Controller:** The controller is an extendable function object. A controller is contained in the server and the servers router calls the controller function “handleRequest(req, res, action, sid)”. The controller will ultimately call the function defined by the action parameter. It is necessary for the router to call this generic function instead of the action defined in the url, because we need to remove the burden of having to deal with the request and response parameters from the developer. Additionally, the url may still need to be parsed to populate any model parameter specified by the developer. At this time the controller will perform other functions such as checking the last edit date of the view, setting the content type, status code, etcetera, of the response. The controller has response & request members that are accessible to the derived controller class.

**Model:** The model is an extendable function object. A model can be defined in a controller, and be reference inside of the view page template. A model may also be specified as a parameter in the controller function. When the developer chooses to use a defined model function as a parameter, the developer must assign the function to the function of the controller which will use the model as a parameter. Example: controller.logon.model = modelFunction;

The model has a setParameters function, which is used by the controller to populate the model before passing into the controller function.

**ViewPage:** A way to preload templates into memory? Declaring content type?

**Demo:**