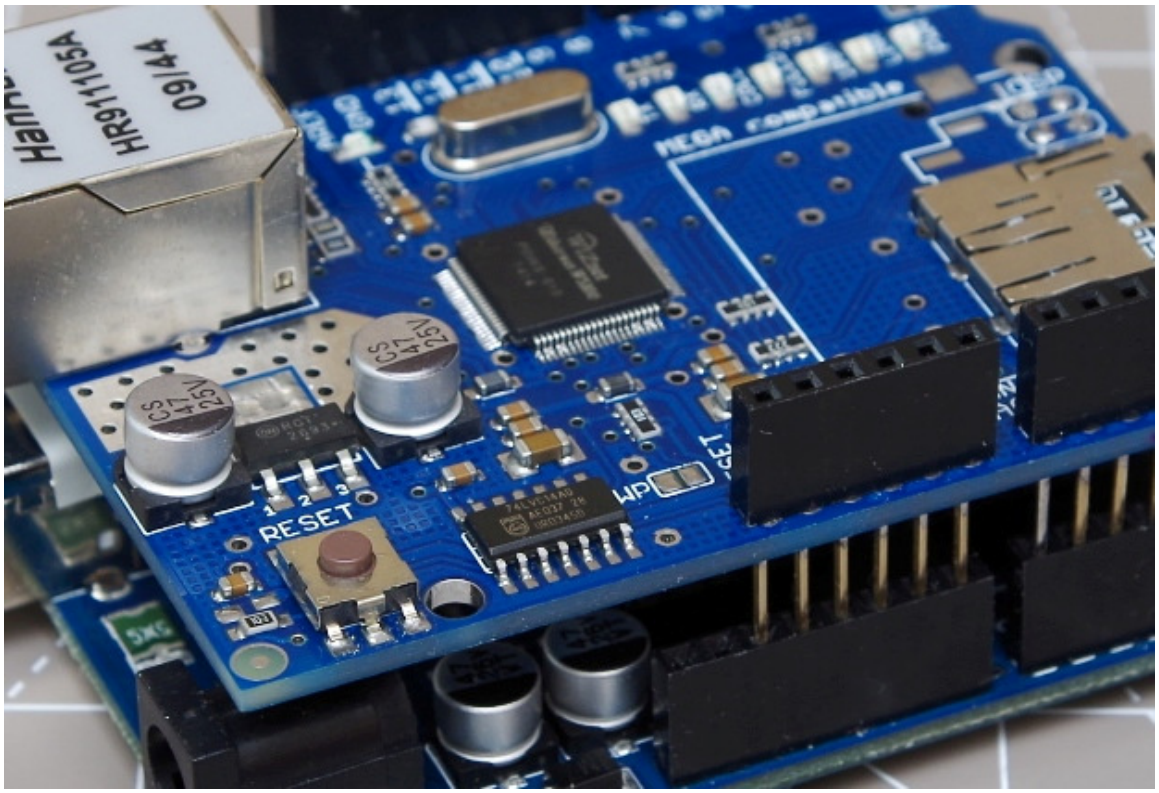
Code Storming

Arduino Ethernet MVC Web Server



By Kashif Baig

Arduino Ethernet MVC Web Server

The Arduino MVC web server has been tested to work with Arduino IDE v 1.8.2, Wiznet 5100 Ethernet shield, Arduino R3 Uno and Mega 2560, and boasts the following:

- implements model, view, controller design
- view builder with asp.net like scripting syntax
- memory efficient and responsive
- view markup can reside in program memory or SD card
- implements RESTful urls
- easy to process HTTP GET and POST requests
- easy to implement simple JSON or Xml web services
- browser caching for CSS, JavaScript, and images
- field by field or line by line processing of text data files

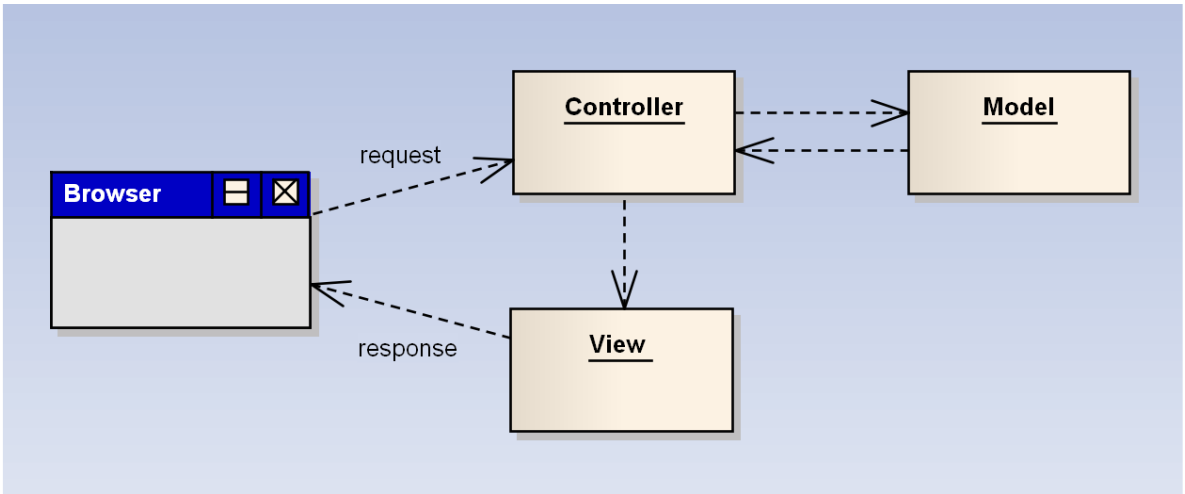
The unique advantage of using our Arduino web server is that web pages and web service responses can be built using a scripting style similar to asp.net. If you've developed web pages using server side scripting before, then this approach will be familiar. The source code is available at the link below, and is provided as is (with no warranty).

http://files.cohesivecomputing.co.uk/Arduino-Ethernet-MVC-Webserver.zip

Be sure to change the MAC and IP address in the source code to values that are correct for your network configuration, and copy 'web' and 'data' folders to the root of an SD card. Please watch the short demonstration video and read on before uploading the source code to your Arduino.

https://youtu.be/mpW0HVeAyIM

Model View Controller in a Nutshell



Browser requests are received by the controller, which analyses the request to determine what action to take. The controller then interacts with a model (responsible for web page / service / application logic), and renders an appropriate view (html or json) back to the browser.

Developing Model Classes

A model class is where you would write code to act on the request parameters and posted html form values. You might also develop code to read sensors, control actuators and perform file IO. Models are also used by view classes for generating responses to the browser.

Building Views with 'Server Scripting'

A view class is where you would write code to generate HTML or JSON markup. If you've developed web pages using asp or asp.net, then it will become apparent that using our online MVC View Builder will dramatically improve productivity in the development of complex web pages that use our Arduino MVC web server:

http://mvc-view-builder.cohesivecomputing.co.uk/

It is recommended that you use a text editor like notepad++ for developing the view HTML (or JSON) source, and name your files with a '.asp' file extension.

Paste the contents of your .asp file in to our online MVC view builder and press the Build View Code button. Copy and paste the output code in to a file in your Arduino project. If your view source depends on a Model class, be sure it exists (or develop it if it doesn't), and add the necessary code in the controller class method performAction() to render the view to the internet browser. Examine the Hello World project if you are not familiar with the MVC design pattern.

At the top of view .asp files are a number of directives which are described in the table below:

Directive	Description
@rem	Any single line remark or comment. Can appear multiple times.
@viewname	A valid C++ identifier used for generating view class names.
@inmemory	If true, view markup is stored in program memory. If false, a markup file is generated for storing on an SD card under folder 'web/'. Storing markup in memory offers the highest performance at the expense of large amounts of program memory.
@minify	If true, view markup is compacted by removing leading and trailing whitespace, and blank lines. For this to work, the correct content type directive must be specified.
@contenttype	Either html, xml or json
@modeltype	The C++ type used for the model. This directive is optional.
@include	A C++ file to include for compilation purposes. Directive can appear multiple times.

All view classes must inherit from the ViewBase class and implement the onRender() method.

A view doesn't need to generate the entire HTML or JSON response. It is possible to organise views in a modular way to achieve a good degree of code reuse, with performance critical views stored in program memory, and others on SD card. Examine the coding examples to see how to do this.

The Controller

The performAction() method in file Controller.h is where interaction with model and view classes occur. The argc and argv parameters represent the argument count and an array of variables representing parts of the request URL as follows:

http://<Arduino IP address>/arg0/arg1/arg2/ …

If parameter isHttpPost is true, you must continue to read from the request stream to process posted data until there are no more bytes to read (see coding examples). The performAction() method must write a HTTP response header before rendering a view. If a view has no accompanying application logic, it is not necessary to have a corresponding model class for it.

Serving Static Content

Static content such as CSS, JavaScript and image files must reside in the 'web/' folder of the SD card and must follow the FAT 8.3 file naming convention. In a browser these files are referenced as follows:

http://<Arduino IP address>/Content/<file.ext>

It recommended that large images and JavaScript libraries be referenced from content delivery servers to take the strain away from the Arduino.

What Next?

It should be possible to support cookies by implementing code to process them in the Controller class, and have the Controller class act as a mediator for the model classes.

Connect with me at uk.linkedin.com/in/kashifbaig if you find the Arduino web server useful or are seeking professional collaboration.