Arm Framework Documentation

Overview

The Arm framework is a lightweight, Model/View/Controller (MVC), web application framework, for the Lasso 9 language. Arm takes only minutes to setup, and construct, an MVC scaffold for your custom web application. Arm is also easy to upgrade within an existing Arm-based application, and further abstracts design from development with a unique HTML-only template structure.

Model/View/Controller (MVC) Structure

Model/View/Controller (MVC) is an architecture that separates the application logic, from the user interface, and the interaction with stored data. This permits independent development, testing, and maintenance of each area of concern. This MVC architecture is most apparent in the file structure. The following is a representation of a very basic arm framework scaffolding.

```
_controllers/
        error_cont.lasso
        home_cont.lasso
_libraries/
        app_yourapp.lasso
        arm.lasso
_models/
        home_mod.lasso
_templates/
        default.lasso
_views/
        error_view.lasso
        home_view.lasso
index.lasso
```

Determining Which Controller to Call

The arm framework relies on a URL rewrite mechanism, that directs all requests to a single file (index.lasso), and places the original request into a query argument with a name of “q” (the letter Q). The first segment of the original request path is used to determine which controller to call. For example, if the original request was for “/about”, the “about_cont” controller, stored at “/_controllers/about_cont.lasso”, would be called.

If no path segments are available, or the first path segment doesn't map to an existing controller, the -defaultpath, and -unknownpath, controllers are called respectively. To see how these paths are specified, see the [arm] type, under Object Reference below.

Setup

Arm takes only minutes to setup, and construct, an MVC scaffold for your custom web application.

1. Setup a URL-rewrite mechanism, using the method of your choice, such that all requests are silently redirected to the index.lasso file within your root application directory, and the original request path is appended as a query argument named “q”.

Using mod_rewrite in Apache 2.2, rewrite syntax would look something like the following.

```
RewriteEngine on
RewriteCond /file-path-to-root/%{REQUEST_FILENAME} !^.*\.LassoApp [NC]
RewriteCond /file-path-to-root/%{REQUEST_FILENAME} !-f
RewriteRule ^(.*)$ /index.lasso?q=$1 [PT,L,NC,NE,NS,QSA]
```

2. Open a window Terminal window, navigate to the directory that contains the ArmSetup script, and initiate the arm setup assistant by entering the following:

```
> sudo lasso9 ArmSetup
```

You will be prompted with several questions. When you have answered all of the questions, your Arm framework scaffold will be automatically created.

Object Reference

[arm]

Description

The core of the arm framework, the arm type is used to call the custom application. The completed response is returned when cast to a string.

Syntax

arm(-defaultpath='/_controllers/main_cont.lasso',
-unknownpath='/_controllers/404_cont.lasso')

arm(-defaultpath='/_controllers/main_cont.lasso',
-unknownpath='/_controllers/404_cont.lasso',
-dontcleanhtml)

Required Parameters

-defaultpath	The controller to call if no controller is specified by the request path.
-unknownpath	The controller to call if the controller specified by the request path does not exist.

Optional Parameters

-dontcleanhtml	Instructs arm to not minimize the resulting HTML. Useful during development, and when minimizing the output alters the semantics.

[arm_controller]

Description

The arm_controller is the progenitor of all controllers within an arm-based application. Neither it, nor any of it's descendants, will likely ever be called by custom written code.

Syntax

arm_controller

[arm_data]

Description

arm_data is a type used internally to store commonly needed values, used across many types and methods. arm_data is essentially a way around inheritance and composition as the only means to communicate.

Syntax

arm_data

[arm_data->set]

Description

This is a method to store a name/value pair in the arm_data type.

Syntax

arm_data->set(key=value)

Required Parameters

key/*value*	A pair, containing the value to be stored, and a key to reference it by later.

[arm_data->get]

Description

A method to retrieve a valued, stored in the arm_data type. Returns the associated value.

Syntax

arm_data->get(key)

Required Parameters

key	The reference key, associated with a value, set earlier.

[arm_model]

Description

The arm_model is the progenitor of all models within an arm-based application. It is likely an arm-

based web application will need to call one of its descendants, by custom written code within a controller.

Syntax

arm_model

[arm_requestpath]

Description

The arm_requestpath type provides easy access to the original request path. The original request path is returned when cast as a string.

Syntax

arm_requestpath

arm_requestpath(*arg*)

Optional Parameters

arg	Instructs arm_requestpath on the name of the get-argument that holds the original request path. Defaults to “q” when not specified. If the specified get-argument does not exist, the actual file path will be used in it's place.

[arm_requestpath->asarray]

Description

Returns the original request path as an array, with each path segment as a separate item within the array.

Syntax

arm_requestpath->asarray

[arm_requestpath->first]

Description

Returns the first segment of the original request path. Typically, this segment represents the controller to use.

Syntax

arm_requestpath->first

[arm_requestpath->get]

Description

Returns the path segment indicated by the provided index.

Syntax

arm_requestpath->get(*index*)

Required Parameters

index	The index of the path segment to be returned.

[arm_requestpath->insert]

Description

Alters the original request path. This can cause problems with the execution of an arm-based web application. The use case for this method should be very rare. Alters the object in place, and returns no value.

Syntax

arm_requestpath->insert(*value, index*)

Required Parameters

value	The value to be inserted into the request path. Requires a string.
index	The position to insert the new value. Requires an integer.

[arm_requestpath->last]

Description

Returns the last segment of the original request path.

Syntax

arm_requestpath->last

[arm_requestpath->replace(*value*, *index*)]

Description

Alters the original request path. This can cause problems with the execution of an arm-based web application. The use case for this method should be very rare. Alters the object in place, and returns no value.

Syntax

arm_requestpath->replace(*value, index*)

Required Parameters

value	The value to be inserted into the request path. Requires a string.
index	The existing position to replace with the new value. Requires an integer.

[arm_requestpath->second]

Description

Returns the second segment of the original request path.

Syntax

arm_requestpath->second

[arm_requestpath->size]

Description

Returns the number of segments in the original request path.

Syntax

arm_requestpath->size

[arm_requestpath->third]

Description

Returns the third segment of the original request path.

Syntax

arm_requestpath->third

[arm_view]

Description

The arm_view is the progenitor of all views within an arm-based application. It is likely an arm-based web application will need to call one of its descendants, by custom written code within a controller.

Syntax

arm_view

[arm_view->author]

Description

Replaces the content-value of any existing author meta-tag within the template. If no author meta-tag exists in the template, this method will have no effect.

Syntax

arm_view->author(*value*)

Required Parameters

value	The value to insert into the author-metadata. Must be of type string.

[arm_view->canonical]

Description

Replaces the href-value of any existing canonical link-tag within the template. If no canonical link-tag exists in the template, this method will have no effect.

Syntax

arm_view->canonical(*value*)

Required Parameters

value	The value to insert into the canonical link-tag href-attribute. Must be of type string.

[arm_view->defaultvalue]

Description

The value replacement tags within the template that are not otherwise utilized are replaced with. Defaults to an empty string, thus erasing unused replacement tags.

Syntax

arm_view->defaultvalue(*value*)

Required Parameters

value	The value used to replace unused replacement tags. Must be of type string.

[arm_view->description]

Description

Replaces the content-value of any existing description meta-tag within the template. If no description meta-tag exists in the template, this method will have no effect.

Syntax

arm_view->description(*value*)

Required Parameters

value	The value to insert into the description meta tag. Must be of type string.

[arm_view->javascript]

Description

Places an HTML <script> tag, with the indicated src-attribute, in the final HTML, just prior to the closing </body> tag. Can be called multiple times to place multiple <script> constructs into the final HTML. Does not effect any HTML <script> tags hard coded on the template.

Syntax

arm_view->javascript(-src='*file-path*', -attributes=map('*attribute-name*'='*value*'), -replace=true)

arm_view->javascript('*script*', -replace=true)

Required Parameters

-src	The URL for a javascript file. Not required if a script parameter is provided. Must be a type string.
script	The javascript code to be included in the final HTML. Should not include the surrounding HTML <script> tags. Not required is a -src parameter is provided. Must be a type string.

Optional Parameters

-attributes	Provides a means of adding arbitrary attributes to the HTML <script> tag. Only works with the -src form of the method. Must be a type map.
-replace	Instructs arm_view to replace, rather than append to, all prior uses of this method, with the contained script.

[arm_view->jquery]

Description

Places the passed script into a document.ready function. Can be called multiple times to place multiple

constructs into the document.ready function. Does not load jQuery itself, which should be either hard coded in the template, or called using a arm_view->javascript method.

Syntax

arm_view->jquery('script', -replace=true)

Required Parameters

script	The jQuery code to be included in the final document.ready function. Should not include the document ready function itself. Must be a type string.

Optional Parameters

-replace	Instructs arm_view to replace, rather than append to, all prior uses of this method, with the contained script.

[arm_view->keywords]

Description

Replaces the content-value of any existing keywords meta-tag within the template. If no keywords meta-tag exists in the template, this method will have no effect.

Syntax

arm_view->keywords(*value*)

Required Parameters

value	The value to insert into the keywords meta tag. Must be of type string.

[arm_view->style]

Description

Places an HTML <style> tag, with the indicated src-attribute, in the final HTML, just prior to the closing </head> tag. Can be called multiple times to place multiple <style> constructs into the final HTML. Does not effect any HTML <style> tags hard coded on the template.

Syntax

arm_view->style(-src='*file-path*', -replace=true)

arm_view->style('*style*', -replace=true)

Required Parameters

-src	The URL for a CSS file. Not required if a script parameter is provided. Must be a type string.

script	The CSS code to be included in the final HTML. Should not include the surrounding HTML <style> tags. Not required if a -src parameter is provided. Must be a type string.

Optional Parameters

-replace	Instructs arm_view to replace, rather than append to, all prior uses of this method, with the contained CSS.

[arm_view->template]

Description

Defines the template to use as a starting point in creating the final HTML document that is returned to the requesting client.

Syntax

arm_view->template('*file-path*')

Required Parameters

file-path	The file-path to an HTML template. Must be a string type.

[arm_view->title]

Description

Replaces the content of any existing title node within the template. If no title node exists in the template, this method will have no effect.

Syntax

arm_view->title(*value*)

Required Parameters

value	The value to insert into the title node. Must be of type string.

[arm_view->*_unknowntag*]

Description

Ad hoc methods that replace replacement-tags in the template – formatted as <!-- $method-name --> – with passed values, or HTML template fragments.

Syntax

arm_view->*method-name*(-fragment='*file-path*')

arm_view->*method-name*('*value*')

Required Parameters

value	The value to replace a replacement tag with in the template. Not required if a -fragment parameter is present. Must be a string type.

Optional Parameters

-fragment	Defines the file-path to an HTML fragment, to use as a template in a sub-view. After calling, the same method name can be used to access this fragment as if it was a view type of its own. Must be a string type.

Support

The arm framework is released without a guarantee of support. That being said, I encourage all questions regarding using the arm framework be directed to the LassoTalk mailing list. I will try my best to monitor that list for questions, between work and home time. If I don't respond within a reasonable period, please try contacting me at solutions@douglasburchard.com.

Roadmap

There is currently no specific roadmap for development of the arm framework. However, I have several projects developed using arm, and several in development. Arm will continue to evolve as a framework based on my own needs, and those needs expressed to me by other developers. If you have a suggestion, critism, or just a random thought, please post to the LassoTalk mailing list, or email me directly at solutions@douglasburchard.com. I will place everything into a queue, towards developing a future roadmap.

The core framework will remain lightweight. However, I foresee an arm-extensions library being added, or an API for multiple extension libraries. Such that pre-built functionality may be added to an application scaffold a la cart, at the time of initial setup.