LassoFusebox 3.0 Specification

Original by Hal Helms
Adapted for Lasso by Rich Tretola, April 5, 2003
Updated by Tami Williams, June 26, 2008

This document also contains an edited (by Tami Williams) version of:
*"**Fusebox 3 improves on its predecessor**"*
by Brian Kotek | May 01, 2002 7:00:00 AM
http://articles.techrepublic.com.com/5100-10878_11-1044897.html

*Also look at the "**Fusebox 3.0 Newbie Guide - PHP edition (DOC)**"*
http://bombusbee.com/downloads/files/FuseboxNewbieGuideV3PHP.zip

Adapted from:

Fusebox 3 improves on its predecessor

by Brian Kotek | May 01, 2002 7:00:00 AM

Source: http://articles.techrepublic.com.com/5100-10878_11-1044897.html

The flow of things

Here's a step-by-step description of what happens in a Fusebox application.

1. A user visits your site by going to http://www.mysite.com/.
2. Your web server's default page, index.lasso, is called.
3. Index.lasso includes the core frozen LassoFusebox file (fbx_LassoFusebox3).
4. The core frozen LassoFusebox file includes the fbx_Circuits file, which provides aliases and directory paths to all the circuits (subdirectories) nested beneath it.
5. The core frozen Fusebox file includes the fbx_Global file, which contains our global application settings. One of these settings defines a default fuseaction to use if none is specified (as in this case). It also includes the root fbx_Settings file, which contains other application variables .
6. The core frozen Fusebox file is given the default fuseaction of ContactApp.list_Contacts by the fbx_Global file.
7. The core frozen Fusebox file dissects the fuseaction. It sees that the first part (the circuit alias) is ContactApp. From previously parsing fbx_Circuits, it knows that ContactApp is the alias for the root, so it includes the fbx_Switch file in the root directory (the same directory as the core fbx_LassoFusebox3 file).
8. The fbx_Switch file looks through its case block for a match to the action "list_Contacts" (the second part of the default fuseaction).
9. After finding the match for the action "list_Contacts", the fbx_Switch file includes the following fuses:
   - qry_getAllContacts: finds all the contacts in the database.
   - dsp_AllContacts: displays a list view of the found records. It has links for adding, editing and deleting records.
10. The core frozen Fusebox file grabs the HTML output generated by the two fuses but does not output this to the user yet. Instead, it temporarily stores this output in a variable.
11. The core frozen Fusebox file looks in the fbx_Layouts file and includes a layout file containing a header, footer and a placeholder for any content generated by the circuits. It inserts the variable that is holding the output from the two fuses into this placeholder spot, essentially wrapping the header and footer around the content that has been generated.
12. The user is shown the final page output consisting of a header, a table of found contact records with links to add, edit and delete, and a footer, as one coherent page.

In an application with more than one circuit that also has nested layouts, here's what happens after step 12:

After the output from the qry_getAllContacts and dsp_AllContacts fuses is captured and temporarily stored in a variable, the core frozen Fusebox file looks for an fbx_Layout file in the child circuit's directory. If one exists, the core frozen Fusebox file uses it to find the child circuit's layout file, and inserts the output here, inside this layout. But it still doesn't output anything; it keeps all this data stored in a variable. Then the core frozen Fusebox file grabs the fbx_Layout file from the root directory (the one that provides the global header and footer) and wraps it around the content that was generated by this nested circuit. In the end, we would get a final page that has a header, the secondary layout elements, a table of found contact records with links to add, edit and delete, and a footer.

LassoFusebox

Introduction

Fusebox is a development methodology for web applications. Its name derives from its basic premise: a well-designed application should be similar to the fusebox in a house. In a fusebox, if one fuse is blown, the rest of the fuses continue to work. Each fuse has its own defined job, and Fuse A does its job without any help from Fuses B, C, or D. Similarly, in a Fusebox application, if one section of the application "breaks", the rest of the application should continue to work.

Fuseactions

In a Fusebox application, every action that an application must handle is referred to as a Fuseaction (aka Exit Fuseactions or XFAs). For example, an e-commerce application would include the following possible Fuseactions:

- viewCart
- viewCatalog
- addItem
- checkout

Rather than having fuseactions hard coded into exit points, like this:

```
<form name="contact" action="index.lasso?fuseaction=home.delete"  method="post">
```

Fusebox 3 replaces these hard coded references with XFAs:

```
<form name="contact" action="[$XFA_Delete]" method="post">
```

Naming Fuseactions

Each XFA is a compound fuseaction, consisting of the circuit name (the directory alias defined in fbx_Circuits), a dot separator, and the circuit request (instruction). The value of these fuseactions is set in the fbx_Switch file:

```
[Select: $fuseaction]
        [Case:'Welcome']
                [Var: 'XFA_Edit' = ($self) + 'home.edit']
                [Var: 'XFA_Delete' = ($self) + 'home.delete']
                [Var: 'XFA_New' = ($self) + 'home.new']
                [FBX_Include: 'qry_getContacts.inc]
                [FBX_Include: 'dsp_Contacts.inc']
        [Case]
                <p>This is the default case tag. I received a fuseaction called
                "[$fuseaction]" and I don't know what to do with it.</p>
[/Select]
```

Note: The variable, self, is not part of the Fusebox 3 specification, but is used by many developers to refer to the root circuit's default page and is set in the fbx_Settings file.

snippet from fbx_Settings.inc

```
[var: 'self' = 'index.lasso?fuseaction=' ]
```

The references to XFAs are resolved at run time, allowing for ease of reuse of both fuses and entire circuits.

Fuses

To perform each Fuseaction, a Fusebox application uses a series of Lasso pages. Each page is referred to as a Fuse. To perform the Fuseaction “viewCart” mentioned above, the user's cart must first be queried and then the results must be displayed. Therefore, the Fuses called for the Fuseaction “viewCart” would be:

- qry_getCart.inc
- dsp_Cart.inc

Types of fuses

Fuses can be grouped into a finite number of types. The most common are:

- Display fuses, prefixed with dsp_, which are used to show information to the user.
- Query fuses, prefixed with qry_, which contain SELECT, INSERT, UPDATE, DELETE or other types of queries.
- Action fuses, prefixed with act_, which contain any other type of activity (like sending email).
- Form fuses, prefixed with frm_, which contain forms.
- Validation fuses, prefixed with val_, which are used for form validation.
- Value List fuses, prefixed with vl_, which contain value list items.
- Ajax fuses, prefixed with ajax_, which are used for Ajax–specific code.

Frm_, val_, vl_, and ajax_ are not official Fusebox prefixes.

Additionally, layout files, which are not technically fuses, have a naming convention; they are prefixed with lay_.

Sample fbx_Switch file

We have used many new terms, which can be confusing. Before looking at the files used in a Fusebox application, here is a short sample of code from an fbx_Switch file, showing Fuseactions and Fuses in use. After reviewing, you should have a better idea of what Fuseactions and Fuses are.

```
[Select: $fuseaction]
        [Case: 'showInputForm']
                [Var: 'XFA_add' = ($self) + 'admin.add']
                [Var: 'XFA_edit' = ($self) + 'admin.edit']
                [Var: 'XFA_delete' = ($self) + 'admin.delete']
                [FBX_Include: 'frm_InputForm.inc']
        [Case: 'showEmployeeList']
                [FBX_Include: ' qry_EmployeeList.inc']
                [FBX_Include: ' dsp_EmployeeList.inc']
        [Case]
                <p>This is the default case tag. I received a fuseaction called
                "[$fuseaction]" and I don't know what to do with it.</p>
[/Select]
```

Explanation of the sample fbx_Switch file

In the above file, the Fuseactions are “showInputForm” and “showEmployeeList”. (Don't worry about the [Var: 'XFA_… = …] statements in the “showInputForm” fuseaction, they will be explained later.) The FBX_Include statements are the Fuses for the Fuseactions. The Fuseaction that is required can be passed to the application through a URL variable.

Core Files

Fusebox 3 is built around a library of core files, prefixed with fbx_. These include:

- fbx_LassoFusebox3.inc: A non-editable file which contains the main Fusebox code.
- fbx_Library.inc: Adds custom tags that are needed by the Fusebox core file. It is included within the core file. This file is unique to the Lasso port of Fusebox 3.0.
- fbx_CustomTags.inc: This file is provided for the user to add their own custom tags. This file is unique to the Lasso port of Fusebox 3.0.
- fbx_Sessions.inc: Extends the Fusebox core file to allow Lasso sessions to operate properly.
- fbx_Circuits.inc: This file provides circuit-to-directory mappings.
- fbx_Globals.inc: This file allows default, application-wide variables to be set in the root circuit.
- fbx_Settings.inc: This file allows variables to be set at each circuit. Variables set by parent circuits are inherited by their descendants and may be overridden.
- fbx_Switch.inc: This file processes the fuseaction sent to the application.
- fbx_Layouts.inc: This file determines the layout file to be used for a particular circuit.

fbx_LassoFusebox3.inc

This file should be called by your root circuit's default file (index.lasso, default.lasso, etc.) like this:

```
[Include: 'fbx_LassoFusebox3.inc']
```

The most important thing to keep in mind when using this file is that it is the engine of the application. It controls the entire application, based on the values you set in other files.

At a very high-level, this file does the following:

- Sets a number of variables and initializes some structures.
- Includes the fbx_Library file and the fbx_CustomTags file.
- Converts all action_params to variables.
- Includes the fbx_Sessions file.
- Includes the fbx_Circuits file, which translates the folder structure of the application to a simple structure named Circuits. This makes referring to folders, subfolders and sub-subfolders very easy.
- Creates a reverse lookup of the circuits just defined.
- Includes the fbx_Globals file from the root application folder.
- Gets the Fuseaction and the circuit that was sent to the application.
- Includes the fbx_Settings file, from the root application folder, then includes the fbx_Settings files from all the subfolders in top to bottom order. This allows child fbx_Settings files to overwrite values set in the parent fbx_Settings file.
- Executes the Fuseaction in the correct circuit's fbx_Switch file using the values stored in the $fuseaction and $_Circuits variables. All output is stored in the $layout variable.
- Includes the fbx_Layouts files from all the subfolders in bottom to top order. These files call the individual layout files, which output the contents of the $layout variable.

fbx_Library.inc

The fbx_Library file adds custom tags that are needed by the Fusebox core file. It is included within the core file. This file is unique to the Lasso port of Fusebox 3.0.

fbx_CustomTags.inc

The fbx_CustomTags file was added to allow users to add their own custom tags. This file is unique to the Lasso port of Fusebox 3.0.

fbx_Sessions.inc

The fbx_Sessions file was added to allow Lasso sessions to work properly within the Fusebox architecture. This file should be located within the root directory and all sessions should be declared within this file.

fbx_Circuits.inc

This file provides directory to circuits mapping. It aliases all the subfolders and sub-subfolders in the application. The reserved structure $_Circuits contains key/value pairs where the key is the circuit alias and the value is the directory path beginning with the root circuit.

The fbx_LassoFusebox3 file uses the fbx_Circuits file to resolve compound fuseaction names. Note that the circuit names do not need to be the same as the directory names. There is only one fbx_Circuits file in an application.

The following is a sample of the code in the fbx_Circuits file:

```
[$_Circuits->(Insert:'Home'='myApp')]
[$_Circuits->(Insert:'Admin'='myApp/Administration')]
[$_Circuits->(Insert:'Rules'='myApp/Administration/Rules')]
[$_Circuits->(Insert:'Front' = 'myApp/FrontEnd')]
```

This allows all references to specific folders to be made using aliases. To call the Fuseaction named “myFuseaction” in the folder myApp/Administration/Rules, you would call the index.lasso file as follows:

```
index.lasso?fuseaction=Rules.myFuseaction
```

fbx_Globals.inc

The fbx_Globals file is used by the Fusebox application to set the default fuseaction and other application-wide default values. There must be only one fbx_Globals file and it must reside in the root folder of the application.

fbx_Settings.inc

The fbx_Settings file is used by the Fusebox application to set default values. There must be one fbx_Settings file in the root folder of the application. An individual circuit can have it’s own fbx_Settings file if default values need to be set for files in that circuit, or if a child circuit needs to overwrite a default value in a parent circuit.

fbx_Switch.inc

The fbx_Switch file is probably the simplest of the core files. It decides what files to include on the page, based on the Fuseaction.

Here is a sample of an fbx_Switch file:

```
[Select:$fuseaction]
        [Case:'Welcome']
                [FBX_Include: ($self) + 'qry_getContacts.inc']
                [FBX_Include: ($self) + 'dsp_Contacts.inc']
        [Case]
        <p>This is the default case tag. I received a fuseaction called "[$fuseaction]"
        and I don't know what to do with it.</p>
[/Select]
```

fbx_Layouts.inc

It is often desirable to have a different 'look' for each section of a Web application. Controlling the look of each circuit is the purpose of the fbx_Layouts file. There should be at least one fbx_Layouts file in an application. However, it can be safely omitted, in which case, no layout will be applied. Any circuit that has its own layout requirements should have its own fbx_Layouts file.

An fbx_Layouts file is responsible for setting two variables, $layoutDir and $layoutFile.

$layoutDir points to a directory (if it exists) in which layout files are kept. $layoutFile points to the file to be used for the layout. If you create a layout file, it must, at minimum, reference the variable, $layout.

LassoFusebox 3 API

Fusebox 3 exposes a series of variables to help in writing code:

Public API Variable	Type	May be set?	Description
Fusebox.isRootCircuit	BOOLEAN	NO	Is the directory of the file currently being operated on by fbx_LassoFusebox3 the same as the root circuit of the app?
Fusebox.isTargetCircuit	BOOLEAN	NO	Is the directory of the file currently being operated on by fbx_LassoFusebox3 the same as the target circuit of the app?
Fusebox.fuseaction	STRING	NO	The simple fuseaction extracted from the attributes.fuseaction of form circuit.fuseaction passed in
Fusebox.circuit	STRING	NO	The simple circuit extracted from the attributes.fuseaction of form circuit.fuseaction passed in
Fusebox.rootCircuit	STRING	NO	The circuit alias of the root circuit of the app
Fusebox.targetCircuit	STRING	NO	The circuit alias of the target circuit of the app. Usually this is the same as Fusebox.circuit ($circuit) unless you've made a circuits definition error
Fusebox.thisCircuit	STRING	NO	The circuit alias of the directory of the file currently being operated on by fbx_LassoFusebox3
Fusebox.LayoutFile	STRING	YES	The layout file to be applied to this circuit after the fuseaction and its fuse(s) are done creating their content
Fusebox.LayoutDir	STRING	YES	The relative path from the target circuit to where the layout file to be applied to this circuit is located. If no special layout directory has been created, this should be set to an empty string
Fusebox.CurrentPath	STRING	NO	The relative directory path of the file currently being operated on by fbx_LassoFusebox3, relative from the root of the application (i.e. the root circuit). Example: dir1/dir2/dir3/
Fusebox.RootPath	STRING	NO	The relative directory path of the file currently being operated on by the core frozen Fusebox code, relative to the root of the application (i.e. the root circuit). Example: ../../../
Fusebox.layout	STRING	NO	This variable captures the content generated to that point in preparation for wrapping a layout file around it (as defined by Fusebox.layoutFile). This variable must be inside each layout file in order for content to be passed up to the next level of nested layouts

Fusedoc

Fusedoc is a documentation system/program definition language for documenting fuses. The Fusedoc uses XML syntax wrapped in a Lasso comment and is normally place on top of the fuse file itself. Here is a sample Fusedoc for a display login file:

```
<fusedoc fuse="frm_login.lasso" language="Lasso" version="3.0">
    <responsibilities>I am a form that lets users log in.</responsibilities>
    <properties>
        <history author="Rich Tretola" date="12/10/2002" role="Architect" type="Create" />
    </properties>
    <io>
        <in>
            <string name="self" />
            <string name="XFA_submitForm" />
        </in>
        <out>
            <string name="fuseaction" scope="formOrUrl" />
            <string name="userName" scope="formOrUrl" />
            <string name="password" scope="formOrUrl" comments="password field" />
        </out>
    </io>
</fusedoc>
```

The Fusedoc XML root element has three sub-elements: responsibilities, properties, and io. Of these, only responsibilities is required.

Responsibilities use the first person to have the fuse describe what it is responsible for. Properties is a more generic, catchall section that has three possible sub-elements: history, property, and note. IO (short for input/output) contains in and out sub-elements.

Fusedoc: responsibilities

Each Fusedoc will have a plain English explanation as to what the Fuse is meant to do. This tag does not have any attributes, the English explanation goes in between the opening and closing tags.

Fusedoc: properties

1. Fusedoc: properties: history
The history element has the following attributes:

- author: free text
- date: free text
- email: free text
- role: free text
- type: create | update

The history tag can be used as a tagset with free text between the tags:

```
<history author="Rich Tretola">
This is just a sample valid history element.
</history>
```

2. Fusedoc: properties: property
The property element has the following attributes:

- name: free text
- value: free text

The property tag is an empty tag set.

3. Fusedoc: properties: note
The note element has the following attributes:

- author: free text
- date: free text

The note tag can be used as a tagset with free text between the tags:

```
<note author="Rich Tretola">
This is just a sample valid note element.
</note>
```

Fusedoc: io

4. Fusedoc: io: in/out
Both in and out elements have no attributes. Both elements accept the following sub-elements:

- string
- number
- boolean
- list
- structure
- array
- recordset
- cookie
- datetime
- file

a) Fusedoc: io: in/out: string
The string element has the following attributes:

- name: free text
- scope: application | attributes | caller | cgi | client | form | formorurl | request | server | session | url | variables
- comments: free text
- default: free text
- mask: free text
- oncondition: free text
- optional: true | false

b) Fusedoc: io: in/out: number
The number element has the following attributes:

- name: free text
- scope: application | attributes | caller | cgi | client | form | formorurl | request | server | session | url | variables

- comments: free text
- default: free text
- precision: integer | decimal
- oncondition: free text
- optional: true | false

c) Fusedoc: io: in/out: boolean
The boolean element has the following attributes:
- name: free text
- scope: application | attributes | caller | cgi | client | form | formorurl | request | server | session | url | variables
- comments: free text
- default: free text
- oncondition: free text
- optional: true | false

d) Fusedoc: io: in/out: list
The list element has the following attributes:
- name: free text
- scope: application | attributes | caller | cgi | client | form | formorurl | request | server | session | url | variables
- delims: comma | free text
- comments: free text
- oncondition: free text
- optional: true | false
- default: free text

e) Fusedoc: io: in/out: structure
The structure element has the following attributes:
- name: free text
- scope: application | attributes | caller | cgi | client | form | formorurl | request | server | session | url | variables
- comments: free text
- oncondition: free text
- optional: true | false

The structure element contains one or more of the following sub-elements:
- string
- number
- boolean
- datetime
- array
- structure
- recordset
- list

f) Fusedoc: io: in/out: array
The array element has the following attributes:
- name: free text
- scope: application | attributes | caller | cgi | client | form | formorurl | request | server | session | url | variables
- comments: free text
- oncondition: free text
- optional: true | false

The array element contains one or more of the following sub-elements:

- string
- number
- boolean
- datetime
- array
- structure
- recordset
- list

g) Fusedoc: io: in/out: recordset

The recordset element has the following attributes:

- name: free text
- scope: application | attributes | caller | cgi | client | form | formorurl | request | server | session | url | variables
- primarykeys: free text
- mask: free text |
- comments: free text
- oncondition: free text
- optional: true | false

h) Fusedoc: io: in/out: cookie

The cookie element has the following attributes:

- name: free text
- domain: free text
- expires: free text
- path: free text
- secure: true | false
- value: free text
- comments: free text
- oncondition: free text
- optional: true | false

i) Fusedoc: io: in/out: datetime

The datetime element has the following attributes:

- name: free text
- scope: application | attributes | caller | cgi | client | form | formorurl | request | server | session | url | variables
- mask: free text |
- comments: free text
- default: free text
- oncondition: free text
- optional: true | false

j) Fusedoc: io: in/out: file

The file element has the following attributes:

- path: free text
- action: read | write | append | overwrite | delete | exists | module | include
- comments: free text
- oncondition: free text
- optional: true | false