Fusebox 3 Help

Introduction

At first glance, Fusebox 3 may seem quite different from Fusebox 2. Certainly, there are some notable differences, but the core philosophy and concepts from Fusebox 2 remain, enhanced and extended.

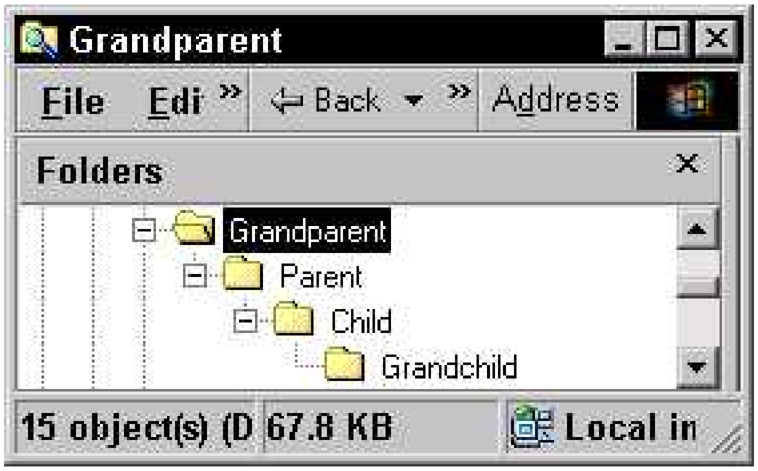
Fusebox 3 introduces/formalizes several key features. These include:

- a nested model for communication between circuits
- a nested model for layouts
- XML-based documentation/program definition language (Fusedoc)
- a defined core set of key Fusebox files
- mechanism for parameterizing exit fuseactions (XFAs)
- a defined structure ("Fusebox") that serves as a public API
- a directory-to-circuit mapping scheme
- a compound fuseaction syntax (*circuitname.circuitrequest*)

Fusebox 3 Help

Nested Circuits

In Fusebox 2, circuits were largely independent of each other. In Fusebox 3, this changes. Circuits can now exist in a parent/child relationship, with the child inheriting from the parent. Nesting can be as deep as is needed. Here is an example of a very simple application that has circuits nested four deep:



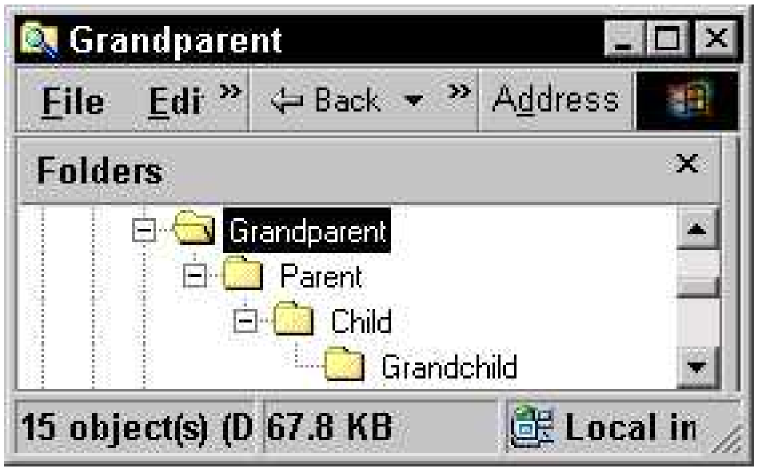
Circuits are defined through the FBX_Circuits.cfm file. This mechanism allows for nesting of layouts, as well.

[Home]

Fusebox 3 Help

Nested Layouts

In Fusebox 3, the optional FBX_Layouts.cfm file defines what, if any layout files, should be called by each circuit. As an example, take the following application directories layout:



If there is a call to grandchild.helloWorld, the core Fusebox file, FBX_Fusebox_CFxx.cfm, first traverses down the circuit structure from grandparent to grandchild, executing any FBX_Settings.cfm files available in the "fuseaction path". Then it calls the grandchild's FBX_Switch.cfm file and executes that code, which is not output, but is rather saved into the variable, Fusebox.layout. This variable is then available to the layout file (if any) specified in grandchild's FBX_Layouts.cfm file.

The contents of this code is not output, but is saved into the variable, Fusebox.layout, overwriting what was there. It is then passed up the directory to the child circuit. The child can have its own layout file, which, once executed, is available as Fusebox.layout -- and so on right up the ladder. If no Fusebox.layout file exists, FBX_Fusebox_CFxx.cfm simply outputs Fusebox.layout.

[Home]

Fusebox 3 Help

Fusedoc

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

```
<fusedoc fuse="dsp_login.cfm" language="ColdFusion" version="2.0">
  <responsibilities>
    I am a form that lets users log in.
  </responsibilities>
  <properties>
    <history author="hal helms" date="22Oct2001" 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, the only one required is responsibilities.

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

[Home]

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="hal helms">
 This is just a sample valid history element.
</history>
```

[Back to Fusedoc]

Fusebox 3 Help

Fusedoc: properties: property

The property element has the following attributes:

- **name**: free text
- **value**: free text

The property tag is an empty tag set:

[Back to Fusedoc]

Fusedoc: properties: note

The history 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="hal helms">
 This is just a sample valid note element.
</note>
```

[Back to Fusedoc]

Fusebox 3 Help

Fusedoc: io: in

Fusedoc: io: 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

[Back to Fusedoc]

Fusedoc: io: in: string

Fusedoc: io: 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
- **format**: **cfml** | wddx
- **mask**: free text
- **oncondition**: free text
- **optional**: true | **false**

[Back to IO]

Fusedoc: io: in: number

Fusedoc: io: 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
- **format**: **cfml** | wddx
- **precision**: integer | decimal
- **oncondition**: free text
- **optional**: true | **false**

[Back to IO]

Fusebox 3 Help

Fusedoc: io: in: boolean

Fusedoc: io: 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**

[Back to IO]

Fusedoc: io: in: list

Fusedoc: io: 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
- **format**: **cfml** | wddx
- **oncondition**: free text
- **optional**: true | **false**
- **default**: free text

[Back to IO]

Fusedoc: io: in: structure

Fusedoc: io: 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
- **format**: **cfml** | wddx
- **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

[Back to IO]

Fusedoc: io: in: array

Fusedoc: io: 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
- **format**: **cfml** | wddx
- **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

[Back to IO]

Fusedoc: io: in: recordset

Fusedoc: io: 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
- **format**: **cfml** | wddx
- **oncondition**: free text
- **optional**: true | **false**

[Back to IO]

Fusedoc: io: in: cookie

Fusedoc: io: 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**

[Back to IO]

Fusedoc: io: in: datetime

Fusedoc: io: 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**

[Back to IO]

Fusebox 3 Help

Fusedoc: io: in: file

Fusedoc: io: out: file

The file element has the following attributes:

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

[Back to IO]

Fusebox 3 Help

Core Files

Fusebox 3 introduces several core files, prefixed with FBX_. These include:

- **FBX_Fusebox_CF*xx*.cfm**: (where *xx* represents the version of ColdFusion server supported.) Replaces many of the custom tags used in Fusebox 2. This file should be treated as "frozen" and should not be changed if you wish your applications to be Fusebox 3-compliant.
- **FBX_Circuits.cfm**: This file provides circuit-to-directory mappings.
- **FBX_Layouts.cfm**: This file determines the layout file to be used for a particular circuit.
- **FBX_Switch.cfm**: This file processes the fuseaction sent to the application.
- **FBX_Settings.cfm**: This file allows variables to be set at each circuit. Variables set by parent circuits are inherited by their descendants and may be overridden.

[Home]

Fusebox 3 Help

FBX_Fusebox_CF*xx*.cfm

This file should be called by your home circuit's default file. If you know the version of ColdFusion you're running, you can either call it directly, like this...

```
<cfinclude template="FBX_Fusebox_CF50.cfm">
```

...or you can allow ColdFusion to determine this for you:

```
<!--- include the core FuseBox --->
<cflock scope="SERVER" timeout="2">
  <cfset serverVersion = Val( ListGetAt( Server.ColdFusion.ProductVersion, 1 ) )>
  <cfset dotVersion = Val( ListGetAt( Server.ColdFusion.ProductVersion, 2 ) )>
</cflock>

<cfif serverVersion LT "5">
  <cfif dotVersion is "5">
    <cfinclude template="fbx_fusebox30_CF45.cfm">
  <cfelse>
    <cfinclude template="fbx_fusebox30_CF40.cfm">
  </cfif>
<cfelse>
  <cfinclude template="fbx_fusebox30_CF50.cfm">
</cfif>
```

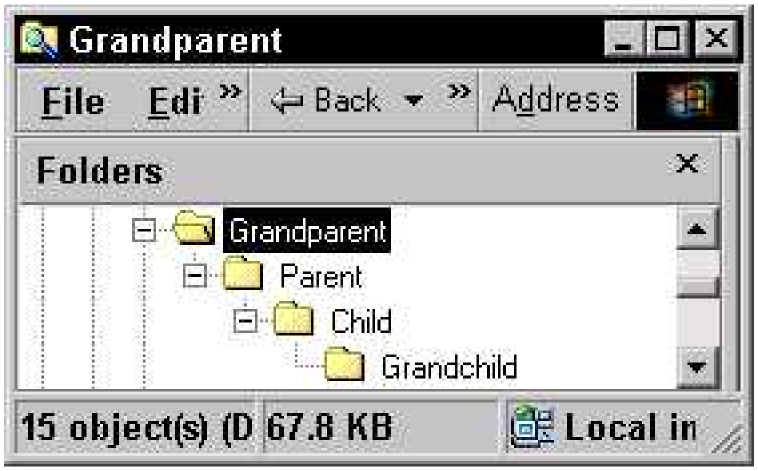
[Back to Core files]

Fusebox 3 Help

FBX_Circuits.cfm

This file provides directory to circuits mapping. The reserved structure Fusebox.Circuits contains key/value pairs where the key is the circuit name and the value is the directory path beginning with the home circuit.

A sample FBX_Circuits.cfm file is provided for the following application:



```
<cfscript>
   Fusebox.Circuits.home = Grandparent;
   Fusebox.Circuits.parent = Grandparent/Parent;
   Fusebox.Circuits.child = Grandparent/Parent/Child;
   Fusebox.Circuits.grandchild = Grandparent/Parent/Child/Grandchild;
</cfscript>
```

The FBX_Fusebox30_CFxx.cfm file uses the FBX_Circuits.cfm file in resolving compound fuseaction names.

Note that the circuit names do not need to be the same as the directory names.

[Back to Core files]

Fusebox 3 Help

FBX_Layouts.cfm

This file is responsible for setting two variables, Fusebox.layoutDir and Fusebox.layoutFile.

Fusebox.layoutDir points to a directory (if it exists) in which layout files are kept. Fusebox.layoutFile points to the file to be used for layout. This file can be safely omitted, in which case, no layout will be applied. If you create a layout file, it must, at minimum, dereference the variable, Fusebox.layout.

Here is an example layout file that applies a copyright at the bottom of the page:

```
<cfoutput>
  #Fusebox.layout#
</cfoutput>
<br>
<cfinclude template="dsp_copyrightInfo.cfm">
```

[Back to Core files]

Fusebox 3 Help

FBX_Switch

Fusebox 3 separates fuseaction handling into this separate file. Other than that, it is identical to what was used in Fusebox 2.

Here is a sample snippet from an FBX_Switch.cfm file:

```
<cfswitch expression="#Fusebox.fuseaction#">
   <cfcase value="login">
      <cfset XFA.register = "Guest.new">
      <cfset XFA.submitForm = "Login.validate">
      <cfinclude template="dsp_login.cfm">
   </cfcase>
</cfswitch>
```

Note that in Fusebox 3, the <cfswitch> statement evaluates the variable Fusebox.fuseaction.

[Back to Core files]

Fusebox 3 Help

FBX_Settings.cfm

This file replaces app_globals.cfm, app_locals.cfm, and myGlobals.cfm. This file would typically be used to do such things as run the <cfapplication> tag, set the variable, self, set a datasource, etc.

[Back to Core files]

Fusebox 3 Help

Exit Fuseactions (XFAs)

Fusebox 3 introduces the concept of exit fuseactions. Rather than having fuseactions hardcoded into exit points, like this...

```
<form action="index.cfm?fuseaction=Users.addUser">
```

...Fusebox 3 replaces these hardcoded references with XFAs:

```
<form action="index.cfm?fuseaction=#XFA.submitForm#">
```

The value of these fuseactions is then set in the FBX_Switch.cfm file:

```
<!--- snippet from FBX_Switch.cfm file --->
<cfswitch expression="#Fusebox.fuseaction#">
   <cfcase value="addUser">
      <cfset XFA.submitForm = "Users.addUser">
      <cfinclude template="dsp_NewUser.cfm">
   </cfcase>
</cfswitch>
```

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

[Home]

Fusebox 3 API

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

Public API variable	Type	May be set?	Description
Fusebox.isCustomTag	BOOLEAN	NO	Is this fusebox being called as a custom tag?
Fusebox.isHomeCircuit	BOOLEAN	NO	Is the directory of the file currently being operated on by FBX_Fusebox_CFxx.cfm the same as the home circuit of the app?
Fusebox.isTargetCircuit	BOOLEAN	NO	Is the directory of the file currently being operated on by FBX_Fusebox_CFxx.cfm 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.homeCircuit	STRING	NO	The circuit alias of the home 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 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_Fusebox_CFxx.cfm
Fusebox.thisLayoutFile	STRING	YES	The layout file to be applied to this circuit after the fuseaction and its fuse(s) are done creating their content
Fusebox.thisLayoutDir	STRING	YES	The relative path from the target circuit 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_Fusebox_CFxx.cfm, relative from the root of the application (ie the home 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 (ie the home circuit). Example: ../../../

Fusebox.layout	STRING	NO	This is the variable used by cfsavecontent or its equivalent custom tag that captures the generated content 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
Fusebox.SuppressErrors	BOOLEAN	YES	FBX_Fusebox_CFxx.cfm has some abilities built in to suppress native ColdFusion errors and instead give its best guess at what it wrong with your application (as it relates to Fusebox). Default value is always FALSE, which therefore generates native ColdFusion errors. You may want to set it to TRUE while you set up your application in Fusebox style and let it default to FALSE when you're confident that your Fusebox is set up correctly but are testing for coding errors in your fuses unrelated to Fusebox.

[Home]

Fusebox 3 Help

Naming Fuseactions

The syntax for naming fuseactions has changed. In Fusebox 2, the fuseaction was simply the circuit request. Under Fusebox 2, a link to add an item to a ShoppingCart circuit might look like this:

```
<a href="../ShoppingCart/index.cfm?fuseaction=addItem">
```

In Fusebox 3, an exit fuseaction for the fuse is defined in FBX_Switch.cfm. The value of that XFA is a compound fuseaction, consisting of the circuit name, a dot separator, and the circuit request.

```
<---snippet from FBX_Switch.cfm --->
<cfcase value="xxx">
   <cfset XFA.buyItem="ShoppingCart.addItem>
   <cfinclude template="xxx.cfm">
</cfcase>
```

```
<---snippet from xxx.cfm --->
<a href="#self#?fuseaction=#XFA.buyItem#">
```

The circuit name refers to the directory alias defined in FBX_Circuits.cfm.

NOTE: The variable, self, is not part of the Fusebox 3 specification, but is used by many developers to refer to the home circuit's default page.

[Home]