Fusebox 5.5 XML Cheat Sheet

The XML changes in Fusebox 5.0 mostly brought a standard and more solid feel to the Fusebox XML language. Fusebox 5.5 brought the ability to not use the XML grammar, so there are very few changes between 5.0 and 5.5. Changes between Fusebox 5.0 and 5.1 grammar and 5.1 and 5.5 grammar are outline at the end of this document.

Fusebox.xml

circuits

Defines folders of fuses and their relationships with each other.

```
<circuits>
  <circuit alias="any_name" path="my_circuit" parent(optional)/>
  ...
</circuits>
```

classes

Defines objects that are used in the application. Does not create any instances of them. Considered not the hot thing to use since ColdSpring and other similar bean factories.

```
<classes>
  <class alias="Employee" classpath="com.company.Employee" type="component"
[constructor="init"]>
  ...
</classes>
```

parameters

name/value pairs for important fusebox parameters

```
<parameters>
  <parameter name="fuseactionVariable" value="fuseaction" />
  <parameter name="defaultFuseaction"  value="main.welcome" />
  <parameter name="precedenceFormOrUrl" value="form"/>
  <parameter name="mode"
value="development-circuit-load|development-full-load|production"/>
  <parameter name="password" value="secure"/>
  ...
</parameters>
```

globalfuseactions

Defines any fuses that are executed at the beginning and ending of every request to the fusebox app.

```
<globalfuseactions>
  <appinit>
    <fuseaction action="main.appinit" />
    <fuseaction action="blog.start" />
  </appinit>
  <preprocess />
  <postprocess />
</globalfuseactions>
```

plugins

Includes files at specified places in the processing of a request. Put plugin cfm files in the plugins folder.

```
<plugins>
  <phase name="preProcess">
    <plugin name="pluginName" template="file" />
  </phase>
  <phase name="preFuseaction" />
  <phase name="postFuseaction" />
  <phase name="fuseactionException" />
  <phase name="postProcess" />
  <phase name="processError" />
</plugins>
```

Circuit.xml

circuit

Defines a circuit for organizing fuseactions

```
<circuit access="public|private|internal" xmlns:lexicon="path/to/lexicons/">
  <prefuseaction />
  <fuseaction name="myFA" />
  ...
  <postfuseaction />
</circuit>
```

fuseaction

Defines a fuseaction for direct calling via URL or do action.

```
<fuseaction name="myFA" access="public|internal|private" permissions="custom value">
  <do>, <set>, <include> etc ...
</fuseaction>
```

do

Executes a fuse like a new request. Beware of recursion when dealing with pre/post fuseactions.

```
<do action="circuit.fuseaction|localFuseaction" [contentvariable="content"]
[append="true|false" for contentvariable] [overwrite="true|false" for contentvariable]>
  [<parameter name="paramName" value="#paramVal#" /> parameters exist only for the life
of this action]
</do>
```

In order to make the do action a dynamic value, you have to invoke a dynamic do:

<invoke object="myFusebox" methodcall="do('circuit.fuseaction'[, contentvariable][, append][, returnOutput])" />

if

Conditional processing. Somewhat limited in the way that there is no 'else if'. Include one or both true/ false blocks in any order.

```
<if condition="myVar GT 4">
  <true>
    <do action="something"/>
  </true>
  <false/> <!-- not required -->
</if>
```

include

Includes a file, aka a Fuse. File should be from the current folder (circuit). Does not require the file extension, but is recommended.

```
<include template="myfile" [required="true|false" throw error if not found]
[contentvariable="content"] [append="true|false" for contentvariable]
[overwrite="true|false" for contentvariable] [circuit="circuit" for implicit circuits]>
  [<parameter name="paramName" value="#paramVal#" /> parameters exist only for the life
of this action]
</include>
```

The syntax for including a file through an implicit circuit is `<include circuit="circuitName" template="fileName[.cfm]" />`

instantiate

Creates an instance of an object referenced in fusebox.xml's classes section. This is frequently not used in favor of ColdSpring or other abstract bean factories.

```
<instantiate object="application.blog" class="blog (defined in fusebox.xml classes)">
  <argument value="#val#" /> (numbered arguments for init method)
  <argument name="" value="" /> (named arguments for init method)
</instantiate arguments="#params#">
```

invoke

Invokes a method from an object that has been instantiated. Does not implicitly rely on the <instantiate> tag. Sometimes this is not used in favor of `<set value="#component.method()#" />`.

```
<invoke [object|class|webservice]="reference" method="myMethod"
[returnvariable="myObject"]>
  <argument value="#val#" /> (numbered arguments)
  <argument name="arg" value="#val#" /> (named arguments)
</invoke>
```

loop

Creates a simple loop.

```
<loop condition="i lt 3">
<loop query="myQuery">
<loop list="apples,tomatoes,lemons,cheese" index="i">
<loop from="1" to="100" index="i">
<loop collection="#myStruct#">
```

relocate

Forwards the user to another request.

```
<relocate [url|xfa="location"] [addtoken="true|false"]
[type="client|server|javascript|moved"]>
```

set

Sets a variable. Requires #'s for dynamic values.

```
<set name="variables.name" value="value" [overwrite="true|false"] />
```

xfa

Defines exit points, can be used in href= or form action=, etc., or in the <relocate> tag. XFAs should either be in the format of "circuit.fuse" or "fuse" if pointing to the current circuit.

```
<xfa name="myXfa" value="circuit.fuse" [overwrite="true|false"]>
```

prefuseaction

Defines the section for events that fire before any fuses from the circuit are called. Beware of possible recursion issues, for example, if a prefuseaction specifies a <do> verb with an action in the same circuit, the prefuseaction will fire again before it calls the action, which will call the do verb again, recursing forever.

postfuseaction

Defines the section for events that fire after any fuses from this circuit are called. Again, beware of possible recursion issues.

Fusebox 5.1 General Notes

Lexicons (custom tags)

Define with `<circuit xmlns:myLex="myLex/" xmlns:cf="cf/">`
Use with `<myLex:myTag attribute1="abc" />` only within the circuit where the lexicon has been declared.
This above example looks for the file approot/lexicons/myLex/myTag.cfm

XFAs

Defines an exit point for controlling a page's outbound links. It is typical to define XFAs in a prefuseaction.

XML:

`<xfa name="home" value="main.welcome" />`
`<xfa name="save" value="userFormSave" />` (missing "circuit.", so fuseaction must be in the same circuit)

In the display file:
`<form action="#myself##xfa.save#">`
`<a href="#myself##xfa.home#>home</a>`

variables.myFusebox

Contains: thisCircuit, thisFuseaction, originalCircuit, originalFuseaction

fusebox.init.cfm

Should consist of (at minimum):

```
self = "index.cfm";
myself = "#self#?#application.fusebox.fuseactionVariable#=";
```

A display file could then have `<a href="#myself#circuit.fuse">` or `<a href="#myself##xfa.link#">`

Access Modifiers

Circuit and fuseaction tags have access modifiers:
Public = Can be called from a browser
internal = Can only be called from fusebox (via <do>)
private = Can only be called from the current circuit

File Name Prefixes

File name prefixes should largely be out of the scope of best practices once the application employs CFCs. Nearly all .cfm files are for display while other actions and queries should exist in respective, organized CFCs. That said:
dsp - a display file - something that renders to the browser
frm - a form - a type of display file
qry - a single query
act - an action file

Differences between Fusebox 5.1 and 5.5

Dynamic do

Differences between Fusebox 5.0 and 5.1

Plugins can have custom attributes
relocate has type=javascript for browser-based relocating
relocate has type=moved for a 301 redirect
relocate has xfa= to follow a preestablished exit point

This document was lovingly created in April 2008 by Nathan Strutz, http://www.dopefly.com/.