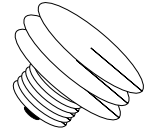


**Here's a sample from
THE Definitive Book:**



Fusebox: Methodology and Techniques

**by:
STEVE NELSON AND CRAIG GIRARD**

Fusion Authority and JM Publishers, Inc. have announced that the first Fusebox book, by Steve Nelson and Craig Girard, will be published in November. Steve and Craig are known experts in this exciting ColdFusion programming methodology. (In fact, Steve is one of the original initiators of the whole concept!) A free sample chapter is available in PDF format below.

What is Fusebox? Just as a house needs a blueprint, a program needs a specification. Fusebox is a method for building a ColdFusion applications. Originally created a way to collaborate efficiently on projects, it is much more now. Fusebox solves problems like finding a way to write object-oriented code and to make scalable applications, while at the same time making it easier to read and document code. In the end Fusebox delivers an idea that was much more than anyone expected.

Now Fusion Authority readers have the first look at a sample chapter of this new book! (This chapter will also be available on the free CD-Rom being distributed at the CFUN-2k conference on July 29 and 30 in Bethesda, Maryland.)

**E-Book Release Date:
Mid-August 2000**

**Print Book Release Date:
November 2000**

The electronic (.pdf) edition of

The print edition of

**FUSEBOX:
METHODOLOGY AND TECHNIQUES**

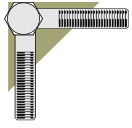
**FUSEBOX:
METHODOLOGY AND TECHNIQUES**

will be available from the following websites:

www.fusebox.org
www.fusionauthority.com
www.jmpublishers.com

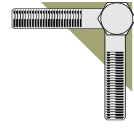
*will be available for sale from the same sites
and better book stores everywhere*

**Order in advance and save:
Advance orders are being taken now, at discounted prices!
Visit any of our participating websites for more information.**



CHAPTER 2:

Basic Fusebox Techniques



Overview

Defining the Terms

In the first chapter, you learned the need for organized code, especially when working in groups. We discussed the history of the **Fusebox Methodology**, and the Introduction, if you didn't skip it, answered a few commonly asked questions. Both chapters constantly refer to Fusebox as a “methodology”, which *dictionary.com* defines as:

meth•od•ol•o•gy

1. a. A body of practices, procedures, and rules used by those who work in a discipline or engage in an inquiry; a set of working methods: the methodology of genetic studies; an opinion poll marred by faulty methodology.
b. The study or theoretical analysis of such working methods.

Usage Note: “*Methodology*” can properly refer to the theoretical analysis of the methods appropriate to a field of study or to the body of methods and principles particular to a branch of knowledge.

Fusebox fits this definition perfectly. It is a body of practices, procedures and rules used by those who work in web development. It is also the study or theoretical analysis of such working methods. There is a core set of recommended Fusebox techniques that, if strayed from, could mean that “technically” you are not developing in Fusebox. However it is not an exact science. Many developers use Fusebox methodologies as a base and then add their own styles on top of them. Many then share their findings with the community of Fusebox developers: if the technique gains popularity, it may become part of the core methodologies.

With that in mind, let's define a few terms and gain an understanding of Fusebox. This chapter will begin to lay the foundation for you to build on. Remember our analogy to designing and building a house. There is the home itself, and there are the circuits that power it. Fusebox uses a similar structure:

Home Application

A **Home Application** is the overall application, or the root directory of your application. It is usually the first place a user will go when on the site. The home application is generally named after the system we are creating, although it does not need to be. The home application is a container for other applications, called “Circuit Applications”.

Circuit Application

Circuit applications are sub-applications of a home application, or sub-directories of a root directory. Each directory is its own application and is said to ‘extend’ the home application. Each sub-application has its own functions and responsibilities. The home application, together with its circuit applications, create the whole application.

A good example of a site making use of home and circuit applications is *ebags.com*. This site contains various home and many circuit applications. *Ebags.com/search* allows you to search for products, while *ebags.com/info/testimonials* allows users to view, add, edit and delete testimonials. Unless the user looks at the URL, they may never be aware that they have been using several applications.

This is similar to a real life home. Whether a person plugs a television into an outlet in the kitchen or in the living room, they still get power, because they are in their home (assuming they paid the power bill). The TV still goes on.

Introducing the Fuseaction

What Is an “Action”?

Whenever a user interacts with a web application, the server performs an **ACTION**. This *action* is whatever the user is allowed to do. They may wish to add an item to a shopping cart, delete an item from a shopping cart or purchase the items in their shopping cart. Each *action* is the user commanding the server to do something, or a single **command** to the server.

The Internet is a “stateless” application. In other words, the user does not stay connected to any server as they use the Web. When a user makes a request of the server, the user is logged in, the server processes the request, returns the output to the user, and logs the user out. Because of this, an action must be sent to the server with every request. If an action is not sent, the server will not know what to do.

Typically, actions are defined by the particular page being requested. The particular page the user is viewing usually contains a Hypertext Link or Form Action Attribute pointing to the next file in the flow of the application. If a user was removing all the items from their shopping cart, a typical Universal Resource Locator (URL) may look like this:

<http://www.fusebox.org/deletecart.cfm>

Fusebox defines a way to organize all the ‘actions’ of an application, called “**FUSEACTIONS**”.

What Is a Fuseaction?

One page of an application could possibly link to one or more different actions. These actions are capable of being sent to the server three different ways: URLs, Form Input Variables and Browser cookie variables. Fusebox recommends sending actions to the server in either Form or URL variables, because they are the only variables that are specific to each page.

Fusebox defines each user action as a **fuseaction**. The term “Fuseaction” is a reserved word in the Fusebox Methodology. The word “Fuseaction” is unique and descriptive, and its purpose is to command the server and give the programmer a quick description of what it does. Each Fuseaction is made up of one or more “**Fuses**”.

Introducing the "Fuse"

Just as a household fusebox has different fuses to control different circuits, so too does a web application have different components to control different functions. One file may display a form for the user to enter information into, another may enter that information into a database and another may send an e-mail. Each file the Fusebox calls, or any file included by the *index.cfm* file, is called a **FUSE**.

A bunch of fuses just bouncing around a fusebox isn’t going to do us much good. In order to make any use of our fuses, we need to organize them.

Types of Fuses

The Fusebox Specification defines five (5) types of *Fuses*.

Application Files: Application files are files that pertain to an application or applications as a whole. Fusebox defines two types of application files:

- **App_globals.cfm**: contains information that is needed globally, for the home application and all of its circuit applications. For example, if a variable defining the data source of a database is required by a home application and its circuit applications, this variable would be set in *app_globals.cfm*.

- **App_locals.cfm**: contains information that is specific to a circuit application.

Display Files: These files return, or display, information to the user. They can contain CFML, SQL and HTML. The SQL queries contained in these files only retrieve information from the database. They do NOT update or delete any records. These files could be used for displaying the results of a database query or presenting the user with a form.

Action Files: These files do not display any information to the user. They are for many different purposes, but are most commonly used to change information on the server. Action files can contain CFML and SQL. The SQL queries are allowed to insert, update and delete data within a database. The CFML could be used to change other data, such as writing to a file or making changes to the data in an LDAP server. These files are not limited to database changes; they can be used for doing those tasks that do not fit particularly well into a display file.

Query Files: These files contain SQL Queries. They are usually reserved for queries referenced in multiple files. Instead of rewriting the query each time, the file is inserted when needed, using the CFINCLUDE tag.

Cold Fusion Custom Tags(CF/CFX Tags): These files are ColdFusion extensions.

Introducing the "Fusebox"

Every household contains a *fusebox*. This fusebox controls the flow of electricity through a house. The electricity does not flow randomly, but instead is controlled by the fusebox. The fusebox uses *fuses* to moderate the electricity in each room. Each room has its own fuse and can function independently of all the other rooms.

The Fusebox specification runs on this philosophy. An application's *index.cfm* file acts as the Fusebox. This file controls the Fuseactions and Fuses, and determines the flow of the application. The application's "Fuses" are the files that are plugged into the Fusebox, the *index.cfm* file, to complete an action.

Each directory in an application contains one *index.cfm* file. These files have one task: Processing Fuseactions. Remember, a household's electricity flows through the fusebox, on through the fuses and into separate rooms of the house. Each *index.cfm* file directs the programming flow the same way: Every action requested by a user is routed through the *index.cfm* file. The *index.cfm* file evaluates the action being requested and invokes the Fuses necessary to complete the Fuseaction.

Conclusion

This chapter introduced the common terms and definitions used within the Fusebox Methodology. We began by defining the word "methodology". Fusebox is a methodology for the web development community. It is a body of practices, procedures and rules to organize your code. Many developers begin with the basic Fusebox techniques and eventually add their own methods and procedures on top of that foundation.

Each application has a home application and can have one or more circuit applications. Each application contains an *index.cfm* file. This file is the Fusebox and controls the flow of the application. Each call to the application goes through a Fusebox; which processes the requested Fuseaction by calling the Fuses required to complete it.

The next chapter will explain the Basic Fusebox Techniques.