

# Fusebox 5.51 Tutorial

ColdFusion Developers Network

12/2/2009  
Em Português  
Livre distribuição

Ricardo Parente

# Fusebox 5.51 Tutorial

Eu trabalho com o framework Fusebox desde sua versão 2. Agora estamos na versão 5.51. Decidi escrever este tutorial para ajudar alguns amigos que não “entraram” ainda na onda do Fusebox mas estão ansiosos para se juntar à turma, somente não sabem como começar. Eu recomendo os livros do Jeff Peters sobre Fusebox para aqueles que realmente querem tirar todas as vantagens desta framework:

- Fusebox 5 & Flip Master-Class ColdFusion Applications  
<http://www.cafepress.com/protonarts.126485280>
- How to Drive Fusebox 5.5  
<http://www.cafepress.com/protonarts.297794563>

## MVC - Model View Controller

Antes de entrarmos no framework, vamos falar um pouco a respeito de MVC (Model, View, Controller). Para aqueles que não conhecem MVC, é um método de separação das camadas de apresentação, ação e controle da aplicação a fim de se obter um jeito melhor de programar com segurança e uma melhor manutenção.

Todas as chamadas externas (vindas do web URL) são direcionadas à camada de controle (Control). O Control então decide quais as ações a tomar, tais como executar uma query, um cálculo, etc... Tais execuções são feitas na camada Model, que não dá acesso ao mundo externo. Então o Control submeterá os resultados à camada View (presentation layer) que montará o conteúdo a ser exibido ao public. Basicamente, Basically, todos os endereços públicos dos links em seu website estarão chamando a camada Control. Logo, todas as suas queries e paginas de ação estarão no Model e suas paginas de display estarão na camada View, fácil de achar e manter. Exemplo: Se voce tiver que mostrar uma lista de propriedades imobiliárias, o Control chamará uma query para obter os dados e então chamará a página de display na camada View para exibir as propriedades trazidas pela query.

## Fusebox

Fusebox é uma metodologia para se desenvolver aplicações web. Ela protege sua aplicação forçando todas as chamadas a serem feitas através de um único template: index.cfm, que age como um hub, com analogia à caixa de fusíveis onde todas as conexões elétricas são feitas. Passando um atributo chamado "fuseaction", voce diz à framework Fusebox qual ação voce quer executar. Logo, comparando com a caixa de fusíveis elétrica, a framework se divide em circuitos e fusíveis (circuits & fuses). As pastas de sua aplicação serão chamadas "circuits" e os templates ColdFusion sero chamados de "fuses".

Neste tutorial estarei referenciando à Fusebox 5.51 XML, e não a versão sem XML (tipo de OO). Vamos começar por baixar os arquivos "core" da versão atual [Fusebox 5.51 Core Files](#). Selecione o primeiro link: **Official FB5.5.1/CFMX core files (v5.5.1)**.

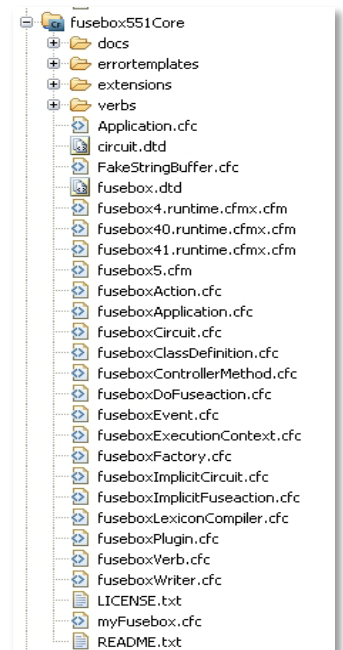


Figure 1.1

# Fusebox 5.51 Tutorial

Expanda o conteúdo do arquivo zip para uma pasta chamada "fusebox5" dentro da raiz de sua aplicação web **caso voce não tenha acesso** ao site de administração do ColdFusion. Caso contrário, expanda o conteúdo para uma pasta fora da raiz de sua aplicação e mapeie-a como "/fusebox5" no site de administração. Fazendo assim, você estará habilitado a ter várias aplicações usando a mesma pasta Fusebox. Your Fusebox core structure should be like *Figure 1.1* side.

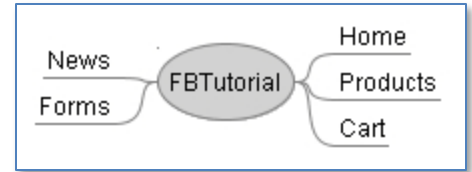


Figure 1.2

Vamos criar uma aplicação web para uma loja de instrumentos musicais chamada **FBTutorial**.

Aqui estão os requerimentos iniciais da aplicação:

- Home Page com artigos, notícias
- Página de Contato - formulário
- Página de Produtos - catálogo
- Shopping Cart

Eu conheço dois jeitos de se criar a estrutura MVC para uma aplicação Fusebox. Um deles é muito usado, criando-se 3 pastas: `_model`, `_view` and `_controller` (os underscores so para posicionar as pastas no topo das demais na aplicação), e dentro daquelas pastas coloca-se os circuitos Fusebox. Acho esse jeito mais complicado e menos modular. Qualquer manutenção requer que você olhe diversas pastas dentro de outras pastas, etc... (*figure 1.3*)

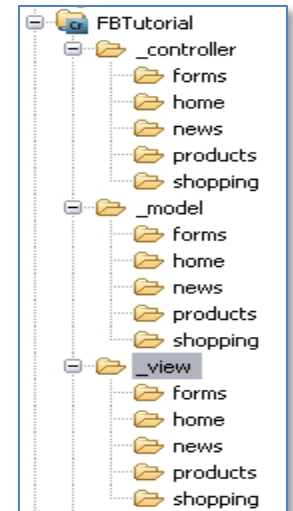


Figure 1.3

Se você quiser tornar um circuito em um módulo que possa ser copiado para outra aplicação, a melhor maneira será criar os circuitos primeiro, e então dentro de cada circuito criar as pastas "model" e "view", tendo o próprio circuito atuando como "control".

Logo, escolhi a segunda estrutura, criando as pastas de circuitos e então as sub-pastas model e view. (*Figure 1.4*).

Aqui está a forma em que criaremos a estrutura da aplicação:

- FBTutorial
  - circuits
    - forms
    - home
    - news
    - products
    - shopping

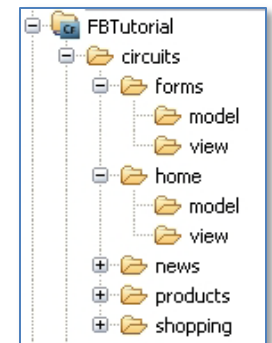


Figure 1.4

Agora, dentro de cada pasta de circuito, criaremos as pastas "model" e "view". Você decide se quiser adicionar o prefixo "\_" (underscore), desde que não teremos mais pastas dentro dos circuitos.

Após criar as pastas necessárias, precisamos criar os arquivos de controle vazios em cada pasta. Esses arquivos serão chamados "circuit.xml.cfm",

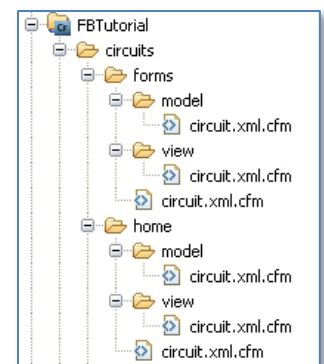


Figure 1.5

# Fusebox 5.51 Tutorial

que são arquivos XML mas com a terminação .cfm para que não possam ser lidos pelo browser sem serem processados pelo ColdFusion. Apenas crie arquivos vazios por agora. *Figure 1.5* mostra a nova estrutura com os arquivos XML.

Não se preocupe com os arquivos vazios, iremos preenchê-los mais tarde. Estou apenas preparando a estrutura da aplicação antes de começarmos a codificar.

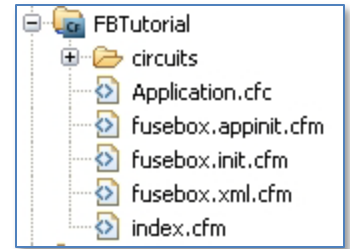


Figure 1.6

Agora, vamos criar os arquivos requeridos e opcionais do Fusebox, na raiz da aplicação. Começando com o `Application.cfc` ("A" maiúsculo), `fusebox.appinit.cfm`, `fusebox.init.cfm`, `fusebox.xml.cfm` e `index.cfm` (crie esses arquivos vazios e iremos trabalhar neles a seguir. Sua estrutura deverá ser como a *Figure 1.6*.

## Raiz da Aplicação

Vamos começar a trabalhar agora com os arquivos Fusebox na raiz da aplicação. Primeiramente, o componente *Application.cfc* que é sempre o primeiro a executar em cada request.

Os métodos básicos no component `Application.cfc` são:

- `onApplicationStart`,
- `onApplicationEnd`,
- `onSessionStart`,
- `onSessionEnd`,
- `onRequestStart`.

Voce pode ter métodos adicionais como `onRequestEnd`, `onError...` mas agora estamos interessados no método Fusebox: `onFuseboxApplicationStart`. Este método é chamado quando a framework Fusebox inicia. Ele substitui o template `fusebox.appinit.cfm` na raiz da aplicação. Ento voce pode resolver usar este método em vez do template, é menus um arquivo para manter.

## Aqui está o nosso `Application.cfc`:

```
<cfcomponent extends="fusebox5.Application" output="false">
<cfscript>
this.name = right(REReplace(expandPath('.'), '[^A-Za-z]', '', 'all'), 64);
this.applicationTimeout = createTimeSpan(0,6,0,0);
this.sessionManagement = true;
this.sessionTimeout = createTimeSpan(0,0,30,0);
this.clientManagement = true;
this.mainDSN = "myDSN";
this.webmasterEmail = "myEmail@myDomain.com";
</cfscript>
<!-- trap non-index.cfm requests - must be outside onXxxYyy() handlers -->
<cfif right(cgi.script_name, len("index.cfm")) neq "index.cfm" and
right(cgi.script_name, 3) neq ".cfc">
    <cflocation url="index.cfm" addtoken="no" />
</cfif>
<cffunction name="onApplicationStart" returntype="boolean" output="false">
```

## Fusebox 5.51 Tutorial

```
<cfreturn True>
</cffunction>
<cffunction name="onApplicationEnd" output="false">
    <cfargument name="applicationScope" required="true" />
</cffunction>
<cffunction name="onSessionStart" output="false">
</cffunction>
<cffunction name="onSessionEnd" output="false">
</cffunction>
<cffunction name="onFuseboxApplicationStart">
    <cfset super.onFuseboxApplicationStart() />
    <!--- code formerly in fusebox.appinit.cfm --->
    <cfif not structKeyExists(application,"mainDSN")>
        <cflock name="mainDSN" type="exclusive" timeout="10">
            <cfset application.mainDSN = this.mainDSN />
        </cflock>
    </cfif>
</cffunction>
<cffunction name="onRequestStart">
    <cfargument name="targetPage" />
    <cfscript>
        super.onRequestStart(arguments.targetPage);
        //code formerly in fusebox.init.cfm
        self = myFusebox.getSelf();
        myself = myFusebox.getMyself();
        if (listFirst(CGI.SERVER_NAME, ".") == "www") {
            FUSEBOX_PARAMETERS.mode = "production" ;
        } else {
            FUSEBOX_PARAMETERS.mode = "development-circuit-load" ;
        }
        // request scope
        request.applicationName = this.name;
        request.mainDSN = this.mainDSN;
        request.webmasterEmail = this.webmasterEmail;
        request.baseDir = GetDirectoryFromPath(GetBaseTemplatePath());
        request.componentPath = "assets.components.";
        request.imagePath = "assets/images/";
        request.cssPath = "assets/css/";
        request.jsPath = "assets/js/";
        request.flashPath = "assets/multimedia/";
        request.videoPath = "assets/multimedia/";
        request.ppsPath = "assets/multimedia/";
    </cfscript>
</cffunction>
</cfcomponent>
```

## Fusebox 5.51 Tutorial

---

Agora vem o truque. Desde que o Fusebox core tem também um Application.cfc com os mesmos métodos, precisamos estender o nosso Application.cfc com o do Fusebox e fazer uso da chamada "call super" para invocar aqueles métodos, precisamos estender o nosso Application.cfc com o do Fusebox e fazer uso da chamada "call super" para invocar aqueles métodos primeiro afim de depois podermos usar os nossos.

Na primeira linha, nós extendemos o component. Lembre-se de mapear /fusebox5 no administrador do ColdFusion para a pasta onde voce salvou os arquivos core.

```
<cfcomponent extends="fusebox5.Application" output="false">
```

O método *onFuseboxApplicationStart* deverá ter a chamada "call super":

```
<cffunction name="onFuseboxApplicationStart">
    <cfset super.onFuseboxApplicationStart() />
    ... ..
```

O método *onRequestStart* também deve ter a chamada "call super":

```
<cffunction name="onRequestStart">
    <cfargument name="targetPage" />
    <cfscript>
        super.onRequestStart(arguments.targetPage);
    ... ..
```

Abaixo, um código que estaria anteriormente no *fusebox.init.cfm*:

```
self = myFusebox.getSelf();
myself = myFusebox.getMyself();
if (listFirst(CGI.SERVER_NAME, ".") == "www") {
    FUSEBOX_PARAMETERS.mode = "production" ;
} else {
    FUSEBOX_PARAMETERS.mode = "development-circuit-load" ;
}
```

## Fusebox.init.cfm

Basicamente nós escrevemos todo o código necessário para iniciar com o `Application.cfc`, agora vamos nos concentrar no template Fusebox chamado `fusebox.init.cfm`. Esse template é incluído pelo framework no início de cada requisição (request). Ele é incluído com uma tag `cfsilent`, logo não pode gerar nenhuma saída. A intenção de uso é para inicialização e manipulação de variáveis Fusebox a cada requisição.

Voce pode, por exemplo, configurar `attributes.fuseaction` para sobrepor o `fuseaction` default. Fusebox 5.1 acima – configure implicitamente as variáveis Fusebox.

Voce tambem pode modificar aqui a variável de localização própria:

```
<cfset myFusebox.setSelf("/myapp/start.cfm") />
```

Por hora, apenas escreva o código seguinte no seu template `fusebox.init.cfm`:

```
<cfprocessingdirective suppresswhitespace="true">
  <!--- some future code here --->
</cfprocessingdirective>
```

## Index.cfm

O template `index.cfm` terá apenas um comando:

```
<cfinclude template="/fusebox5/fusebox5.cfm" />
```

Numa aplicação Fusebox, o template `index.cfm` não tem nenhum outro uso além de incluir o framework.

## Fusebox.xml.cfm

O último arquivo que temos a trabalhar na raiz da aplicação é o ***fusebox.xml.cfm***, que é o arquivo que controla todos os circuitos, parâmetros, classes, plugins e ações globais do Fusebox.

No bloco "circuits", nós definimos os caminhos e aliases (apelidos) para os circuitos. Você pode escolher qualquer nome como um alias para um circuito, não precisa ser o mesmo nome da pasta. Você verá abaixo que estamos chamando de "cart" o circuito na pasta "shopping".

O bloco "parameters" contém a configuração padrão do Fusebox, e é customizável.

Aqui está o nosso arquivo *fusebox.xml.cfm*:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fusebox>
<!--
Example fusebox.xml control file. Shows how to define circuits, classes,
parameters and global fuseactions.
This is just a test namespace for the plugin custom attribute example
-->
<fusebox xmlns:test="test">
  <circuits>
    <!-- Home circuit -->
    <circuit alias="home" path="circuits/home/" parent="" />
    <circuit alias="mhome" path="circuits/home/model/" parent="home" />
    <circuit alias="vhome" path="circuits/home/view/" parent="home" />
    <!-- Forms circuit -->
    <circuit alias="forms" path="circuits/forms/" parent="home" />
    <circuit alias="mforms" path="circuits/forms/model/" parent="forms" />
    <circuit alias="vforms" path="circuits/forms/view/" parent="forms" />
    <!-- News circuit -->
    <circuit alias="news" path="circuits/news/" parent="home" />
    <circuit alias="mnews" path="circuits/news/model/" parent="news" />
    <circuit alias="vnews" path="circuits/news/view/" parent="news" />
    <!-- Products circuit -->
    <circuit alias="products" path="circuits/products/" parent="home"/>
    <circuit alias="mproducts" path="circuits/products/model/" parent="products" />
    <circuit alias="vproducts" path="circuits/products/view/" parent="products" />
    <!-- shopping circuit -->
    <circuit alias="cart" path="circuits/shopping/" parent="home" />
    <circuit alias="mcart" path="circuits/shopping/model/" parent="cart" />
    <circuit alias="vcart" path="circuits/shopping/view/" parent="cart" />
    <!-- Layout circuit -->
    <circuit alias="layout" path="circuits/layout/" />
  </circuits>
  <classes>
  </classes>
  <parameters>
    <parameter name="defaultFuseaction" value="home.main" />
    <!-- you may want to change this to development-full-load mode: -->
```

## Fusebox 5.51 Tutorial

```
<parameter name="mode" value="development-full-load" />
<!-- change this to something more secure: -->
<parameter name="password" value="" />
<parameter name="strictMode" value="true" />
<!-- These are all default values that can be overridden:
<parameter name="fuseactionVariable" value="fuseaction" />
<parameter name="precedenceFormOrUrl" value="form" />
<parameter name="scriptFileDelimiter" value="cfm" />
<parameter name="maskedFileDelimiters"
    value="htm,cfm,cfml,php,php4,asp,aspx" />
<parameter name="characterEncoding" value="utf-8" />
<paramater name="strictMode" value="false" />
<parameter name="allowImplicitCircuits" value="false" />
<parameter name="debug" value="false" />
-->
</parameters>
<globalfuseactions>
    <preprocess>
</preprocess>
    <postprocess>
</postprocess>
</globalfuseactions>
<plugins>
    <phase name="preProcess">
</phase>
    <phase name="preFuseaction">
</phase>
    <phase name="postFuseaction">
</phase>
    <phase name="fuseactionException">
</phase>
    <phase name="postProcess">
</phase>
    <phase name="processError">
</phase>
</plugins>
</fusebox>
```

Nós definimos os circuitos no bloco "circuits" dando um alias para o caminho (path). Na aplicação, nós chamaremos o *index.cfm* passando circuit, fuse e fuseaction que quisermos executar.

Example:

```
index.cfm?fuseaction=home.main
index.cfm?fuseaction=products.showProduct&productID=1234
index.cfm?fuseaction=cart.addToCart
```

Com Fusebox, tudo é construído no raiz da aplicação. O *index.cfm* carrega o Fusebox e este prepara o request a ser servido sem que o cliente veja em que pasta os templates (páginas) estão.

# Fusebox 5.51 Tutorial

Nós adicionamos um circuito chamado "layout" que fará o manuzeio de todos os layouts para nossa aplicação. Sendo parte da camada de apresentação (presentation layer), não teremos que criar a pasta "model". Figura 1.7 mostra uma versão atualizada da nossa estrutura.

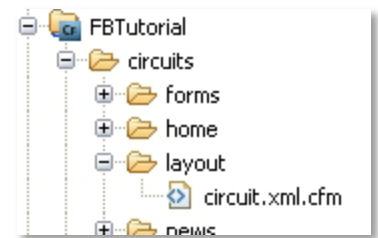


Figure 1.7

## Circuit.xml.cfm

Agora nós vamos olhar o arquivo *circuit.xml.cfm* que é colocado em todos os circuitos.

Vamos começar com o circuito principal "home", o controller " /circuits/home/circuit.xml.cfm":

Por agora, não teremos nenhuma ação a chamar, somente uma página de exibição (display) para mostrar a homepage. Então nosso arquivo *circuit.xml.cfm* terá o seguinte código:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- home -->
<circuit access="public" xmlns:cf="cf/">
  <fuseaction name="main">
    <do action="vhome.main" />
  </fuseaction>
</circuit>
```

Voce deve ter notado que estamos chamando uma ação em um outro circuito, o "vhome", na camada "view" do MVC, para exibir a homepage. Se precisássemos de calcular alguma coisa ou obter alguma informação de um banco de dados afim de popular aquela homepage, então nós chamaríamos a camada "model" antes da "view".

A chamada "index.cfm?fuseaction=home.main" levará ao circuito "vhome" na camada "view". Subsequentemente aquela fuseaction "main" (vhome.main), incluirá o template "dspMain.cfm" com o conteúdo da homepage, e então o Fusebox montará a página inteira para o ColdFusion enviar ao browser.

Note que o atributo "access" para este circuito foi configurado como "public". Isto significa que ele pode ser acessado pelo web browser, é um circuito público.

Aqui está o arquivo *circuit.xml.cfm* na camada "view":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE circuit>
<!-- vhome -->
<circuit access="internal" xmlns:cf="cf/">
  <fuseaction name="main">
    <include template="dspMain.cfm" required="true" />
  </fuseaction>
</circuit>
```

## Fusebox 5.51 Tutorial

---

Neste caso, o atributo "access" está configurado como "internal" significando que este circuito pode ser acessado (chamado) apenas por um outro circuito, ele não responderá a chamadas do web browser.

A fuseaction "main" incluirá o template *dspMain.cfm*.

Somente para teste, crie o arquivo *dspMain.cfm* com o seguinte código HTML:

```
<h1>Hello World</h1>
```

Agora, vamos ver o site. Se voce tiver ColdFusion Developer Edition instalado em sua máquina, navegue para: "http://localhost/FBtutorial". Voce não precisa especificar nenhum atributo, porque o arquivo de configuração do Fusebox (*fusebox.xml.cfm*) se responsabilizará em chamar a fuseaction default "home.main".

Voce deve receber a página "Hellow World".

## Layouts

Gostaria que voce prestasse uma atenção em particular ao circuito "layout". Este circuito não será chamado por qualquer página sequer. Ele é um circuito interno, basicamente um circuito "view" que montará todo o conteúdo da página dentro de um template de layout que escolhermos.

Agora que voce tem sua página "Hello World" funcionando, vamos brincar um pouco com layouts. Suponhamos que voce queira ter blocos de cabeçalho, conteúdo e rodapé e então inseri-los no seu template de layout. (Figure 2.1). Primeiro voce terá que criar os conteúdos do cabeçalho, principal e rodapé. Vamos ver como isso é feito.

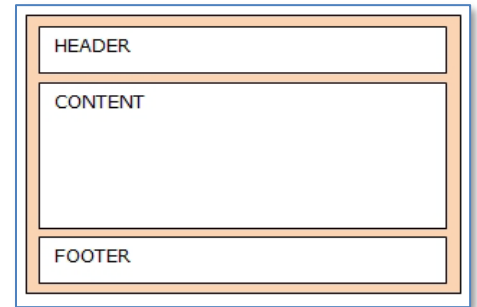


Figure 2.1

No template *fusebox.xml.cfm* existe uma tag `<globalfuseactions>`, seguida por duas tags `<preprocess>` e `<postprocess>`. Todas as ações que voce incluir entre essas tags serão processadas ou antes ou depois que o Fusebox montar a página.

Nós configuraremos as ações de cabeçalho e rodapé na tag `<preprocess>` e as ações de layout na tag `<postprocess>`, porque queremos preparar os dados do cabeçalho e rodapé para serem incluídos no layout (se fôrmos usar algum layout).

O código é:

```
<globalfuseactions>
  <preprocess>
    <fuseaction action="vhome.header"
      contentvariable="content.header" />
    <fuseaction action="vhome.footer"
      contentvariable="content.footer" />
  </preprocess>
  <postprocess>
    <fuseaction action="layout.checkLayout" />
  </postProcess>
</globalfuseaction>
```

Note o atributo "contentvariable" no bloco `<preprocess>` na tag `fuseaction`. Isso significa que o Fusebox processará aquela ação e colocará os resultados em uma variável, em vez de exibi-los na tela. Mais tarde, faremos uso daquela variável no context do layout. Estamos chamando ações na camada "view" do circuito "home"

## Fusebox 5.51 Tutorial

---

Então, vamos trabalhar agora no circuito "layout" e preparar o cabeçalho, rodapé e templates de layout.

Crie dois templates ColdFusion chamados *dspHeader.cfm* e *dspFooter.cfm* dentro da pasta "home view": `\circuits\home\home`. E adicione os códigos seguintes:

*dspHeader.cfm*

```
<h2>Fusebox Tutorial</h2>
<hr />
```

*dspFooter.cfm*

```
<hr />
<h4>&copy;2009 Your Name</h4>
```

Agora vamos editar o *circuit.xml.cfm* no circuito "vhome":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE circuit>
<!-- vhome -->
<ircuit access="internal" xmlns:cf="cf/">
  <fuseaction name="main">
    <include template="dspMain" required="true" />
  </fuseaction>
  <fuseaction name="header">
    <include template="dspHeader" required="true" />
  </fuseaction>
  <fuseaction name="footer">
    <include template="dspFooter" required="true" />
  </fuseaction>
</ircuit>
```

Voce notou que não é preciso escrever a extensão do arquivo `.cfm` para o nome dos templates? Fusebox assumirá que voce estará carregando templates ColdFusion se voce não especificar a extensão de arquivo.

Nós temos agora os 3 includes para o bloco "home view", a página principal, que é chamada por default pelo Fusebox, o cabeçalho e o rodapé. As fuseactions do cabeçalho e rodapé são chamadas pelo "preprocess" no *fusebox.xml*.

Execute sua aplicação e voce receberá uma página como a figura 2.2.



**Fusebox Tutorial**

**Hello World**

©2009 Your Name

Figure 2.2

# Fusebox 5.51 Tutorial

---

## Nota do autor:

ColdFusion Developers Network não é uma empresa, nem um website. Apenas um mail server que criei para hospedar as contas de uma comunidade internacional de desenvolvedores ColdFusion.

Até o presente momento, dezembro 2 de 2009, contamos com mais de 40 membros de diversos países como Brasil, USA, Belgium, Netherlands, Portugal, Germany...

Se voce é um CF developer e desejar uma conta gratuita no cfdevelopers.net, envie-me um email com seu nome de usuário desejado que eu criarei sua conta e enviarei instruções de acesso.

Desde já, obrigado pelo suporte à comunidade ColdFusion em geral.

Espero que este tutorial sirva para ajuda-lo a iniciar no framework Fusebox.

Pode copiar, imprimir, repassar este tutorial gratuitamente, não deterei copyright, pois a intenção é divulgar o máximo de conhecimento para a comunidade ColdFusion.

Caso voce tenha dúvidas, não hesite em me enviar um e-mail com suas perguntas. Será sempre um prazer ajudar um "fellow developer" a se tornar mais um entusiasta deste excelente framework.

## Dados Pessoais:

(por favor: NO SPAM !!)

Ricardo Parente

Blogs:

<http://ricardo.parente.us>

<http://ensina.me/coldfusion>

E-mail: [rparente@cfdevelopers.net](mailto:rparente@cfdevelopers.net)

Google Wave: [cfdevelopers@googlewave.com](http://cfdevelopers@googlewave.com)

Google Phone: +1(407) 212-RICA (407-212-7424)