# Mach-II Quick Start Guide

Mach-II 1.1.1 for ColdFusion

Last Updated on November 15, 2006

Matt Woodward
Release Coordinator

## In This Document

## Introduction

Welcome to Mach-II! The purpose of this Quick Start Guide is to help ColdFusion developers who are new to Mach-II get started as quickly and easily as possible. This guide won't teach you everything you need to know about Mach-II, but it will walk you through installing Mach-II and verifying the installation as well as the basic Mach-II concepts you'll need to begin writing your own Mach-II applications. We'll even build a very simple sample application to help you get your feet wet with Mach-II. Before you know it you'll be talking listeners, views, filters, and plugins with great aplomb and impressing friends and strangers alike with your Mach-II knowledge. (Don't worry; we'll get you there gradually.)

To reiterate, this guide is not intended to serve as the be-all and end-all of Mach-II documentation. Think of it as an easy and (we hope) fun way to dive in, install Mach-II, and start getting some hands-on experience with this extremely powerful, flexible ColdFusion framework. This guide will also give you a good foundation for more advanced Mach-II documentation and concepts.

## What is Mach-II?

To discount a couple of points of confusion right away, in our case Mach-II does not refer to a razor or breaking the sound barrier (as an aside, attempting to shave and break the sound barrier at the same time isn't advisable). In the context of the present discussion, Mach-II is a powerful object-oriented ColdFusion framework that will help you build highly flexible, maintainable ColdFusion applications. Sounds great, right? It is! Let's explain a few of these ideas just a bit further.

1. **Mach-II is powerful.**
   Mach-II is used on numerous large, high-traffic sites including portions of adobe.com. Mach-II is based on many tried-and-true principles of application architecture that have been used successfully both in desktop and web application development for years.
2. **Mach-II is flexible.**
   Mach-II allows you and helps you to write highly modular ColdFusion applications, meaning your code is highly reusable, stable and easy to debug.
3. **Mach-II applications are highly maintainable.**
   One of the great benefits of Mach-II is not only in the process of development itself but the end result. If you're lucky, your application will spend a lot more time in maintenance mode than it does in development. By using Mach-II you can not only lengthen the overall life-span of your application, but you'll make the maintenance cycle much less painful.

Is Mach-II the fountain of youth for your ColdFusion applications and the great cure for spaghetti code of which you've been dreaming? It just might be! Once you get the hang of Mach-II you'll be amazed and how much more easily and quickly you can build world-class web applications, as well as how easily you can change and add to these applications because of the best development practices supported by Mach-II. So without further ado, let's install Mach-II!

## Installing Mach-II

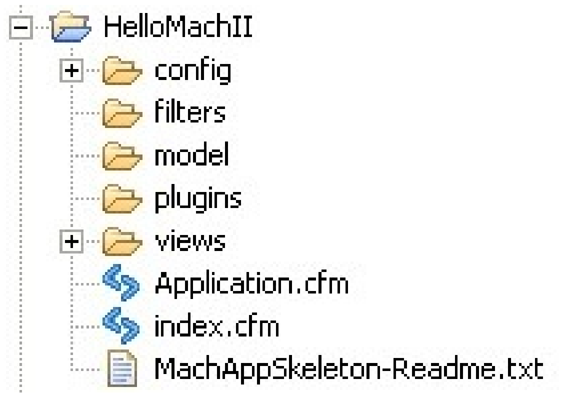
The installation process for Mach-II is so utterly simple "installation" might even be considered a misnomer. In order to install Mach-II simply do the following:

1. Download the framework code from mach-ii.com if you haven't already.
2. Unzip the framework to your web root. For example, on Windows the default web root is [DRIVE]:\Inetpub\wwwroot, or if you're using ColdFusion's built-in web server, your web root is likely [DRIVE]:\CFusionMX\wwwroot. If you're on a Unix system of some sort this will vary. The end result of this step is that you should have the directory MachII (note: no hyphen) in your web root, and inside this directory you'll find the Mach-II framework files. If you wish to place the Mach-II framework files elsewhere that's fine as well, you'll just have to create a mapping called "MachII" in the ColdFusion administrator and point this mapping to the location where you placed the framework files.
3. If you use sandbox security on your ColdFusion server, you may have to add the framework's directory to your sandbox otherwise ColdFusion may throw an java security exception.

Yes, it's really that simple. At the end of the day Mach-II itself is nothing more than a ColdFusion application, so it operates and behaves just like any other ColdFusion application.

## The Mach-II Application Skeleton

Typically you'll start each of your Mach-II applications from a set of files forming an application template typically referred to as the "skeleton files." If you haven't done so already please download the Mach-II application skeleton from http://www.mach-ii.com. Unzip the skeleton files to a directory called "HelloMachII" in your web root. When you look inside this directory it should look something like this:



```
HelloMachII
    config
    filters
    model
    plugins
    views
    Application.cfm
    index.cfm
    MachAppSkeleton-Readme.txt
```

Let's take a look at each of these files and directories briefly.

- **/config/mach-ii.xml:**
  This is the XML control file for your Mach-II applications. You'll define events, views, listeners, and several other things in this file. Think of this as the roadmap that tells your application where to go and what to do.
- **/filters:**
  This is where you'll put filters, which are CFCs that you can use to filter specific events, meaning some sort of code is run prior to the event proceeding. For example, you might use a filter on particular events to make sure someone is logged in. We won't be covering filters in this quick start guide.
- **/model:**
  This is where you'll put the CFCs that form the business logic layer and/or domain model of your application. You can actually put your CFCs anywhere, but as you're learning Mach-II and for smaller applications, this model directory works just fine. For larger applications you may find yourself organizing things a bit differently.
- **/plugins:**
  Plugins can be thought of as cousins to the filters we described above. Whereas filters are used to filter specific events, plugins contain code that is executed prior to every event in your application. We won't be covering plugins in this quick start guide.
- **/views:**
  This is where you'll put all the view pages for your application, which are the CFML pages that are actually presented to the user and with which the user interacts. You'll see a couple views in your views directory already. `exception.cfm` is called when an exception is thrown by your application, and we'll see how in a moment.
- **Application.cfm:**
  This file is more or less a traditional Application.cfm file, but it's largely stripped down since in Mach-II applications we do things that might otherwise be in Application.cfm in a slightly different way. That's not to say you can't use Application.cfm in exactly the same way you're used to using it, but in many Mach-II applications all you'll see in Application.cfm is the `<cfapplication>` tag containing the application name and the other typical application settings such as session management, etc.
- **index.cfm:**
  As we'll see in a moment, index.cfm is the file through which all requests to your Mach-II application are routed. This is a relatively simple file that mainly serves as the entry point for your application and includes the necessary Mach-II code from the framework itself. We won't worry about all the details in this quick start guide, but one line in this file we'll make you aware of right away is lines 3 and 4:

```
<!--- Set the configuration mode (when to reload): -1=never, 0=dynamic, 1=always --->
<cfset MACHII_CONFIG_MODE = 1 />
```

These lines control whether the Mach-II framework itself, and consequently the components of your application that Mach-II loads into RAM for performance reasons, is reloaded on every request (a setting of 1), only when the mach-ii.xml file changes (a setting of 0), or never (a setting of -1). You'll use the -1 setting for production mode, and of course this will give your applications quite a performance boost. In development you'll either just have the framework reload every time, or for larger applications or applications for which reloading every time will cause a problem of some kind, dynamic reloading (0) is the way to go.

Taken as a whole, these files will be the core set of files that you'll use as the starting point for all your Mach-II applications, and it's our sincere hope that after building a couple of bits and pieces in this guide you'll want to build many, many more! For now we're going to dig in and have you build your first Mach-II application. Yes, that's right, you're going to do it right here, right now, and in mere moments you'll see just how simple this can be.

## Hello Mach-II! (Come on, you knew this was coming.)

Now that Mach-II is successfully in your webroot or a mapped equivalent let's make sure everything's working as it should. In the grand tradition that software development historians believe started with the hieroglyphs of ancient Egypt (OK, maybe it actually only goes back to about 1974, but in software development terms that pretty much *is* ancient Egypt), let's create a "Hello World" application in Mach-II. This will be a quick and easy way to better familiarize yourself with the Mach-II application skeleton as well as get a taste of how Mach-II applications work.

Yes, it's obligatory. You HAVE to build a Hello World application in order to truly understand a technology. Don't worry, it's simple, it's painless, and it actually is a great way to walk you through what's involved with building a real Mach-II application.

**Step 1: Name Your Application**
First, open up your text editor of choice and navigate to your HelloMachII directory (as a reminder, this is the directory to which you unzipped the Mach-II skeleton). Next, open up the Application.cfm file that's provided with the base Mach-II skeleton and change `name="APPLICATION_NAME"` to something a bit more meaningful. Let's just be obvious and type "HelloMachII" in there. The new Application.cfm file should contain only this line:

```
<cfapplication name="HelloMachII" sessionmanagement="yes" />
```

**Step 2: Set the Application Root and Default Event Name**
We'll get into this in more detail later, but the basic gist of how Mach-II works is that events are announced and subsequently "stuff happens." Your Mach-II applications will get all their information about the "stuff" that happens within each event from an XML configuration file located in the config directory. Again in your favorite text editor open up the mach-ii.xml file in the config directory of your Mach-II application skeleton.

First and foremost, don't panic! This file may look a bit intimidating but it's actually quite straight-forward. Let's focus on just the bits and pieces we need in order to create our Hello World application. In other words, don't get bogged down in the details at this point because we'll be looking at this file in greater detail later.

For starters let's take a look at lines 11-12 of mach-ii.xml, which should look something like this in the base skeleton:

```
<property name="applicationRoot" value="/APPLICATION_ROOT" />
<property name="defaultEvent" value="DEFAULT_EVENT" />
```

The applicationRoot property defines where this particular Mach-II application lives, which in this case since we placed a directory called HelloMachII in the web root would be in the directory /HelloMachII, so go ahead and update that information. Your new line 11 should look like this:

```
<property name="applicationRoot" value="/HelloMachII" />
```

Next we need to define a default event. Mach-II applications don't know what to do if they don't have an event to execute, so to avoid forcing your users type an event into the URL of your site you define a default event that will be executed when an event name isn't present in the URL.

For this application there will only be two events, one of which is the event that is called when an exception occurs, and that's already been taken care of for us. To define your default event update line 12 to read as follows:

```
<property name="defaultEvent" value="sayHello" />
```

By using a default event you ensure that if someone goes to the root of your application and doesn't pass a page name and event name in the URL, the application will still respond.

**Step 3: Create the Default Event**
Another topic we'll discuss in greater detail very shortly is how Mach-II separates the business logic of your applications from the presentation layer, which Mach-II refers to as the "view." (Not coincidentally this corresponds to the "view" in the "Model-View-Controller" design pattern which, unless you've been doing FORTRAN development for the last several years, you've probably at least heard of by now. We'll cover MVC as it applies to Mach-II in greater detail in a future document.)

Perhaps the most basic of all Mach-II events is one that simply displays a view, which is precisely what we'll be dealing with here. Since you're already in the XML configuration file (or if you're not please get there), let's first define our view, then we'll create the actual view page itself.

First let's tell Mach-II what's going to happen in the "sayHello" event that we defined as the default event in the step above. Events are handled within Mach-II applications by a tag in the configuration file called (appropriately enough) the event-handler. On line 43 of the mach-ii.xml file you should see the following:

```
<event-handler event="DEFAULT_EVENT" access="public">
    <!-- any legal elements -->
</event-handler>
```

First let's update the `DEFAULT_EVENT` text and change that to our default event name, which is `sayHello`. Next we need to tell Mach-II what happens within this event. All we really want to have happen in this event is for our application to say hello to us (yes, we're that lonely, OK?), which is really just a matter of telling the application to display a view. Displaying a view in this event is as simple as replacing <view-page name="home"/> with the following:

```
<view-page name="hello" />
```

This tells Mach-II that when the sayHello event is called, we're going to display a view page called hello. Note that there is no .cfm at the end of the view page name because we aren't calling the CFML page directly, we're calling a view by a name that we give it; you'll see how we do this in the next step. Your new event-handler should look like this:

```
<event-handler event="sayHello" access="public">
    <view-page name="hello" />
</event-handler>
```

**Step 4: Define and Create the View**
How does Mach-II know which page is called hello? How does Mach-II display views? I'm glad you asked these questions! Views are defined in the `page-views` section of the mach-ii.xml configuration file. The `<view-page...>` command within the event-handler tag tells Mach-II the name of the page to display, and the flip-side of this is the corresponding page-view declaration contained in the page-views node of the XML configuration file.

In the mach-ii.xml file you downloaded with the base application skeleton you'll see this on line 54:

```
<page-view name="home" page="/views/home.cfm" />
```

The name attribute of the page-view tag corresponds to the names that are used in the event-handler tag, which in our case is "hello." The page attribute tells the application where to find the actual page to use for the display, which typically will be in the views directory of your application. As with many of the files in your Mach-II applications your views can actually go anywhere, but for convenience let's leave them in the views directory for now. Even though we haven't yet created the CFML page that will be used as our view, go ahead and update line 54 as follows:

```
<page-view name="hello" page="/views/hello.cfm" />
```

Note that because of the application root we've defined for the application, the initial slash in the page attribute actually points to the *application* root, not the web root. Just be aware of this because ColdFusion Components (CFCs)—yes, we'll get there in another guide!—play by slightly different rules.

Creating the view is as simple as making a CFML page called hello.cfm and placing it in the views directory for your application. Create a new page in your editor and enter or copy and paste the following:

```
<html>
    <head>
        <title>Hello Mach-II!</title>
    </head>
    <body>
        <h3>Hello Mach-II!</h3>
    </body>
</html>
```

You probably already know what's next …

**Step 5: Say Hello, Mach-II!**
Before we call the event and see our view in action (patience, young Mach-II learner), let's review what you did to get to this point:

1. You gave your application a name in the Application.cfm file. If you want your Mach-II applications to be independent from one another, just as with any other ColdFusion application you'll need to give each application a unique name in the Application.cfm file (or you may also use the new Application.cfc available in ColdFusion 7).

2. You set the application root (i.e. the directory containing your application) and default event name. The default event is the event that gets called when there is no event explicitly declared in the URL or announced by some other means.
3. You defined the details of the default sayHello event. This consisted of adding a view-page command to the event-handler for the `sayHello` event.
4. You defined the specific CFML page that corresponds to the view named hello and created the actual CFML page that will be displayed when the hello view is called, in this case as a part of the sayHello event.

And now the moment you've all been waiting for! Enter this URL into your browser (adjust as needed to match your particular configuration):
http://localhost/HelloMachII

Or if you want to explicitly declare the event in the URL, you may also enter the following (which is equivalent due to the default event):
http://localhost/HelloMachII/index.cfm?event=sayHello

If everything is set up correctly you should see "Hello Mach-II!" displayed in your browser. Feels like 1974 all over again. (For you youngsters, 1974 was the year that the first Hello World program was created in C at AT&T Labs. I still say we can probably trace this tradition back to ancient Egypt if we try hard enough.)

Bravo! You just created your first Mach-II application. Don't worry if you don't understand much about how this all works at this point. Believe it or not you probably understand more than you think you do.

## What Did I Just Do?

What you just accomplished is taking your first step into the wonderful world of ColdFusion application development with the Mach-II framework. Even though this was a very simple way to begin, many of the principles you learned in building the Mach-II version of Hello World will be used in even the most complex Mach-II development. Armed with your new Mach-II knowledge you can now move on to "Mach-II: An Introduction to the Event Object and Listeners" which is available for download (or if you're one of those "early adopter" types and it isn't there yet, it will be very shortly) at http://www.mach-ii.com This next step in your Mach-II education will teach you all about the event object and listeners, which are a key part of building real-world Mach-II applications. Go download it now!

## About The Author:

Matt is a Principal IT Specialist for the Office of the Sergeant at Arms at the United States Senate in Washington, D.C. He has a Bachelor of Music degree from the University of Nebraska, a Master of Music degree from Wichita State University, and a Master of Science degree in Computer Information Systems from the University of Phoenix.

Matt has been working with ColdFusion since 1996. He is the Release Coordinator for the Mach-II framework and is a member of the editorial board for the ColdFusion Developer's Journal.

You can contact Matt at matt@mach-ii.com.

## Copyright: