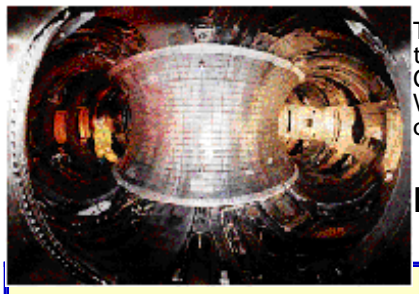
WarpCore

Posted At : July 27, 2007 5:07 PM | Posted By : Jon Hartmann
Related Categories: Experiments, Frameworks



The name's probably already taken, but what the hell. WarpCore is the name I've given to a very simple implementation of an implicit invocation controller scheme in ColdFusion. The concept is simple: some variable is set that contains an event, the WarpCore handles the registration of new drives (controllers), and the WarpCore controls activating functions in those drives.

Declaring the WarpCore

```
<cfscript>
    url.event = "welcome";

    WarpCore = CreateObject("component", "cfc.warpCore").init(
        trigger: "url.event"
    );

    WarpCore.addDrive("controller1");
    WarpCore.addDrive("controller2");
    WarpCore.addDrive("controller3");

    WarpCore.execute();
</cfscript>
```

Example Drive

```
<cfcomponent name="controller1">

    <cffunction name="welcome">
        Welcome from Controller 1<br />
        <cfset setEvent('leave')>
    </cffunction>

    <cffunction name="bye">
        Good-bye from Controller 1<br />
    </cffunction>

</cfcomponent>
```

The WarpCore

```
<cfcomponent name="warpCore">

    <cffunction name="init" returntype="warpCore" output="false">
        <cfargument name="trigger" type="string" required="false" default="request.func"/>

        <cfscript>
            variables.core = StructNew();
            variables.trigger = arguments.trigger;
            variables.cap = 10;
```

```
        return this;
    </cfscript>
</cffunction>

<cfscript>
    function addDrive (sControllerPath) {
        var objController = CreateObject("component", sControllerPath);
        var arrEvents = StructKeyArray(objController);
        var iEvents = ArrayLen(arrEvents);
        var i = 0;

        if (iEvents gt 0) {
            for(i = 1; i lte iEvents; i = i + 1) {
                if (NOT StructKeyExists(variables.core, arrEvents[i])) {
                    StructInsert(variables.core, arrEvents[i], ArrayNew(1));
                }

                ArrayAppend(variables.core[arrEvents[i]], objController[arrEvents[i]]);
            }
        }
    }

    function execute() {
        var eCurrent = "";
        var iCount = variables.cap;
        var iEvents = 0;
        var x = 0;
        var executor = "";

        do {
            eCurrent = getEvent();

            if (StructKeyExists(variables.core, eCurrent)) {
                iEvents = ArrayLen(variables.core[eCurrent]);

                for (x = 1; x lte iEvents; x = x + 1) {
                    executor = variables.core[eCurrent][x];
                    executor();

                    if (getEvent() neq eCurrent) {
                        continue;
                    }
                }
            }
            iCount = iCount - 1;
        } while ((getEvent() neq eCurrent) AND (iCount gte 0));
    }

    function getEvent() {
        return evaluate(variables.trigger);
    }

    function inspect() {
        return variables.core;
    }

    function setEvent (value) {
        "#variables.trigger#" = arguments.value;
    }
</cfscript>
```

```
</cfcomponent>
```

Any way, thats my 2 hour implementation of an II control scheme like those in Mach-II and Model Glue, just without the XML. I know its ugly: I've already got ideas on better ways to handle the setup. The key thing to me is that I want to be able to detect and register event handlers without resorting to predefining them in XML. I love XML, but I just hate needing to write large chunks of code in XML when I can just have the system automatically register the events for me.