

Building Advanced Workflows with ColdSpring

Process & Methodology

Dan Skaggs

<http://dan.skaggsfamily.ws>

dan.skaggs@web-meister.com



CF.Objective()

Agenda

Hopefully what you expected



CF.Objective()

Agenda

- Two Faces of ColdSpring
 - Dependency Injection
 - Application Configuration
- The Objective
 - Document management system
- The Execution
 - Using ColdSpring to define workflows
- The Inevitable
 - Managing workflow changes

About Me

You know you want to know



CF.Objective()

About Me

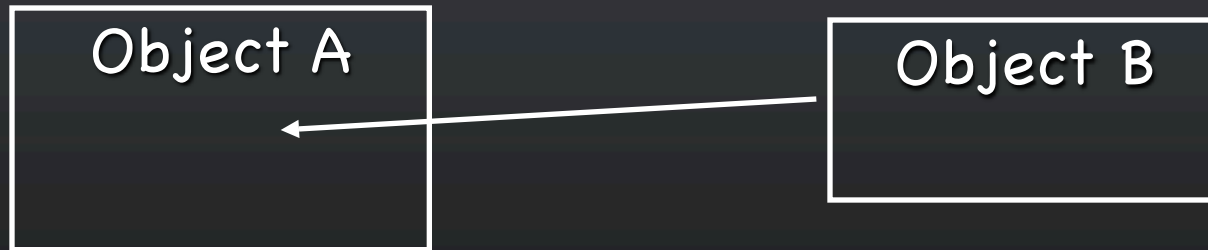
- President, Web-Meister Designs
- ColdFusion developer since 2000
- Started with CF 4.5 and Spectra 1.0
- Independent consultant since 2006
- Contributing Partner to Model-Glue framework
- Amateur (Ham) Radio enthusiast
- Avid shooter

Two Faces of ColdSpring



CF.Objective()

Dependency Injection

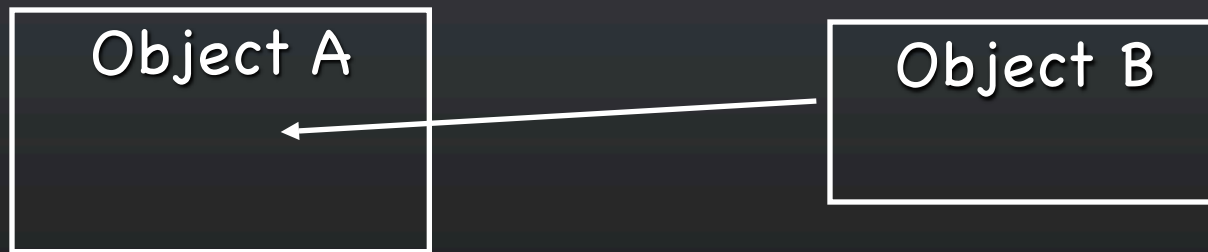


Before ColdSpring

```
<cfset ObjectB = createObject("component", "ObjectB").init( ) />
```

```
<cfset ObjectA = createObject("component", "ObjectA").init( Object B ) />
```

Dependency Injection

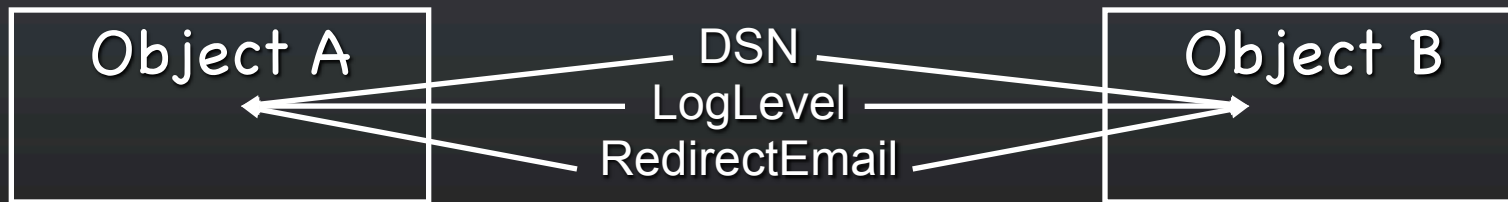


After ColdSpring

```
<bean name="ObjectA" class="path.to.ObjectA">
  <constructor-arg name="ObjectB">
    <ref bean="ObjectB" />
  </constructor-arg>
</bean>
```

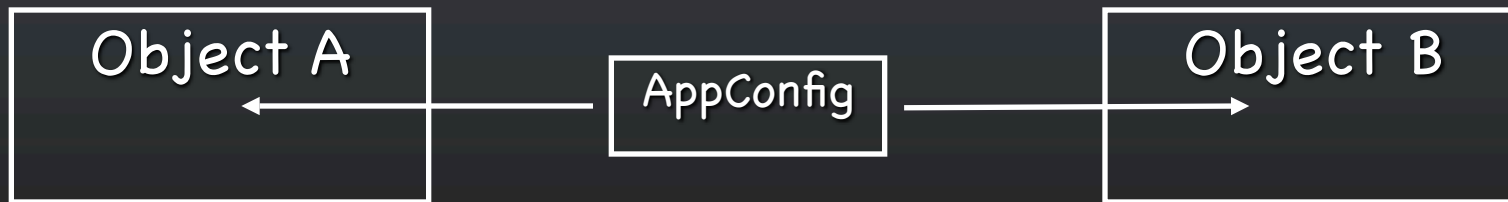
```
<cfset objA = Application.BeanFactory.getBean("ObjectA") />
```

Application Configuration



```
<bean name="ObjectA" class="path.to.ObjectA">  
  <constructor-arg name="DSN">  
    <value>${DSN}</value>  
  </constructor-arg>  
  <constructor-arg name="LogLevel">  
    <value>${logLevel}</value>  
  </constructor-arg>  
  <constructor-arg name="RedirectEmail">  
    <value>${redirectEmail}</value>  
  </constructor-arg>  
</bean>
```

Application Configuration



```
<bean name="AppConfig" class="coldspring.beans.factory.config.MapFactoryBean">
  <property name="SourceMap">
    <map>
      <entry key="DSN"><value>${DSN}</value></entry>
      <entry key="logLevel"><value>${logLevel}</value></entry>
      <entry key="redirectEmail"><value>${redirectEmail}</value></entry>
    </map>
  </property>
</bean>
```

```
<bean name="ObjectA" class="path.to.ObjectA">
  <constructor-arg name="config">
    <ref bean="AppConfig" />
  </constructor-arg>
</bean>
```

The Objective

What We're Trying to Solve



CF.Objective()

Document Management System

- Simple document actions (upload, review, approve, reject, archive, unarchive)
- Overly encapsulated to illustrate concept
- No MVC frameworks - “controller” type functions in CFM files
- Fake document database using WDDX written to the filesystem
- Why? So you can extract and run/play

Business Process

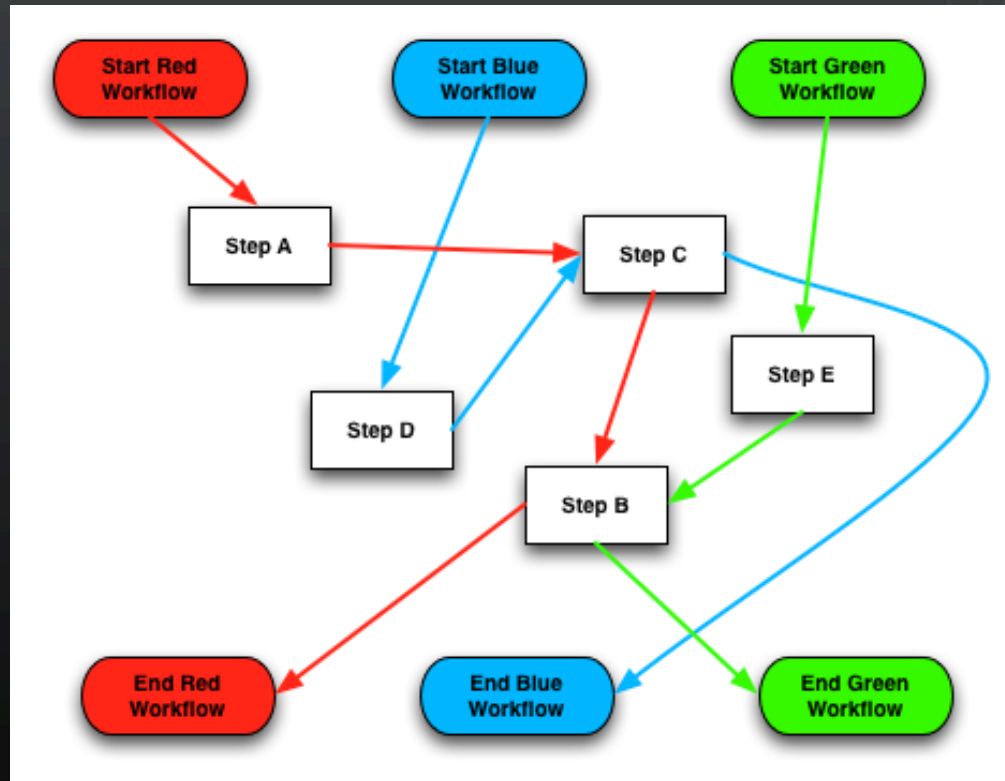
A user uploads a new document into the system

A user reviews the document and rejects or approves it

A user may archive a document when it is no longer needed

A user must be able to “unarchive” a document that is mistakenly moved to the archive

Log entries must be created on each action



The Execution

How We're Going to Solve It



CF.Objective()

Workflow Object Concept

- Initialized by ColdSpring with metadata and an array of steps
- No logic contained in the workflow bean
- Create as many workflow beans as necessary in ColdSpring
- Can contain n steps

Individual Workflow Config

```
<bean id="CreateDocument" class="model.Workflow">
  <constructor-arg name="WorkflowConfig">
    <map>
      <entry key="Name"><value>CreateDocument</value></entry>
      <entry key="Description"><value>Add a new document to the system</value></entry>
    </map>
  </constructor-arg>

  <constructor-arg name="Steps">
    <list>
      <map>
        <entry key="delegate"><ref bean="DocumentManager" /></entry>
        <entry key="action"><value>renameFile</value></entry>
      </map>
      <map>
        <entry key="delegate"><ref bean="DocumentManager" /></entry>
        <entry key="action"><value>save</value></entry>
      </map>
      <map>
        <entry key="delegate"><ref bean="LogManager" /></entry>
        <entry key="action"><value>logDocumentCreated</value></entry>
      </map>
    </list>
  </constructor-arg>
</bean>
```

Workflow Processor Concept

- Initialized by ColdSpring with a structure of Workflow objects
- `process(workflowName)` used to initiate a workflow
- `process()` method only executes if the `workflowName` attribute matches a pre-defined workflow
- Maintains an array of result messages that each step in a workflow can add to for an audit trail of the entire workflow

Workflow Processor Config

```
<bean id="WorkflowProcessor" class="model.WorkflowProcessor">
  <constructor-arg name="registry">
    <map>
      <entry key="CreateDocument"><ref bean="CreateDocument" /></entry>
      <entry key="ApproveDocument"><ref bean="ApproveDocument" /></entry>
      <entry key="RejectDocument"><ref bean="RejectDocument" /></entry>
      <entry key="ArchiveDocument"><ref bean="ArchiveDocument" /></entry>
      <entry key="RestoreDocument"><ref bean="RestoreDocument" /></entry>
    </map>
  </constructor-arg>
</bean>
```

Application Walk- Through and Demo



CF.Objective()

The Inevitable

Handling Workflow Changes



CF.Objective()

The boss wants what?

- Changes to workflows are an eventuality
 - If workflow steps already exist, simply change the XML configuration and reinitialize the app
 - New workflow actions are plugged into the XML configuration once coded and tested
- Let's see how we'd add an email notification to selected steps of the workflow

Workflow Change Demo



CF.Objective()

Benefits and Drawbacks

Benefits

- All workflow definitions are kept in one place rather than scattered throughout the CFML code
- Less chance of introducing bugs when adding/updating workflows
- Easily test workflows outside normal application operation
- Reuse your existing model objects / methods

Drawbacks

- More up-front planning needed for architecture setup
- All values/objects needed at start of workflow

Questions / Comments?

No Tomatoes, Please



CF.Objective()

Ask Me Later

- Email: dan.skaggs@web-meister.com
- Blog: <http://dan.skaggsfamily.ws>
- Twitter: dskaggs
- GTalk: dan.skaggs@web-meister.com