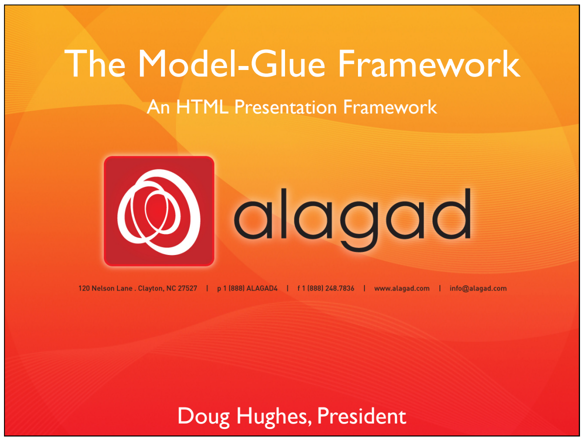


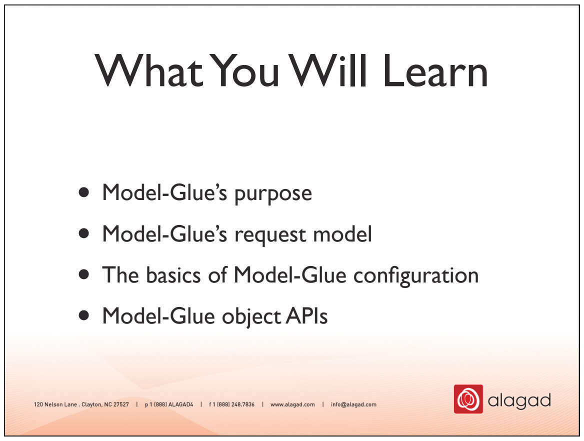
The Model-Glue Framework

An HTML Presentation Framework

alagad

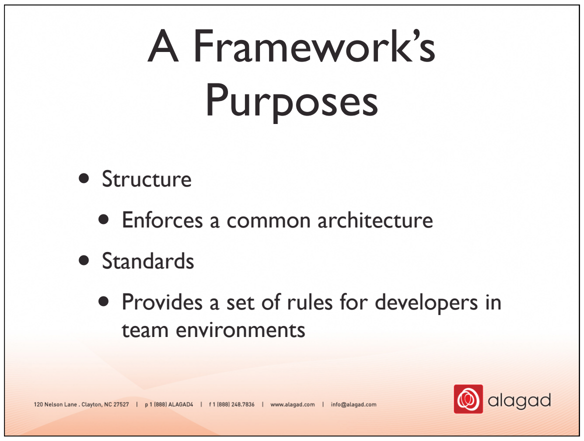
120 Nelson Lane . Clayton, NC 27527 | p 1 (888) ALAGAD4 | f 1 (888) 248.7836 | www.alagad.com | info@alagad.com

Doug Hughes, President



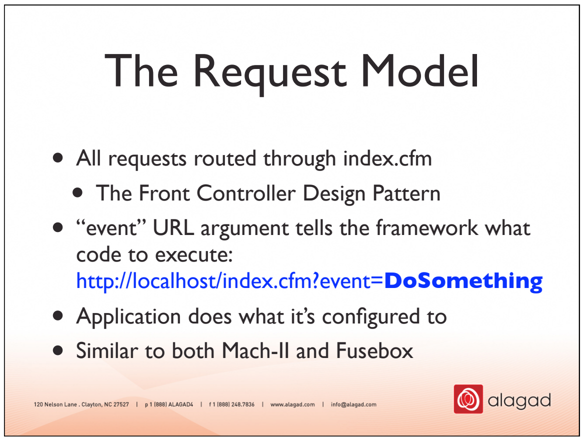
What You Will Learn

- Model-Glue's purpose
- Model-Glue's request model
- The basics of Model-Glue configuration
- Model-Glue object APIs



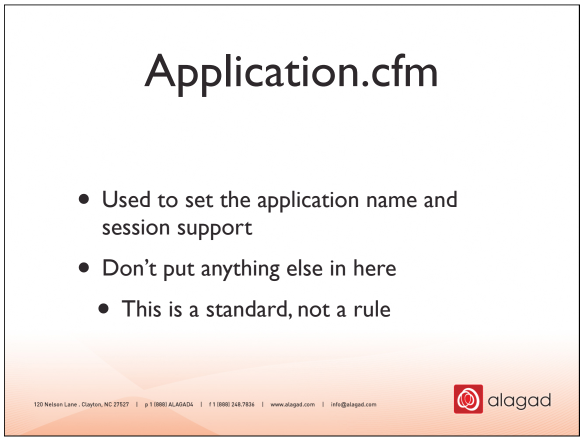
A Framework's Purposes

- Structure
  - Enforces a common architecture
- Standards
  - Provides a set of rules for developers in team environments



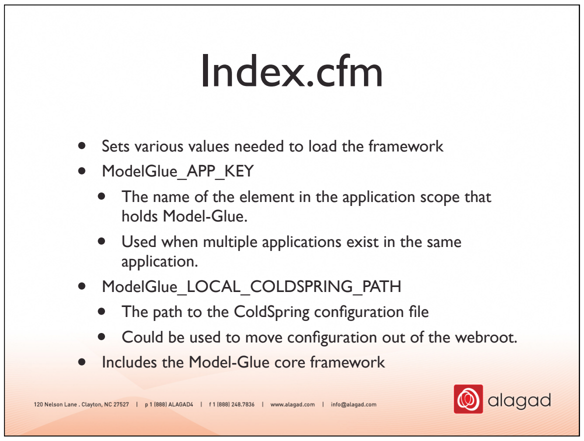
The Request Model

- All requests routed through index.cfm
  - The Front Controller Design Pattern
- "event" URL argument tells the framework what code to execute: http://localhost/index.cfm?event=**DoSomething**
- Application does what it's configured to
- Similar to both Mach-II and Fusebox



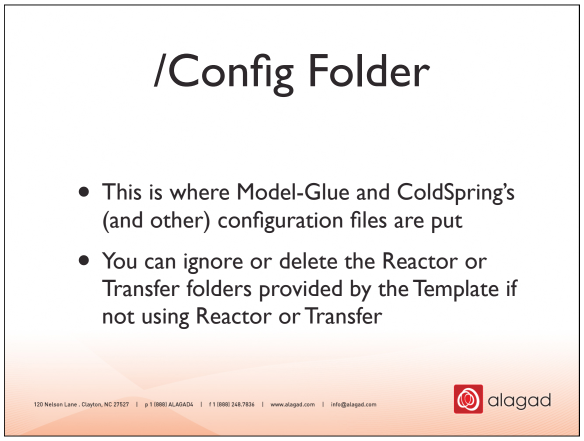
Application.cfm

- Used to set the application name and session support
- Don't put anything else in here
  - This is a standard, not a rule



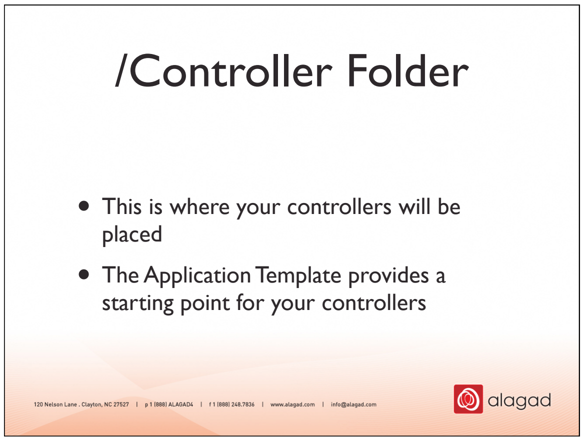
Index.cfm

- Sets various values needed to load the framework
- ModelGlue_APP_KEY
  - The name of the element in the application scope that holds Model-Glue.
  - Used when multiple applications exist in the same application.
- ModelGlue_LOCAL_COLDSPRING_PATH
  - The path to the ColdSpring configuration file
  - Could be used to move configuration out of the webroot.
- Includes the Model-Glue core framework



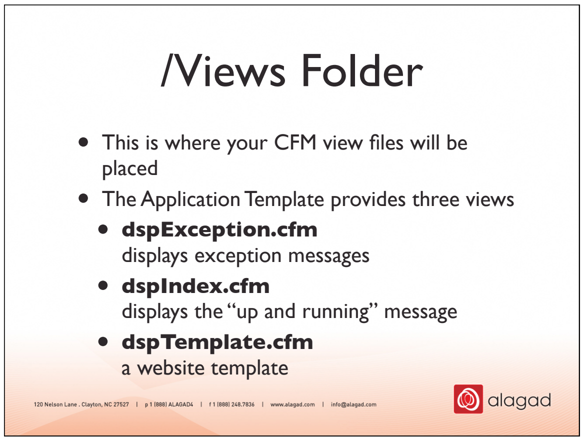
/Config Folder

- This is where Model-Glue and ColdSpring's (and other) configuration files are put
- You can ignore or delete the Reactor or Transfer folders provided by the Template if not using Reactor or Transfer



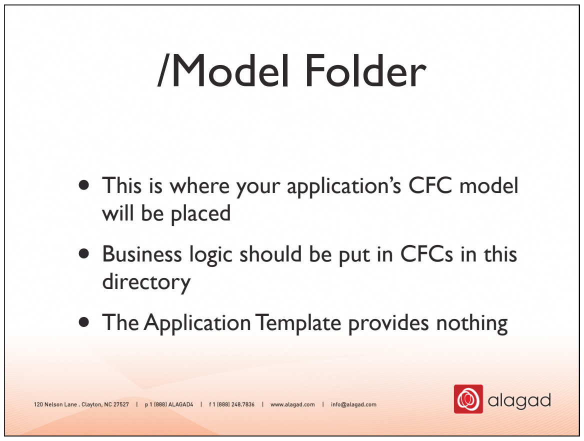
/Controller Folder

- This is where your controllers will be placed
- The Application Template provides a starting point for your controllers



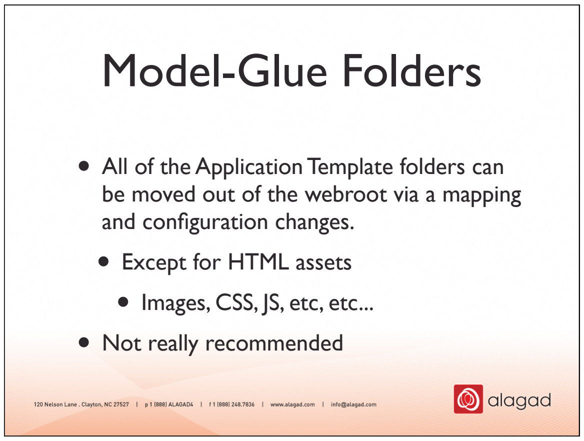
/Views Folder

- This is where your CFM view files will be placed
- The Application Template provides three views
  - **dspException.cfm**
    displays exception messages
  - **dspIndex.cfm**
    displays the "up and running" message
  - **dspTemplate.cfm**
    a website template



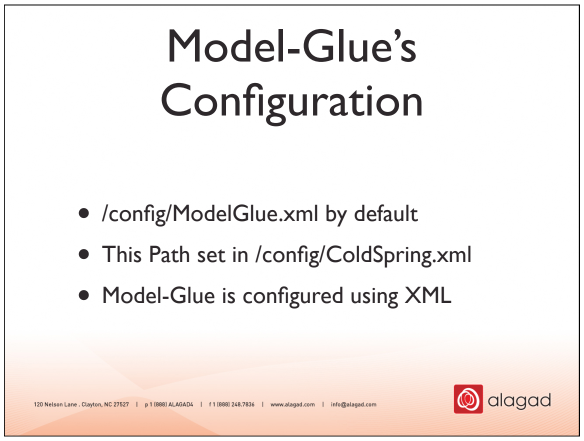
/Model Folder

- This is where your application's CFC model will be placed
- Business logic should be put in CFCs in this directory
- The Application Template provides nothing



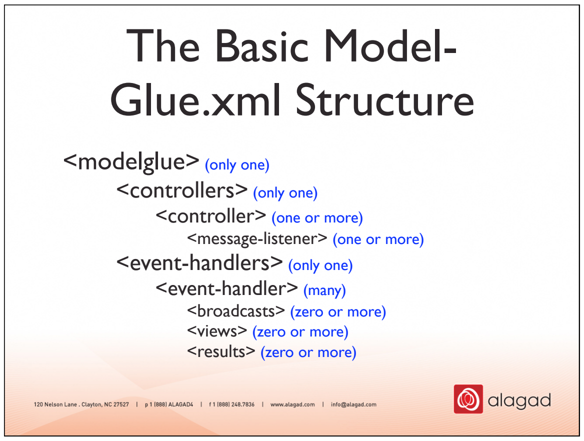
Model-Glue Folders

- All of the Application Template folders can be moved out of the webroot via a mapping and configuration changes.
  - Except for HTML assets
    - Images, CSS, JS, etc, etc...
- Not really recommended



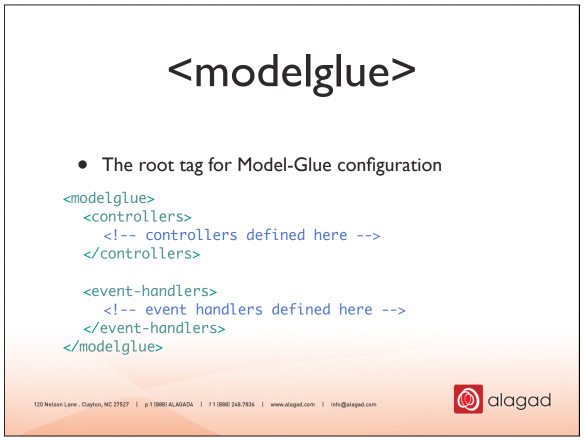
Model-Glue's Configuration

- /config/ModelGlue.xml by default
- This Path set in /config/ColdSpring.xml
- Model-Glue is configured using XML



The Basic Model-Glue.xml Structure

```
<modelglue> (only one)
    <controllers> (only one)
        <controller> (one or more)
            <message-listener> (one or more)
    <event-handlers> (only one)
        <event-handler> (many)
            <broadcasts> (zero or more)
            <views> (zero or more)
            <results> (zero or more)
```

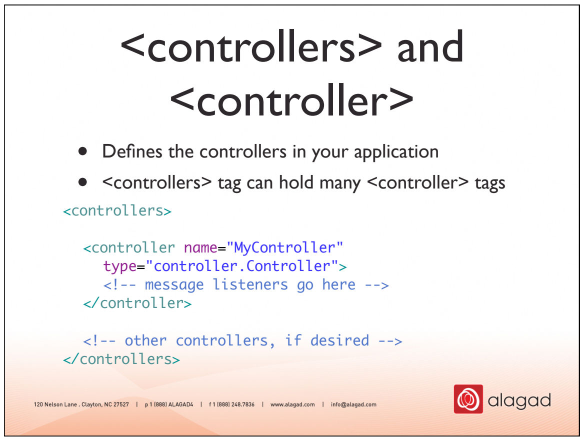


<modelglue>

- The root tag for Model-Glue configuration

```
<modelglue>
  <controllers>
    <!-- controllers defined here -->
  </controllers>

  <event-handlers>
    <!-- event handlers defined here -->
  </event-handlers>
</modelglue>
```



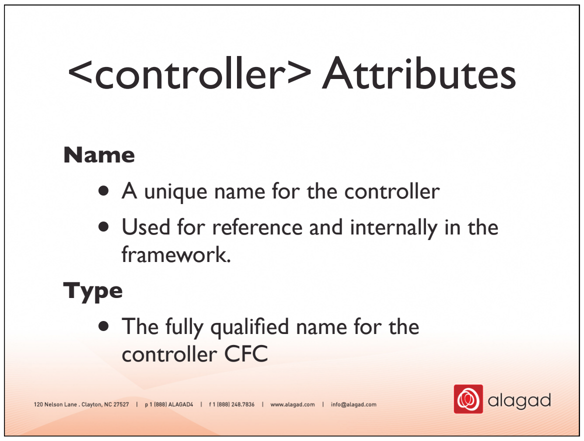
<controllers> and <controller>

- Defines the controllers in your application
- <controllers> tag can hold many <controller> tags

```
<controllers>

  <controller name="MyController"
    type="controller.Controller">
    <!-- message listeners go here -->
  </controller>

  <!-- other controllers, if desired -->
</controllers>
```



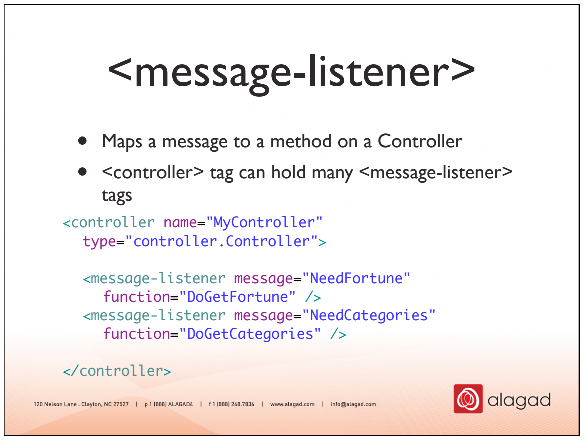
<controller> Attributes

Name

- A unique name for the controller
- Used for reference and internally in the framework.

Type

- The fully qualified name for the controller CFC



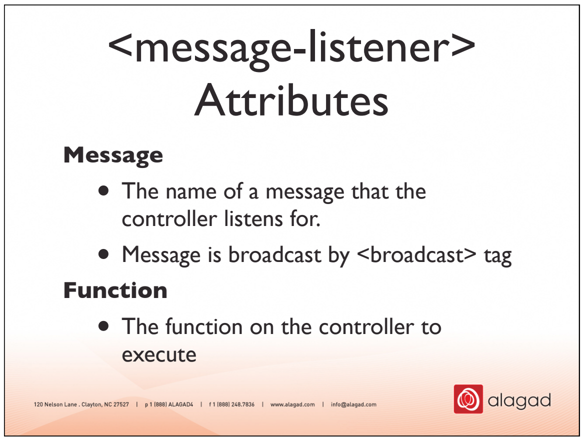
<message-listener>

- Maps a message to a method on a Controller
- <controller> tag can hold many <message-listener> tags

```
<controller name="MyController"
  type="controller.Controller">

  <message-listener message="NeedFortune"
    function="DoGetFortune" />
  <message-listener message="NeedCategories"
    function="DoGetCategories" />

</controller>
```



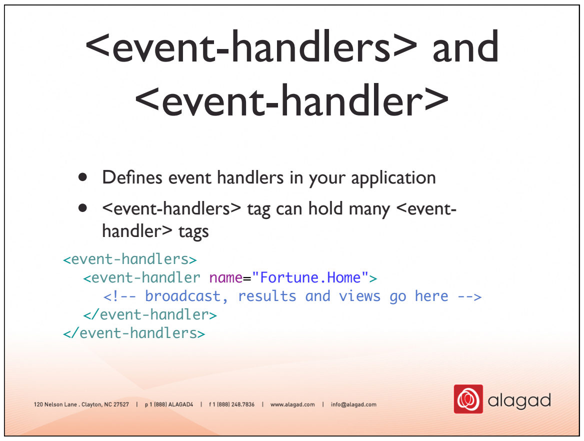
<message-listener> Attributes

Message

- The name of a message that the controller listens for.
- Message is broadcast by <broadcast> tag

Function

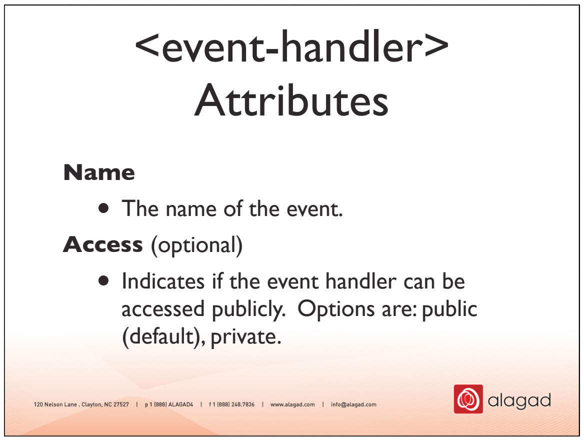
- The function on the controller to execute



<event-handlers> and <event-handler>

- Defines event handlers in your application
- <event-handlers> tag can hold many <event-handler> tags

```
<event-handlers>
  <event-handler name="Fortune.Home">
    <!-- broadcast, results and views go here -->
  </event-handler>
</event-handlers>
```



<event-handler> Attributes

Name

- The name of the event.

Access (optional)

- Indicates if the event handler can be accessed publicly. Options are: public (default), private.

<broadcasts> and <message>

- Defines messages that the event will broadcast.
- Controllers listen for these messages
- <broadcasts> tag can have many <message> tags

```
<event-handler name="Fortune.Home">
   <broadcasts>
      <message name="NeedCategories" />
   </broadcasts>
   <!-- results and views go here -->
</event-handler>
```

<message> Attributes

Name

- The name of the message
- This name is what is listened for by <message-listener> tags

<results> and <result>

- Results map to other event handlers
- Results are set in the controller (or one unnamed)
- <results> tag can hold many <result> tags

```
<event-handler name="Fortune.Home">
   <!-- broadcasts go here -->
   <results>
      <result name="example" do="Fortune.Example" />
      <result do="view.template" />
   </results>
   <!-- views go here -->
</event-handler>
```

<result> Attributes

(abridged)

Do
- Specifies an event handler to execute

Name (optional)
- The name of the result.
- Can choose not to provide this for one result
- If not provided the result is always added.
- Good for site wide templates.
- Results are specified in the controller by name.

Redirect (optional)
- Indicates if result should redirect. Defaults to false.
- Redirect terminates execution immediately.

Append (optional)
- A list of values from the viewState to append to the url (more on this later)

<views> and <include>

- Defines views that the event will display
- <views> tag can have many <include> tags

```
<event-handler name="Fortune.Home">
   <!-- broadcasts and results go here -->
   <views>
      <include name="body" template="dspIndex.cfm" />
   </views>
</event-handler>
```

<include> Attributes

Name

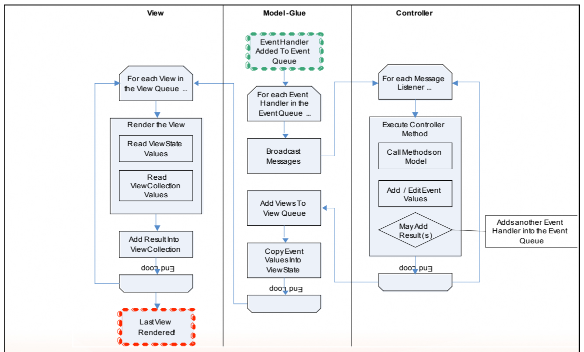
- The name of the rendered view in the viewCollection

Template

- The name of a CFM file to include from the view mappings. (More on this later)

Append

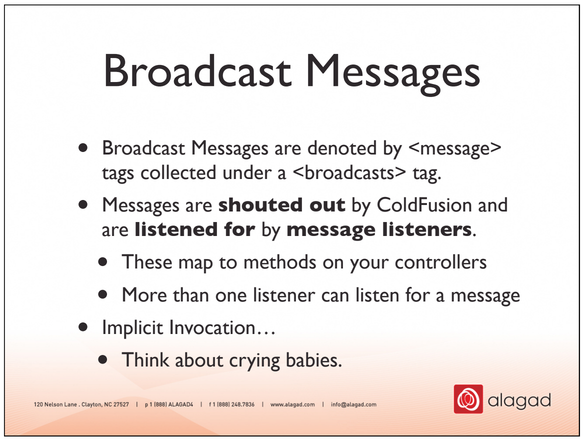
- Indicates is includes with the same name should be appended to each other


View
Model-Glue
Controller
Event Handler Added To Event Queue
For each View in the View Queue ...
For each Event Handler in the Event Queue ...
For each Message Listener ...
Render the View
Read ViewState Values
Read ViewCollection Values
Broadcast Messages
Execute Controller Method
Call Methods on Model
Add / Edit Event Values
Add Views To View Queue
May Add Result(s)
Adds another Event Handler into the Event Queue
Add Result Into ViewCollection
Copy Event Values Into ViewState
End Loop
End Loop
End Loop
Last View Rendered

Request Flow

Event Handlers

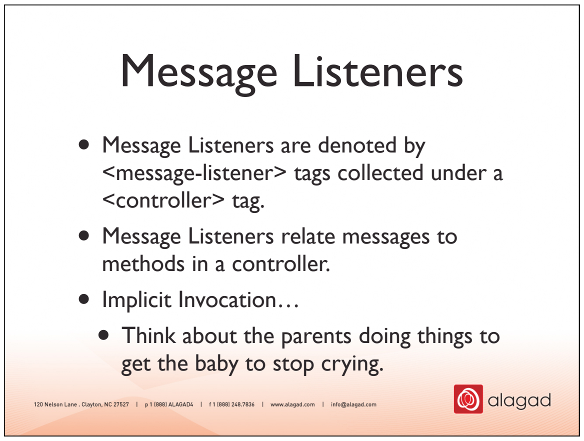
- Denoted by the <event-handler> tag.
- Roughly equivalent to a "page".
- Requested by event url argument.
- Implicitly does "stuff" via broadcast messages
- Branches via results
- Display "stuff" via views



Broadcast Messages

- Broadcast Messages are denoted by <message> tags collected under a <broadcasts> tag.
- Messages are **shouted out** by ColdFusion and are **listened for** by **message listeners**.
  - These map to methods on your controllers
  - More than one listener can listen for a message
- Implicit Invocation…
  - Think about crying babies.

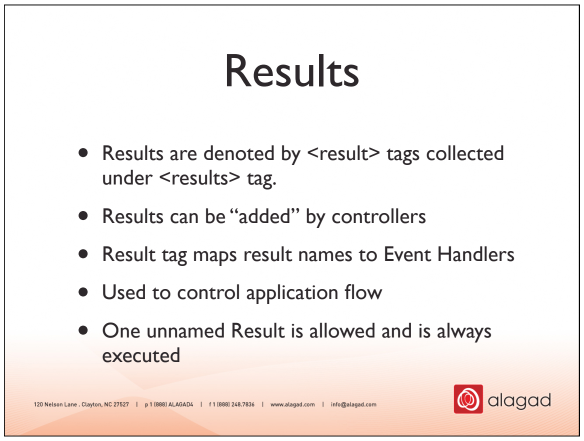
alagad



Message Listeners

- Message Listeners are denoted by <message-listener> tags collected under a <controller> tag.
- Message Listeners relate messages to methods in a controller.
- Implicit Invocation…
  - Think about the parents doing things to get the baby to stop crying.

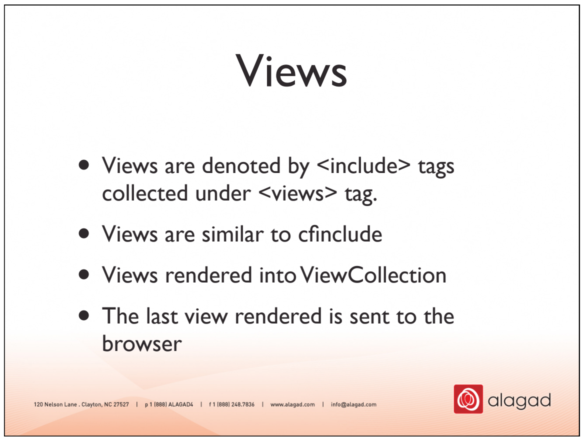
alagad



Results

- Results are denoted by <result> tags collected under <results> tag.
- Results can be “added” by controllers
- Result tag maps result names to Event Handlers
- Used to control application flow
- One unnamed Result is allowed and is always executed

alagad



Views

- Views are denoted by <include> tags collected under <views> tag.
- Views are similar to cfinclude
- Views rendered into ViewCollection
- The last view rendered is sent to the browser

alagad

The Event Object

- Holds form and URL variables
  - Form has precedence by default
- Controller can add and manipulate values in the ViewState.
- The Event object exists while event handlers are being executed

alagad

The Event API

(Abridged)

Event.setValue(name, value)

- Returns nothing

Event.getValue(name, default)

- Returns any type

Event.valueExists(name)

- Returns boolean

alagad

The ViewState Object

- ViewState is populated from the Event Object
- ViewState exists while views are being rendered

alagad

The ViewState API

(Abridged)

ViewState.getValue(name, default)

- Returns any value

ViewState.exists(name)

- Returns a boolean

alagad

The ViewCollection

- As views are rendered they're added into the ViewCollection
- Other views can get rendered views from the ViewCollection

The ViewCollection API

(Abridged)

ViewCollection.getView(name)

- Returns a string of HTML

ViewCollection.exists(name)

- Returns a boolean

Controlling Application Flow With Results

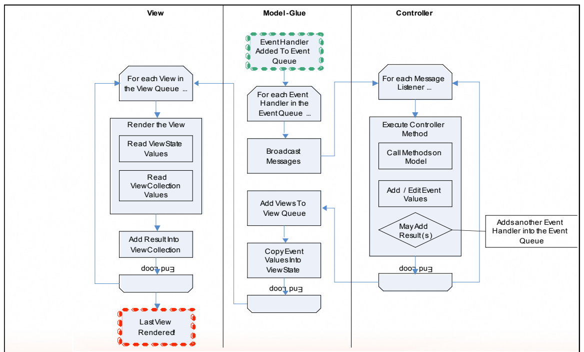
- In any controller you can add a result to the Event:

```
<cfset arguments.event.addResult("ResultName") />
```

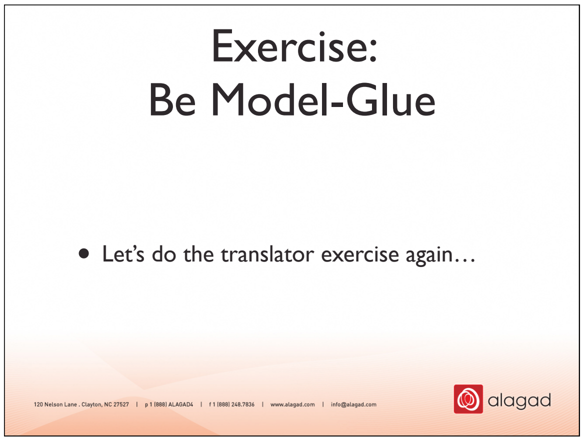
- Result Name maps to a result tag in the Event Handler:

```
<result name="ResultName" do="EventName" />
```

- Results enqueue the specified event
- All results are processed before any views

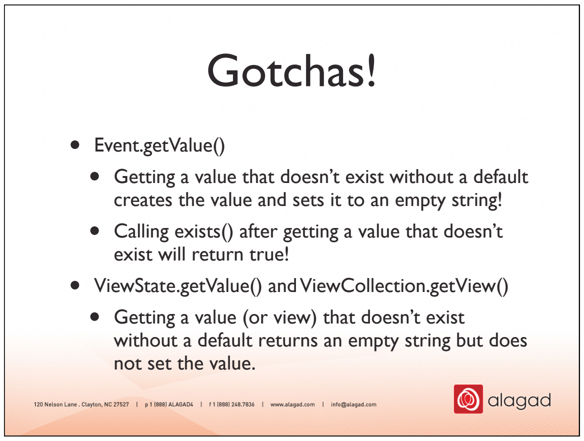

View
Model-Glue
Controller
EventHandler Added To Event Queue
For each View in the View Queue ...
Render the View
Read ViewState Values
Read ViewCollection Values
Add Result Into ViewCollection
End Loop
Last View Rendered
For each Event Handler in the EventQueue ...
Broadcast Messages
Add Views To View Queue
Copy Event Values Into ViewState
End Loop
For each Message Listener ...
Execute Controller Method
Call Methods on Model
Add / Edit Event Values
May Add Result(s)
Adds another Event Handler into the Event Queue
End Loop

Request Flow



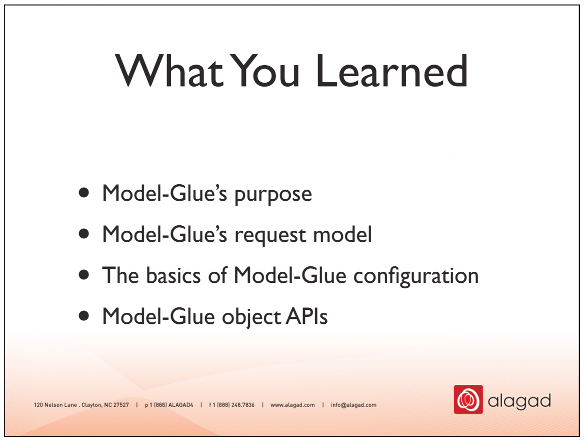
Exercise: Be Model-Glue

- Let’s do the translator exercise again…



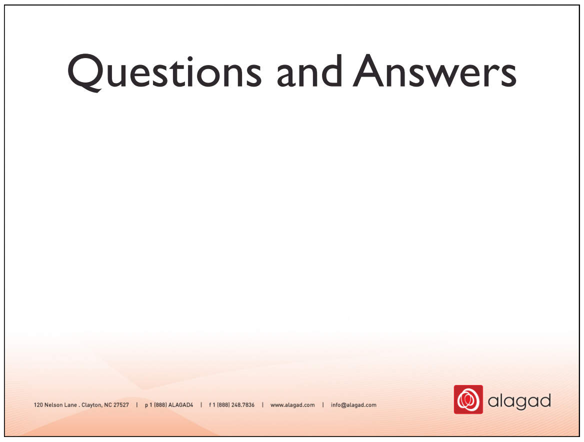
Gotchas!

- Event.getValue()
  - Getting a value that doesn’t exist without a default creates the value and sets it to an empty string!
  - Calling exists() after getting a value that doesn’t exist will return true!
- ViewState.getValue() and ViewCollection.getView()
  - Getting a value (or view) that doesn’t exist without a default returns an empty string but does not set the value.

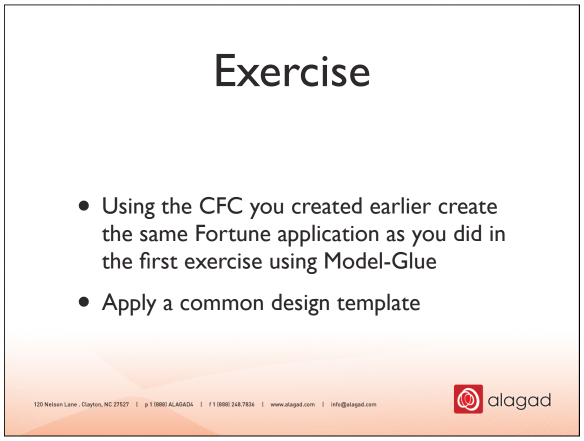


What You Learned

- Model-Glue’s purpose
- Model-Glue’s request model
- The basics of Model-Glue configuration
- Model-Glue object APIs

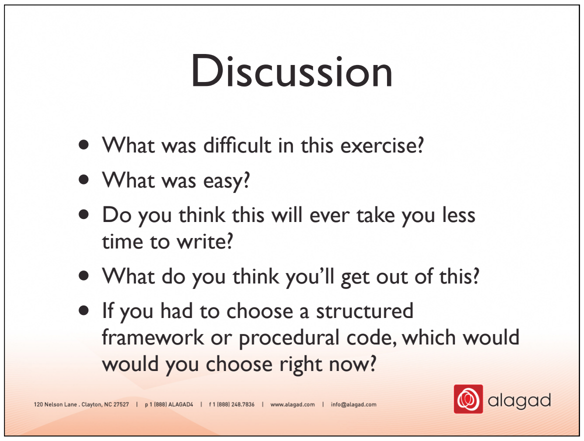


Questions and Answers



Exercise

- Using the CFC you created earlier create the same Fortune application as you did in the first exercise using Model-Glue
- Apply a common design template



Discussion

- What was difficult in this exercise?
- What was easy?
- Do you think this will ever take you less time to write?
- What do you think you'll get out of this?
- If you had to choose a structured framework or procedural code, which would would you choose right now?