Unit 3: Building Templates

Unit Objectives

After completing this unit, you should be able to:

- Plan a template strategy for your site
- Convert an HTML-based design into a ColdFusion Custom Tag
- Deploy a design into a CommonSpot template
- Define editable regions in a template
- Register an external CSS file with a template

Unit Topics

- Defining your template architecture
- Building a Base Template
- Implementing your Graphical Design
- Implementing CSS-based designs
- Making Styles Available to Contributors
- Implementing a Printer Friendly Format

Defining your template architecture

Every page in a CommonSpot site is created from a template which, guided by logic embedded on a ColdFusion page, and settings retrieved from the content database, defines the default look and feel for a new page. New templates can also be created from existing templates. By layering existing templates on top of each other, a new template can inherit its look and feel as well as other attributes from all of the layered templates.

Using templates to build new pages allows content authors to concentrate on the site content without having to worry about maintaining design consistency. Since most content authors do not create templates in CommonSpot, the site administrator or the developer(s) must create these “site page blueprints” for content contributor use. As a CommonSpot developer it is your job to engineer these templates to be as simple, flexible, and intuitive for your contributors as possible.

When considering how to design your templates, let the following four rules be your guide:

1. **Implement the fewest number of editable regions possible:** The more editable regions you have, the slower a page will render on an author server. Fast executing pages usually results in happy contributors. Happy contributors are the key to keeping your content updated on a timely basis.
2. **Keep the implementation flexible: A** user should be able to toggle between template layout variations without having to copy and paste content..
3. **Keep it simple**: Engineer your pages to automatically adapt to inheriting look and feel based on where they are placed in a site.
4. **Think about the future**: Sites are typically redesigned every two years. Is your implementation flexible enough to implement a design change without requiring user intervention to modify the content of existing pages?

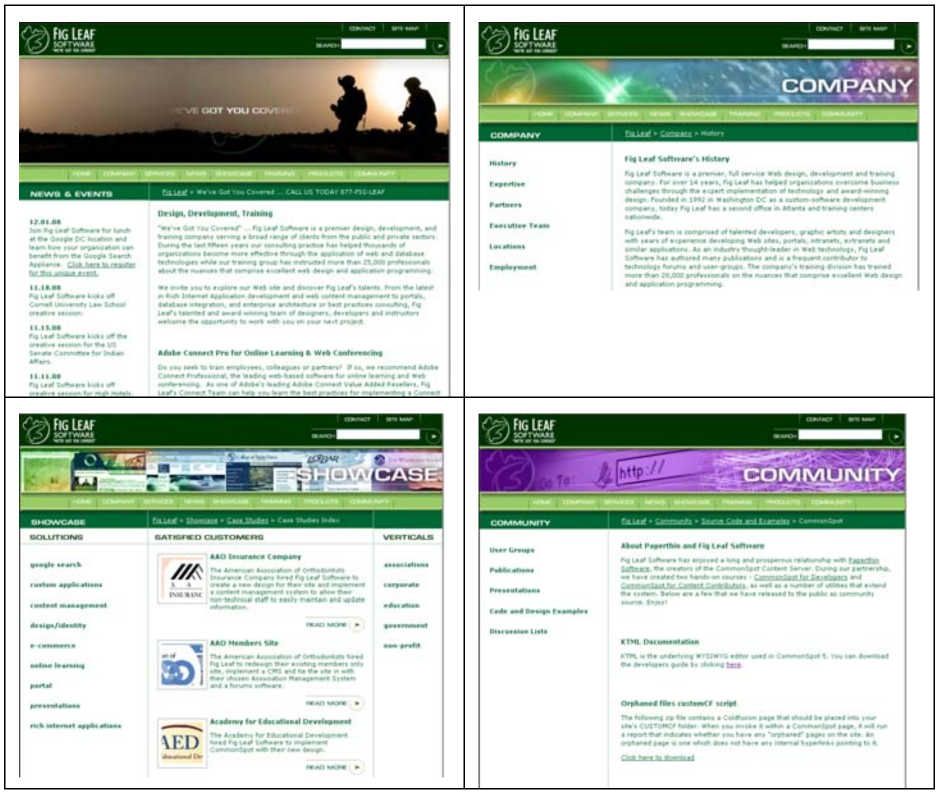
Identifying base templates

Before you can start implementing templates, you need to decide how many base templates you will need to codify. As their name implies, Base templates are the lowest-level of templates within the CommonSpot hierarchy and are coded using a combination of basic ColdFusion tags as well as two CommonSpot tags that deploy the CommonSpot contributor toolbar and identify editable regions.

Most web sites can be implemented using just two or fewer base templates.

- One base template for your site's home page
- One base template for every other content page in the web site

When evaluating how many base templates you require, compare the pages in your site. All pages that have a similar look and feel should be tied into a single base template as this will help to minimize the amount of work required to implement a redesign.



***www.figleaf.com* was implemented using a single base template**

Identifying Editable Regions

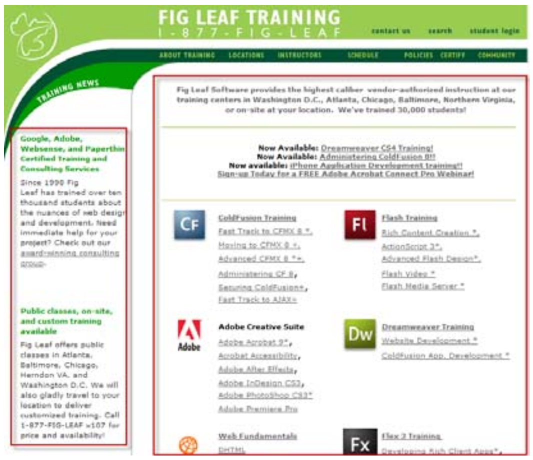
Once you have identified the number of templates required to implement your site you must then identify which areas of a web page should be editable from within CommonSpot. All page content falls into one of the following categories:

1. **Static content**
   The content does not change under any circumstances. Your site framing graphics and copyright notifications typically fall into this category. This content is typically coded on the base template and as such, only a developer would be able to change it.

2. **Dynamic content that only changes at page creation time**
   This can include page header graphics that identify where your page is located, as well as breadcrumb navigation. Typically this program logic is embedded directly in the base template where only a developer would be able to change it.

3. **Dynamic content that changes frequently**
   This may include content being pulled dynamically from a database, ERP system, or some other back-office resource. This can be implemented as an editable element on a page where either a developer or contributor can control the content.

4. **Content that is editable by contributors**
   This is implemented as one of CommonSpot's editable elements and typically represents most framed content of the site. **A page should have no more than SIX editable areas in order to give it the fastest rendering time possible in author mode**.



Google, Adobe, Websense, and Paperthin Certified Training and Consulting Services

Since 1990 Fig Leaf has trained over ten thousand students about the nuances of web design and development. Need immediate help for your project? Check out our award-winning consulting group.

Public classes, on-site, and custom training available

Fig Leaf offers public classes in Atlanta, Baltimore, Chicago, Herndon VA, and Washington D.C. We will also gladly travel to your location to deliver customized training. Call 1-877-FIG-LEAF x107 for price and availability!

Fig Leaf Software provides the highest caliber vendor-authorized instruction at our training centers in Washington D.C., Atlanta, Chicago, Baltimore, Northern Virginia, or on-site at your location. We've trained 30,000 students!

Now Available: Dreamweaver CS4 Training!
Now Available: Administering ColdFusion 8!!
Now available: iPhone Application Development training!!
Sign-up Today for a FREE Adobe Acrobat Connect Pro Webinar!

ColdFusion Training
Fast Track to CFMX 8 *,
Moving to CFMX 8 +,
Advanced CFMX 8 *+,
Administering CF 8,
Securing ColdFusion+,
Fast Track to AJAX+

Flash Training
Rich Content Creation *,
ActionScript 3*,
Advanced Flash Design*,
Flash Video *
Flash Media Server *

Adobe Creative Suite
Adobe Acrobat 9*,
Acrobat Accessibility,
Adobe After Effects,
Adobe InDesign CS3,
Adobe PhotoShop CS3*
Adobe Premiere Pro

Dreamweaver Training
Website Development *
ColdFusion App. Development *

Web Fundamentals
DHTML

Flex 2 Training
Developing Rich Client Apps*,

***http://training.figleaf.com* has just two editable areas**

Walkthrough 3-1: Analyzing your design

In this walkthrough you will analyze the design given to you by your graphic designers in order to identify the number of base templates and editable regions that you will need to implement.

Steps

1. Open a web browser and navigate to http://[serverip]/solution
2. Browse through the pages in Read mode with your instructor and discuss the number of base templates required to implement this design
3. Based on your survey of the pages in the Solution site, answer the following questions:

   Which elements are static?

   Which elements would only change at page creation time?

   Which content changes frequently?

   Which content should be editable by contributors?

4. Repeat step 3 for the following web sites:

   http://www.figleaf.com
   http://www.nps.gov
   http://lsr.nisgroup.com

-- End of Walkthrough --

Building a Base Template

When you define a new CommonSpot website, the following two templates will automatically be defined and registered for you within the CommonSpot Administrator:

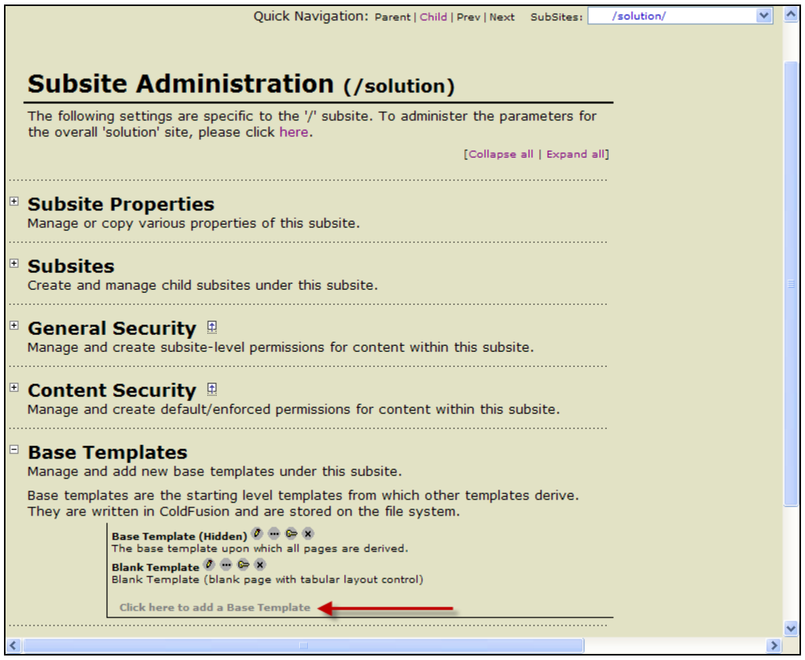
- Base Template: /templates/template-basepage.cfm
- Blank Template: /templates/template-blanklayout.cfm

Base templates are CFM files containing the following items:

- Static HTML. Note that CommonSpot will automatically generate the <html>, <head>, and <body> tags for you.
- Code that invokes the CommonSpot menu bar
- Code that defines dynamic, editable regions on each page

Registering a Base Template

Base templates should be placed in the root /templates/ folder of your site. Once there you need to register it within the CommonSpot Administrator under the Base Templates heading as depicted below:


Quick Navigation: Parent | Child | Prev | Next SubSites: /solution/

Subsite Administration (/solution)

The following settings are specific to the '/' subsite. To administer the parameters for the overall 'solution' site, please click here.

[Collapse all | Expand all]

Subsite Properties
Manage or copy various properties of this subsite.

Subsites
Create and manage child subsites under this subsite.

General Security
Manage and create subsite-level permissions for content within this subsite.

Content Security
Manage and create default/enforced permissions for content within this subsite.

Base Templates
Manage and add new base templates under this subsite.

Base templates are the starting level templates from which other templates derive. They are written in ColdFusion and are stored on the file system.

Base Template (Hidden)
The base template upon which all pages are derived.

Blank Template
Blank Template (blank page with tabular layout control)

Click here to add a Base Template

CommonSpot Subsite Administration

Creating a Simple Base Template

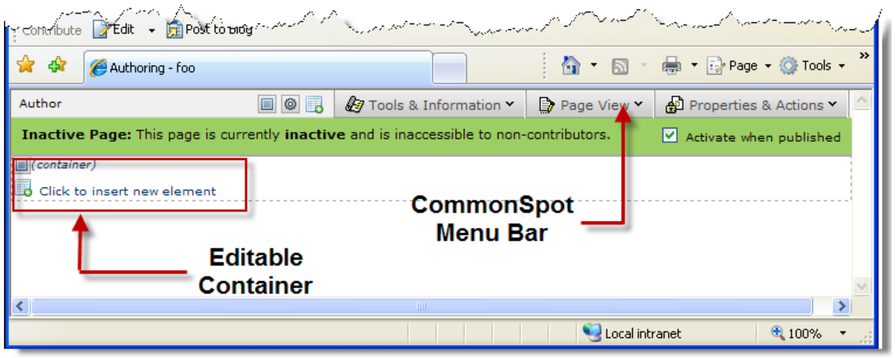
All base templates contain a <cfinclude> tag that displays the CommonSpot menu bar, along with at least one <cfmodule> tag call that defines an editable region.

Therefore, the simplest base template that you could possibly produce would be similar to the following:

```
<!--- Page mode ui (the three CommonSpot Indicators) --->
<CFINCLUDE TEMPLATE="/commonspot/pagemode/pagemodeui.cfm">

<!--- dynamic, editable area --->
<CFMODULE
TEMPLATE="/commonspot/utilities/ct-render-named-
element.cfm"
ElementType="container"
elementName="myeditablecontent">
```

Note that in order to have static HTML render on a page, it must be wrapped in <cfoutput> tags, regardless of whether that HTML contains any pound-sign delimited ColdFusion expressions. Conversely, a <cfmodule> call to ct-render-named-element.cfm should NOT be wrapped in <cfoutput> tags.



Inactive Page: This page is currently inactive and is inaccessible to non-contributors.

Activate when published

Implementation of simple template

Defining the CommonSpot Menu Bar

The CommonSpot menu bar is invoked through the <cfinclude> tag described below. It will automatically appear on all pages derived from the base template and can only be hidden from view by configuring options listed under the General Security heading in the CommonSpot Administrator. While you can place this <cfinclude> tag anywhere within your base template, it is typically deployed as the first line of code in the file.

```
<CFINCLUDE TEMPLATE="/commonspot/pagemode/pagemodeui.cfm">
```

Defining Editable Areas

Editable areas in a template are defined by placing a <cfmodule> tag call to ct-render-named-element.cfm in the appropriate location on your page. The general syntax for defining an editable region is the following:

```
<CFMODULE
TEMPLATE="/commonspot/utilities/ct-render-named-element.cfm"
ElementType="container"
elementName="myeditablecontent">
```

The elementName attribute can be any arbitrary label that you wish to supply, as long as it is a unique name on the template. In order words, two or more invocations to ct-render-named-element should not have the same element name.

The ElementType dictates the type of content that you can place in the editable region.

Container Elements

Container elements allow contributors to add multiple CommonSpot elements to a page via the CommonSpot element gallery. Container elements are typically reserved for use on base templates.

Type	**Description**
Container	Provides for the arrangement of elements using either CSS or element layout properties. In addition, the Container element significantly improves the scheduling of container elements
Layout	A rectangular table with a specified number of rows and columns. Individual cells within the table may contain other elements. Note that the use of this element has largely been deprecated in favor of the Container element.
Schedule	Different elements based on date/time, audience, subsite, or page category. Customers need to purchase the schedule/personalize feature as part of their license for this option to function properly.
MultiSection	A single or multiple repeating sections. Sections include a Title, description and separator description. Sub-elements may be embedded within the defined sub-sections

Navigation Elements

Navigation elements are used to automatically generate different types of hyperlinks that allow a user to easily traverse your web site.

Note that since the contents of navigation elements do not typically change often, you may consider deploying custom ColdFusion code in your base template in order to reduce the number of dynamic elements per page.

Type	**Description**
Breadcrumb	A horizontal list of breadcrumb links displaying the subsite hierarchy of the current page
Linkbar	A list of text based links. This type of element is often used as a navigational menu at the bottom of a page
FacetNav	Renders facet-based navigation based on CommonSpot's taxonomy subsystem
PageIndex	An index of pages based on specified filter parameters. Note that page indexes can be processor intensive and should be used with care.
Link	A simple link
LinkList	Displays either a drop down list of links or an expanded list of links

Text Elements

Text elements enable contributors to easily add content to their web pages. Note that in all of the modules listed below, an identical effect can be achieved by using the general purpose Textblock WYSIWYG control alone.

Type	**Description**
Textblock	A text block with rich formatting options, including CSS style selection, content structure, inserting images, and hyperlinking
Textblock-nohdr	The same as Textblock, but with the header text disabled by default
Date	Typically used to output the date that the page was last modified
TextaroundImage	Text flowed around a specified image. Several image positioning options are provided
WordConvert	An uploaded Word document as HTML, requires MS Word installed on the server.
Tb-simple	A simple text block, no wysiwyg options
Tb-simple-nohdr	A simple text block without a header
Bl-tb	A single column of bulleted text blocks. The lists can be categorized and titled with an appropriate header
BulletedList	A bulleted list of hyperlinks or text blocks. The lists can be categorized and titled with an appropriate header
HTML	Uploaded HTML within a textblock
Custom:name	Renders a custom element that allows a contributor to enter structured content via a tabbed-based form that is defined by an administrator

Multimedia Elements

Multimedia elements allow contributors to upload rich media including file-based video, Powerpoint presentations, PDF files, and Flash. These elements generally require uses to have the appropriate plug-ins installed in order to view the uploaded files.

Type	**Description**
Image	A single image and associated caption text. Rendering options support definition of width, height, fly-over image and status bar text, and associated hyperlink. This function can also be accomplished by using a textblock.
Image-nohdr	An Image with the header text hidden by default
ImageGrid	A vertical or horizontal series of images. You cannot define a custom render handler for an ImageGrid at this time.
Video	A Windows Media Video (WMV) clip. Requires the user have WMV compatible player such as Windows Media Player.
Quicktime	A quicktime movie. Users must have either iTunes or an appropriate player installed on their machines to view this content.
PDF	An embedded PDF file. End-users must have the appropriate version of Adobe Reader installed on their machines to view the uploaded PDF file.
Powerpoint	If Microsoft Powerpoint is installed on the server, this element will render a PPT file as a series of jpg images along with navigational controls. Users have the ability to download the original PPT file to their machines
Flash	A SWF file. Users must have the appropriate version of the Flash player installed on their local machines.

Search Elements

Search elements interface with the Verity search engine that ships with ColdFusion. Depending on your CF license, you may be restricted to indexing a maximum of either 125K or 250K pages/files. Due to the inherent CPU utilization involved with creating and maintaining Verity collections we advise customers with more than 1000 pages of content to integrate a Google Search Appliance instead.

Type	Description
SearchForm	The specified components of a search form
SearchResults	The results of a full-text search

Page Set Elements

Page Sets enable you to bind closely related pages into a single object whose navigational controls can be rendered using the following elements:

Type	Description
pstoc	The Table of Contents for a page set
psnav	Navigational links for a page set
psindex	An index of page sets

Simple Form and Datasheet Elements

The Simple Form element allows a non-technical user to define a data entry form on a page whose input is both saved to CommonSpot's database as well as e-mailed to the user accounts of their choosing. The data is extractable through the Datasheet element.

Type	Description
Form	A simple form
Datasheet	A datasheet that presents form data as well as data from any relational database where a JDBC driver has been defined.

Miscellaneous Elements

Out of the following elements, the only one of real import is the CustomCF element which allows the contributor to select and execute a CF file created by a developer.

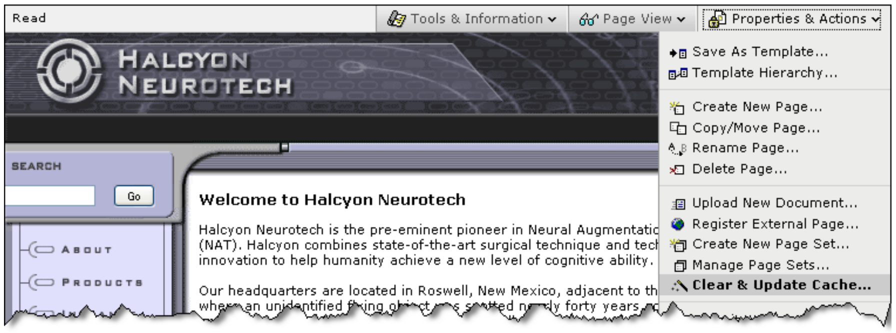
Type	**Description**
CustomCF	The output of a custom ColdFusion module
Webprocess	the output of a Web process or page
Tabbar	A horizontal row of tabs
Label	A label or heading which is typically used to denote a new section within the page
Task	A Task List

Caching Generated Output

By default the output from CommonSpot pages and their associated elements are cached on disk in the /commonspot-data/ directory (see unit 2). Caching significantly reduces the amount of processing time and database interaction required to execute each request. The disk cache is only updated under the following circumstances:

- A change to an element has been made by a contributor
- A contributor performed a Clear & Update Cache operation on a page
- An Administrator cleared the cache from the CommonSpot Administrator

Fundamentally this means that any time you make a change to your base template logic you will need to perform a clear cache operation to see the results on derived pages.



Welcome to Halcyon Neurotech

Halcyon Neurotech is the pre-eminent pioneer in Neural Augmentatic (NAT). Halcyon combines state-of-the-art surgical technique and tech innovation to help humanity achieve a new level of cognitive ability.

Our headquarters are located in Roswell, New Mexico, adjacent to th

Clear & Update Cache on a page

In addition there may be instances where you have dynamic content executing in a customCF script element that you want cached for a specific period of time. For instance, you might want to delay the retrieval of an RSS Feed by 15 minutes. In this circumstance, you can use the KeepCacheTimeSpan attribute of the ct-render-named-element tag as indicated below:

```
<CFMODULE
 template="/commonspot/utilities/ct-render-named-element.cfm"
 elementName="TopNewsStories"
 elementType="customcf"
 keepCacheTimeSpan="#createTimeSpan(0,0,15,0)#">
```

If an identical script were running on each page derived from the template then you also have the option to have all instances of that element pull results from the same location in memory by stipulating a memorycachename attribute as indicated below:

```
<CFMODULE
 template="/commonspot/utilities/ct-render-named-element.cfm"
 elementName="TopNewsStories"
 elementType="customcf"
 keepCacheTimeSpan="#createTimeSpan(0,0,15,0)#"
 Memorycachename="TopNewsStoriesCache">
```

Since the KeepCacheTimeSpan and the MemoryCacheName attributes cause CommonSpot to persist generated content in memory you should carefully consider the potential impact to your JVM memory heap before implementing this feature.

Walkthrough 3-2: Creating a Base Template

In this walkthrough you will create a base template from which all pages in the Halcyon Neurotech web site will be defined.

Steps

1. In Dreamweaver, create a new, empty ColdFusion file
2. Save the file into your /dev/templates/ directory as template-halcyon.cfm
3. At the top of the file, insert the following code to deploy the CommonSpot menu bar:

```
<cfinclude template="/commonspot/pagemode/pagemodeui.cfm">
```

4. After the code from step 3, insert a call to ct-render-named-element to deploy a breadcrumb navigation element as indicated below:

```
<CFMODULE
TEMPLATE="/commonspot/utilities/ct-render-named-element.cfm"
ElementType="breadcrumb"
elementName="breadcrumbnav">
```

5. After the code from Step 4, insert another call to ct-render-named-element that deploys a container object. Use an elementName of `Content`.
6. Save the file

Register the Template

7. Open your CommonSpot Administrator by navigating to http://[serverip]/dev/admin.cfm
8. Log-in as **admin-commonspot**
9. Click on **Base Templates**
10. Click on the link labeled **Click here to add a Base Template**
11. Click on the New button to add a new template category named **Halcyon Specific**

12. Fill out the form as indicated below:

Template Category:	Halcyon Specific
Template Name:	Halcyon Base Template
Description:	Base template for Halcyon Neurotech
Template Name:	template-halcyon.cfm

13. Click OK

Create a page from the base template

14. Redirect your browser to http://[server ip]/dev/createpage.cfm
15. Click Create New Page
16. Click on Halcyon Specific and choose the Halcyon Base Template
17. Enter the following information into the Create New Page dialog box:

Name:	Basic Template
Title:	Basic Template
Title Bar Caption:	Basic Template
Description:	Base Template for Halcyon Site

18. Click Next

Save the page as a template layer

19. From the CommonSpot menu bar, select **Properties & Actions > Save as Template**
20. Click the **Next** button
21. In the Save Current Page as a Template dialog box, enter the following information:

Category:	**Halcyon Specific**
Template Name:	**Halcyon Base**
Description:	**Basic Halcyon Template with Content Container**

22. Click **Finish**
23. Press **OK** to view the template
24. From the **Properties & Actions** menu, select **Submit Template for Public Use**

Hide the Base Template

25. Return to the CommonSpot Administrator by going to /dev/admin.cfm
26. Click on **Base Templates**
27. Click on the pencil icon adjacent to Halcyon Base Template
28. Turn on the checkbox to **Hide Template in template gallery**
29. Click **OK**

Create the Index Page

30. Using your web browser, return to /dev/createpage.cfm
31. Create a new page named index using the Halcyon Base template and place it in the root subsite (/). Note that you currently only see the the breadcrumb navigation element and a container object that are both undefined.

-- End of Walkthrough --

Implementing your Graphical Design

Now that you have implemented your base template, you must implement the graphics that will wrap your editable region. As a best practice we recommend abstracting your HTML into a ColdFusion CFML custom tag. This approach has the following advantages:

- Keeping the graphical layout characteristics of your site in a separate file will make it easier to modify in the event that a redesign is required.
- You can use the custom tag on non-commonspot pages to ensure a consistent look and feel across your applications
- Abstracting the HTML into a separate file helps simplify your base template logic, making it easier to maintain for other developers not as familiar with your implementation.

In the case of our implementation for Halcyon Neurotech, our overall goal is to deploy our design using the following syntax:

```
<cf_Halcyon>
<!--- Page mode ui (the three CommonSpot Indicators) --->
<CFINCLUDE TEMPLATE="/commonspot/pagemode/pagemodeui.cfm">

<!--- dynamic, editable area --->
<CFMODULE
TEMPLATE="/commonspot/utilities/ct-render-named-
element.cfm"
ElementType="container"
elementName="myeditablecontent">
</cf_Halcyon>
```

Creating CFML Custom Tags

Custom tags are pieces of code that can be called from many places in one or more applications. Like UDFs and CFCs, they let you encapsulate code. They differ from functions, however, in how they are defined and called as well as how they share data with the calling page.

Until the release of ColdFusion MX, custom tags were the primary architectural building block for most ColdFusion applications. CFCs did not yet exist and native ColdFusion tags could not be called from within UDFs. Now, however, CFCs can be used to encapsulate logic that was previously contained in custom tags.

A CFML custom tag is simply a ColdFusion template. What makes it different from any other ColdFusion page is the way it is stored and referenced from other templates—known as the *calling templates*. Data can be passed out of custom tags, though not as elegantly and cleanly as with CFCs.

Hence, the current recommendation for building your application is to:

- Use components to encapsulate all business logic.
- Use custom tags whenever you need to build reusable display elements—entities that display only the data passed in and do not pass any data back out.

Commenting Custom Tags

Including detailed comments in custom tag code facilitates reuse by other developers and applications.

Comments should include:

- Description of what the custom tag does.
- List of attributes, names, and descriptions, and whether they are required.
- Usage notes, including sample usage code.
- Author's name and e-mail address.
- Creation date.
- Chronological list of changes.

Here is a sample of comments to include at the top of each custom tag:

```
<!---
AUTHOR:
DATE:
PURPOSE:
ATTRIBUTES:
  Name="..." -required
USAGE NOTES:
   <cf_foo name="value">
MODIFICATION LOG:
DATE    AUTHOR   MODIFICATION
====    =======   ================================
```

--->

Passing attributes as name/value pairs

As you learned in the *Fast Track to ColdFusion MX* course, you call the custom tag foo.cfm, using the following notation:

```
<cf_foo>
```

To pass an attribute to this custom tag, you specify the name of the attribute and the associated value within the call:

```
<cf_foo attribute1="value1">
```

You pass multiple attributes as name/value pairs:

```
<cf_foo attribute1="value1" attribute2="value2" …>
```

Normal CFML coding rules apply when you create attributes:

- They are not case-sensitive.
- You can list them in any order within a tag.
- You must separate name/value pairs with a space.
- You must enclose the values in quotation marks (").
- You should use descriptive attribute names so they are easy to reference in calling pages.

In addition, note that:

- The attributes can be complex data types. For example, you can pass an array or a structure to a custom tag.
- Just as in the case of UDFs, structures and queries are passed by reference. All other data types are passed by value.

Referencing attributes

You refer to attributes passed into a custom tag using the attributes prefix.

For example, if attribute1 is passed into the custom tag, you reference it as follows:

```
<cfoutput>
The value of the first attribute is:
#attributes.attribute1#
</cfoutput>
```

Like all the other variable scopes, the attribute's scope is a structure and you can manipulate it using structure functions.

<cf_> syntax

The simplest way to call a custom tag is the method you have been using; you take the name of the file that contains the custom tag code (in this case, foo.cfm), remove the CFM extension, and add the prefix "cf_", as follows:

```
<cf_foo attribute1="value1" attribute2="value2" ...>
```

ColdFusion looks for the file in the following places in this order:

- The current directory of the calling page.
- The paths and subdirectories of those paths specified in the ColdFusion Administrator. The paths are searched in order of listing in the ColdFusion Administrator.

Nested processing

When you use the start and end tag notation, ColdFusion keeps track of whether it is the first time or second time through the custom tag. You can use this information to carry out different processing inside the custom tag depending on whether it is the first or second time through. This capability becomes useful when you have code between the custom tags:

```
<cf_foo>
   [other nested code...]
</cf_foo>
```

Now, instead of writing two related custom tags—one for the processing on each side of the nested code—you can use one custom tag that executes one set of instructions when it is initially executed and another set the second time through, after the nested code has executed. In addition, the variables inside the custom tag are not lost between calls; the same data is accessed both times in the custom tag.

Custom tag instance data

When any custom tag executes, ColdFusion keeps data related to the tag instance in the thisTag structure. You can access the thisTag structure from inside the custom tag. The built-in variables or keys of the thisTag structure are listed in the following table:

Variable	Description
ExecutionMode	Start, end, or inactive. Specifies the execution mode of the custom tag.
HasEndTag	True or false. Used for code validation to make sure the tag has an end tag if it is supposed to.
GeneratedContent	The HTML content that is generated by the tag.
AssocAttribs	A structure containing all of the attributes for any nested custom tags.

Processing modes

Although the start and end modes were mentioned above, you can actually consider three modes of processing:

- **Start mode** – When the custom tag is processed the first time. This is when it is called in the start tag, <cf_foo>.
- **Inactive mode** – When the nested code is processed, but no processing occurs in the custom tag itself.
- **End mode** – When the custom tag is processed a second time. This is when it is called in the end tag, </cf_foo>.

You can check to see which mode the custom tag is in using the variable thisTag.ExecutionMode:

```
<cfif thisTag.ExecutionMode is "start">
   [some code]
<cfelse>
   [some other code]
<cfif>
```

Modifying the <head> section of a document

When implementing an HTML design, consider the following:

- Code that needs to be placed in the <head> section of a document needs to be placed in a [template name].head file within the /templates/ directory
- A variable named headstruct is passed into the .head file. You can modify headstruct to dynamically change facets of the <head> section
- You can modify the <doctype> tag by editing the variable ServerInfo.DocTypeString in the /commonspot/keys/servervars.cfm file.

Placing content in the <head> section with .HEAD files

In order to place code in the generated <head> section of a document, you must create a file in the /templates/ directory with the same name as your base template, but with a .head file extension, i.e. template-halcyon.head.

In order to execute JavaScript code as part of a <body onLoad> event, you must create a variable in your .HEAD file named request.onLoadScript. This variable should contain the JavaScript that you want to execute as part of a <body> onLoad event handler.

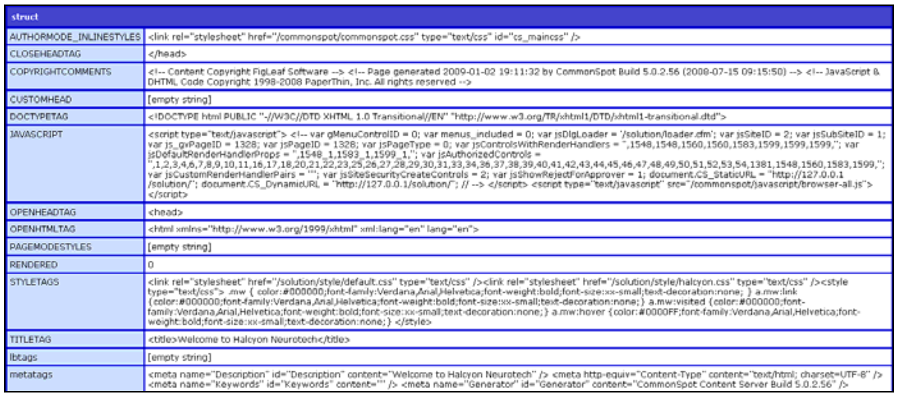
Do not forget to wrap the contents of a .head file with <cfoutput> tags if you want the code to render at runtime.

A typical .head file might resemble the following:

```
<!--- link-in custom JavaScript -→
<cfoutput>
<script language="Javascript"
src="#application.basehref#scripts/ui.js"></script>
</cfoutput>
<cfsavecontent variable="request.onLoadScript">
   <cfoutput>
   MM_preloadImages();
   </cfoutput>
</cfsavecontent>
```

Working with headstruct

The headstruct variable enables you to modify the <html>, <head>, <title>, metadata, and <style> tags for a generated CommonSpot page. The screenshot below is a <cfdump> of the headstruct variable from inside the .head file:



struct	
AUTHORMODE_INLINESTYLES	<link rel="stylesheet" href="/commonspot/commonspot.css" type="text/css" id="cs_maincss" />
CLOSEHEADTAG	</head>
COPYRIGHTCOMMENTS	<!-- Content Copyright FigLeaf Software --> <!-- Page generated 2009-01-02 19:11:32 by CommonSpot Build 5.0.2.56 (2008-07-15 09:15:50) --> <!-- JavaScript & DHTML Code Copyright 1998-2008 PaperThin, Inc. All rights reserved -->
CUSTOMHEAD	[empty string]
DOCTYPETAG	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
JAVASCRIPT	<script type="text/javascript"> <!-- var gMenuControlID = 0; var menus_included = 0; var jsDlgLoader = '/solution/loader.cfm'; var jsSiteID = 2; var jsSubSiteID = 1; var js_gvPageID = 1328; var jsPageID = 1328; var jsPageType = 0; var jsControlsWithRenderHandlers = ",1548,1548,1560,1560,1583,1599,1599,1599,"; var jsDefaultRenderHandlerProps = ",1548_1,1583_1,1599_1,"; var jsAuthorizedControls = ",1,2,3,4,6,7,8,9,10,11,16,17,18,20,21,22,23,25,26,27,28,29,30,31,33,34,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,1381,1548,1560,1583,1599,"; var jsCustomRenderHandlerPairs = ""; var jsSiteSecurityCreateControls = 2; var jsShowRejectForApprover = 1; document.CS_StaticURL = "http://127.0.0.1/solution/"; document.CS_DynamicURL = "http://127.0.0.1/solution/"; // --> </script> <script type="text/javascript" src="/commonspot/javascript/browser-all.js"></script>
OPENHEADTAG	<head>
OPENHTMLTAG	<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
PAGEMODESTYLES	[empty string]
RENDERED	0
STYLETAGS	<link rel="stylesheet" href="/solution/style/default.css" type="text/css" /><link rel="stylesheet" href="/solution/style/halcyon.css" type="text/css" /><style type="text/css"> .mw { color:#000000;font-family:Verdana,Arial,Helvetica;font-weight:bold;font-size:xx-small;text-decoration:none; } a.mw:link {color:#000000;font-family:Verdana,Arial,Helvetica;font-weight:bold;font-size:xx-small;text-decoration:none;} a.mw:visited {color:#000000;font-family:Verdana,Arial,Helvetica;font-weight:bold;font-size:xx-small;text-decoration:none;} a.mw:hover {color:#0000FF;font-family:Verdana,Arial,Helvetica;font-weight:bold;font-size:xx-small;text-decoration:none;} </style>
TITLETAG	<title>Welcome to Halcyon Neurotech</title>
lbtags	[empty string]
metatags	<meta name="Description" id="Description" content="Welcome to Halcyon Neurotech" /> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" /> <meta name="Keywords" id="Keywords" content="" /> <meta name="Generator" id="Generator" content="CommonSpot Content Server Build 5.0.2.56" />

The contents of headstruct

Additionally, you can completely customize the rendering of the <head> section by setting the variable headstruct.rendered equal to 1. You will, of course, be required to supply your own custom output as part of the .head file to replace CommonSpot's generated head section to maintain the validity of the page.

Using <cfhtmlhead>

The ColdFusion tag <cfhtmlhead> can be used to output content dynamically into the <head> section of a document. <cfhtmlhead> is particularly useful when developing scripts containing external JavaScript library references that contributors can deploy using the CommonSpot customcf element. For example, the following code located in a customcf module references an Adobe Spry javascript. This code only needs to be included if the customcf module is deployed:

```
<cfsavecontent variable="SpryIncludes">
<cfoutput>
   <script
   src="/SpryAssets/SpryValidationTextField.js"
   type="text/javascript"></script>
</cfoutput>
</cfsavecontent>
<cfhtmlhead text="#SpryIncludes#">
```

Walkthrough 3-3: Implementing a Design

In this walkthrough you will convert an html page into a CFML custom tag and then deploy it into a CommonSpot template.

Steps

1. In Dreamweaver, open the file /dev/ui/index.htm
2. Save the file as /dev/templates/halcyon.cfm
3. Remove the HTML, HEAD, TITLE, and META tags as CommonSpot will generate those for you at runtime.
4. Use CFML comment tags to comment out the <body> tag. Delete the </body> tag
5. Wrap the page with a <cfif>…</cfif> block,
   Set the condition to <cfif thistag.executionmode is "Start">
6. Replace the string Content goes here with a <cfelse> tag
7. Wrap <cfoutput> tags around the interior of the <cfif> and <cfelse> blocks
8. Press Ctrl+F and perform the following search and replace operations:

   a) Replace # with ##
   b) Replace inter_images/ with #variables.imgpath#

9. At the top of the file, insert the following code:

```
<cfparam name="attributes.url" default="/">
<cfif attributes.url contains "/intranet/">
  <cfset variables.imgpath =
                 application.uipath & "intra_images/">
<cfelse>
  <cfset variables.imgpath =
                 application.uipath & "inter_images/">
</cfif>
```

10. Save the file and browse it. You should see the first part of the page window dressing.

Create a .head file

11. Using Dreamweaver, create a new, empty ColdFusion page and save it to /templates/template-halcyon.head
12. Inside the template-halcyon.head file, insert the following code to invoke the loading of a JavaScript:

```
<cfoutput>
<script language="JavaScript"
src="#application.basehref#ui/scripts/halcyon.js">
</script>
</cfoutput>
```

13. Using Dreamweaver, return to the /templates/halcyon.cfm custom tag
14. Cut the contents of the JavaScript block and paste them into the file /ui/scripts/halcyon.js.
15. Replace the ## in the MM_preloadImages() function with a #
16. Save the file
17. Return to /templates/halcyon.cfm
18. Copy the MM_preloadImages() function call from the <body> onLoad handler and paste it into the template-halcyon.head file using the following syntax:

```
<cfsavecontent variable="request.onLoadScript">
 <cfoutput>
 MM_preloadImages ( … );
 </cfoutput>
</cfsavecontent>
```

19. Save the file
20. Return to /templates/halcyon.cfm
21. Delete the <script>…</script> tags
22. Save the file

Deploy the Custom Tag to your Template

23. Using Dreamweaver, return to /dev/templates/template-halcyon.cfm
24. Wrap the contents of the file with the following syntax:

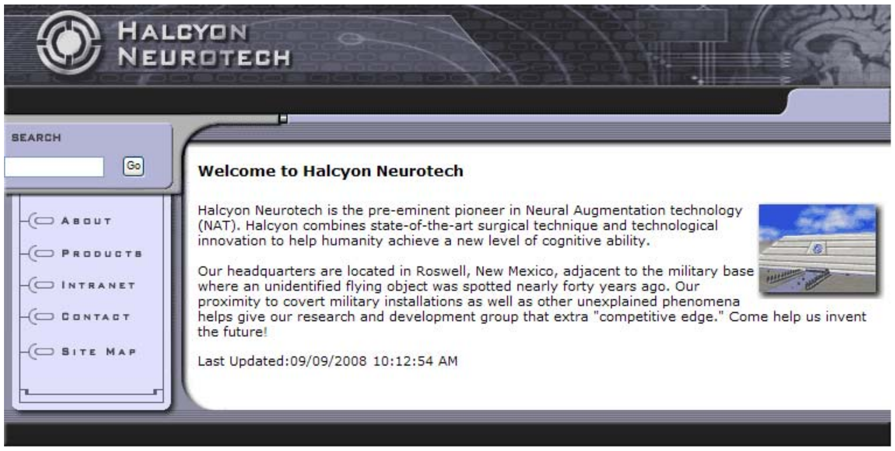
```
<cf_halcyon url="#request.subsite.url#">
   <!--- program code --->
</cf_halcyon>
```

25. Save the file
26. Using your browser, return to /dev/index.cfm. Note that you may not yet see the results of implementing the custom tag UI.

27. Select **Properties & Actions > Clear and Update Cache**
28. You should now see the Halcyon graphics wrapping your page.

Add content to the page

29. Using CommonSpot, add a Formatted Text Block and Date element to the page.
30. Copy and paste the text from {coursefiles}/Other Documents/Home Page Text.txt into the Formatted Text Block
31. Insert the image {coursefiles}/Other Documents/Halcyon_hq.gif into the Formatted Text Block
32. Configure the date element to display the Date of Last Revision.
33. Your final result should resemble the following screenshot:



Welcome to Halcyon Neurotech

Halcyon Neurotech is the pre-eminent pioneer in Neural Augmentation technology (NAT). Halcyon combines state-of-the-art surgical technique and technological innovation to help humanity achieve a new level of cognitive ability.

Our headquarters are located in Roswell, New Mexico, adjacent to the military base where an unidentified flying object was spotted nearly forty years ago. Our proximity to covert military installations as well as other unexplained phenomena helps give our research and development group that extra "competitive edge." Come help us invent the future!

Last Updated:09/09/2008 10:12:54 AM

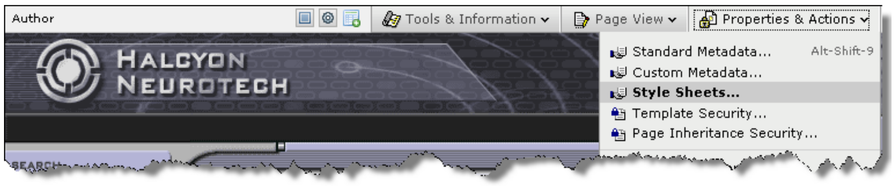
-- End of Walkthrough --

Implementing CSS-based designs

CommonSpot 5.x fully supports the deployment of 100% CSS-based layouts. Using CSS instead of tables to lay out your page is one of the basic tenants of accessibility and efficient design. Note that there are some variations between how different browsers on different operating systems support CSS-based positioning. Be sure to add time to your test plan to test across all of your target browsers prior to launch.

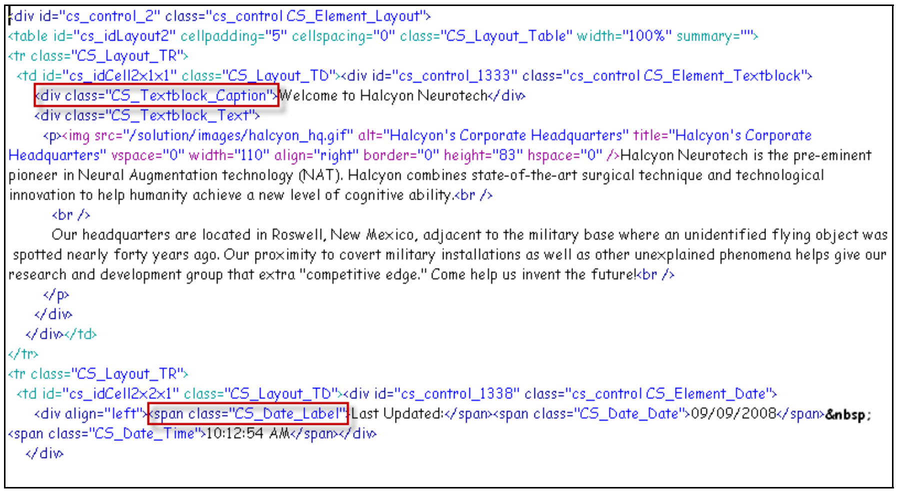
Linking to Style Sheets

Style sheets (.CSS files) should be placed in your site's /style/ directory and is typically linked-in via the CommonSpot UI from your base template.



Style Sheet support is activated from the Properties & Actions menu

Once stylesheets are active for a page, CommonSpot will wrap all of its elements on the page with a series of <div> and <span> tags that reference CSS classes.



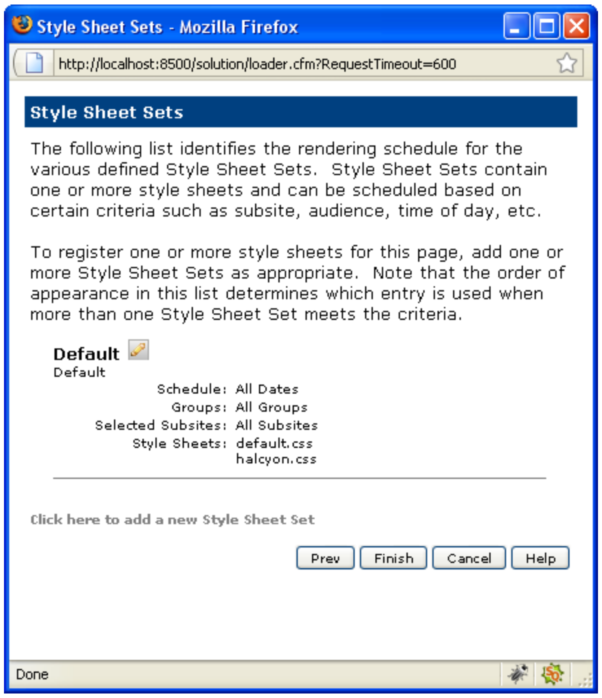
```
<div id="cs_control_2" class="cs_control CS_Element_Layout">
<table id="cs_idLayout2" cellpadding="5" cellspacing="0" class="CS_Layout_Table" width="100%" summary="">
<tr class="CS_Layout_TR">
  <td id="cs_idCell2x1x1" class="CS_Layout_TD"><div id="cs_control_1333" class="cs_control CS_Element_Textblock">
     <div class="CS_Textblock_Caption">Welcome to Halcyon Neurotech</div>
     <div class="CS_Textblock_Text">
       <p><img src="/solution/images/halcyon_hq.gif" alt="Halcyon's Corporate Headquarters" title="Halcyon's Corporate
Headquarters" vspace="0" width="110" align="right" border="0" height="83" hspace="0" />Halcyon Neurotech is the pre-eminent
pioneer in Neural Augmentation technology (NAT). Halcyon combines state-of-the-art surgical technique and technological
innovation to help humanity achieve a new level of cognitive ability.<br />
         <br />
         Our headquarters are located in Roswell, New Mexico, adjacent to the military base where an unidentified flying object was
 spotted nearly forty years ago. Our proximity to covert military installations as well as other unexplained phenomena helps give our
research and development group that extra "competitive edge." Come help us invent the future!<br />
       </p>
     </div>
   </div></td>
</tr>
<tr class="CS_Layout_TR">
  <td id="cs_idCell2x2x1" class="CS_Layout_TD"><div id="cs_control_1338" class="cs_control CS_Element_Date">
     <div align="left"><span class="CS_Date_Label">Last Updated:</span><span class="CS_Date_Date">09/09/2008</span>&nbsp;
<span class="CS_Date_Time">10:12:54 AM</span></div>
   </div>
```

CommonSpot outputs its elements with <div> and <span> tags referencing specifically named style classes

Style Sheet Sets

Multiple Style Sheets are grouped into Style Sheet Sets. These sets can, in turn, be scheduled to be invoked based on the following criteria:

- Time of Day
- User Group Membership
- Page Category
- Page Location (Subsite)
- Browser Type
- Variable Comparison


Style Sheet Sets - Mozilla Firefox
http://localhost:8500/solution/loader.cfm?RequestTimeout=600

Style Sheet Sets

The following list identifies the rendering schedule for the various defined Style Sheet Sets. Style Sheet Sets contain one or more style sheets and can be scheduled based on certain criteria such as subsite, audience, time of day, etc.

To register one or more style sheets for this page, add one or more Style Sheet Sets as appropriate. Note that the order of appearance in this list determines which entry is used when more than one Style Sheet Set meets the criteria.

Default
Default

Schedule:	All Dates
Groups:	All Groups
Selected Subsites:	All Subsites
Style Sheets:	default.css halcyon.css

Click here to add a new Style Sheet Set

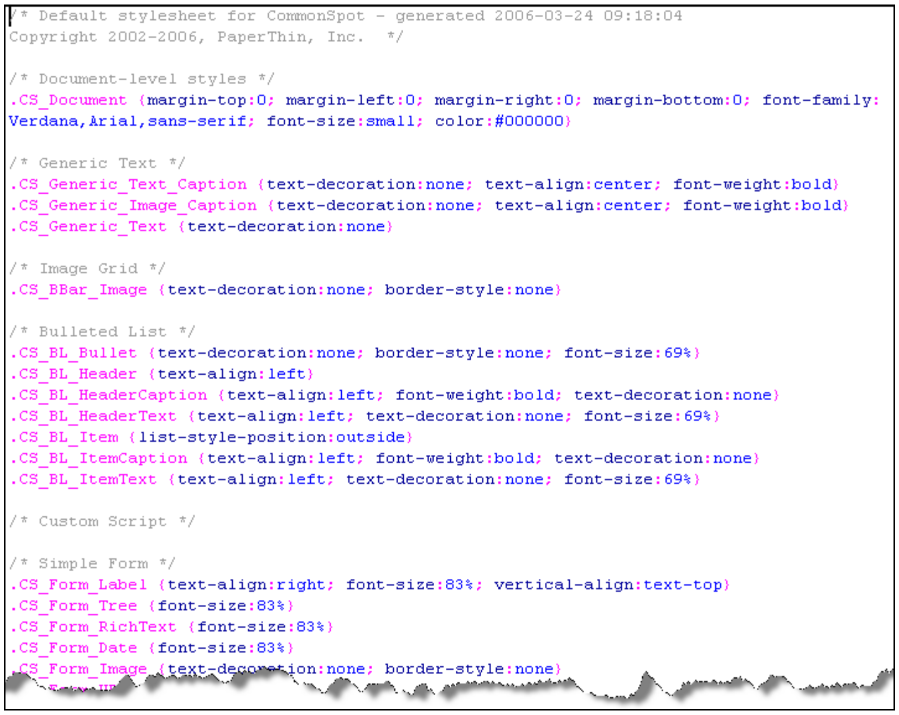
Style Sheet Sets can be scheduled based on a number of criteria

Default.css

The typographic definitions for all CommonSpot element classes are located in the default.css file that CommonSpot automatically places in the /style/ directory at the time that you created your site.

Body tag style attributes such as leftmargin and topmargin attributes should be handled at the style sheet level by modifying or defining the .CS_Document style class.

You should modify this CSS file to format the default output of CommonSpot elements. Note, however, that Paperthin may add to the style listing in the future as they add additional product features. This may require you to run a diff operation after upgrading to merge in any changes.



```
/* Default stylesheet for CommonSpot - generated 2006-03-24 09:18:04
Copyright 2002-2006, PaperThin, Inc.  */

/* Document-level styles */
.CS_Document {margin-top:0; margin-left:0; margin-right:0; margin-bottom:0; font-family:
Verdana,Arial,sans-serif; font-size:small; color:#000000}

/* Generic Text */
.CS_Generic_Text_Caption {text-decoration:none; text-align:center; font-weight:bold}
.CS_Generic_Image_Caption {text-decoration:none; text-align:center; font-weight:bold}
.CS_Generic_Text {text-decoration:none}

/* Image Grid */
.CS_BBar_Image {text-decoration:none; border-style:none}

/* Bulleted List */
.CS_BL_Bullet {text-decoration:none; border-style:none; font-size:69%}
.CS_BL_Header {text-align:left}
.CS_BL_HeaderCaption {text-align:left; font-weight:bold; text-decoration:none}
.CS_BL_HeaderText {text-align:left; text-decoration:none; font-size:69%}
.CS_BL_Item {list-style-position:outside}
.CS_BL_ItemCaption {text-align:left; font-weight:bold; text-decoration:none}
.CS_BL_ItemText {text-align:left; text-decoration:none; font-size:69%}

/* Custom Script */

/* Simple Form */
.CS_Form_Label {text-align:right; font-size:83%; vertical-align:text-top}
.CS_Form_Tree {font-size:83%}
.CS_Form_RichText {font-size:83%}
.CS_Form_Date {font-size:83%}
.CS_Form_Image {text-decoration:none; border-style:none}
```

Excerpt from /Style/Default.css. Note the .CS_Document class

Walkthrough 3-4: Implementing CSS

In this walkthrough you will link to the existing default.css file and modify it to format the default behavior of two CommonSpot elements.

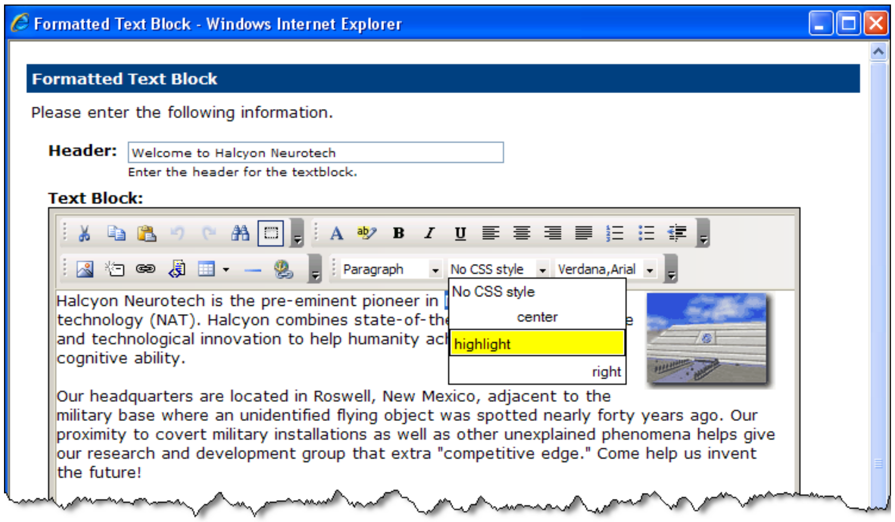
Steps

1. Return to /dev/index.cfm in your web browser
2. Select **Properties & Actions > Template Hierarchy**
3. Click on **Halcyon Base**. You will be redirected to your base template.
4. Select **Page View > Author**
5. Select **Properties & Actions > Style Sheets**
6. Click on **Use Style Sheet and prevent element style override**
7. Click Finish.
8. Return to /dev/index.cfm. You should notice that your fonts have changed.
9. In Dreamweaver, open /dev/style/default.css
10. Modify the **.CS_Document** class with the following directives:
font-family: Arial;
font-size: 1 ems;
11. Modify the **.CS_Textblock_Caption** class with the following directives:
font-size: 1.1 ems;
text-decoration: underline;
12. Modify the **.CS_Textblock_Text** class with the following directives:
font-size: 0.9 ems;
13. Modify all of the classes that start with **.CS_Date**, changing their font-sizes to 0.8 ems
14. Save the file
15. Re-browse /dev/index.cfm. You should see that the format of your content has changed again.

-- End of Walkthrough –

Making Styles Available to Contributors

Despite the fact that you have enabled Style Sheets on your web site, you still need to register specific style classes for users to select during the content creation process. In most cases this will simply require you to register a few styles for users to select from in the WYSIWYG editor.


Formatted Text Block - Windows Internet Explorer

Formatted Text Block

Please enter the following information.

Header: Welcome to Halcyon Neurotech
Enter the header for the textblock.

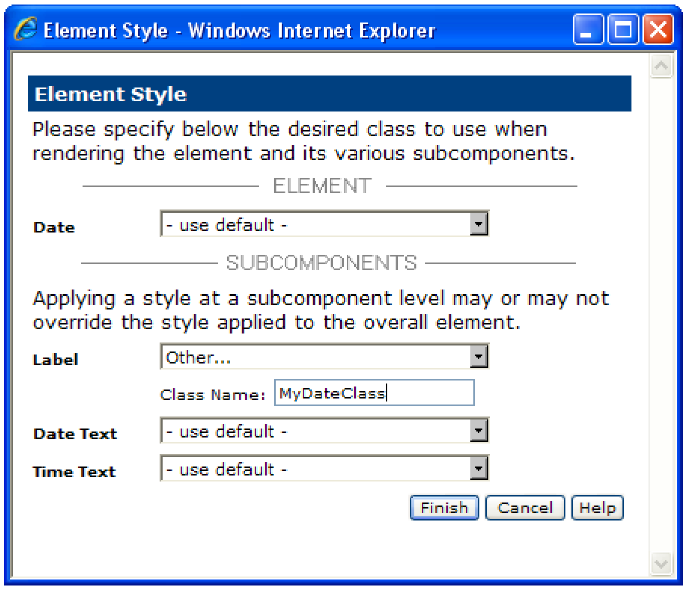
Text Block:

Halcyon Neurotech is the pre-eminent pioneer in
technology (NAT). Halcyon combines state-of-the
and technological innovation to help humanity ach
cognitive ability.

Our headquarters are located in Roswell, New Mexico, adjacent to the military base where an unidentified flying object was spotted nearly forty years ago. Our proximity to covert military installations as well as other unexplained phenomena helps give our research and development group that extra "competitive edge." Come help us invent the future!

Styles Must Be Registered for Use in Specific Contexts

More complex implementations may allow contributors to choose from a broad range of styles and apply them to different subcomponents of CommonSpot elements.


Element Style - Windows Internet Explorer

Element Style

Please specify below the desired class to use when rendering the element and its various subcomponents.

ELEMENT

Date - use default -

SUBCOMPONENTS

Applying a style at a subcomponent level may or may not override the style applied to the overall element.

Label Other...

Class Name: MyDateClass

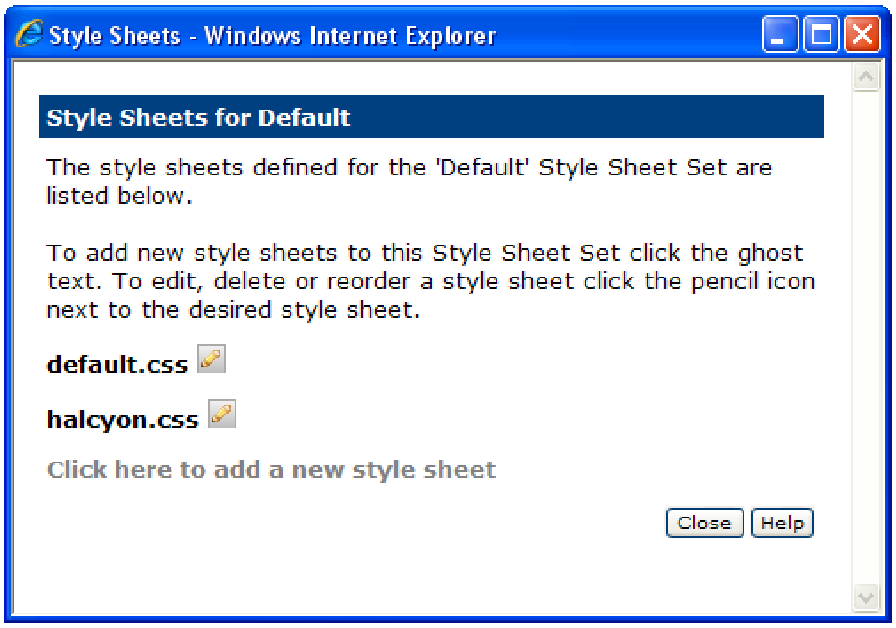
Date Text - use default -

Time Text - use default -

Style classes can be applied to components of an element

Creating Custom Style Classes

While you can technically define additional style classes in the /style/default.css file, the best practice is to instead create a second CSS file in the /style/ directory and register it with the Base Template.



Style Sheets for Default

The style sheets defined for the 'Default' Style Sheet Set are listed below.

To add new style sheets to this Style Sheet Set click the ghost text. To edit, delete or reorder a style sheet click the pencil icon next to the desired style sheet.

default.css

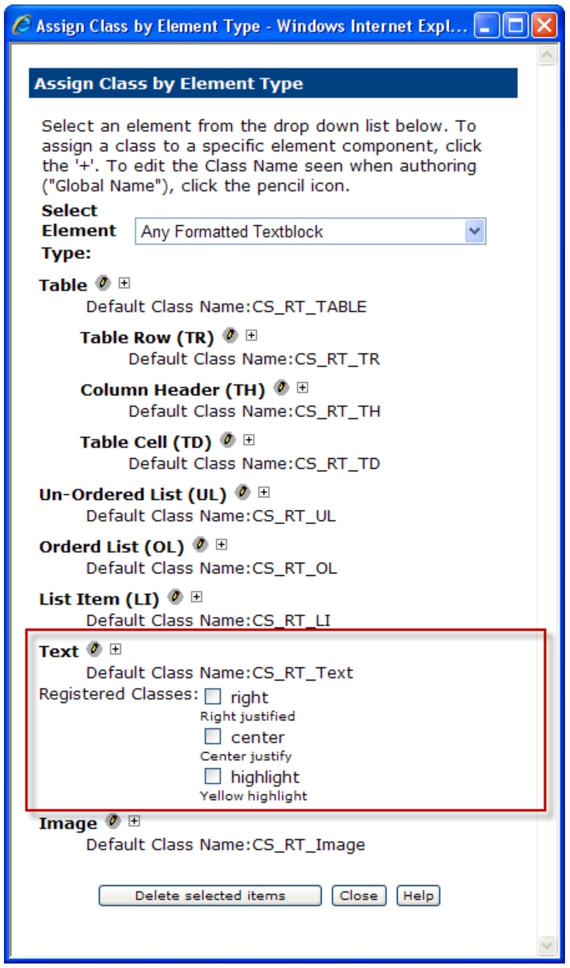
halcyon.css

Click here to add a new style sheet

Multiple Style Sheets can be bound to a Style Sheet Set

Registering Style Classes with CommonSpot

In order for style classes to be deployable through the CommonSpot GUI they must first be registered within the CommonSpot Administrator. During this process you first register the style class name and then indicate the locations of where its use is allowed. While this process may seem tedious it is necessary in order to prevent contributors from assigning styles to elements for which they were not intended. Also, consider that you will only need to register styles if there are two or more styles that can apply to a specific CommonSpot element. Otherwise, simply modifying the default.css file as previously mentioned should meet your design requirements.



Assign Class by Element Type

Select an element from the drop down list below. To assign a class to a specific element component, click the '+'. To edit the Class Name seen when authoring ("Global Name"), click the pencil icon.

Select Element Type: Any Formatted Textblock

Table
Default Class Name:CS_RT_TABLE

Table Row (TR)
Default Class Name:CS_RT_TR

Column Header (TH)
Default Class Name:CS_RT_TH

Table Cell (TD)
Default Class Name:CS_RT_TD

Un-Ordered List (UL)
Default Class Name:CS_RT_UL

Orderd List (OL)
Default Class Name:CS_RT_OL

List Item (LI)
Default Class Name:CS_RT_LI

Text
Default Class Name:CS_RT_Text
Registered Classes: ☐ right
Right justified
☐ center
Center justify
☐ highlight
Yellow highlight

Image
Default Class Name:CS_RT_Image

Typically you will need to register additional styles for use in the WYSIWYG editor

Walkthrough 3-5: Registering CSS Styles

In this walkthrough you will create and register styles for use by contributors within the CommonSpot WYSIWYG editor.

Steps

Inspect the Style Sheet

1. Using Dreamweaver, open {coursefiles}/ui/halcyon.css and review its contents with your instructor.

Register an additional style sheet

2. Using Windows Explorer, copy the file {coursefiles}/ui/halcyon.css to /dev/style/
3. In your web browser, navigate to /dev/index.cfm
4. Select **Properties & Actions > Template Hierarchy**
5. Click on **Halcyon Base**. You will be redirected to your base template.
6. Select **Page View > Author**
7. Select **Properties & Actions > Style Sheets**
8. Click **Next**
9. Click on the pencil icon next to Default and select **Style Sheets…**
10. Click on the **Click here to add a new style sheet** link
11. Click on **halcyon.css** and then click **Finish**
12. Click **Close**
13. Click **Finish**

Register the Styles with CommonSpot

14. In your web browser, navigate to /dev/admin.cfm
15. In the Subsites drop-down box, select **dev Site**
16. Click on the heading for **Style Definitions**
17. Click on **Manage and Register Classes**
18. Add the following three classes. Note that the names are case sensitive:

    center
    highlight
    right

19. Close the Manage Classes dialog box

20. Click on **Assign Class by Element Type**
21. Select **Any Formatted Textblock**
22. Click on the **[+]** adjacent to **Text**
23. Ctrl-Click on center, highlight, and right
24. Click **Finish**
25. Click **Close**
26. Click on the **Rich Text Editor Settings** Heading
27. Click on the **Toolbar** tab
28. In the column labeled **On Pages with Style Sheets**, set the following items to **Never Show**:

    Text Color
    Background Color
    Font Type List
    Font Size List
29. Click **OK**
30. Return to /dev/index.cfm in your web browser
31. Edit the formatted text block containing the welcome message for Halcyon Neurotech
32. Notice that the center, highlight, and right styles should now be selectable in the editor.

-- End of Walkthrough --

Implementing a Printer Friendly Format

Using CommonSpot, you have two methods from which you can define the printer friendly format for a page.

- Use the CSS MEDIA attribute of the <link> tag
- Use Base Template logic in conjunction with CommonSpot's request.renderforprint variable

Using CSS to implement a printer-friendly format

The Media attribute of the HTML <link> tag specifies how the document is to be presented on different media. For example, you can create a style sheet that defines the way your page looks when printed or displayed on handheld devices.

Your options for Media are:

- **All** (default) – suitable for all devices
- **Aural** – speech
- **Braille** – for Braille tactile feedback devices
- **Handheld** – PDA, small screen, limited bandwidth
- **Print** – for printed material
- **Projection** – for projector devices
- **Screen** – for computer screens
- **TTY** – teletype, terminals, or portable devices with limited display capabilities
- **TV** – television type devices

By default, CommonSpot will not stipulate any media type as part of its generated link tags, however, you can programmatically modify the variable headstruct.styletags from your .HEAD file in order to inject a MEDIA attribute into the generated <link> tags using the following code snippet:

```
<cfset headstruct.styletags =
     replace(headstruct.styletags,
               "<link ", "<link media=""Screen"" ","ALL")>
```

This now allows you to insert a reference to a printer-specific style sheet using code similar to the following:

```
<cfset headstruct.styletags = headstruct.styletags &
'<link media="print" type="text/css" rel="stylesheet"
href="/style/printerfriendly.css">'>
```

Using Base Template Logic

CommonSpot will automatically regenerate page results on the fly when it senses the existence of a url variable named renderforprint, i.e. index.cfm?renderforprint=1. During the page regeneration process CommonSpot transfers the url.renderforprint variable over to the request scope so you can place conditional code into your base template to include or exclude content based on the value of request.renderforprint as indicated in the example below:

```
<cfif request.renderforprint>

<!--- printer friendly – only output content -→
<CFMODULE
TEMPLATE="/commonspot/utilities/ct-render-named-element.cfm"
Elementtype="layout"
elementName="contentlayout">

<cfelse> <!--- screen mode -→

<CFINCLUDE TEMPLATE="/commonspot/pagemode/pagemodeui.cfm">

<cf_halcyon>

<!--- generate printer friendly link -→
<p align="right">
 <a href="#request.page.filename#?renderforprint=1">
    [Printer Friendly]
 </a>
</p>

<CFMODULE
TEMPLATE="/commonspot/utilities/ct-render-named-element.cfm"
Elementtype="breadcrumb"
elementName="breadcrumb">

<CFMODULE
TEMPLATE="/commonspot/utilities/ct-render-named-element.cfm"
Elementtype="layout"
elementName="contentlayout">

</cf_halcyon>

</cfif> <!--- renderforprint -→
```

Walkthrough 3-6: Implementing Printer Friendly Pages

In this walkthrough you will link to the existing default.css file and modify it to format the default behavior of two CommonSpot elements.

Steps

1. In Dreamweaver, open the file /dev/templates/template-halcyon.cfm
2. Within the <cf_halcyon> block, prior to the first <cfmodule> tag, create a **<CFOUTPUT>**…**</CFOUTPUT>** block.
3. Inside the <cfoutput> block, add a paragraph block with the ALIGN attribute set to RIGHT.
4. Add an anchor tag for the printer-friendly functionality that invokes a JavaScript function popupWindow and calls the current page while passing in a url variable named renderforprint. Set the width and height of the popup window to 1024 and 768, respectively. Your syntax should resemble the following:

```
<cfoutput>
<p align="right"><a
href="javascript:popupWindow('#request.page.filename#?rend
erforprint=1',1024,768)">Printer Friendly</a>
</cfoutput>
```

5. Wrap the contents of your page with a <cfif> block that checks if request.renderforprint is true
6. Place a <cfelse> tag directly after the <cfif> condition.
7. Between the <cfif> and the <cfelse>, copy and paste the <cfmodule> tag in the file that refers to the container element.
8. Save the file
9. Return to your browser and view /dev/index.cfm
10. Select **Properties & Actions > Clear and Update Cache**. The Printer Friendly link should now be visible.
11. Click on the **Printer Friendly** link. Your page will re-generate in a popup window with only the page's content being visible.

-- End of Walkthrough --

Unit Summary

- Your template strategy will ultimately be responsible for the success or failure of your web site
- Most web sites can be implemented using two or fewer base templates
- Keep the number of editable areas on a page to six or fewer
- The most basic Base Template contains a <cfinclude> to deploy the CommonSpot menu bar and a <cfmodule> tag call to ct-render-named-element that defines an editable region
- You can insert code into the <head> section of a page by creating a .HEAD file in the /templates/ directory
- Turn your page design into a CFML custom tag to reduce its complexity.
- CommonSpot pages are generated from the template hierarchy
- Template layers are just a visual abstraction for passing parameters to the ct-render-named-element tags
- CommonSpot supports the implementation of CSS positioned designs.
- Link stylesheets to your base template through the CommonSpot UI.
- You can implement printer-friendly output by either modifying the headstruct variable or by adding logic to your base template leveraging the request.renderforprint variable.

Unit Review

1. What is the purpose of a base template?

2. What operation must you perform on your pages in order for them to display the results from changes to a base template?

3. What is the name of the variable that gives you access to the content that CommonSpot will use in generating the <head> section of a page?

4. What style class must you change to influence the output of content between the <body> tags?

5. CommonSpot wraps all of its elements in <div> and <span> tags that reference specific style classes (true/false)

6. What are the advantages and disadvantages to encapsulating your design in a CFML custom tag?

Lab 3: Adding Dynamic Elements to a Template

In this lab, you will add a CommonSpot search form to your Halcyon Site and format its behavior through a template. You will also register and deploy a style for use with the CommonSpot date element.

Objectives

After completing this lab, you should be able to:

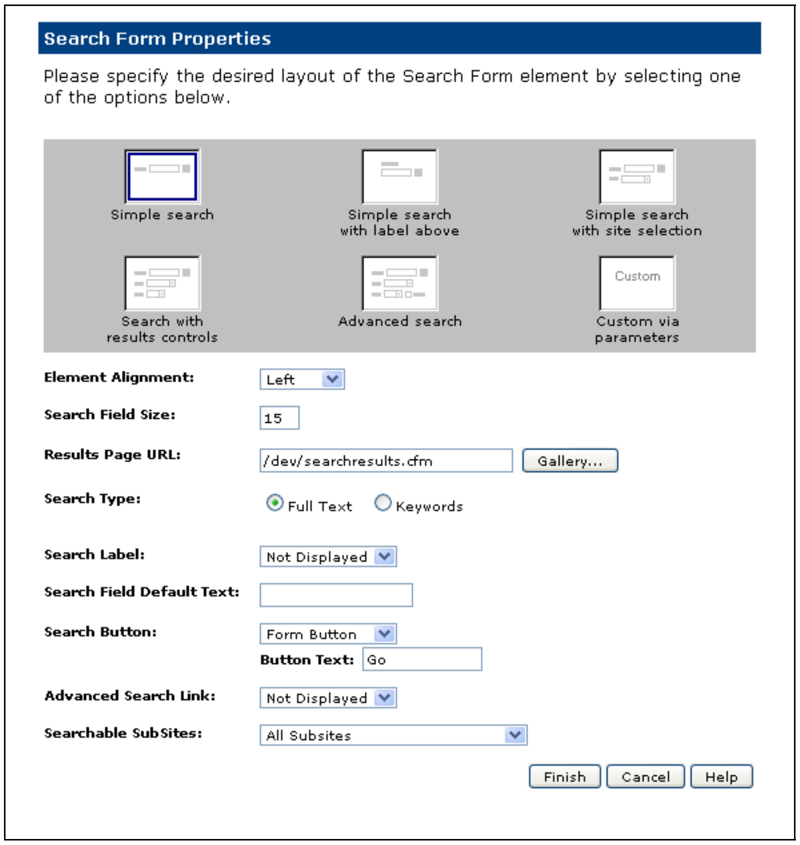
- Use ct-render-named-element to define an editable region
- Insert code to dynamically change the page banner based on where a page is located in the site
- Define and apply a style to an element at a template-level.

Steps

Define an editable region

1. Using Dreamweaver, open the file /dev/templates/halcyon.cfm.
2. Where indicated the the comment <!—start lab 3 →, delete the form field between the comment tags.
3. Between the comment tags, insert a call to ct-render-named-element that deploys a CommonSpot **searchform** element.
4. Place a </cfoutput> tag directly *before* the searchform element
5. Place a <cfoutput> tag directly *after* the searchform element.
6. Save the file
7. In your web browser, navigate to /dev/index.cfm
8. Perform a clear & update cache operation. The CommonSpot search form should now be visible
9. Select **Properties & Actions** > **Template Hierarchy**
10. Select the lowest level template available
11. Go to **Page View** > **Author Mode**

12. Click on the gear icon above the search form
13. Select Edit
14. Fill out the form as indicated below:


Simple search
Simple search with label above
Simple search with site selection
Search with results controls
Advanced search
Custom
Custom via parameters
Gallery...
Finish
Cancel
Help

Search Form Properties

Please specify the desired layout of the Search Form element by selecting one of the options below.

Element Alignment:	Left
Search Field Size:	15
Results Page URL:	/dev/searchresults.cfm
Search Type:	[✓ Full Text] Keywords
Search Label:	Not Displayed
Search Field Default Text:	
Search Button:	Form Button
Button Text:	Go
Advanced Search Link:	Not Displayed
Searchable SubSites:	All Subsites

15. Click Finish
16. Submit the page for publication
17. Return to /dev/index.cfm. You should see your changes have affected the home page.

Implement a dynamically rendered banner using base template logic

18. Return to Dreamweaver and open /dev/templates/halcyon.cfm
19. Locate the <img> tag that references the image banner.gif
20. Replace the <img> tag with the contents of {coursefiles}/other documents/banneroutput.cfm
21. Save halcyon.cfm
22. Using CommonSpot, create a new page named index.cfm in the /about/ directory using the Halcyon template. You should notice that the appropriate banner displays at the top of the generated page.

Register a style

23. In your web browser, go to /dev/admin.cfm
24. Click on the hyperlinked word **here**
25. Click on the heading for **Style Definitions**
26. Click on **Manage and Register Classes**
27. Click on **Click to Add a New Class**
28. Enter a name of **datetext**
29. Enter a description of **Formatted Date**
30. Click **Finish**
31. Click **Close**
32. Click on **Assign Class by Element Type**
33. Select **Date** from the drop-down list
34. Click on the **[+]** adjacent to Date
35. Select **datetext**
36. Click **Finish**
37. Click **Close**
38. Return to /dev/index.cfm
39. Go to Author mode
40. Click on the gear icon above the date element
41. Click on **Style**
42. Select **datetext** from the first drop-down list
43. Click **Finish**. Note that the formatting of the date has changed and that no submission for publication is required.

-- End of Lab --