Content Server

Version: 6.3

FirstSite II: Use, Reuse, and Replication

Document Revision Date: Dec 2, 2005

FatWire SOFTWARE

FirstSite II: Use, Reuse, and Replication
Document Revision Date: Dec 2, 2005
Product Version: 6.3

FatWire Technical Support
www.fatwire.com/Support

FatWire Headquarters
FatWire Corporation
330 Old Country Road
Suite 207
Mineola, NY 11501
www.fatwire.com

Table of
Contents

List of

Figures

About This Guide

This guide serves as an introduction to Content Server through the use and study of FirstSite. FirstSite is a Content Server starter site created by FatWire Engineers to demonstrate the best practices for developing new sites with Content Server. It includes several step-by-step exercises for managing content in Content Server and covers basic design principles used in developing FirstSite.

Who Should Use This Guide

This guide was written especially for administrators, developers, and content providers who are new to Content Server. For content providers, technical proficiency is not required. For developers and administrators, it is assumed that the reader has a basic understanding of Java, JSP, XHTML, and other internet technologies.

How This Guide Is Organized

This guide is divided into three parts. Part 1 (Chapters 3-5) introduces users to Content Server and its major features. Part 2 (Chapters 6 and 7) guides developers through the FirstSite framework and discusses how to reuse it. Part 3 (Chapter 8) instructs administrators on how to replicate FirstSite in order to use it as a foundation for new sites.

This guide contains the following chapters:

- Chapter 1, “Introduction” - provides a high level overview of the Content Server product family and its features, including a full list of common terms used throughout this guide.
- Chapter 2, “Content Server FirstSite” - includes an introduction to FirstSite and instructions for how to get started.
- Chapter 3, “Creating Content in FirstSite” - Introduces users to assets and includes exercises on how to create new assets in FirstSite.
- Chapter 4, “Workflow & Publishing” - provides an overview of Content Server’s workflow feature.
- Chapter 5, “Personalization with FatWire Engage” - brief introduction to FatWire Engage, which allows you to create targeted messages for online site visitors.
- Chapter 6, “Data Modeling” - information about asset data models and flex assets, mainly for developers.
- Chapter 7, “Page Design and Development” - a discussion on modular page design strategy and template design for developers.
- Chapter 8, “Site Replication” - outlines the procedure for replicating FirstSite.
- Appendix A, “FirstSite Configurations” - a breakdown of FirstSite default values and settings
- Appendix B, “Access Permissions” - provides an overview of how to manage users in Content Server.

Supporting Publications

This guide is not to be used in place of the Content Server user, developer, or administrator guides. All topics covered in this guide are discussed in further detail each of the guides mentioned above. These guides are available online at:

`http://e-docs.fatwire.com/CS`

The Developer’s Tag Reference should also be kept on hand for Part 2 of this guide. The tag reference can also be found online at the web address shown above.

The documentation website is password-protected; you will need to obtain a password from FatWire Technical Support. For Technical Support contact information, see the following website:

`http://www.fatwire.com/Support/contact_info.html`

Chapter 1

Introduction

Projects with basic requirements built on a reusable architecture can help your organization manage risk by shortening the deployment cycle and allowing users to become familiar with the system quickly. FirstSite is a Content Server starter site that offers standard templates and documents to ensure that the fundamental features of your own site are implemented the right way. With Content Server FirstSite, you can develop and manage your own site efficiently, reducing development time, deployment cycles, and cost.

This chapter contains the following sections:

- About Content Server
- Basic Content Management Concepts
- Types of Users
- Recommended Approach to this Guide
- Steps for Creating a New Site

About Content Server

The Content Server (CS) product family is a database-driven, high-performance content management and delivery system. You can use Content Server to manage and deliver content on websites and other formats (WAP, for example). The Content Server product family includes the following:

- **Content Server**
  The core application upon which all the content applications are built. It is the operating system that powers the entire CS product family, stores the content you are managing, and serves content to your online site.
- **Satellite Server**
  A remote caching program that allows your site to scale as it grows, and can prevent users from requesting pages before they have been pre-generated.
- **Engage**
  An application that enables your marketing team to divide your market into segments of customers, for the purpose of targeting those segments with personalized promotional or marketing messages.
- **DocLink**
  A document management application that allows you to upload files to the Content Server database through a simple drag-and-drop interface integrated with Windows Explorer.
- **Analytics**
  An application that captures information related to site events. This information is then used to generate detailed reports for analysis by site administrators or content providers.

This guide focuses mainly on the Content Server application and includes overviews of DocLink and Engage. To learn more about Satellite Server and Analytics, you can refer to the *Content Server Developer's Guide*.

When you are using Content Server as your content management system, you and your team can work with up to four different systems:

- **Development system -** where the developers and designers plan and create the online site.
- **Management system** - (also referred to as "the staging environment") hosts content management sites where the content providers develop the content that is delivered to the online site. Using workflow and revision tracking, you track and monitor content until it is are approved and published to the delivery system.
- **Delivery system** - (also referred to as "the production environment") where the content or products that you are selling are served to your visitors or customers. In other words, it hosts the online site.
- **Testing system -** where the Quality Assurance group tests the performance of both the management system and the delivery system.

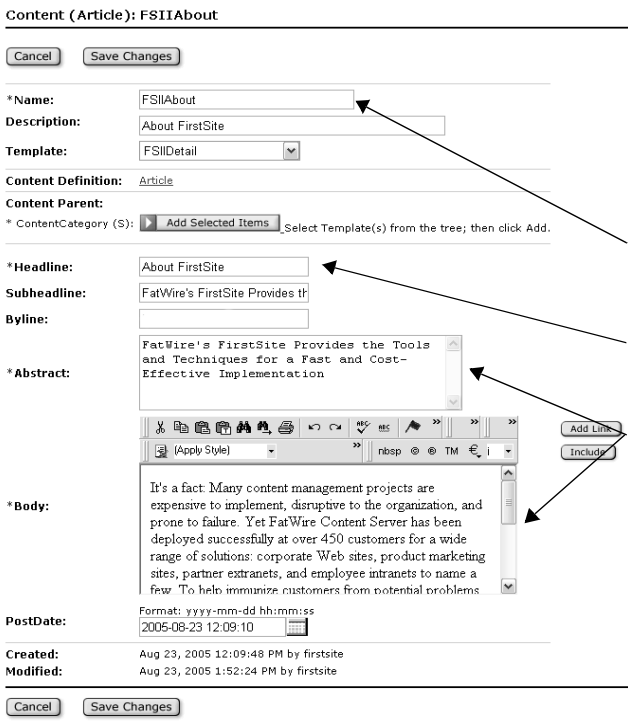
Our focus, for the purposes of this document, is mainly on the management and development systems. For a more detailed explanation on the Content Server components, refer to the *Content Server Developer's Guide*.

Basic Content Management Concepts

This section explains how Content Server defines and treats content. It explains terms such as “assets,” and “asset types,” which are used throughout this guide.

Assets and Asset Types

Content in Content Server is referred to as “assets.” Assets are divided into different types. The difference between an asset and an asset type is explained below.



An **asset** is an object created by a user populating the fields of a content-entry form, similar to the one at the left. In this example, the asset is an article, defined by field values entered by the user:

a specific name (FSIIAbout in this example)

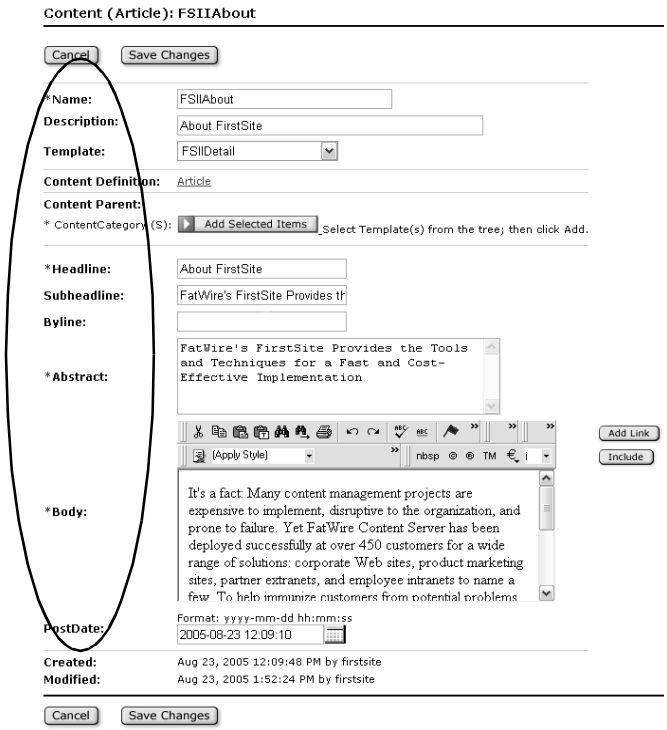
a specific headline (About FirstSite in this example)

a specific abstract and body

Field values define the asset. When saved, the asset is stored in Content Server’s database. The asset can be edited, inspected, deleted, duplicated, placed into workflow, tracked through revision tracking, searched for, and delivered to the online site.

An asset is categorized as either a **content** asset or a **design** asset. Content assets are those that visitors can read, examine, or purchase. This includes content such as articles, products, or photos. Design assets are used to organize content assets or format the code used to display content assets. Most design assets are built-in to Content Server and are available to you provided they have been enabled by your administrator. These built-in assets, also known as core assets, include Page, Collection, Query, Template, Recommendation, CSElement, and SiteEntry assets. With exception of the Page and Recommendation assets, design assets are generally used by developers.

During the process of designing your online site, you and others on your team create the assets that you need in order to represent the content for your site. In Content Server, assets are based on one of two models: **basic** or **flex**. The asset model chosen to represent the data displayed on your site depends on the nature of the data. For content providers, the distinction between the two models is not relevant. The majority of the functions you perform for managing content are the same, regardless of the asset model type. For developers, asset models are discussed in greater detail in Part 2.



Like any asset, the asset in our example is an instance of an asset type, defined in the content-entry form by the names of the fields (listed at the left of the screen): Name, Headline, Subheadline, Byline Abstract, Body, and PostDate. **Field names define the asset type.**

An **asset type**, is a specification—a set of field names that are created by the administrator (and the system) in order to define the nature of the asset type: a news article, a job ad, an HTML document, for example.

Many kinds of asset types can be created for Content Server. Each asset type has its own content-entry form, formatted as shown on this page, but with a unique set of fields. Developers design asset types for you to work with, creating and editing assets of those types.

From Content-Entry Forms to the Online Site

Content-entry forms are interfaces designed for business users to provide and edit content for their online site(s). A content-entry form has a well-defined relationship to the online site and to the database, which Content Server uses to store content. The relationship is illustrated in Figure 1 and Figure 2, and explained below. The figures illustrate the relationship of a content-entry form to a dynamic online site.

When you populate a content-entry form—for example, the “Article” form in Figure 1—and save the content, Content Server stores the content in its database (step 1 in Figure 1). After you approve the asset for dynamic publishing, for example, you or the CS administrator publishes the asset to the delivery system, where a duplicate database accepts the asset (step 2 in Figure 1). Finally, when the asset is ready for delivery to the online site, its content is drawn from the database by code, formatted by the code, laid out by the code (step 3 in Figure 1) and delivered to the online site by code (step 4 in Figure 2).

The relationship of a content-entry form to the online site is a straightforward one: A content-entry form accepts raw content for storage in the database; at display time, the site displays the content, but in client-ready format.

Content-entry forms offer their users major advantages:

- Users don't need to learn the specifics of the Content Server’s database.

  Content that you enter into a form is stored in the database. Content that you retrieve is read from the database and displayed in an editorial version of the content-entry form.

  Because a content-entry form is a standard interface to the variety of databases that Content Server supports, it spares users from having to learn the specifics of any

database in particular. If one database is replaced with another, the switch is transparent to users.

- Users don't need to know HTML or other markup languages.

  No content-entry form requires its users to format the content they enter or edit. Formatting is accomplished by code, which is created by developers to meet the online site designer's specifications. Content providers remain focused on the content they are providing and its quality.

- Users know what kind of information is expected from them.

  In content-entry forms, field names prompt users for certain kinds of information: a phone number, a job description, a file name, and so on.

- Reusability and consistency are maximized.

  Each unit of content that is entered into a form can be reused as many times as necessary, in as many formats as necessary, in as many locations within the online site as necessary. Reusability ensures consistency across the site by eliminating the need for re-creating information each time the information must be used.

Figure 1: A content-entry form

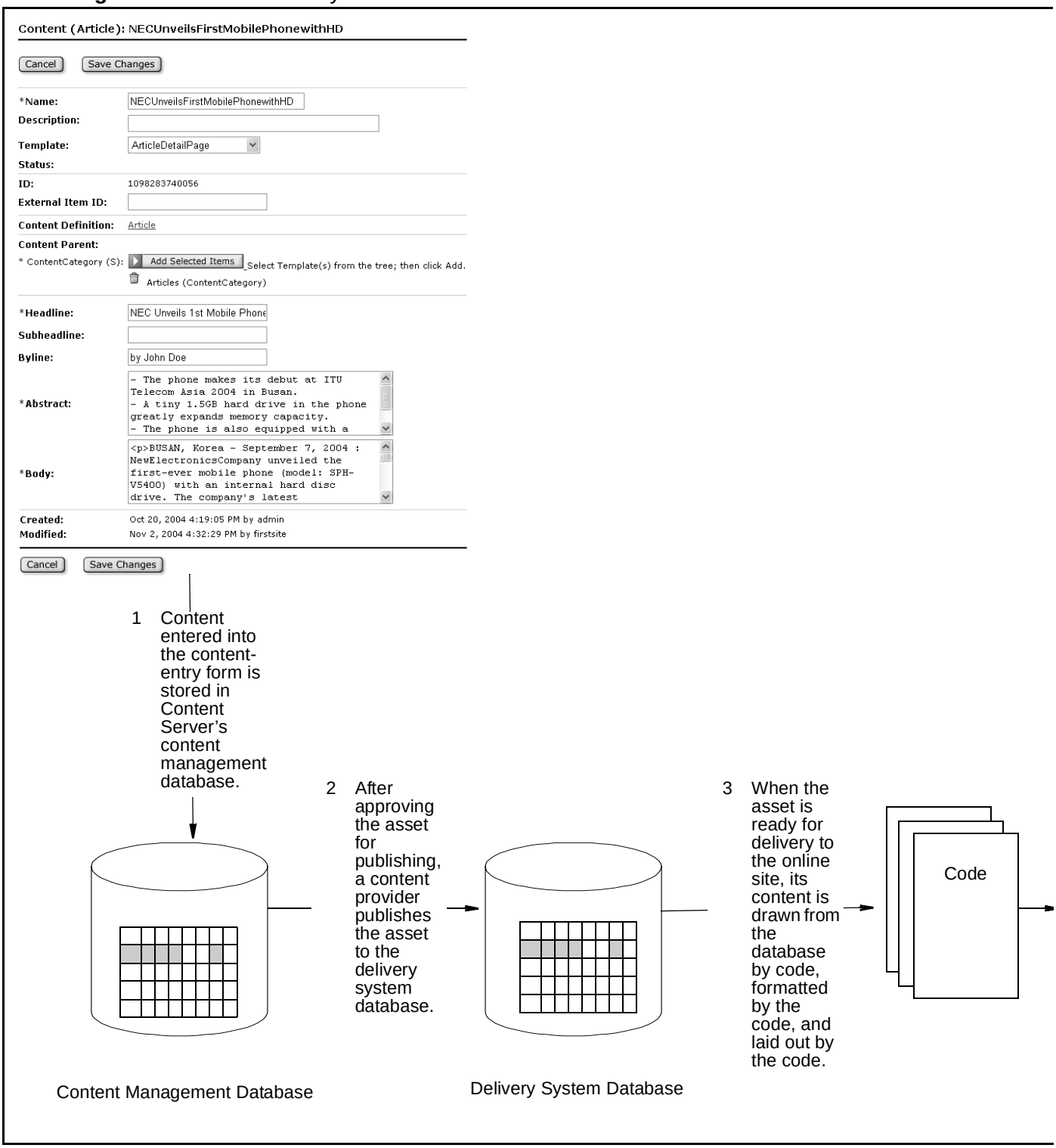
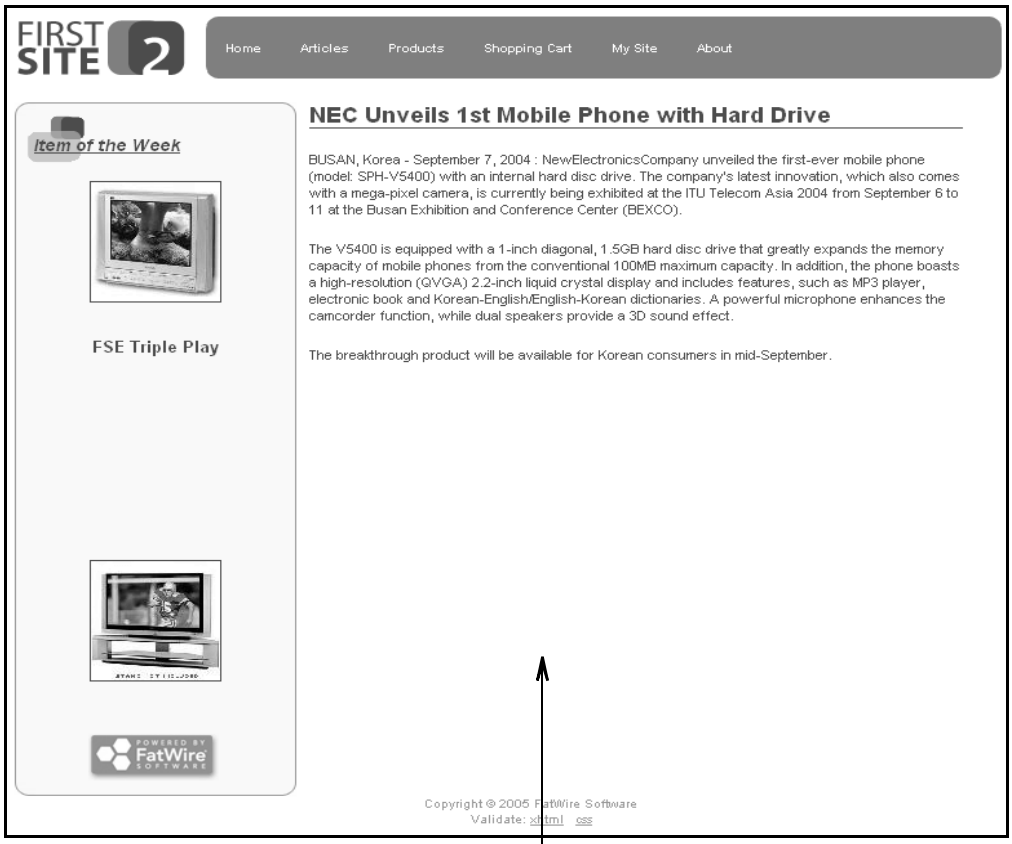


Figure 2: The online site



4 Formatted content is delivered by code to the online site.

FirstSite Asset Types

An **asset type** is a definition or specification that determines the characteristics of assets of that type. The FirstSite Store Schema module comes with definitions for three content asset families (FirstSite modules are discussed in Chapter 2, "Content Server FirstSite"). A complete list of FirstSite asset types can be found in "FirstSite Configurations," on page 97. The following table lists the content asset types available with the FirstSite Store Schema module:

Table 1: FirstSite Store Schema Asset Types

Asset Type	Description
Content	For creating articles and page detail assets
Product	For creating product items
Media	For creating multimedia assets such as product images, site images, and movie clips

New asset types can be created by developers when the need arises. Developers may refer to the *Content Server Developer's Guide* for information on creating new asset types.

Types of Users

In general, there are three types of Content Server users, each with specific functions. The following is a brief description of their jobs:

- **Administrators**
  Administrators are responsible for configuring your CM site and the interfaces you work with. They also manage Content Server user accounts and establish user roles (more information on user accounts and user roles can be found in Appendix B, "Access Permissions").
- **Developers**
  Developers create the content management framework. This includes the data model, asset types, and rendering templates that determine how to display content.
- **Content providers**
  Content providers are users of the framework created by developers. They create assets to populate the site with content.

Although this guide was written primarily for developers, content providers may also take away some helpful points for creating content from Part 1 of this guide.

Recommended Approach to this Guide

This guide is divided into three parts. Part 1 contains information and exercises that can be useful to all users. After successfully completing the exercises in Part 1, readers will have learned how to:

- create, edit, and preview assets using the standard interface
- create content assets using CS-Desktop

- create media and document assets using DocLink
- use personalization to get the most out of your online site (available for FatWire Engage customers)

Part 2 of this guide covers topics such as, modular page design, data modeling, and templates, all of which are important for developers. It is recommended that developers work through Part 1 of this guide prior to moving on to Part 2. This enables developers to gain a full understanding of how assets are created and rendered. By the end of Part 2 of this guide, readers will have learned:

- how the modular page design strategy is applied to FirstSite
- which data model is used for FirstSite and how this model can apply to their own sites
- how to create a new flex asset family
- how content created in Part 1 is rendered to the online site

In Part 3, Content Server administrators are introduced to Site Launcher. This Content Server tool is used to replicate existing CM sites in order to help speed up the deployment cycle of new sites. By the end of Part 3 of this guide, readers will have successfully replicated FirstSite.

Steps for Creating a New Site

The following steps outline the general process for creating a new site with Content Server:

1. **Design the site**
   Determine what is required for your new online site. This includes defining the goals that must be met by the site, what kind of information will be displayed, who the target audience is, etc. Finally, you must determine what types of assets will be needed, then design them accordingly (determine what attributes you will need). Design your workflow processes.
2. **Replicate FirstSite (Administrators)**
   FatWire recommends that you use FirstSite as a foundation for creating new sites. By following guidelines and coding practices outlined by FatWire Engineers, you will achieve optimal performance by Content Server. For information on replicating FirstSite, see Chapter 8, “Site Replication.”
3. **Create New Assets Types**
   If the FirstSite asset types do not meet your requirements, you can modify them to fit your needs or design and create new assets or asset families. If you are using the Flex Asset Model, you will create flex attributes, flex parent definitions, flex asset definitions, and flex filters at this stage. See Chapter 6, “Data Modeling” for more information.
4. **Design and Create New Templates**
   If the FirstSite Template framework does not meet your requirements, design and create new templates and elements to achieve your goals. If you created new asset types, you must create five new templates in order for that asset to be rendered. For more information, refer to Chapter 7, “Page Design and Development.”

5. **Add Content (Content Providers)**
   Once you have defined your asset types and determined how your assets will be rendered, content can now be added to the site. For information on how to add content to your online site see, Chapter 3, “Creating Content in FirstSite.”
6. **Test Your Site**
   Always thoroughly test your site before publishing to your delivery system.
7. **Publish Your Site**
   After publishing your site to your delivery system, you are ready to launch!

Chapter 2

Content Server FirstSite

FirstSite is a Content Server starter site that was developed by FatWire to ensure that you start your project on the right path. It is a pre-fabricated CM site that contains its own generic content (including products and images) and templates for rendering the site. Users can log on and begin using FirstSite like any other CM site. Whether your goal is to learn the basics about Content Server or to roll out a new online site, FirstSite can help you achieve it quickly and with ease.

This chapter contains the following sections:

- FirstSite Objectives
- FirstSite Modules
- The FirstSite Online Site
- Getting Started

FirstSite Objectives

FirstSite can be used in the following ways:

- **As a Learning Tool**

  Using this guide, readers embark on a self-guided tour of Content Server and FirstSite, learning to navigate Content Server and use its tools for managing content by completing several step-by-step exercises. After successfully completing the exercises in this guide, readers should feel comfortable with using Content Server to manage content on a site and be familiar with Content Server terminology.

- **As a Foundation**

  The FirstSite Core modules, which are the base for all other modules, can be used as a foundation for developing applications. The FirstSite Core and the FirstSite Engage Core modules can be duplicated using Content Server site replication tool to serve as the foundation for your project or added to, by installing additional modules. The Core module includes a few standard templates and support for site navigation and site structure to help get your project started the right way.

FirstSite Modules

FirstSite can consist of up to ten individual modules. Depending upon the needs of your business, your Content Server installation may or may not include all ten modules listed below in Table 2. FirstSite will function regardless of which modules are installed on your system as long as the FirstSite Core module is installed. Also note that some modules depend on others in order to function properly. Throughout this guide, you will be informed of which modules are needed in order to complete certain tasks. The following is a list of the available components, their descriptions, and their dependencies:

Table 2: FirstSite Components

Component	Description	Dependencies
FirstSite Core	The FirstSite Core creates the asset type and template framework. It includes support for site navigation (using typeless templates and a few core asset types), a workflow, and user login/logout hooks for use with other components. This module does not include any content asset type definitions or sample data.	Content Server
FirstSite Store Schema	The FirstSite Store Schema module includes the flex families and asset types for media, products, and content, as well as all of the templates required to render them, including support for shopping carts. Also included are: tree tabs, start menus, workflow, and other infrastructure needed to work with the schema.	FirstSite Core

Table 2: FirstSite Components

Component	Description	Dependencies
FirstSite Store Demo Data	This module installs the demo data and sample users into the store schema, not including any demo data that is specific to FatWire Engage.	FirstSite Store Schema
FirstSite Engage Core	This module extends the FirstSite Core and adds typless templates that have support for FatWire Engage. It also includes personalization templates, enhanced recommendation assets, and enhanced user login/logout code for use with other components. The Engage Core does not contain any data.	FatWire Engage FirstSite Core
FirstSite Engage Store Schema	Includes tree tabs, start menus, and infrastructure needed to add Engage functionality to the Store Schema module including shopping cart and discount support.	FirstSite Store Schema FirstSite Engage Core
FirstSite Engage Store Demo Data	This module includes sample data and users specific to FatWire Engage.	FirstSite Engage Store Schema
FirstSite Document Schema	This module includes the FirstSite Document Management Schema (i.e. the flex families and asset types for document management), as well as workflow start menus.	FirstSite Core
FirstSite Document Demo Data	This module includes demo data and users for the FirstSite Document Schema.	FirstSite Document Schema
FirstSite URI Assembler	This module provides a URI assembler that works within the FirstSite template naming convention to provide short, optimized URLs. The URLs are not guaranteed to have no query string, but the number of variables in it will be reduced as much as possible.	Content Server
FirstSite Local User Demo Add-on	This module uses flex assets to model individual users. It includes login pages and user profile management code. It is present primarily for demo purposes, but also shows how to connect users to the FirstSite Core and Engage Core modules.	FirstSite Core

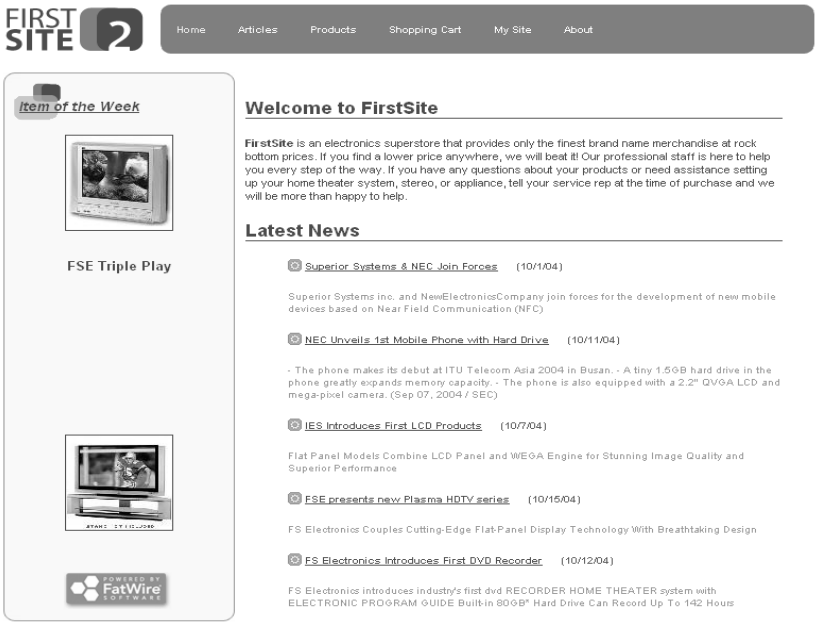
In general the Core modules create the foundation. The Schema modules are “quick start” modules for developers. The Demo Data modules demonstrate a working site. Both the Core and Schema modules require developers to complete some modifications before they are ready for content providers.

For more information on sample users, roles, and default settings packaged with each of the modules listed above, see Appendix A, “FirstSite Configurations.”

The FirstSite Online Site

This Content Server starter site is an online retail site that includes articles, images, a product catalog with several product entries, documents, and sample user accounts. Figure 3 shows the home page, or end result, of the FirstSite CM site.

Figure 3: FirstSite home page preview



The FirstSite model is meant to help you feel comfortable with using Content Server tools to create your online site. Even if your industry is not retail, you can still take the experience you gain with FirstSite and apply it to your own online site.

Getting Started

1. Start your Content Server JumpStart Kit:
   a. Locate the CLICKMEFIRST.exe file.
   b. Double-click on the file to start your Content Server instance.
   c. Point your Internet Explorer browser to the following URL:
      `http://localhost:7001/cs/Xcelerate/LoginPage.html`
2. At the Content Server Login screen, enter the following username and password and click Login:

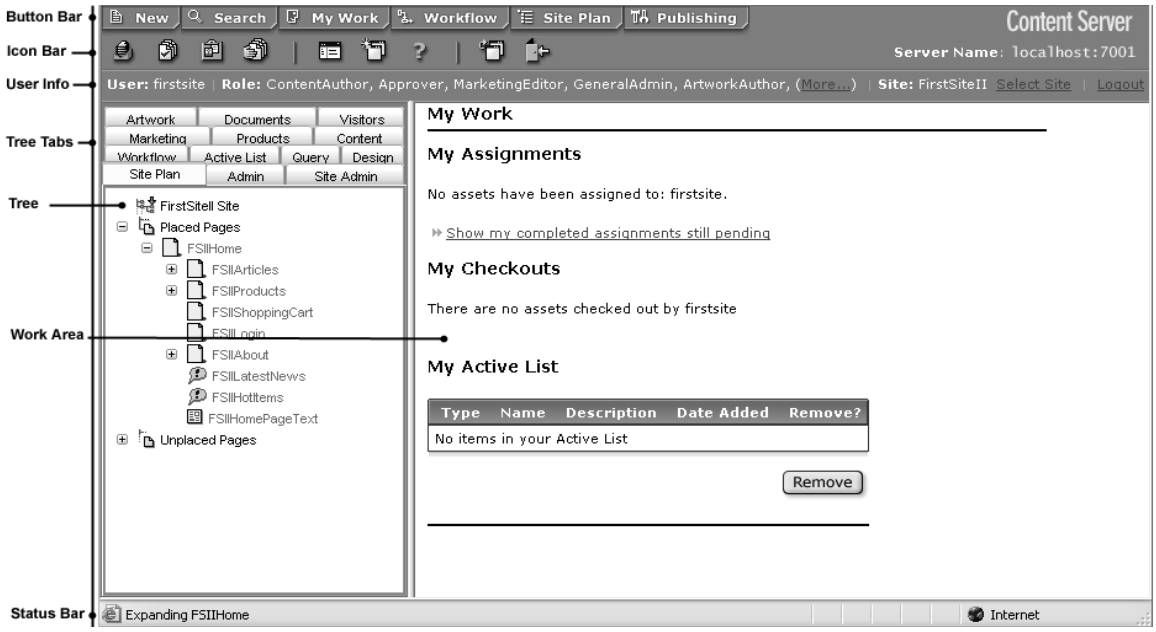
   Username: `firstsite`

   Password: `firstsite`

   If your Content Server JSK includes other sites in addition to FirstSite, you will be brought to the **Site Selection** screen after successfully logging in. You can work with only one site at a time, but once you are logged in, you can switch between sites if other sites to which you have access rights are available.

3. Select **FirstSiteII** from the list of sites to work on if it does not immediately load. Your workspace will look similar to the one shown in Figure 4.

Figure 4: Content Server Interface.



Generally, the appearance of your Content Server interface is dependent upon how your administrator configures your site and what roles you are assigned as a user. A role in Content Server determines your job function and limits your access to other areas of the site. For example, a **ContentAuthor** is limited to creating articles. When a **ContentAuthor** logs in to the CS interface they do not see any other tree tab except for the Content and Media tabs. For the purposes of this guide, we are using the user **firstsite**, who has been assigned all available roles for this site.

For information about different views, toolbars, and customization of your Content Server interface, refer to your *Content Server User's Guide*.

Preview FirstSite

To make it easier for you to refer back to the Preview version of FirstSite, bookmark the the page for future reference:

1. Click the **Site Plan** tab in the left panel.
2. Expand ⊞ Placed Pages by clicking on the ⊞.
3. Right-click on ⊞ FSIIHome and select **Preview** from the menu. The Preview version of the FSIIHome page will render in a new browser window.
4. In the top right section of the new browser window, click "Preview in Full Window." When the page has finished loading, bookmark this page for future reference.

Part 1

Using FirstSite

Part 1 contains information that is helpful to all users. It includes the following chapters:

- Chapter 3, “Creating Content in FirstSite”
- Chapter 4, “Workflow & Publishing”
- Chapter 5, “Personalization with FatWire Engage”

Chapter 3

Creating Content in FirstSite

The type of content that you manage with Content Server depends on the nature of your organization. For example, a news site might produce articles, photos, and video clips. A human resources department might manage job postings and personnel policies. An online retailer might offer product descriptions, special offers, and coupons. In Content Server, articles, clips, coupons, product descriptions, photos, and so forth are referred to as assets. Your end goal for your assets is to move them from your management system to your delivery system so that your visitors can view them.

This section shows you how to add content to FirstSite through a series of step-by-step exercises. From these exercises, you will gain an understanding of how content is created and you will become comfortable with navigating the Content Server interface.

In order to complete the exercises in this chapter, you must have the following FirstSite modules installed:

- FirstSite Core
- FirstSite Store Schema
- FirstSite Store Demo Data
- FirstSite Document Schema
- FirstSite Document Demo Data

This section includes the following sections:

- Creating Assets
- Alternative Ways to Add Content

Creating Assets

In this section, you will use the Content Server standard interface to create the following assets in FirstSite:

- Content Asset - a new article
- Product Asset - a new product placed under a new product brand
- Static List Recommendation Asset - grouping assets in a recommendation list
- Page Asset - creating and placing a page in the Site Plan tree

Note

The same assets can be created by alternative methods independent of the standard Content Server interface. These methods are summarized in the next section, "Alternative Ways to Add Content," on page 35.

By completing the exercises in this section, you will gain an understanding for how content is created and managed in Content Server. Developers will learn in Part 2 of this guide how to create the asset types and content-entry forms that content providers use.

Content Asset

The Content asset type is used to represent articles. In the following exercise, you will create a new article using the Content asset type.

To create a new content asset

1. Log in to the Content Server interface by following the steps outlined in "Getting Started," on page 22.
2. Click 

in your button bar. This will load the **Start Menu** in your workspace.
3. In the **Start Menu**, select **New Content**.
4. In the "Choose Assignees" screen, select **firstsite**. Click 

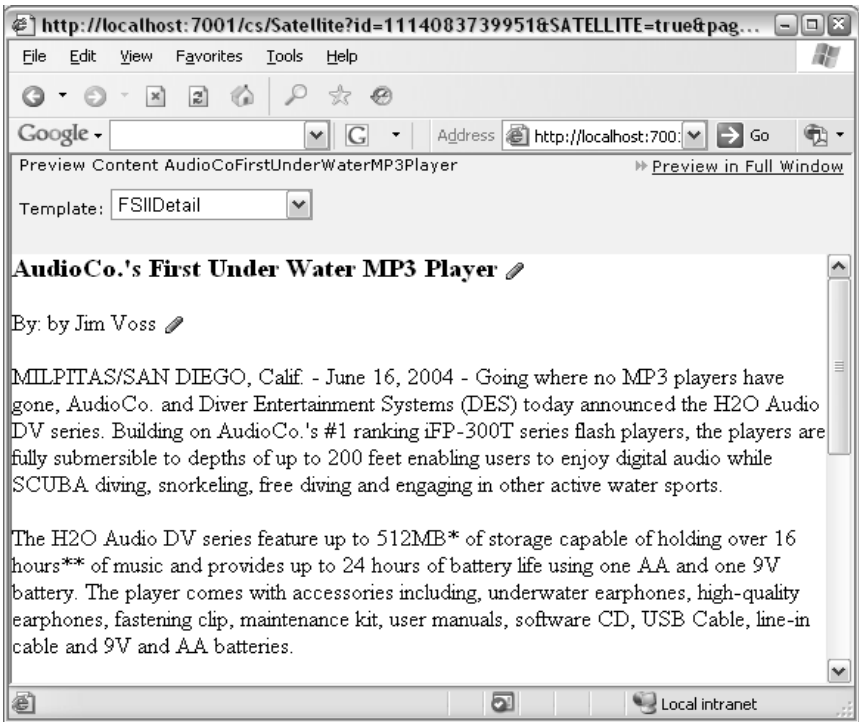
.
5. In the form that appears in your workspace, fill in the appropriate information for each of the fields.
   a. In the "Name" field, enter `MyNewArticle`.
   b. Enter in a short description for your new article in the "Description" field.
   c. For "Template", select **FSIIDetail**.
   d. For "Content Parent,"
      1) Click on the "Content" tab in the tree panel.
      2) Single-click on **FSII Articles** to select it
      3) In your workspace, click 

to add **FSII Articles** as a "Content Parent."
   e. Complete the other fields as you see fit. Fields that are marked with a red asterisk (*) are required and must be filled in before you submit the form.
6. When all required fields are filled in, click 

.

7. When your content asset is saved, you are brought to the content asset's "Inspect" screen. Click Preview. Your new content asset should look similar to the one shown in Figure 1.

Figure 1: Preview of the new article asset.



After the article is created, it is listed under the **Content** tab of the tree panel.

For Developers

The content-entry screen is a manifestation of the data model. Data models are discussed in Chapter 6, "Data Modeling."

The discussion on how the FSIIDetail template renders article assets can be found on page 79.

Product Asset

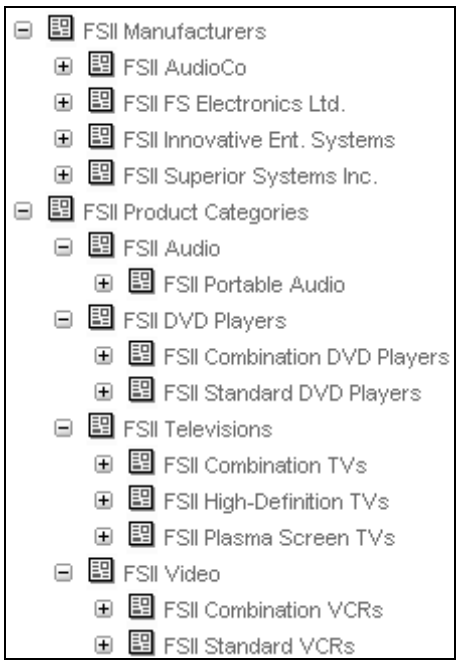
The FirstSite Product Catalog represents the hierarchal structure that can be created in Content Server. If your industry is not retail, you can use this model to represent arbitrary hierarchal structures used to organize and classify your assets based on common properties. Another example of a hierarchal a flex asset structure can be found in our discussion about "FatWire DocLink: Integration with MS Windows Explorer," on page 50.

For Developers

The FirstSite Product Catalog mentioned here and the hierarchal structure discussed on page 50 are examples of the flex asset model. For more information regarding asset data models and flex assets, see Chapter 6, "Data Modeling" on page 61.

In the tree panel, click on the **Products** tab. You should see various products categorized under different brands and electronics types (Figure 2).

Figure 2: View of the Products list for FirstSite



In the following exercise, you will add a new product asset, a `NewElectronicsCompany Flat Panel Plasma TV`, to the product list. You will first add the `NewElectronicsCompany` brand category to our product catalog before creating the new product asset.

To add the `NewElectronicsCompany` brand to the Product Catalog

1. In the Content Server interface button bar, click New
2. In the **Start Menu**, select **New Product Parent**.
3. In the "Choose Assignees" screen, select **firstsite**. Click Set Assignees
4. In the "Name" field, enter `FSII NewElectronicsCompany`.
5. For "Product Parent Definition," select **FSII Manufacturer**. Click Continue.
6. Enter in a brief description for your new product parent in the "Description" field
7. Select **FSIIDetail** for the "Template."
8. For "Product Parent,"
   a. Click the **Products** tab in the tree panel.

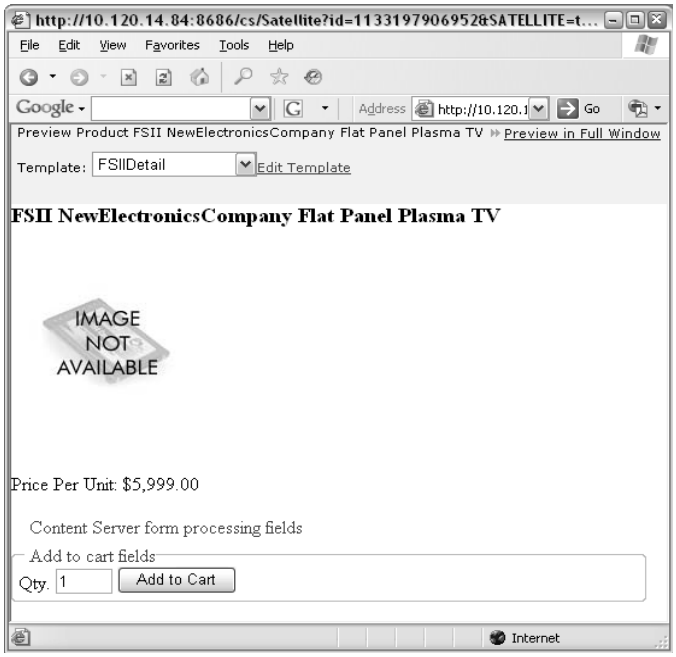
   b. Single-click on **FSII Manufacturers** to select it.
   c. In your workspace, click Add Selected Items to add **FSII Manufacturers** as the product parent.
9. You can enter a brief description for the `NewElectronicsCompany` in the "FSII ManufacturerDescription" field.
10. For "FSII ManufacturerLogo", click on the **Choose Visually** link.
   a. From the thumbnail selection, select the **FSII Image_Not_Available.jpg** thumbnail.
11. When you are finished filling in the form, click Save. Your new product parent will appear in the "Products" tree under **FSII Manufacturers**.

Now we can add the `NewElectronicsCompany Flat Panel Plasma TV` to our product list.

To add a new product to the list

1. In the button bar, click New
2. In the **Start Menu**, select **New Product**.
3. In the "Choose Assignees" screen, select **firstsite**. Click Set Assignees.
4. In the form that appears,
   a. Name your new product `FSII NewElectronicsCompany Flat Panel Plasma TV`.
   b. Enter a brief description for your new product in the "Description" field.
   c. Select **FSIIDetail** for "Template."
   d. For the "Product Parent" section:
      1) Under **FSII Manufacturers** in the **Products** tree, single-click **FSII NewElectronicsCompany** to select it.
      2) Back in your workspace, click Add Selected Items to add `FSII NewElectronicsCompany` as a product parent.
      3) To add a product category parent, expand the **Product Categories** item in the **Products** tree. Under **FSII Televisions**, single-click **FSII Plasma Screen TV**.
      4) In your workspace, click Add Selected Items again for "Product Parent."
   e. Next, enter the "FSII SKU", "FSII LongDescription", and "FSII Price" for the product in their respective fields.
   f. For "FSII Image", select **image_Not_available.jpg** from the menu.
5. Click on Save when you are finished. You will then see your product added to the product list.
6. After your product is saved, you are brought to the product's "Inspect" view. In the action bar, click Preview to see how the product is rendered to the online site.

Figure 3: Preview of the NEC Flat Planel Plasma TV



Static List Recommendation Asset

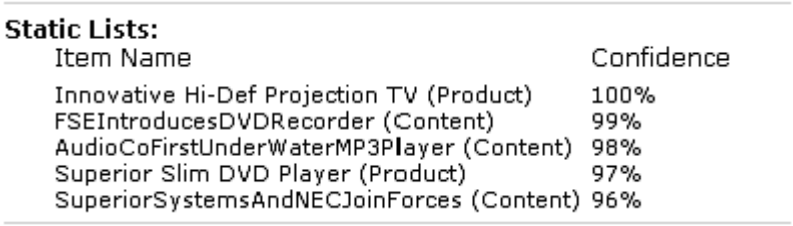
Recommendations are assets that determine which products or content should be featured or “recommended” on a rendered page. There are three types of recommendations in Content Server, static list recommendations, related items recommendations, and dynamic recommendations. Only static list recommendations are available to you without FatWire Engage. A static list recommendation asset holds a pre-selected list of assets that do not have to be of the same type. For example, you can create a static list recommendation and configure it to hold assets of type Content and/or Product. Using recommendations will give you the ability to determine what content is displayed, as well as the order in which it is displayed.

An example of a recommendation that contains both Content and Product assets is the **HotItems** recommendation.

To view the HotItems recommendation

1. In your **Site Plan** tree tab, expand the Placed Pages item by clicking on the ⊞.
2. Expand the FSIIHome item.
3. Double-click on FSIIHotItems to inspect the details of this recommendation item in your workspace.
4. Examine the section labeled **Static Lists**.

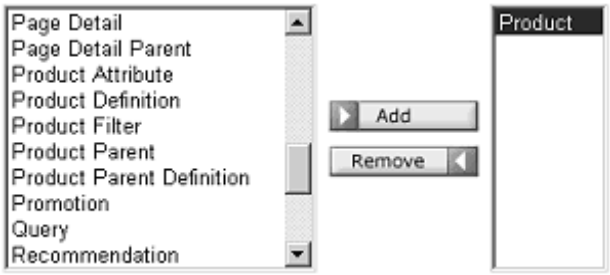
Figure 4: Details displayed for the HotItems recommendation item



The values you see for Confidence represent a scaling or weighting factor used by a recommendation to determine the order in which the assets appear. The Confidence value is used in determining the order in which assets will be displayed.

To create a new recommendation:

1. Log in to the Content Server interface (follow steps in "Getting Started," on page 22).
2. In the button bar of the Content Server interface, click New
3. Select **New Recommendation** from the **Start Menu**.
4. In the "Choose Assignees" screen, select **firstsite**. Click Set Assignees.
5. Enter `NewRecommendation` in the "Name" field.
6. For the "Description," enter `New Recommendation`.
7. For "Mode", select **List**.
8. Click Continue.
9. For "Options,"
   a. Select, "**Does NOT Bring back children of returned assets**."
   b. The options regarding promotions are related to personalization features offered by FatWire Engage. You can select: "**Promotions cannot override this recommendation**."
   c. The options for selecting which assets the recommendation applies to is important. Since we are creating a recommendation list of articles and products, select the option: "**This recommendation applies to the following assets:**"
   d. Single-click on **Product** to select it and click on Add.



   e. Single-click on **Content**, and then click Add.
10. Click Continue.
11. Add items to the recommendation

a. Click on the **Product** tab in the tree panel. Select any products from the tree and click Add Selected Items to add them to the recommendation

b. From the **Content** tab, select any article and add it to the recommendation as well.

c. You can change the order of the items by selecting items and moving them up and down by clicking the respective arrow buttons under **Display Order**.

12. Click Save.

13. When you are finished, your recommendation will then be listed under the **Marketing** tab of the tree panel if you have installed the FirstSite Engage Schema.

Page Asset

A page asset is a design asset that represents pages or classes of pages on a website. A page asset stores references to other assets. Page assets do not represent each rendered page on the site, but rather, they represent different sections of the site determining the overall structure and organization.

Creating a Page Asset

To create a new page asset

1. Click New in the button bar.
2. Select **New Page** from the **Start Menu**.
3. In the "Choose Assignees" select **firstsite**. Click Set Assignees.
4. In the "Name" field, enter `MyNewPage`.
5. For "Subtype" select **Standard**.
6. For "Template", select **FSIILayout**.
7. In "New Page" form, scroll down to the section labeled **Contains**.
   a. You can associate a recommendation list by selecting the recommendation you created above from the **History** tab in the tree panel.
   b. After you have selected your recommendation list, click Add Selected Items.
8. Complete the rest of the form and click Save.
9. Your new page will now appear in the **Unplaced Pages** portion of the **Site Plan** tree.

Placing a Page

To place pages in the site

1. Determine what the parent node will be for your new page and right-click on that node in the **Site Plan** tree under **Placed Pages**
   - Similarly, you can click on **Site Plan** from the button bar and navigate down to the node you wish to add your page to, then click on "**Place page under this site**". If you choose this technique, you can skip to Step 3.
2. From the popup menu, select **Place Page**. This will load the Place Page form in your workspace.
3. In the Place Page form, determine the rank of your new page (the order it would appear with regards to other child pages that may exist for the parent node).

4. Click Save.
5. You should now see your new page appear in the **Placed Pages** portion of the **Site Plan** tree. To preview your new page,
   - Right-click on the Page and select **Preview** from the menu, OR
   - Double-click on the asset to load its "Inspect" view in your workspace and select Preview from the action bar.

   Notice in your preview that your page now appears in the top navigation bar.

Deleting a Page

To delete a page

1. First, remove the page from the Placed Pages portion of the Site Plan tree
   a. Right-click on the parent node and select **Place Pages** from the popup menu.
   b. Mark the check box under **Remove?** for the page you wish to remove.
   c. Click Save. Your removed page is now placed in the **Unplaced Pages** portion of the **Site Plan** tree.
2. In the **Unplaced Page** section, right-click on the page you want to delete and select **Delete**.

Alternative Ways to Add Content

Content Server provides users with three alternatives for creating/editing content: InSite Editor, FatWire DocLink, and CS-Desktop. In this section we also discuss additional standard interface features that you use to create content in Content Server.

InSite Editor

Content providers who are infrequent users of the Content Server interface can edit or approve content directly from a rendered (Preview) version of an asset through the InSite Editor tool. InSite Editor is a browser-based tool that allows the user to select a rendered asset and make minor changes to and/or approve the asset from this continuous previewed version. It simplifies the process for editing and approving assets, as well as allows business users who are inexperienced with the Content Server interface to accomplish their goals of delivering content to their customers.

In order to use this tool, developers must code the rendering templates appropriately. Additionally, InSite Editor can be configured for specific users. For more information on configuring InSite Editor, refer to the *Content Server Administrator* and *Developer's Guides*.

Using InSite Editor

Open Internet Explorer for Windows and return to your bookmarked Preview version of FirstSite that you saved in "Preview FirstSite," on page 23. When the page is finished loading, you may be prompted to log in (see Figure 5 below).

Figure 5: InSite Editor login window

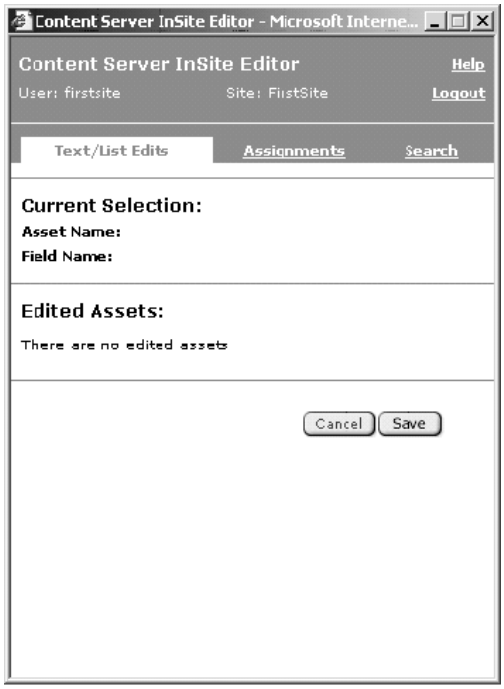
Enter the user name and password for FirstSite:

User Name: `firstsite`

Password: `firstsite`

The InSite Editor browser window opens when you are viewing an editable page. From this window (Figure 6), you manage all of your inline editing activities.

Figure 6: The InSite Editor window



There are three work areas in the InSite Editor window:

- Asset Summary display
- Assignments
- Search

Asset Summary Display

When you point to an editable field on a page, this area displays a summary of identifying information specific to the asset that is associated with that field. If you make changes to a

field, use the Save button at the bottom of the asset summary display to commit those changes to the Content Server database.

Assignments
The Assignments area is the second tab of the InSite Editor window, where your current workflow assignments are listed.

Search
The Search feature is located in the third tab of the InSite Editor window. This feature allows you to search the site for assets by name or description. The search results area displays up to 10 matching assets. Use the previous (<<) and next (>>) icons to scroll through the results.

For more information on using InSite Editor, see the *Content Server User's Guide.*

CS-Desktop: Integration with Microsoft Word

Content Server Desktop is a tool integrated with Microsoft Word that allows you to create assets by entering text and mapping it to specific fields. These fields are the fields in the asset creation form that would be filled in if you were using the Content Server interface. If you do not have CS-Desktop installed and you are using a Content Server Jumpstart Kit, you can find the self-extracting executable file at the following URL: `http://localhost:7001/cs/Xcelerate/DownloadPage.html`

CS-Desktop is accessible in MS Word through the Content Server toolbar. If the toolbar is toggled off, go to the menu bar and click on **View > Toolbars > Content Server**.

To log on to CS-Desktop,

1. Start Microsoft Word.
2. Click on **Content Server** in the Content Server toolbar.
3. Select **Login…**

When you are prompted, enter in your username, password, and the server URL for Content Server.

Username: `firstsite`
Password: `firstsite`
Server URL: `http://localhost:7001/cs/`

4. Click **OK** to connect.

Figure 7: Content Server toolbar when a user is not logged in.

Once you are logged in, the three list menus (Site List, Type List, and Subtype List) are populated dependent upon the sites you have enabled for CS Desktop and have access to.

Figure 8: Content Server toolbar when user “firstsite” is logged in.

Field markers appear as square brackets that enclose selected text when you choose the marker from the Field Markers list. The ! that appears next to a field marker in the list denotes a required field that must be specified prior to saving the asset to Content Server. The ✓ signifies that the field has been set. These markers map text that you enter in your Word document to fields of the asset type. In Word terminology, these field markers are called "bookmarks." You can use the light bulb icon to toggle their appearance on or off. When the light bulb is yellow, they are on; a white bulb means they are off. Using this toggle is a shortcut to selecting **Tools** > **Options** > **View** > **Show Bookmarks** in the menu bar.

FatWire DocLink: Integration with Microsoft Windows Explorer

FatWire DocLink is an application integrated with Windows Explorer that allows you to create content for your website and save it to Content Server without having to log in to the standard interface. DocLink provides a simple drag-and-drop interface for uploading files that would be represented by flex assets in Content Server. This includes text files, formatted documents, image files, and spreadsheets. In order to upload files using DocLink, your flex assets must be configured to work with DocLink. For more information on enabling assets for DocLink, see the *Content Server Administrator’s Guide*.

If you are using a Content Server Jumpstart Kit, DocLink is provided with your Jumpstart Kit as a self-extracting executable file. You can find self-extracting executable file at the following URL: `http://localhost:7001/cs/Xcelerate/DownloadPage.html`

Note

If you are using FirstSite on a full Content Server installation, you must have the FirstSite Document Schema component and FatWire DocLink installed in order to complete the exercises in this section.

Once you have DocLink installed on your machine, the DocLink icon will appear on your desktop. When you double-click on the DocLink icon, you will be prompted to enter a Login URL and your account information for the Content Server interface. To log in, enter the following information:

Login URL: `http://<servername>:<port>/cs/servlet/ContentServer?pagename=OpenMarket/CSClient/InitialURLs`

Username: `firstsite`

Password: `firstsite`

After logging in successfully, the DocLink window will appear as shown in Figure 9. The folder structure in DocLink reflects the hierarchy of the flex family enabled to work with DocLink. You can now upload files to your site by using the drag and drop method. To edit the details for any of the files uploaded, you can log in to the Content Server interface and edit them as you would for any other asset.

Figure 9: View of DocLink

Media Assets

You can easily add photos, audio files, video files, and other types of media to your site using FatWire DocLink. The following exercise outlines the process for adding an image to Content Server using DocLink.

For this exercise, you will upload an image to FirstSite. This is so that you can understand the process of creating a Media asset and associating it with a product in the product list. You will need to find a suitable image to upload to Content Server for this exercise.

To create a Media asset,

1. Log in to FatWire DocLink using the information from above.
2. Open **FirstSiteII** > **Media** > **FSII Product Images**
3. In a new Windows Explorer window (double-click on **My Computer**), navigate to location where you stored your image.
4. Drag and drop your image to the **FSII Product Images** folder in DocLink.
5. Click the in the toolbar to log out.

If uploaded successfully, your file will appear in DocLink. After you have completed uploading files, you can then log in to the Content Server interface to edit the asset and fill in other information as you would any other asset.

To associate your image with a product

1. Log in to the Content Server interface.
2. In the tree panel, click on the **Products** tab.
3. Go to **FSII Manufacturers** > **FSII FS Electronics Ltd.**
4. Double-click on **FSE Plasma Screen TV** to load the Inspect view in your workspace.
5. In your workspace, click on Edit.
6. Scroll down to **FSII Image**.
7. Click on the **Choose Visually …** link.
8. The ImagePicker will open in a new window. Find the image you uploaded through FatWire DocLink.
9. Click on the image to select it. The ImagePicker window will close and you should see the image selected appear for the **Image** field.

10. Click Save.

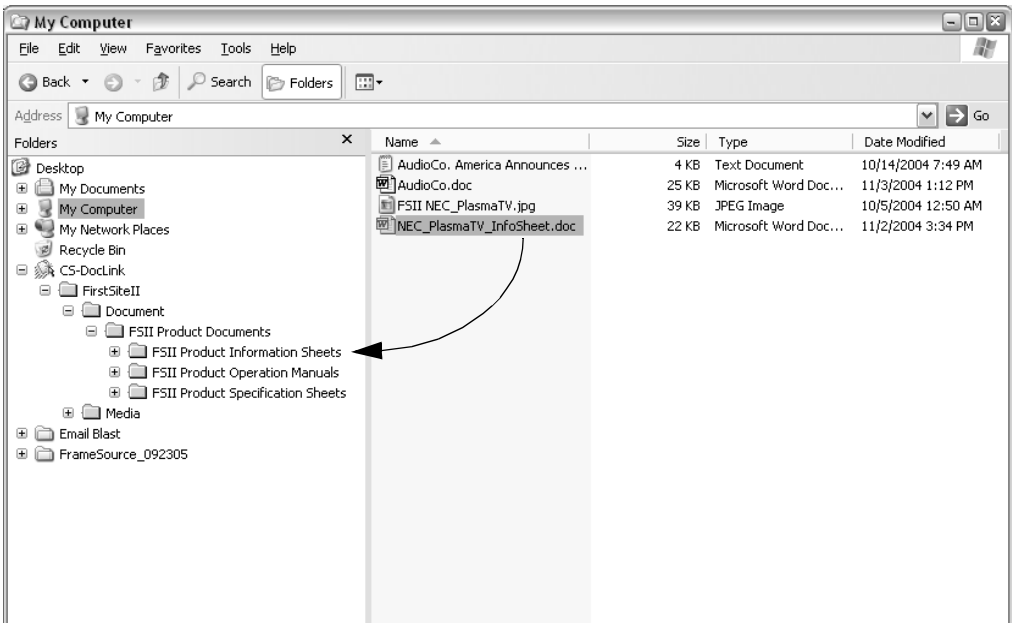
Document Assets

DocLink can also be used to upload Document assets. The Document asset is used to represent Microsoft Word documents, spreadsheets, text files, PDF documents, and other files of a similar nature. Its flex asset hierarchy creates a folder-like categorization that is apparent when you are using DocLink. The Document asset is also unique in that it converts documents to HTML when you upload them to Content Server through DocLink or the standard interface. The Document flex family is available with the FirstSite Document Schema module.

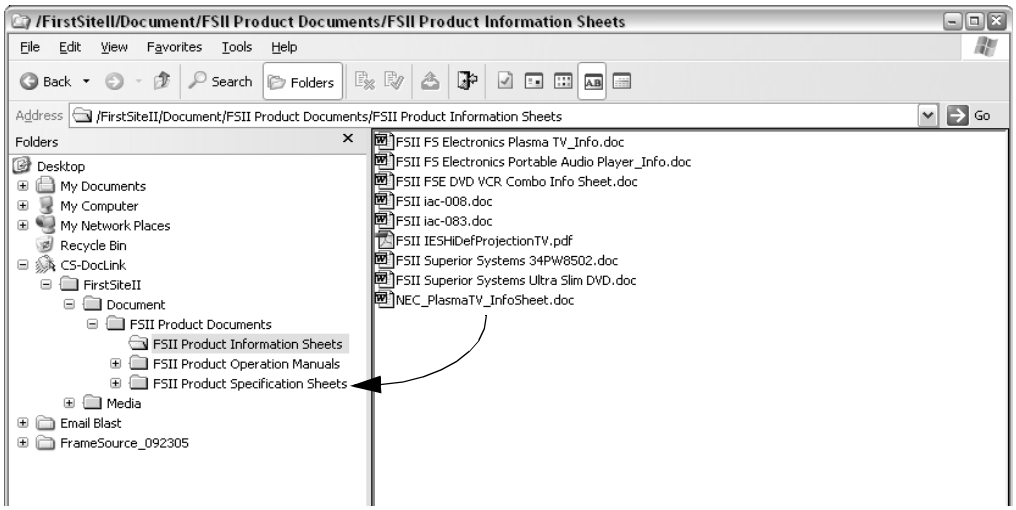
In the following exercise, we will use the Document asset to demonstrate another feature of DocLink. This feature allows you to upload a single document and classify it under multiple categories. If you were to make changes to that document in one location, it is reflected across all the other locations where that document is listed. In other words, you can specify multiple parents for the Document asset in DocLink. For this exercise, you must have the FirstSite Document Schema module install and you will need a document (either MS Word or PDF) to upload to Content Server.

To create a Document asset,

1. Log in to Content Server DocLink using the information from above.
2. Configure the Windows Explorer window to show the Folders panel.
   a. In the Menu bar of the DocLink window, go to **View** > **Explorer Bar** > **Folders** (a checkmark will appear next to it)
3. Open the folders: **FirstSiteII** > **Document** > **FSII Product Documents** > **FSII Product Information Sheets**.
4. Navigate to the location where your document is stored. This document will be uploaded to Content Server.
5. Drag and drop your document to the **FSII Product Information Sheets** folder in DocLink.



6. Open the **FSII Product Information Sheets** folder to make sure that your document has been uploaded successfully.
7. Click and drag your document file from the **FSII Product Information Sheets** folder to the **FSII Product Specification Sheets** folder in the left panel.



8. When you are prompted to select a **Drop Action**, select **Link**.
9. Open the **FSII Product Specification Sheets** folder and you should see your document file.
10. Click in the toolbar to logout.

If you log in to the Content Server standard interface and inspect the your document file, you will see that the file has two parents listed, **FSII Product Information Sheets** and **FSII Product Specification Sheets**. To remove one, edit the file and delete one of the parents.

Attribute Editors

There are tools in Content Server that help simplify content entry for creating or editing assets using the standard interface. Custom attribute editors, created by developers, are used to determine how data is entered for an asset (typically flex assets). Content Server comes packaged with the following attribute editors:

- **Checkboxes**
  Formats the input style of an attribute as a set of checkbox options.
- **Radio buttons**
  Formats the input style of an attribute as a set of radio buttons.
- **Pull down menu**
  Formats the input style of an attribute as a select field with a drop down list.
- **TextArea**
  Formats the input style of an attribute as a textbox with radio buttons for the user to specify whether or not the textbox will allow embedded links.
- **TextField**
  Formats the input style of an attribute as a single text field.
- **EWebEditPro**
  Invokes the EWebEditPro HTML editor. See below for more detailed information.

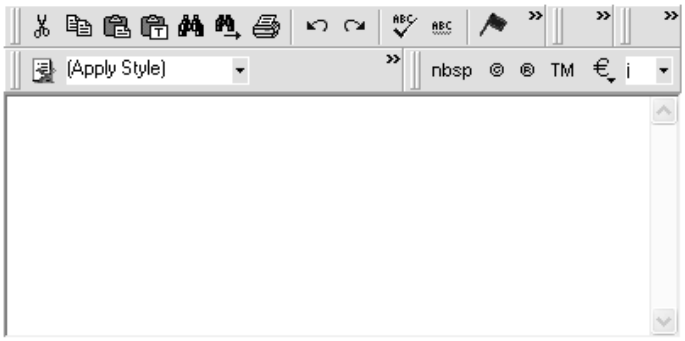
- **PickAsset**
  If the attribute type is an asset, this attribute editor allows you to select assets from the tree.
- **Remember**
  Formats the input style of an attribute as a popup window that displays all the asset currently in the user's ActiveList and History tabs.

Ektron eWebEditPro

eWebEditPro is an easy to use, browser-based content editor that enables anyone to author HTML documents and data. It is a third-party application provided by Ektron, Inc. that is compatible with Fatwire Content Server products. A text field configured to use eWebEditPro will allow you to input text and format it as you would if you used Microsoft Word. Both basic assets and attributes for flex assets can be created using eWebEditPro enabled text fields. An evaluation license is available with your JumpStart Kit.

Figure 10 below shows some of the tools you can use to format data when you use eWebEditPro. You can customize your view of the toolbars to hide tools that are not frequently used.

Figure 10: eWebEditPro text field in Content Server.



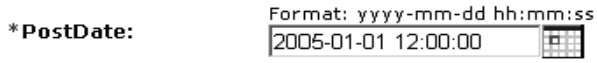
Custom Attribute Editors

In addition to the attribute editors packaged with Content Server, FirstSite includes two custom attribute editors.

The DatePicker Attribute Editor

The DatePicker Attribute Editor formats the input style for dates and time. By default, when you create a new Content asset (or any asset that requires you to enter a date), the date and time are set to the date and time at which you are creating the new asset. You can change the date manually by entering it in the proper format in the field, or you can use the DatePicker.

Figure 11: PostDate field in an asset creation form.



To enter a date using the DatePicker, click on the  icon. A calendar window will appear, allowing you to change the month and year, and select a date from the calendar.

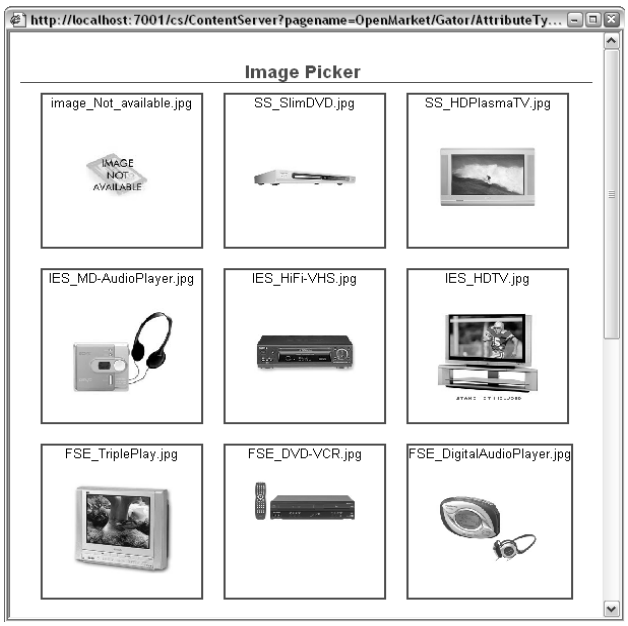
Figure 12: View of the DatePicker calendar window

Once you have selected a date, the calendar will disappear and the date selected is populated to the field in the proper format.

The ImagePicker Attribute Editor

The ImagePicker attribute editor simplifies the process for selecting images for articles, products, and other assets that include images in their rendered display. When you create or edit a Product asset for example, the **Image** field allows you to select an image visually using the ImagePicker. The ImagePicker opens in a new window and displays all the images that have been uploaded for FirstSite as thumbnails. You can browse through the thumbnails to find the image you want. When you have found the image, click on it and the ImagePicker window closes. The image then appears in the Product asset form.

Figure 13: View of the ImagePicker tool.



Note for Developers

The FirstSite ImagePicker attribute editor is not replicable without manual enhancement. In particular, following replication, you have to edit the attribute editor and modify the XML file contained therein so that the new field names reflect the corresponding field names on the replicated site.

Chapter 4

Workflow & Publishing

This chapter provides an overview of workflow. This Content Server feature helps to manage collaborations within a team of content contributors.

This chapter contains the following sections:

- Workflow Overview
- Publishing Methods
- Publishing Approval

Workflow Overview

Several individuals can participate in the gathering, reviewing, and delivery of website content to the rendered site. The process in which content is passed from one person to another upon completion of a task in a specified order is referred to as **workflow**. Content Server users are assigned roles. Tasks are assigned to the different roles, as opposed to individual users. Content Server has a number of pre-defined roles such as, GeneralAdmin, SiteAdmin, and WorkflowAdmin. FirstSite provides additional roles (see Appendix A, "FirstSite Configurations" for a list of available roles). You can create additional roles to customize Content Server to workflows that better suit your business environment. The workflow feature in Content Server also allows you to generate reports to track the progression of content and user assignments in workflow.

Sample Workflow

Figure 14: Graphical depiction of the approval process for Content.

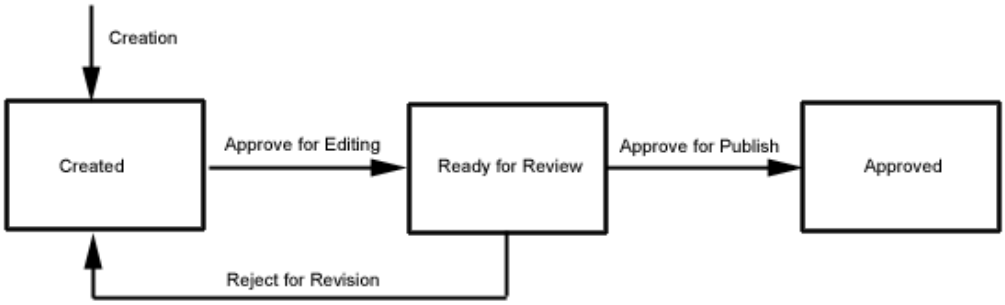


Table 1: Example steps and states in a workflow process.

State	Step	Description	Moves to State
none	Creation	An article is created, initiating the workflow process	Created
Created	Approve for Editing	The article is sent for review.	Ready for Review
Ready for Review	Reject for Revision	The article is rejected and is sent back for revisions.	Created
	Approve for Publication	The article is accepted and approved for publication.	Approved

For additional information on Workflow, please refer to the *Content Server User's Guide*.

Publishing Methods

Content Server lets you serve either dynamic content, static content, or a combination of the two, on your delivery system. There is a default publishing method (or delivery type) for each. You can also convert content to files that can be served on a system other than Content Server. There is also a default publishing method for converted content.

- **Mirror to Server.** This publishing method copies your assets from the Content Server database tables on your management system to the Content Server database tables on your delivery system. Your web server can then generate pages dynamically when visitors request them. This is known as a dynamic delivery type.
- **Export to Disk.** This publishing method renders all of your assets into static HTML files according to the templates provided by the design team. Those files can then be uploaded to the web server of your delivery system. This is known as a static delivery type.
- **Export Assets to XML.** This publishing method converts your assets into individual XML files and stores them in a directory. These asset files can then be published to an external content delivery system that is designed to handle them. This is known as an external delivery type.

The administrator configures publishing delivery types and destinations, and declares which sites have access to a given destination. For information about setting up target destinations, how HTML files are named during an export, and so on, see the *Content Server Administrator's Guide*.

Publishing Approval

Before assets can be published, they must be approved for publishing. Assets have dependencies: basic assets have named and unnamed asset associations; flex assets have associations as well as definitions and parent definitions. Requiring approval is a safeguard against publishing an asset whose dependent assets are not ready to be published. This prevents broken links on the delivery system.

For more information on publishing, refer to the *Content Server User's Guide*.

Chapter 5

Personalization with FatWire Engage

This chapter covers features offered by FatWire Engage, an optional Content Server product. These features allow your company's marketing team to gather data on visitors to your site and develop marketing campaigns targeted to certain visitors based on the data compiled. In order to complete the exercises in this chapter, you must have the following modules installed:

- FatWire Engage
- FirstSite Engage Core
- FirstSite Engage Store Schema
- FirstSite Engage Store Demo Data
- FirstSite Local User Demo Add-on

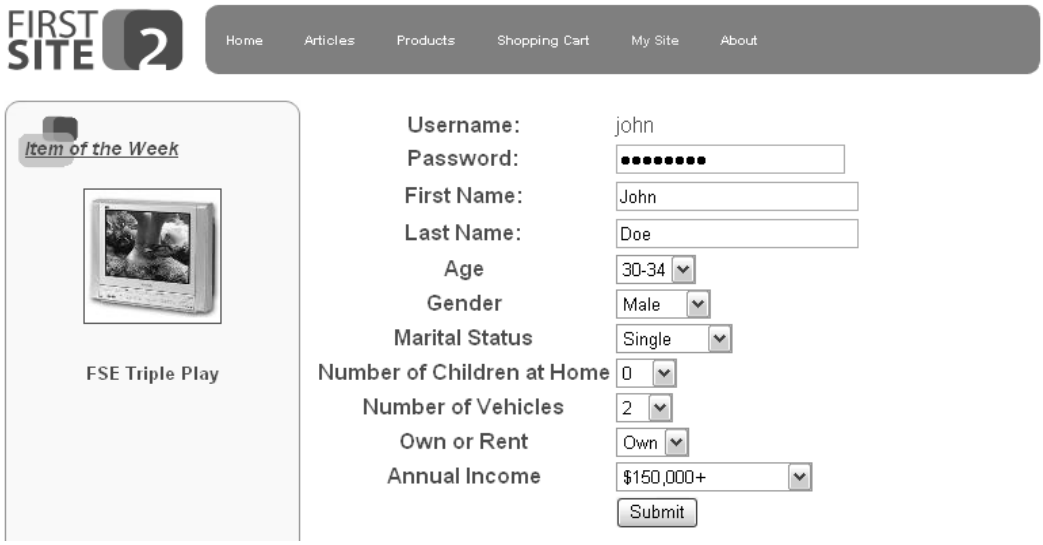
This chapter contains the following sections:

- Visitor Data
- Segments
- Advanced Recommendations
- Promotions
- The Shopping Cart

Visitor Data

Knowing your audience and their interests will help you serve your visitors better. To gather information about your visitors, you use visitor attributes. A visitor attribute represents a single datum value such as age, gender, household income, or state of residence collected from a visitor to your site. To collect visitor data, the visitor completes a form consisting of visitor data fields. The values entered in these fields are collected and used to create visitor attribute assets. Visitor attribute assets are used to define segments, or filtering criteria, by which visitors are grouped. For example, you can set the filtering criteria to include only those who are male between the ages of 20–24, or visitors who are single and live in New York. Your marketing team uses these segments for target marketing campaigns and promotions. For more information on Visitor Data assets and how to create them, see the *Content Server User's Guide*.

Figure 15: Preview of a visitor's profile



Two sample FirstSite visitors were created to help demonstrate Content Server's personalization features. **John** is an affluent single male who owns a house and two cars. **Tracy** is a young married mother of two, who rents her apartment and does not own a car. We will return to these site visitors in the next section.

By using visitor data and segments, you can deliver the right content to the right people at the right time. With that in mind, the same concept can also be applied to any type of site you are creating with Content Server. For example, you might segment people by department and roles, and deliver human resources information depending upon the department and role. Segments are covered in further detail in the next section.

Segments

Segments are assets used to categorize visitors into groups based on similar characteristics. Visitors qualify for segments based on the values of their visitor attributes. The content displayed on site page for a promotion and/or recommendation rely on the segments a visitor qualifies for. Your marketing team can create segments and determine which promotions and recommendations should be associated with specific visitor segments.

Example Segment

Let's take a look at a sample segment already created in FirstSite.

1. Log in to the Content Server interface as the user **firstsite** (password: `firstsite`).
2. Preview the FSIIHome page:
   a. Go to the **Site Plan** tab.
   b. Expand the **Placed Pages** and the **FSIIHome** items.
   c. Right-click on **FSIIProducts** and select **Preview** from the menu.
3. Take note of the items listed for **Hot Products**.
   - FSE Digital Audio Player
   - Innovative HiFi VHS
   - Superior Slim DVD Player
4. Now click on the **My Site** link in the header.
5. Login to FirstSite as the visitor **John** using the following information:
   - Username: `john`
   - Password: `password`

When you have logged in successfully, you will be brought to John's member profile.

6. Click on the **Products** link in the header or left navigation again.
7. Note the items that appear as **Hot Products**:
   - Superior Hi-Def Plasma Screen TV
   - FSE Plasma Screen TV
   - FSE Triple Play

This is an example of how segments work. The **HotProducts** recommendation is associated with a segment called **AffluentYoungSingles** that includes the visitor John. The first list of Hot Products that was displayed consisted of general recommended items. This recommendation list is shown if a user who does not qualify for the segment is viewing the page. After logging in to FirstSite as the visitor John, the list of Hot Products changed since he qualified for the segment. John has been segmented into this group based on the information he provided in his member profile. If you notice, the items recommended to visitors in the segment are of a higher price range since visitors like John are more likely to purchase those items.

You can use segments to present the certain content to visitors based on similar characteristics among your visitors. If your visitors are able to find the right information

quickly and effortlessly, they are more likely to return to your site and perhaps recommend your site to others as well.

In the next section, we will create a segment to display the recommendation list we created in "Static List Recommendation Asset," on page 32 to FirstSite visitors like Tracy.

Creating Segments

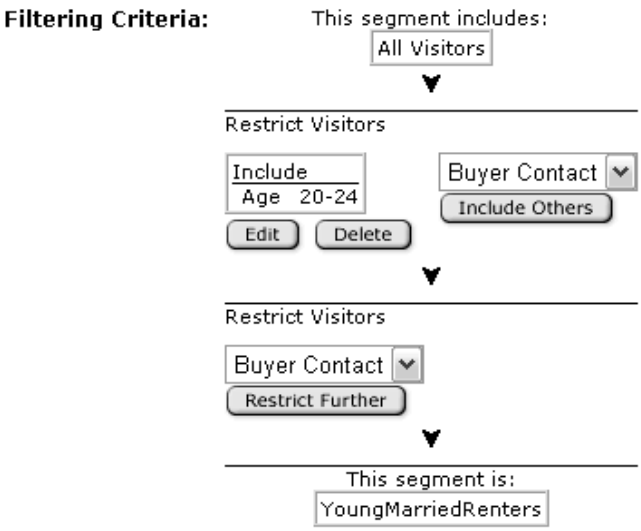
Suppose the FirstSite marketing team wishes to target visitors who are between the ages of 20-29, married, and rent their homes or apartments. The following section will outline how we can accomplish this goal. To create a segment,

1. Click on New.
2. Select **New Segment** from the **Start Menu**.
3. In the "Choose Assignees" screen, select **firstsite**. Click Set Assignees.
4. For "Name", enter `YoungMarriedRenters`.
5. For "Description", enter `Young Married Renters`.
6. Click Continue.
7. In the "Segment Filtering Criteria" screen, we will define the criteria that will qualify or disqualify visitors from our segment.
   a. Select **Profile** from the categories at the top of your workspace.
   b. Select **Age** from the **ProfileCriteria** list.
   c. To set the criterion for Age, select:

   d. When you are finished, click Add This Criterion.
8. You will be brought to the **Edit Segment** screen.

   The **Filtering Criteria** section shows the filtering rules you have created for the segment. From here you can create additional rules to further restrict your segment.



Since we wanted our age range to be 20-29, we will have to define another restriction to include the 25-29 age range.

a. Next to the restriction for the 20-24 age bracket, select **Profile** from the drop down menu and click Include Others.

b. Select **Age** from the **ProfileCriteria** list and select 25-29 for the age bracket:

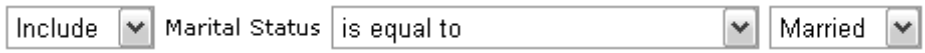
c. When you are finished, click Add This Criterion. You are then brought back to **Edit Segment** screen.

9. For **Filtering Criteria**,

a. In the section beneath the criteria we set for age, select **Profile** from the drop down menu and click on Restrict Further.

b. Select **Marital Status** from the **ProfileCriteria** list.

c. Select the following criteria:



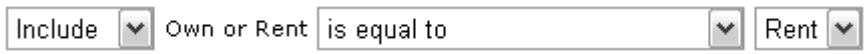
d. When you are finished, click Add This Criterion. You are then brought back to the **Edit Segment** screen again.

10. For **Filtering Criteria**,

a. In the section beneath the criteria we set for marital status, select **Profile** from the drop down menu and click on Restrict Further.

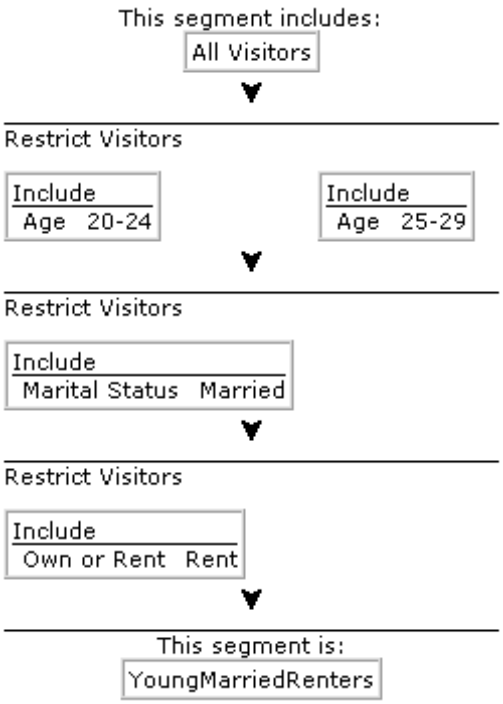
b. Select **Own or Rent** from the **ProfileCriteria** list.

c. Select the following criteria:



d. When you are finished, click Add This Criterion.

11. When you are finished selecting your filtering criteria, click Save. The final segment will appear as follows:



12. Your segment will appear in the **Marketing** tab of the tree panel.

Applying a Recommendation to a Segment

In this section we take the HotProducts recommendation and apply it to the segment we have created above to demonstrate personalization.

1. In your tree, click on the **Marketing** tab.
2. Expand the **Recommendations** item.
3. Double-click on the **HotProducts** recommendation.
4. Click Edit in the action bar.
5. Click Change for **Static Lists**.
6. For "New Static List" select the segment we created in "Segments," on page 51, **YoungMarriedRenters**, from the drop down menu.
7. From the **Products** tab, select the following products (hold down the CTRL button on your keyboard and use the mouse to select multiple items):
   a. AudioCo iAC-083 Portable Audio Player
   b. Superior Slim DVD Player
   c. FSE Triple Play
8. Click the Add Selected Items button for the **YoungMarriedRenters** segment.
9. Click Save.

Now lets preview the site and login as the visitor, Tracy.

1. Click on the **Site Plan** tab in the tree panel,
2. Expand the **Placed Pages** item.
3. Right-click on **FSIIHome** and select **Preview**.
4. On the previewed **FSIIHome** page, click on the **MySite** link in the top navigation, and use the following information to login:

   Username: `tracy`

   Password: `password`
5. After you are brought to Tracy's member profile, click on the **Products** link.
6. On the **Products** page, the **Featured Products** section should display the items we listed above.

Advanced Recommendations

Earlier in the document, you learned about Static List Recommendations. The Static List Recommendation asset type is the only type of recommendation available to you without FatWire Engage. With FatWire Engage, you receive two additional recommendation types, Related Items Recommendations and Dynamic Recommendations.

Related Items Recommendations

A Related Items recommendation is based on relationships that are defined between flex assets. Common relationships between assets are cross-sell and up-sell relationships. For example, if marketers determine that visitors who purchase a television player also purchase a DVD player, a cross-sell related items recommendation is created and rendered pages for a television will list DVD players. After creating a related items recommendation, you then determine the flex assets or flex parent assets to assign the relationship, as well as configure the confidence value that represents the asset's weight or scale in the recommendation.

For information on creating related items recommendations, you can refer to the *Content Server User's Guide*.

Dynamic List Recommendations

A dynamic list recommendation references a element that is coded by a developer to determine the assets to display. For example, a dynamic list recommendation named "New Products" could reference an element that returns only the products added to the site within the last five days. Further discussion on dynamic list recommendations can be found in the *Content Server User's Guide.*

Promotions

A promotion is a merchandising asset that offers a discount value to a visitor based on the products they buy or the segment they belong to. With FatWire Engage, you can create promotions that offer a discount on a single product price, the total shopping cart price, shipping, or combine various discounts.

You can define the following criteria when creating your promotion:

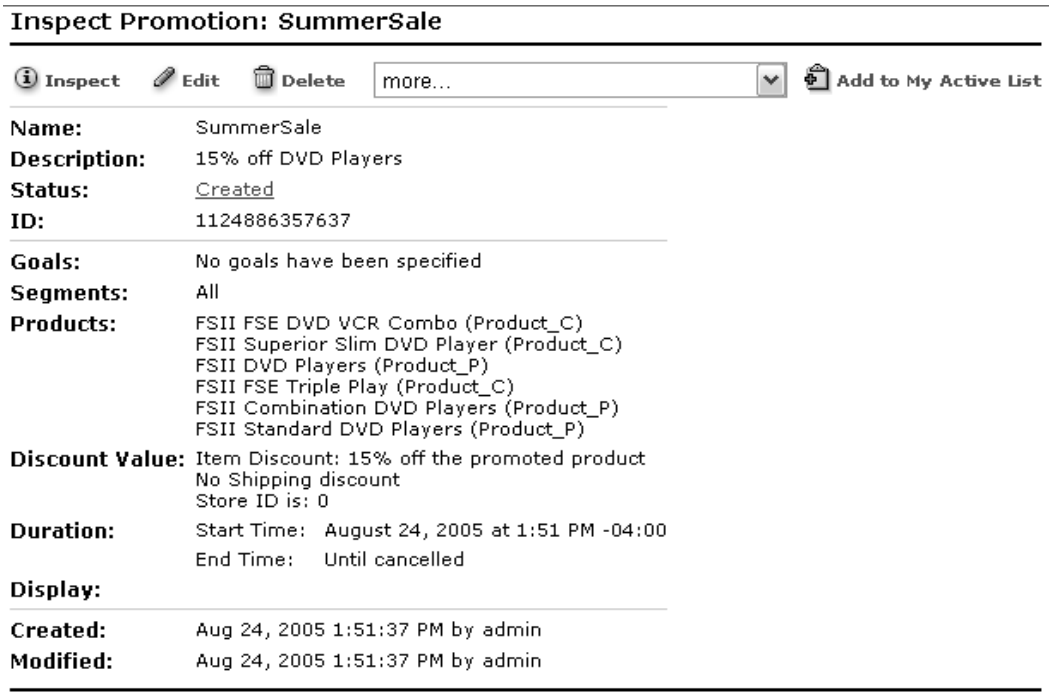
- The products to promote
- The value the customer receives when they purchase a promoted product
- The visitors that qualify for the promotion
- Where the promotion appears on the site
- The duration of the promotion

Promotions use recommendations to deliver promotional content. They are displayed on a site by replacing the recommendations that would normally appear there. When you create your promotion, you select the segment(s) to advertise the promotion to. The promotion will override any recommendation that is applied to the segment(s) you chose. When the recommendation is rendered, FatWire Engage will check to see if there is a promotion associated with it. If one is found, the promotion is rendered using the template applied to the recommendation, as long as you selected the option "Promotions can override this recommendation" when the recommendation was created.

To see an example promotion

1. Log in to the Content Server interface as the user **firstsite** (password: `firstsite`).
2. Click on the **Marketing** tab.
3. Expand the **Promotions** item.
4. Double-click on the **SummerSale** promotion.

Figure 16: SummerSale promotion



This promotion, applied to specific products, has a discount value of 15%. If you previewed any of the products that this promotion applies to, you may notice under “Price Per Unit” it says **Item Discount:** in red text. That is the result of our SummerSale promotion.

The Shopping Cart

An integral component of an e-commerce site is the shopping cart. When a site visitor decides to purchase an item from your site, they place it in their shopping cart and can either process their transaction or continue shopping. With Content Server, you can easily create a shopping cart that fits the needs of your site. You are provided with a set of XML and JSP tags that you can use to collect information on a visitor's purchase and add that information to a cart object.

The shopping cart allows you to perform the following tasks:

- Add and delete cart items in response to customer actions
- Apply existing discounts to cart items and/or the cart total.
- Apply taxes and/or shipping charges.
- Perform currency conversions

If you preview the Superior Slim DVD Player product again and click Add to Cart , you are brought to the shopping cart screen. If you add another item to the cart, it will look similar to Figure 17. The shopping cart will calculate the amount owed, including any discounts to apply based on available promotions a visitor qualifies for.

Figure 17: Preview of the shopping cart after adding two items.

Product	Unit Price	Qty.	Price	Remove
Superior Slim DVD Player	Regular price: $89.99 - Item Discount: $13.50 Your price: $76.49	1.0	$76.49	☐
Innovative Hi-Def Projection TV	$6,999.99	1.0	$6,999.99	☐

Checkout Remove Items

For information on how to implement the shopping cart and how orders are validated and processed, please refer to the *Content Server Developer's Guide.*

Part 2

Reusing FirstSite Components

This part of the document goes into further detail on the FirstSite framework and includes topics such as data modeling, page design, and template creation, and requires technical knowledge of JSP, XHTML, XML or other internet technologies. It includes the following chapters:

- Chapter 6, “Data Modeling”
- Chapter 7, “Page Design and Development”

Chapter 6

Data Modeling

In Part 1 of this guide, we mentioned that there were two asset data models in Content Server. In this chapter, we discuss the differences between the two models and examine the model used in designing FirstSite.

This chapter includes the following sections:

- Assets Types and Asset Models
- Creating a Flex Family
- FirstSite Flex Families

Assets Types and Asset Models

Developers design and create asset types while designing the content management system and online sites. Content providers then create and edit assets of those types. In general, assets perform one of the following three roles:

- Provide content that visitors read and examine on your online sites
- Provide the formatting logic or code for displaying the content
- Provide data structure for storing the content in the Content Server database

The developer's job is to design asset types that are easy for content providers to work with on the management system and that can be delivered efficiently to visitors from the delivery system.

Content Server provides your with two data models for the asset types that you design: **basic** and **flex**.

The Basic Asset Data Model

In general, the data model for basic asset types is one database table per asset type. All basic assets of the same type have the exact same fields (properties) and all assets of a single type are stored in the same database table. When the data for an asset type can be imagined as a spreadsheet, as a simple flat table where each asset of that type is a single record and every record has the same columns, that asset type should use the basic asset model.

The Flex Asset Data Model

In the **Flex Asset Data Model**, flex assets have a supporting family of assets that give them structure. Flex assets can be represented in a hierarchy in which assets inherit attribute values from parent assets. This means that assets of the same type can have different attributes, depending on their heritage. The flex asset itself is the unit of content that you make available to visitors to your site, but the rest of the flex asset family defines its makeup. The flex asset family structure is stored in the Content Server database across several tables where one field in one table will reference row in a different table. FirstSite was designed using the Flex Asset Data Model. The FirstSite Store Schema component has three flex asset families that define the structure of the content, product, and media assets called *Content*, *Product*, and *Media*, respectively.

The Flex Family

A flex family consists of six asset types, five of which must exist for flex assets to function:

- Flex attribute
- Flex parent definition
- Flex asset definition
- Flex parent
- Flex asset
- Flex filters (optional)

While the flex asset is the key, the **flex attributes** are the foundation of the flex asset model. An attribute is a piece of information that represents values such as color, height, author, headline. A flex asset and a flex parent comprises a specific set of attributes. Flex assets inherit attribute values from their parents who inherit attribute values from their parents and so on.

You define the set of attributes that comprise flex assets and flex asset parent by creating "definitions," using the **flex asset definition** and **flex parent definition** asset types. Flex assets can have more than one flex parent. The make up of a flex asset is dependent upon the asset's parents.

In summary, the flex assets and flex asset parents of a flex family are the only visible components to visitors of your site. The other members of a flex family provide data structure for the flex asset. However, because all of the members in the family are assets, you can take advantage of the standard CS features like revision tracking, workflow, search, and so on.

Reusing FirstSite Flex Families

If you can base your data structure on either of the sample flex asset families available to you with the FirstSite Store Schema module, you do not have to create an entire flex family--you can create only the new members that you need. Refer to the *Content Server Developer's Guide* for information on how to edit existing flex families. The next section outlines the procedure for creating new flex families.

Creating a Flex Family

In this section, you will create a new flex family in Content Server for FirstSite Store Locations. The following briefly outlines the steps for creating a new flex family. Later, we will discuss in further detail how each step is accomplished.

To create a new flex family

Step 1: Design Your Flex Family

Determine the structure of your flex family, including what attributes you will need to create.

Step 2: Create a New Flex Family with the Flex Family Maker

The **Flex Family Maker Utility** is located in in the **Admin** tab of the CS standard interface. This utility creates several new database tables and prepares the family for use.

Step 3: Configure the CS Interface for Your New Asset Types

Here, you enable the new asset types for your Content Server site on the development system, create start menu items for your new asset types, place your new flex attributes, flex parent definition, and flex asset definition in the **Design** tab, and create a new tab for your flex parent and flex asset types.

Step 4: Create Your Flex Attributes

There are different types of attributes that you can create. The most common are attributes of type String, blob, or asset. You can also determine how information is entered for these attributes on content-entry forms by using attribute editors. Attribute editors are discussed in Chapter 4, "Alternative Ways to Add Content."

Step 5: Create Flex Filters (Optional)

Flex filters allow you to derive new attribute values based on other information.

Step 6: Create the Flex Parent Definition

Specify which attributes will define the flex parent assets.

Step 7: Create Flex Asset Definition

Specify which attributes will define the flex asset type and its parent(s).

For more detailed information, you can refer to the *Content Server Developer's Guide*.

Step 1: Design Your Flex Family

First, determine what attributes are needed for the flex asset type and its parent(s). Each store location will be represented by a single flex asset. The information we need regarding each store location is listed similar to the following:

- Store Name
- Street Address
- State
- ZipCode
- Phone Number
- Store Hours

Each of the items listed above will be the attributes for our flex asset. In this example, we will create a flex parent for every state. This flex parent will consist of a single attribute, State. Our flex family hierarchy will look as follows:

- New York
  - Store 001
  - Store 002
  - Store 003
- Pennsylvania
  - Store 004
  - Store 005

Step 2: Create a New Flex Family with the Flex Family Maker

To create a new Flex Family

1. In the standard interface, click the **Admin** tab.
2. Expand the **Flex Family Maker** item.
3. Double-click on **Add New Family**. The "Add New Flex Family" form will load in your workspace.
4. In the "Add New Flex Family" form, enter names for the following fields:
   a. For "Flex Attribute"**,** enter `StoreLocation_A`
   b. For "Flex Parent Definition", enter `StoreLocation_PD`
   c. For "Flex Definition", enter `StoreLocation_CD`

d. For “Flex Parent”, enter `StoreLocation_P`

e. For “Flex Asset**”,** enter `StoreLocation_C`

f. For “Flex Filter”, enter `StoreLocation_F`

g. Click Continue.

5. In the next screen of the “Add New Flex Family” form, enter the following information according to the table below:

Form Field	Description	Plural Form
Flex Attribute	`Store Attribute`	`Store Attributes`
Flex Parent Definition	`Store Parent Definition`	`Store Parent Definitions`
Flex Definition	`Store Definition`	`Store Definitions`
Flex Parent	`Store Parent`	`Store Parents`
Flex Asset	`Store`	`Stores`
Flex Filter	`Store Filter`	`Store Filters`

6. Click Add New Flex Family.

Content Server will then create your flex family, creating the necessary database tables and references.

Step 3: Configure the CS Interface for Your New Asset Types

To enable assets for FirstSite

1. In the standard interface, click on the **Admin** tab.

2. Expand the **Sites** item.

3. Expand the **FirstSiteII** item.

4. Expand the **Asset Types** item.

5. Double-click on Enable. The “Enable Asset Types: FirstSite II” will load in your workspace.

6. Select all the **StoreLocation_** asset types that we created in Step 2.

7. Click Enable Asset Types. The next screen will ask which start menu items you would like to create for the assets you are enabling for the site. Check all the items to create start menu items for each of the assets being enabled.

8. Click Enable Asset Types.

To create a new tab (Optional)

1. In the **Admin** tab, double-click on Tree.

2. Click Add New Tree Tab.

3. In the “Add New Tree Tab” form,

a. For “Title”, enter `Store Locations`.

b. For “Tooltip”, enter `Store Locations`.

c. Select **FirstSite II** from the list of "Sites."

d. For "Required Roles," select **Any**.

e. For "Tab Contents," select **Store Parent** and **Store** from the list. Click Add Selected Items to move them to the Selected list. Use the arrow buttons on the right-hand side to reorder the items so that Store Parent is listed first.

4. Click Save.

To add items to the Design Tab (Optional)

1. In the **Admin** tab, double-click on Tree.
2. Click the for **Design** to edit it.
3. For "Tab Contents," select **Store Attribute**, **Store Parent Definition**, **Store Definition**, and **Store Filter** and click **Add Selected Items** to move them to the Selected list. Reorder this group of items so that they are in the order of: Store Attribute, Store Parent Definition, Store Definition, and Store Filter.
4. Click Save.

Step 4: Create Your Flex Attributes

To create a flex attribute

1. Click **New** in the button bar.
2. Select **New Store Attribute**.
3. In the "New Store Attribute" creation form,
   a. For "Name," enter `StoreName`.
   b. For "Description," enter `Store Name`.
   c. For "Value Type," select **string**.
   d. Click Save.
4. Repeat Steps 1-3 for the remaining attributes:
   - StreetAddress
   - State
   - ZipCode
   - PhoneNumber
   - StoreHours

When you are finished creating the attributes, the Store Attributes item in your **Design** tab will appear as follows:

Step 5: Create Flex Filters (Optional)

A flex filter is an asset that processes data contained in a flex asset and sets resultant data as attributes of the original asset. Flex Filters can be used to simplify the creation of flex assets. For example, instead of having to provide the width, height, and thumbnail file for a media asset, you can use a filter to extract that information when the media file is uploaded to Content Server.

A filter consists of a reference to a Java class that implements the IFilter interface, as well as arguments necessary for the filter to function. FirstSite comes with filters including *FieldCopier, ThumbnailExtractor, DocType* and *DocumentTransformation.*

- **DocumentTransformation** takes any document asset over 200 pages and converts it to HTML, storing the HTML file as an attribute of the original document asset.
- **DocType** extracts the file type and MIME type of a document and stores this information as attributes of the original document asset.
- **ThumbnailExtractor** takes images uploaded to Content Server and creates thumbnails (a smaller version of the original), storing the thumbnail file and the thumbnail's width and height as attributes of the original Media asset.
- **FieldCopier** is used to copy one or more of the 13 basic asset field values (or parent asset field values) to fields that are accessible using the standard flex asset API. Otherwise, these attributes can only be accessed by loading the flex asset, which is an operation that is not recommended for an online site.

Note

Frequent database access can severely hinder the performance of your site. In order to ensure that your site remains a high-performance, scaleable application, it is recommended that you use FlexFilters like the FieldCopier when needed.

To create a new flex filter

1. In your CS interface, click **New** in the button bar.
2. From the **Start Menu**, select **New Store Filter**.
3. In the "New Store Filter" creation form,
   a. For "Name" and "Description," enter `StateFieldCopier`
   b. From the "Filter" menu, select **FieldCopier**.
   c. Click 

   d. In the **Arguments** section, from the "Name" menu, select **name**.
   e. For "Value," enter `State`.
   f. Click 
.
4. Click 
.

For more information on Flex assets and creating Flex Filters, refer to the *Content Server Developer's Guide*.

Step 6: Create the Flex Parent Definition

To create a flex parent definition

1. In your CS interface, click  in the button bar.
2. From the **Start Menu**, select **New Store Parent Definition**.
3. In the "New Store Parent Definition" creation form, enter `State` for the "Name" and "Description" fields.
4. In the "Filters" menu, select **StateCopier**. Click **Select** to move it to the Selected list.
5. Click  .

Step 7: Create Flex Asset Definition

To create a flex asset definition

1. In your CS interface, click  in the button bar.
2. From the **Start Menu**, select **New Store Definition**.
3. In the "New Store Definition" creation form,
   a. For the "Name" and "Description" fields, enter `Store`
   b. For "Store Parent Definitions," select **State** and click **Single Value Required**. This will move State to the Selected list.
   c. For **Attributes**, select **PhoneNumber**, **StoreHours**, **StreetAddress**, and **ZipCode** and click **Required**. This will move the attributes to the Selected list. (The two attributes not selected will be used in creating our Store Filters later in the chapter.)
   d. Use the **Display Order** arrows to reorder the Selected list as follows: StreetAddress, ZipCode, PhoneNumber, StoreHours.
   e. In the **Filters** menu, select **StoreNameCopier**. Click **Select** to move it to the Selected list.
4. Click  .

FirstSite Flex Families

FirstSite was designed using the Flex Asset Data Model. Definition asset types, combined with the inheritance of attribute values, enable you to implement extensive categorical hierarchies to better organize your data. The following table lists the FirstSite flex families available to you if you have the following FirstSite modules installed:

- FirstSite Store Schema
- FirstSite Document Schema
- FirstSite Local User Demo

Table 2: FirstSite Flex Families

Flex Family	Description	Module
Content	Represents assets that contain one or more paragraphs of text.	FirstSite Store Schema
Product	Represents products, product categories, and manufacturers. The Product Flex Family is an excellent example of flex asset hierarchy and attribute inheritance.	FirstSite Store Schema
Media	Used to organize media files such as photos. Photos are used for articles and product images.	FirstSite Store Schema
Document	Used to organize assets representing any binary file type such as Microsoft Word documents, spreadsheet files, PDF files, and text files. The document flex parents represent folders. This folder-like hierarchal structure is more apparent when using FatWire DocLink.	Document Schema
Site Visitor	Used to store information about registered visitors to your site.	Local User Add-on

Chapter 7

Page Design and Development

With Content Server, you can easily manage dynamic websites. This chapter introduces the concept of modular page design and how it is used in developing dynamic FirstSite pages.

This chapter contains the following sections:

- Modular Page Design
- FirstSite Templates and CSElements
- Creating Templates
- URL Assemblers

Modular Page Design

Pages are designed using templates, which you can resuse and modify as necessary to render content on your own sites.

FatWire recommends that you design your web pages using a modular page design strategy, where a web page that a website visitor sees is made up of multiple independent pagelets. Modular page design has several benefits:

- It segregates code, which simplifies maintenance.
- It allows you to develop an efficient caching strategy by reusing pagelets and caching others, resulting in better performance.
- It allows you to code common design elements, like navigation for example, one time and use them on multiple pages.

Layout A in Figure 18 depicts a simplified modular page. Each rectangle in Layout A represents a pagelet. A pagelet is the generated output of one or more elements. These pagelets are requested by a containing page, which we refer to as the "Layout Template." The Layout Template determines how the pagelets will appear on the finished page and generates all surrounding navigation, represented by Pagelet A and Pagelet B, for the rendered page.

The modular page design strategy is utilized in FirstSite. Figure 18 shows the graphical representation of how each pagelet is laid out on a rendered FirstSite page (Layout B). Notice the similarities in the positioning of pagelets between Layouts A and B. The FirstSite top navigation and side navigation components in Layout B represent Pagelet A and Pagelet B from Layout A respectively. The main content portion of the rendered FirstSite page (represented by Pagelet C in Layout A) is content unique to that page.

Figure 18: Graphical representation of pagelets rendered on a FirstSite Page.

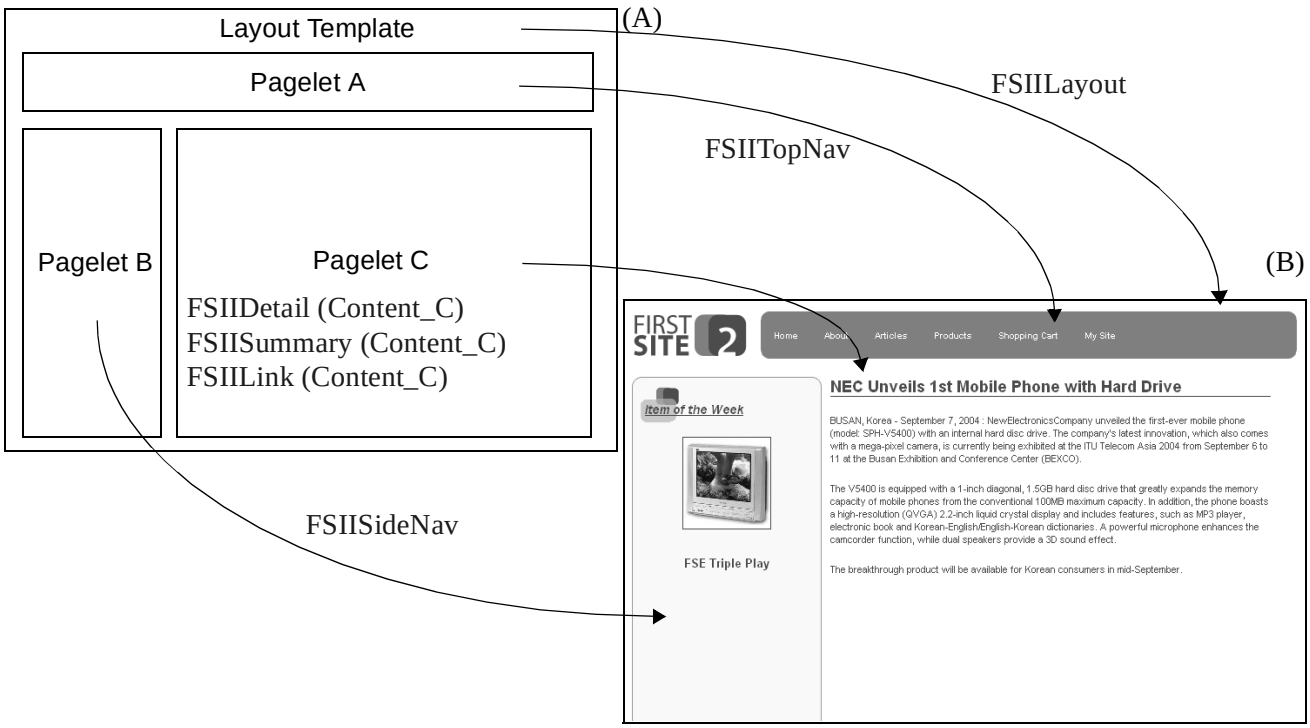
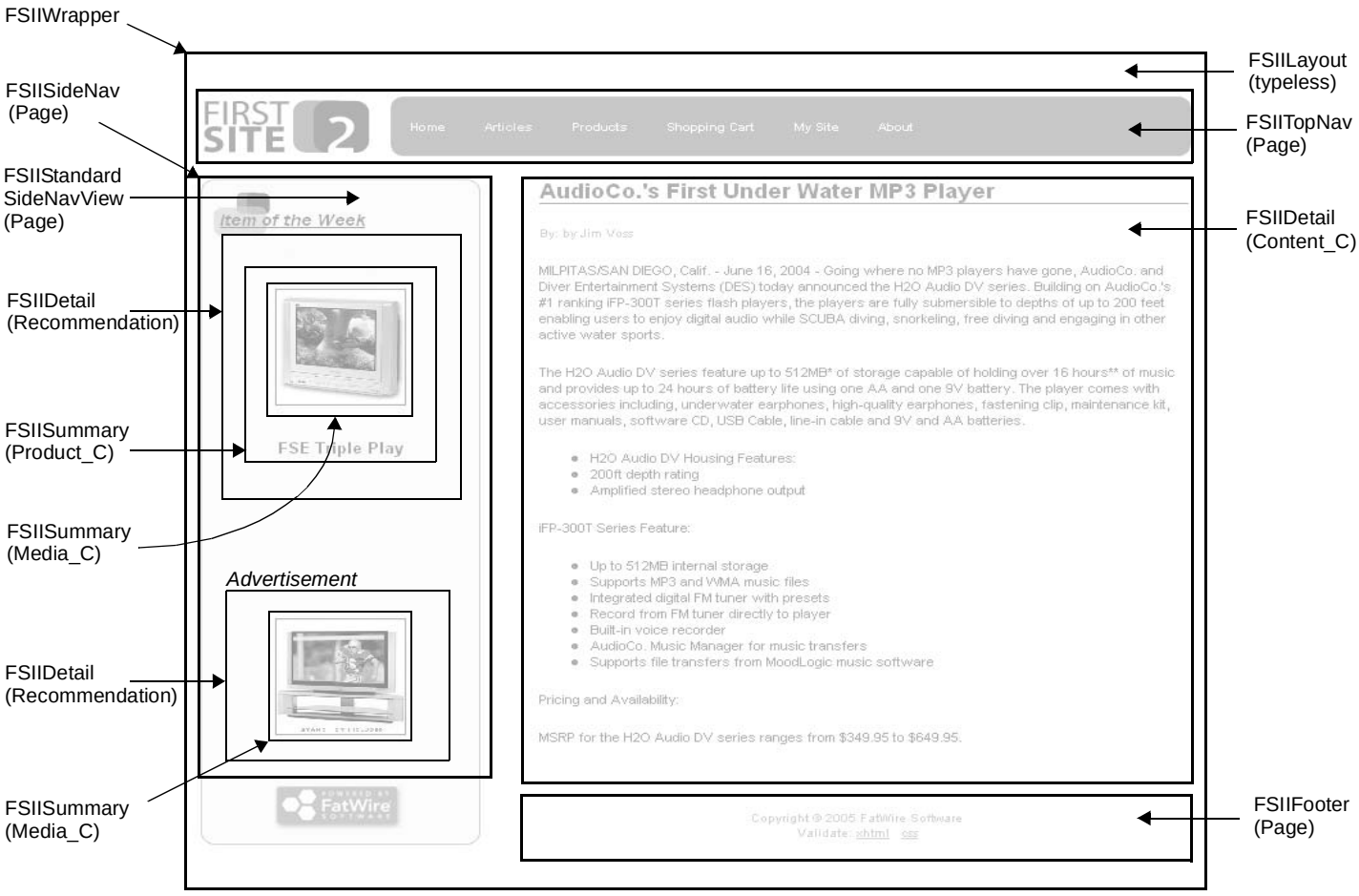


Figure 19 shows a more advanced view of a rendered FirstSite article and represents the call structure of the templates used. The **FSIIWrapper** is the outermost template. The purpose of this wrapper is to perform logic that must always be performed on Content Server, even in a fully cached environment. Such logic includes security checking, search pre-processing, and session management. The **FSIILayout** template is a typeless template that determines the overall structure of the rendered page. It loads the stylesheet and calls to other components that render the surrounding navigation (**FSIITopNav**, **FSIISideNav**, and **FSIIFooter**), content (**FSIIDetail**). Since the subtype of the parent page is *Standard*, the FSIISideNav template dispatches the **FSIIStandardSideNavView** template. The same can be said for FSIIDetail template, which dispatches to FSIIStandard DetailView (not shown).

Within the FSIIStandardSideNavView, two recommendations are rendered, *Item of the Week* and *Advertisement*. The Item of the Week recommendation contains a product, while the Advertisement recommendation contains an image. The product in the Item of the Week recommendation is rendered in summary (only the product name and its image are rendered) using the **FSIISummary (Product_C)** template. In order to render the product image, the **FSIISummary (Media_C)** template is used. For the Advertisement recommendation, the only FSIISummary (Media_C) template is used.

The templates mentioned here are discussed in further detail in upcoming sections.

Figure 19: Advanced layout view depicting FirstSite template calls.



FirstSite Templates and CSElements

In this section, you will learn about FirstSite templates and elements, and how articles are rendered.

Viewing and Copying Templates

For a better understanding of the templates discussed in this section, it is strongly recommended that you follow along with the source code. For your benefit, the source code is heavily commented with detailed explanations for what is happening in the code.

You have several ways to view (and copy) the source code:

- **In the Content Server interface**
  Click on the **Design** tab and expand the **Template** or **CSElement** items
- **Using a text editor of your choice**
  You are not limited to the tools provided by Content Server. You can use a text editor of your choice to open and work with source files.

FatWire recommends that you keep the *Developer's Tag Reference* on hand as well, since our discussion on templates does not go into great detail on how each tag is used.

Template Coding Conventions

To help you understand the FirstSite template source code, the following coding conventions were used:

- **Language**

  The entire set of FirstSite templates is written in JSP.

- **Comments**

  Each significant block of code in a FirstSite template is commented in detail. Comments are enclosed in `<%--` , `--%>` brackets. HTML style commenting (comments enclosed by `<!--` and `-->` brackets) are not appropriate for this purpose because they are streamed to the client. Comments include a hard line break when an 80 character limit is reached. Subsequent lines are indented using spaces until the first character of the next line is at the left edge of the page unless a whitespace streaming reason precludes it.

- **White space issues**

  When Content Server streams a text page, the page may contain a significant amount of "white space"—spaces, carriage returns and tabs—that have no effect on the data that is consumed by the client. The white space is visible when the source code is viewed by the consumer. Excessive white space needlessly increases the size of the response, which ultimately increases bandwidth use. Consequently, it is wise to reduce excessive white space.

  The JSP specification requires that all white space be preserved. Thus a page that looks like this:

```
<%@ page import="my class name"%>
<%@ page import="my class 2"%>
<cs:ftcs>
<p>Hello world!</p>
</cs:ftcs>
```

would have three carriage returns preceeding the <p> because the text is displayed on the third line after the JSP has been interpreted. With more complicated pages, the problems is compounded.

To avoid the issue, consider altering your JSP pages as follows:

```
<%@ page import="my class name"
%><%@ page import="my class 2"
%><cs:ftcs><p>Hello world!</p></cs:ftcs>
```

While this is not as elegant, it will result in page output without any white space prior to the <p> tag. You should use this technique judiciously. Readability is an important factor to consider for your templates. For more information on white space issues, see Appendix C of the *Content Server Developer's Guide*.

- **Tag Attributes**

  Tags with multiple attributes are written on a single line. Soft-wrapping will handle wrapping the tag attributes. This allows developers to easily inspect unwrapped code to determine the flow of control. While this makes it hard to read all attributes without turning on wrapping, the benefit of being able to follow flow of control for novices is more important in FirstSite.

- **Replication**

  In order to be completely replicable, FirstSite templates and elements avoid:

  - Referring to any assets directly by name or id.
  - Referring to attributes directly by name or id.
  - Calling templates directly by name or id.
  - Referring to the site directly by name or id.

Structure vs. Content Templates

FirstSite templates can be thought of as either "structure" or "content" templates.

- **Structure templates** - Render the structure of a page, which include the layout and surrounding navigation. The following are considered structure templates:
  - Typeless Templates
  - Page templates (all templates listed in Page table)
- **Content templates** - All other templates are considered "content" templates. There are five standard names for content templates that need to be created in order for content to be rendered:
  - Detail
  - Head
  - SideNav
  - Summary
  - Link

Structure Templates

In this section, we focus on the structure templates that generate the layout and navigation for rendered FirstSite pages.

Typeless Templates

Typeless templates are used to provide the same infrastructure for rendering an asset, regardless of the underlying type of the asset. For example, a typeless template can be used to draw the navigation part of the page, while relying on a typed template to do the actual rendering of the detail part of the page. By convention, typless template names always begin with a slash ("/").

The following is a list of typeless templates available with the FirstSite Core module:

Table 3: FirstSite Core Typeless Templates

Template Name	Purpose
/FSIILayout	Generates the overall layout of the page, making the appropriate calls to other elements to be rendered on the page
/FSIIDetail	Calls the Detail template directly, with no surrounding layout. This template is used primarily for preview and debugging.
/FSIISummary	Calls the Summary template for a specified asset. This template is used primarily for preview and debugging.
/FSIILink	Calls the Link template for a specified asset. This template is used primarily for preview and debugging.
/FSIISideNav	Calls the SideNav template for a specified asset. This template is used primarily for preview and debugging.

For reference, these templates can be found under **Tables** > **Element Catalog** > /.

FSIILayout

The FSIILayout template is a structure template that determines the order of information presented on a rendered page.

To view the FSIILayout source code

1. Log in to the Content Server standard interface.
2. In the left panel, click on the **Design** tab.
3. Expand the **Template** item.
4. Double-click on **FSIILayout** to open it in your workspace. Observe the "Element Logic" section.

> **Note**
>
> This template is located under **Tables** > **Element Catalog** > /

Standard to all JSP templates and CSElements are the first few lines of code where Content Server tag libraries are imported. These libraries must be imported in order for you to use the tags you need. This template uses the tag libraries `cs`, `ics`, `render`, and `satellite`. Always remember to import additional tag libraries as the need arises.

The next tag, `<cs:ftcs>`, tells Content Server to begin interpreting the code between itself and its closing tag, `</cs:ftcs>`, which can be found at the very end of the element code. These tags are automatically generated upon creation of the element in the standard interface. Failure to include this pair of tags will result in a compilation error.

The first thing any element needs to do is to record itself as a compositional dependency on the rendered page. This will help the cache management module remove this page from cache when the template asset is saved:

```
<ics:if condition='<%=ics.GetVar("tid")!=null%>'>
  <ics:then>
    <render:logdep cid='<%=ics.GetVar("tid")%>' c="Template" />
  </ics:then>
</ics:if>
```

Within the <head> and </head> tags, our stylesheets and any head content that is asset specific is loaded. The `render:lookup` tag is used to determine the correct name of the **FSIIHead** template to call for a given asset. Template mapping is discussed later in the chapter. After the correct template name is retrieved, the `render:calltemplate` tag is used to render the template. After displaying the head content, we then display the body of the page.

Best Practice: divs vs tables

`div` tags are used throughout this template. Each `div` is named, so that the correct CSS styles are applied to the applicable sections. As per W3C guidelines, tables are only used to present tabular data, not for design or layout.

Next, we render the top navigation. The top navigation is a global navigation bar, so there are not many different variations of it. In fact, the only thing that varies from page to page for the top navigation is which entry is highlighted to indicate what page the user is currently viewing. This is done in a separate element.

As we did before, we use the `render:lookup` tag to determine the proper template to render the top navigation bar. Initially, there is only one, but in the event that you replicate your site, more will be created. After retrieving the proper template, the `render:calltemplate` tag is used to render the template.

```
<render:lookup site='<%=ics.GetVar("site")%>' varname="TopNavVar"
  key="TopNav" tid='<%=ics.GetVar("tid")%>'/>
<render:calltemplate tname='<%=ics.GetVar("TopNavVar")%>'
  site='<%=ics.GetVar("site")%>' tid='<%=ics.GetVar("tid")%>'
  slotname="TopNav" c='Page' cid='<%=ics.GetVar("p")%>'
  ttype="Template" />
```

Using the same principles, the side navigation, the main content, and the footer are rendered. The side navigation section displays section-specific content such as product categories for product pages, recommendations for generic pages, and so on. Since the Page asset's SideNav and Detail templates dispatch to cached views, we call the Detail template as an element instead of a pagelet. This increases performance by reducing a round-trip between Satellite Server and Content Server.

FSIITopNav

Since the top navigation is an element reused across multiple rendered pages, the **FSIITopNav (Page)** template contains a element call to the **FSIICommon/Nav/TopNav** CSElement. In this section our focus is on the TopNav CSElement.

> **Note**
>
> For information regarding why the TopNav template calls an element, refer to the source code for FSIILayout and FSIITopNav which can be found under **Tables** > **Element Catalog** > **/** and **Tables** > **Element Catalog** > **Page** respectively.

The TopNav CSElement renders the top navigation bar on rendered FirstSite pages. Navigation is based on the Site Plan for your Content Server site.

To examine this CSElement in the standard interface

1. In the **Design** tab of your Content Server standard interface, expand the **CSElement** item.
2. Double-click on the **FSIICommon/Nav/TopNav** element to open it in your workspace.

After importing all required Content Server tag libraries and logging dependencies (see previous section entitled "FSIILayout"), the site name is displayed:

```
<div id="SiteName">
  <h1>
    <ics:getvar name="site"/>
  </h1>
</div>
```

Next, the Site Plan tree is loaded, based on the **Home** page node. First, the `render:lookup` tag is used to retrieve the name of the Home page node.

```
<render:lookup site='<%=ics.GetVar("site")%>'
  varname="HomePageName" key="HomePage" ttype="CSElement"
  tid='<%=ics.GetVar("eid")%>' match=":x"/>
```

After the name is retrieved, it is used to locate and load the proper node into memory.

```
<asset:load name="HomePage" type="Page" field="name"
  value='<%=ics.GetVar("HomePageName")%>'/>
```

The Home page node ID is extracted from the node loaded in memory, which is then used to create a link in the navigation bar. To render the link, the `render:calltemplate` tag is used to load the **FSIILink** template for page assets.

After the Home page link is rendered, the Home page node is used to retrieve the child nodes so that links to these pages can be created in the navigation bar as well. First, the site plan is loaded into memory using the Home page node.

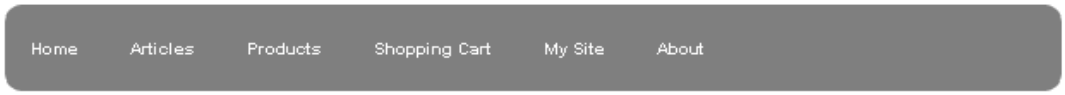
```
<siteplan:load name='HomeNode' nodeid='<%=ics.GetVar(
  "HomePageNodeId" )%>' />
```

Next, the immediate children of the parent node are placed into a list.

```
<siteplan:listpages name='HomeNode' placedlist='SectionPageList'
  level='1' />
```

The list is then looped through and links to each of the pages are created by using logic similar to what was used above.

Figure 20: FirstSite's top navigation



Content Templates

In this section, we discuss how content templates render articles. Content templates also exist for other FirstSite asset types but will not be discussed here. You may wish to examine the other content templates in order to learn how different asset types are rendered. In order to study these templates, you must have the FirstSite Store Schema component installed.

The Content_C asset type represents articles in FirstSite. Among the attributes that make up FirstSite articles are: headline, subheadline, abstract, byline, postdate, and article body. In this section we examine how Content_C assets are rendered in detail (FSIIDetail (Content_C), FSIIHead (Content_C), and FSIISideNav (Content_C)), in summary (FSIISummary (Content_C)), and as a link (FSIILink (Content_C)).

FSIIDetail (Content_C)

This template will render an article in its full form, which includes the headline, subheadline, byline, and article body attributes.

To examine the FSIIDetail template for Content_C assets

1. Click on the **Design** tab in the Content Server standard interface.
2. Expand the **Template** item, and double-click on **FSIIDetail (Content_C)** to open the file in your workspace.

Note

The FSIIDetail template for Content assets can be found under "**Tables** > **Element Catalog** > **Content_C"**

After all the required Content Server tag libraries are imported and dependencies are logged, an assetset is created.

```
<assetset:setasset name="ArticleSet" type='<%=ics.GetVar("c")%>'
  id='<%=ics.GetVar("cid")%>' />
```

Next, we retrieve the attributes that belong to that set.

All attributes can be retrieved using `assetset:getattributevalue(s)`, but if we know in advance that we are going to retrieve more than one attribute then we should instead use the `assetset:getmultiplevalues` tag because it is much more efficient than making multiple calls to `assetset:getattributevalue(s)`. In particular, there is significantly less database access with a single `assetset:getmultiplevalues` call.

Text and blob attributes, however, cannot be retrieved using the `assetset:getmultiplevalues` tag, and they must be retrieved individually. Attribute names can change when a site is replicated, so we have to look up the actual attribute

name in a database table maintained by the template asset. We assigned an arbitrary key to the name of the attribute. The `render:lookup` tag is used to look up asset references.

After all attribute values have been attained, the asset is rendered. First, the headline attribute is rendered. The presence of `insite:edit` tags mean that insite editing has been enabled for this asset and this asset can be edited directly through the **Preview** view in Content Server:

```
<h3>
  <ics:listget
    listname='<%="art:"+ics.GetVar("HeadlineAttrName")%>'
    fieldname="value" output="Headline" />
  <insite:edit assetid='<%=ics.GetVar("cid")%>'
    assetfield='<%="Attribute_"+ics.GetVar("HeadlineAttrName")%>'
    assetfieldvalue='<%=ics.GetVar("Headline")%>'
    assettype='<%=ics.GetVar("c")%>' />
</h3>
```

Since the subheadline and byline are optional attributes, we first check if the values exist for the asset before rendering. Finally, the body attribute is rendered:

```
<div id="body">
   <insite:edit
      assetid='<%=ics.GetVar("cid")%>'
      assetfield='<%="Attribute_"+ics.GetVar("BodyAttrName")%>'
      assetfieldvalue='<%=ics.GetVar("Body")%>'
      assettype='<%=ics.GetVar("c")%>' />
</div>
```

Click on the **Content** tab of the Content Server standard interface and preview one of the articles listed there. The FSIIDetail template renders the article as shown in Figure 19, on page 73.

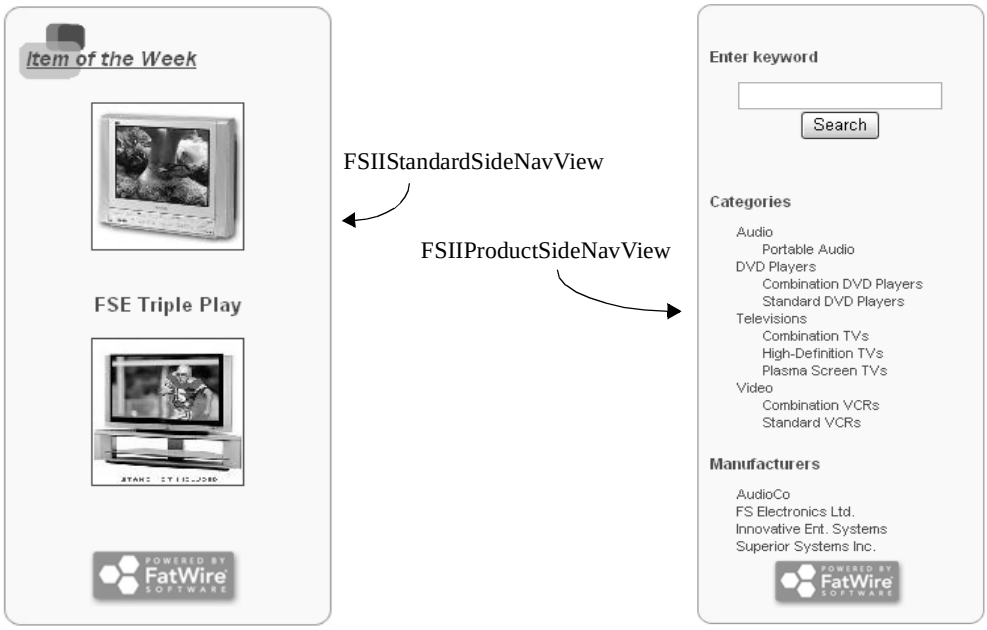
FSIIHead (Content_C)

The FSII Head template contains asset-specific information that should be rendered between the `<head>` and `</head>` tags of a rendered page. For example, the **FSIIHead (Content_C)** template contains the `<title>` tag, which sets the title of the article in the title bar of the internet browser window. The FSIIHead template is necessary for any asset that you create Detail templates for.

FSIISideNav (Content_C)

The content displayed by the FSIISideNav template depends on the subtype of the parent page being rendered. If the subtype is *Product*, the Product view is rendered. The Standard view is rendered in all other cases. In this example, FSIISideNav dispatches to the Page template, **FSIIStandardSideNavView**. This view displays two recommendations, *Item of the Week* and *Advertisements*. The FSIIStandardSideNavView template is located under **Template** in the **Design** tab of the standard interface, or under **Tables** > **ElementCatalog** > **Page**.

Figure 21: Standard and Product SideNavs



A Note on FSII SideNav for Products

If the asset is accessed from a page whose subtype is *Product*, then the content in the side navigation consists all product categories and subcategories. The Product Catalog side navigation can be seen in Figure 21.

This SideNav is a section-specific navigation bar that is rendered on all product pages. The FSIISideNav templates for pages with the subtype *Product*, dispatches the page template, **FSIIProduct SideNavView**. This page template then dispatches the element, **FSIICommon/SideNav/ProductView**. For information on why an element is used, refer to the source code.

Note

To view the source for the FSII Product SideNavView page template, look under **Tables > Element Catalog > Page**.

The ProductView element can be found under **Tables** > **Element Catalog** > **FSIICommon > SideNav**.

To view the ProductView element source code in the CS interface

1. Click on the **Design** tab in the Content Server interface.
2. Expand the **CSElement** item and double-click on **FSIICommon/SideNav/ ProductView** to load it in your workspace.

A searchstate is a set of constraints applied to an assetset based on the attribute values you specify. Searchstates are used to narrow or broaden a search for assets to filter out assets that do not meet your search criteria. In this element, three searchstates are created to return all product categories with children.

The first searchstate, `CategorySearch`, the top-level categories such as Audio, DVD Players, Televisions, and Video. The resultset returned by this searchstate is placed in a list. The list is then looped through in order to retrieve the subcategories of each top-level category. This includes: Portable Audio (Audio), Combination DVD Players (DVD Players), Standard DVD Players (DVD Players), Combination TVs (Televisions), High-Definition TVs (Televisions), Plasma Screen TVs (Televisions), Combination VCRs (Video), and Standard VCRs (Video). In order to retrieve these subcategories, a second searchstate, `SubCategoryIDSearch` is created. The third searchstate retrieves the existing product manufacturers including: AudioCo, FS Electronics Ltd., Innovative Entertainment Systems, and Superior Systems Inc.

This element is fully commented and explains in further detail how the searchstate tags are used. For more information, refer to the source code for this element.

FSIISummary (Content_C)

This template renders an article asset in summary, displaying the headline (as a link), and abstract attributes.

To view the FSIISummary (Content_C) template,

1. Click on the **Design** tab in the Content Server standard interface.
2. Expand the **Template** item, and double-click on **FSIISummary (Content_C)** to open the file in your workspace.

Note

The FSIISummary template for Content assets can be found under **Tables** > **Element Catalog** > **Content_C.**

The FSIISummary template source code is similar to the FSIIDetail template. When the headline is rendered, it is rendered as a link to the asset's Detail view.

URLs are generated using the `render:gettemplateurl` tag. The tag needs a variety of parameters but basically it needs to know the layout page name and the uncached wrapper's page name, plus information about the asset to be rendered by the link.

```
<render:gettemplateurl outstr="aUrl" tid='<%=ics.GetVar("tid")%>'
  slotname="Content_CLinkOnSummaryPage"
  site='<%=ics.GetVar("site")%>' c='<%=ics.GetVar("c")%>’
  cid='<%=ics.GetVar("cid")%>'
  tname='<%=ics.GetVar("LayoutVar")%>'
  wrapperpage='<%=ics.GetVar("WrapperVar")%>' >
    <render:argument name="p" value='<%=ics.GetVar("p")%>'/
</render:gettemplateurl>
```

An alternative to creating the link inline here would be to call the **Link** template. Tradeoffs to consider: Using the link template reuses code. Using an inline link simplifies the control flow.

Next, we render the headline as linked text:

```
<a href='<string:stream variable="aUrl"/>'>
  <string:stream list='<%="art:"+ics.GetVar("HeadlineAttrName")%>'
     column="value" /></a>
```

Note

The `string:stream` tag ensures proper encoding of the URL. This is required for XML compliance.

The rendered headline is followed by the postdate and abstract attributes.

FSIILink (Content_C)

When a content asset is displayed as a link, its headline attribute is displayed as linked text to the asset's Detail view.

To view the source code for the FSIILink template

1. Click on the **Design** tab in the Content Server standard interface.
2. Expand the **Template** item, and double-click on **FSIILink (Content_C)** to open the file in your workspace.

Note

The FSII Link template for Content assets can be found under **Tables** > **Element Catalog** > **Content_C.**

Creating Templates

When you create new flex families in the FirstSite model, you need to create new content templates in order to render them to your online site. Generally, each asset type requires a set of five templates in order to be rendered in the three ways mentioned earlier (in detail, in summary, and as a link). These five content templates are:

- FSIIDetail
- FSIIHead
- FSIISideNav
- FSIISummary
- FSIILink.

If you only plan to render the asset in only one or two of the three possible ways, then you need not create all five templates.

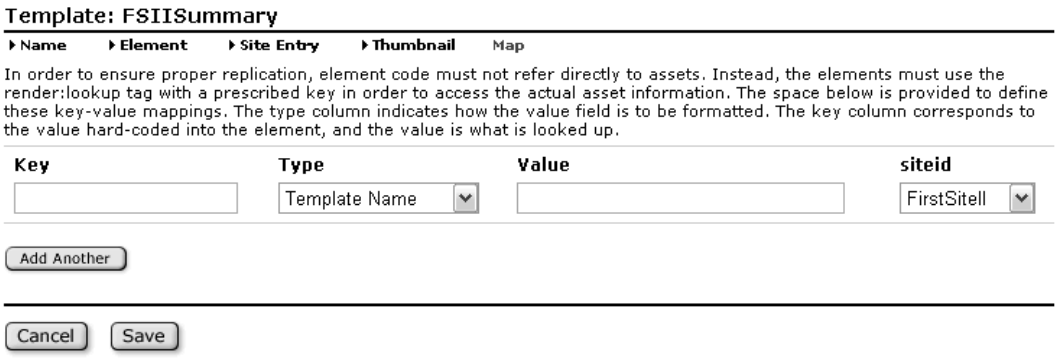
For our Store asset type example (see "Creating a Flex Family," on page 63), we only want to display the store locations (along with their relevant attributes) in a list. The template we will create for this is the FSIISummary template.

This section briefly outlines the procedure for creating a new FSIISummary template using the standard interface. This section does not include a discussion on the source code for this template.

To create a new Summary template

1. In the CS standard interface, click New.
2. From the **Start Menu**, select **New Template**.
3. In the "Choose Assignees" screen, select **firstsite** from the list of users and click **Set Assignees**.
4. In the "New Template" creation form,
   a. Enter `FSIISummary` in the "Name" field. This is the name of your new template.
   b. Enter a brief description for your new template in the "Description" field. This description will help you and other developers to know the purpose of this template.
   c. The "Category" field allows you to specify what section of the page your template will be used for. From the list, select **Subpage Template**.
   d. Next, select the asset type this template applies to. From the list of available asset types, select **Store**.
   e. For "Applies to subtypes:" select **Store**.
   f. Click Continue.
   g. In the next screen, for "Usage," select **Element is used within an HTML page.**
   h. For "Create Template Element?" you can select what language your template will be written in. Click JSP.
   i. In the "Element Logic" text area, you would enter your source code for the FSIISummary template. For now, enter:
   ```
   <h3>New FSIISummary Template</h3>
   ```
   j. Click Continue until you reach the "Map" screen (Figure 22), or click ▸Map in your action bar at the top of your creation form.

Figure 22: The Map screen on a Template creation form



   k. Add the following information in the fields provided. Click Add Another to add additional key/value entries.

Key	Type	Value	siteid
Wrapper	Asset (Type:Name)	SiteEntry:FSIIWrapper	FirstSiteII
StoreAttrName	Asset (Type:Name)	StoreLocation_A:StoreName	FirstSiteII
AddressAttrName	Asset (Type:Name)	StoreLocation_A:StreetAddress	FirstSiteII
ZipAttrName	Asset (Type:Name)	StoreLocation_A:ZipCode	FirstSiteII
PhoneAttrName	Asset (Type:Name)	StoreLocation_A:PhoneNumber	FirstSiteII
HoursAttrName	Asset (Type:Name)	StoreLocation_A:StoreHours	FirstSiteII
StoreAttrType	Asset Type	StoreLocation_A	FirstSiteII
Layout	Template	/FSIILayout	FirstSiteII

The values in the table above are the attributes and templates that you would use in coding your template. Replicable templates must not refer to assets directly in the element code. Template mapping is discussed in further detail in the next section.

l. Click Save.

Template Mapping

In order to ensure proper replication, element code must not refer directly to assets. Instead, the elements must use the `render:lookup` tag with a prescribed key in order to access the actual asset information. In this section, we examine the **FSIISummary (StoreLocation_C)** template to get an idea of how template mapping is being used in the FirstSite framework.

To examine the FSIISummary (StoreLocation_C) template

1. Click on the **Design** tab in the Content Server standard interface.
2. Expand the **Template** item, and double-click on **FSIISummary (StoreLocation_C)** to open the file in your workspace.
3. In your workspace, scroll down to the section labeled **Map**.

Figure 23: Map section for the FSIISummary (StoreLocation_C) template.

Map:

Key	Type	Value
StoreAttrType	Asset Type	StoreLocation_A
AddressAttrName	Asset (Type:Name)	StoreLocation_A:StreetAddress
Wrapper	Asset (Type:Name)	SiteEntry:FSIIWrapper
ZipAttrName	Asset (Type:Name)	StoreLocation_A:ZipCode
HoursAttrName	Asset (Type:Name)	StoreLocation_A:StoreHours
PhoneAttrName	Asset (Type:Name)	StoreLocation_A:PhoneNumber
StoreAttrName	Asset (Type:Name)	StoreLocation_A:StoreName
Layout	Template Name	/FSIILayout

In Figure 23, the **Key** represents the value that is used in the element code to search for its **Value** pair. For example, in the **FSIISummary (StoreLocation_C)**, to render the **Layout**, we first use the `render:lookup` tag to find the **/FSIILayout** template:

```
<render:lookup site='<%=ics.GetVar("site")%>' varname="LayoutVar"
  key="Layout" tid='<%=ics.GetVar("tid")%>'/>
```

The value for the `key` attribute `"Layout"` is used to find the **Template Name**, which is then stored as `LayoutVar`. Available template mappings include:

- **Template Name** - map template name to key value (see prior example)
- **Asset Type** - map asset type to key value
- **Asset (Type:Name)** - map attribute type:name to the key value
- **Asset (Type:ID)** - map attribute type:ID to the key value

For the wrapper page, the **Asset (Type:Name)** template mapping is used. To look up the wrapper page, we use the following:

```
<render:lookup site='<%=ics.GetVar("site")%>' varname="WrapperVar"
  key="Wrapper" tid='<%=ics.GetVar("tid")%>' match=":x"/>
```

If you remember from the template creation example in the previous section, you created a mapping for the Wrapper, giving it the value, `SiteEntry:FSIIWrapper`. The `match=":x"` attribute value pair in the `render:lookup` tag tells Content Server to store only the second half of the `type:name` pair and store is as `WrapperVar` for later use.

Once the FSIILayout template and FSIIWrapper values are found they are then used to construct URLs:

```
<render:gettemplateurl
        outstr="aUrl"
        tid='<%=ics.GetVar("tid")%>'
        slotname="StoreLocation_CLinkOnSummaryPage"
        site='<%=ics.GetVar("site")%>'
        c='<%=ics.GetVar("c")%>'
        cid='<%=ics.GetVar("cid")%>'
        tname='<%=ics.GetVar("LayoutVar")%>'
        wrapperpage='<%=ics.GetVar("WrapperVar")%>' >
   <render:argument name="p" value='<%=ics.GetVar("p")%>'/>
</render:gettemplateurl>
```

This practice of template mapping is a common occurrence across all FirstSite templates. Template mapping not only facilitates replication but it also simplifies template usage by making them completely pluggable. If you created a new Layout template, for example, you simply replace the FSIILayout value for your new layout template.

URL Assemblers

URL Assemblers permit you to change the way a URL is generated. This allows you to have full control over the appearance of links. Together with Content Server URL generation tags, assemblers generate URLs and disassemble the URLs they generate. Two assemblers are installed with Content Server: Query Assembler and QueryAsPathInfo Assembler. The Query Assembler is Content Server's default assembler. A third URL

assembler, the FirstSite URL Assembler, is a part of a separate installable module that can be added on to the FirstSite Core module upon installation.

The FirstSite URL Assembler

The FirstSite URL Assembler module contains a custom URL assembler that takes the long URLs constructed by Content Server and creates shorter, more concise URLs, based on coding practices used in designing FirstSite.

If you elect not to install the FirstSite URL Assembler, FirstSite will still function properly and Content Server reverts back to its default URL Assembler. For more information regarding the FirstSite URL Assembler, refer to the source code located in the CS installer folder under `FirstSiteII/PrettyURL/src`.

For information on creating your own custom URL Assemblers, refer to the *Content Server Developer's Guide*.

Part 3

Replicating FirstSite

This part of the guide contains information about replicating FirstSite using Content Server Site Launcher. Replication can only be performed by administrators. This part contains the chapter:

- Chapter 8, “Site Replication”

Chapter 8

Site Replication

If you want to create a new site on Content Server, FatWire recommends that you replicate FirstSite and use it as your foundation. By replicating an existing Content Server site, you minimize your effort in creating new sites.

This topic is much more advanced than what is presented in this section. For complete information on site replication, including guidelines and procedures, you can refer to the *Content Server Administrator's Guide*. In order to complete the site replication procedure, you must have at least the FirstSite Core module installed.

This chapter contains the following sections:

- Overview
- Preparing for Replication
- Site Replication Steps
- Post-Replication Steps

Overview

The utility for replicating sites on Content Server is called Site Launcher. This utility is not designed for backing up CM sites, but for spinning them off. Site Launcher replicates source sites directly on their native Content Server system, reusing the existing database schema. The sites are replicated quickly and easily, without the need for coding. However, while replication itself is a quick and straightforward procedure, it does require preparation and follow up on the part of the administrator. How much, depends on the content management needs.

Note

Site replication can be carried out only by the GeneralAdmin role. The SiteAdmin role cannot replicate sites, as they have no access to Site Launcher.

Preparing for Replication

In general before you attempt to replicate a site, you need to consider several recommendations regarding the nature of the site and its replicate. You must also make decisions about which components to copy or share, and finally ensure that the source site meets system requirements for replication (For more information on the guidelines, system requirements, and our recommendations, you can refer to the *Content Server Administrator's Guide*).

Our purpose for replicating FirstSite is to keep a copy of the original site for later use. When you are prompted to select which components are copied/shared, you will choose to copy all components.

Naming Assets

Assets that are copied to the new site are named algorithmically, by use of prefixes as follows:

- The prefix for the source is specified when you designate a site as a site launcher source site.
- The prefix for the new site is specified in Site Launcher itself. When specifying a prefix, ensure that it is short and unique.
- If the source asset has a prefix, or the asset type requires a unique name, then a prefix will be used in the name of the new asset. If the source asset has a prefix, the new asset will have the new prefix in place of the old prefix.
- If the source asset does not have a prefix, then the prefix is prepended to the asset name to make the new asset name.

Planning Users

Given that users are not copied or shared to the new site during replication, you will have to manually add them to the new site. Steps for adding users to a new site are outlined below in "Post-Replication Steps" on page 93.

Site Replication Steps

In order to replicate a source site, you must first configure Site Launcher.

To configure Site Launcher

1. In the **Admin** tab, expand the **Sites** node and double-click on **FirstSite**. FirstSite is our source site for replication.
2. In FirstSite's "Inspect" screen, click the link **Configure CS-Site Launcher for this site** (at the bottom of the screen).
3. In the screen "Configure CS-Site Launcher,"
   a. In the section labeled **Enabled Asset Types**, click the **Copy All** button. We are going to copy everything because we want to keep the original site in tact.
   b. Click the Save button to save your configuration options.

Once you have enabled the source site for replication, you can use Site Launcher to create as many copies of the source site as you wish.

To replicate the source site

4. At the top of the "Inspect" form, double-click the Launch Copy... icon.
5. On the "Site Launcher" screen, do the following:
   a. In the "Name" field, enter the name for the new site.
   b. In the "Description" field, enter the description for the new site.
   c. In the "Publish Destinations" panel:
      1) Select the publish destinations you would like to be available for the new site.
      2) Initialize the destinations by selecting the checkboxes, as necessary.
   d. Click the Add Site button to copy the site.

   After the new site is added, Content Server displays a screen that indicates:

   - Which assets were copied or shared. For each asset that is shared, Content Server adds a row in the `AssetPublication` table, indicating the site id (publication id) of the new site. For each asset that is copied, Content Server enters the copy into the table where the original asset is stored.
   - The number of start menu items, workflow processes, and publish destinations that were shared with the new site.

Post-Replication Steps

As previously stated in "Planning Users" on page 92, users are not copied nor shared to the new site and will have to be added manually.

Users

To add the user, firstsite, to your new site

1. In the Content Server standard interface, click on the **Admin** tab.
2. Expand the **Sites** item.
3. Expand the item created for your new site.
4. Double-click on **Users**. This will load the User Role Management screen in your workspace.
5. In the **User Role Management** screen, enter the username, `firstsite`, and click **Select**.
6. Click on **firstsite** to select it.
7. Click the **Edit** button.
8. Highlight all the roles for **firstsite** and click **Save**.

If you would like to establish new users for your new site, follow the guidelines in Chapter 3, “Site Configuration Guidelines,” of the *Content Server Administrator’s Guide*.

Web Resources

Web resources like images and style sheets are not automatically replicated by Site Launcher. You must manually replicate these resources for your new site.

To copy FirstSite web resources to your new site

1. In your JumpStart Kit folder, go to **tomcat5 > webapps > cs**.
2. Create a new folder. Rename this folder to match the name of the new site you created.
3. Copy the contents of the **FirstSiteII** folder and place the copied items into your newly created folder.
4. (Optional) Alter the images and style sheets in the new folder to reflect the desired brand of your new site.

Appendices

The topics covered in the following appendices are typically an administrative topics, however, they would be useful for developers to gain an understanding of FirstSite and Content Server configurations.

This part contains the following appendix:

- Appendix A, “FirstSite Configurations”
- Appendix B, “Access Permissions”

Appendix A

FirstSite Configurations

This appendix lists the default configuration of the FirstSite starter site. The information in the following sections:

- Component Configurations
- Workflow

Component Configurations

This section outlines the default component configurations for available roles, users, tree tabs, and start menu items for the following installable FirstSite modules:

FirstSite Core

This is the module of FirstSite that needs to be installed in order if any other component is to be used. The FirstSite Core can also be used for developing applications upon. This module does not include any content asset type definitions or sample data. The FirstSite Core includes support for site navigation and site structure, built using typeless templates.

Roles

The following roles are installed:

Role	Description
Approver	Users assigned this role can approve assets for publish.
Designer	Users with this role can manipulate the navigation and the Recommendation assets, as well as Templates, CSElements, and SiteEntry assets.

Users

The following describes the roles assigned to each user:

User	Roles	Description
firstsite	Designer, Approver, WorkflowAdmin, SiteAdmin, GeneralAdmin	This is the FirstSite super-user.
fwadmin (password: `xceladmin`)	GeneralAdmin	User is given administrative tasks. With this privilege they can assign themselves whichever roles are desired.

Tree Tabs

The following tree tabs are installed with FirstSite Core:

Tree Tab	Allowed Roles	Asset Types	Description
Active List	Approver, Designer		This tab ships with CS-Direct.
Design	Designer	AttrTypes, CSElement, Query, AdvCols, SiteEntry, Template, Page	This tab is to be used for designing site templates and flex family schemas, as well as content forms. It is not intended for use by content editors.
History	Approver, Designer		This tab ships with CS-Direct
Query	Approver, Designer		This tab ships with CS-Direct.
Site Plan	Approver, Designer		This tab ships with CS-Direct.
Workflow	Approver, Designer		This tab ships with CS-Direct.

Start Menu Configurations

Name	Asset Type	Start Menu Type	Roles
New Attribute Editor	Attribute Editor	New	Designer
Find Attribute Editor	Attribute Editor	Search	Designer
New CSElement	CSElement	New	Designer
Find CSElement	CSElement	Search	Designer
New Page	Page	New	Designer, Approver
Find Page	Page	Search	Designer, Approver
New Query	Query	New	Designer, Approver
Find Query	Query	Search	Designer, Approver
New Template	Template	New	Designer
Find Template	Template	Search	Designer

FirstSite Engage Core

This module extends FirstSite Core and adds typeless templates that have FatWire Engage support. It does not include any FatWire Engage data, other than additional

recommendation asset instances as needed by the new and revised typeless templates that it provides.

Roles

The following roles are installed with the FirstSite Engage Core:

Role	Description
MarketingAuthor	can author promotions, recommendations, and create segments
MarketingEditor	can edit promotions, recommendations, and create segments

Users

The following describes the roles assigned to each user:

User	Roles	Description
firstsite	MarketingAuthor, MarketingEditor	

Tree Tabs

The following tree tabs are installed with the FirstSite Engage Core:

Tree Tab	Allowed Roles	Asset Types	Description
Design		ScalarVals, HistoryVals, HFields	Gives designeres the right to modify visitor data.
Marketing	MarketingAuthor, MarketingEditor	Segments, AdvCols, Promotions	Gives the marketing team the right to ability to work with promotional information
Query	MarketingAuthor, MarketingEditor		
SitePlan	MarketingAuthor, MarketingEditor		

Start Menu Configurations

Name	Asset Type	Start Menu Type	Roles
New Attribute Editor	Attribute Editor	New	Designer
Find Attribute Editor	Attribute Editor	Search	Designer
New CSElement	CSElement	New	Designer
Find CSElement	CSElement	Search	Designer
New Page	Page	New	Designer, Approver
Find Page	Page	Search	Designer, Approver
New Query	Query	New	Designer, Approver
Find Query	Query	Search	Designer, Approver
New Template	Template	New	Designer
Find Template	Template	Search	Designer

FirstSite Store Schema

This module includes the FirstSite Store Schema, including flex families and asset types for media, products, and content.

Roles

The following roles are installed with the FirstSite Store Schema:

Role	Description
ArtworkAuthor	Can author media assets
ArtworkEditor	Can edit media assets
ContentAuthor	Can author content assets
ContentEditor	Can edit content assets
ProductAuthor	Can author product assets
ProductEditor	Can edit product assets

Users

The following describes the roles assigned to each user:

User	Roles	Description
firstsite	ContentAuthor, ContentEditor, ProductAuthor, ProductEditor, ArtworkAuthor, ArtworkEditor	

Tree Tabs

The following tree tabs are installed with the FirstSite Store Schema:

Tree Tab	Allowed Roles	Asset Types	Description
Design		Media_A, Media_PD, Media_CD, Media_F, Content_A, Content_PD, Content_CD, Content_F, Product_A, Product_PD, Product_CD, Product_F	Allows designers to work with the schema of all flex families
Products	ProductAuthor, ProductEditor	Product_P, Product_C	Allows modification of products and product parents
Artwork	ArtworkAuthor, ArtworkEditor, ContentAuthor, ContentEditor, ProductAuthor, ProductEditor	Media_P, Media_C	Allows modification and selection of media and media parents. Product and content people need to see this to pick images for the assets they are creating.
Content	ContentAuthor, ContentEditor	Content_P, Content_C	Allows modification of content and content parents
Query	ArtworkAuthor, ArtworkEditor, ContentAuthor, ContentEditor, ProductAuthor, ProductEditor		

Start Menu Configurations

Name	Asset Type	Start Menu Type	Roles
New Content	Content	New	ContentAuthor
Find Content	Content	Search	ContentAuthor, ContentEditor
New Content Attribute	Content Attribute	New	Designer
Find Content Attribute	Content Attribute	Search	Designer
New Content Category	Content Parent	New	ContentAuthor, ContentEditor
Find Content Category	Content Parent	Search	ContentAuthor, ContentEditor
New Content Definition	Content Definition	New	Designer
Find Content Definition	Content Definition	Search	Designer
New Content Filter	Content Filter	New	Designer
Find Content Filter	Content Filter	Search	Designer
New Content Parent Definition	Content Parent Definition	New	Designer
Find Content Parent Definition	Content Parent Definition	Search	Designer
New Media	Media	New	ContentAuthor, Content Editor, MediaAuthor, MediaEditor, ProductAuthor, ProductEditor
Find Media	Media	Search	ContentAuthor, Content Editor, MediaAuthor, MediaEditor, ProductAuthor, ProductEditor
New Media Attribute	Media Attribute	New	Designer
Find Media Attribute	Media Attribute	Search	Designer
New Media Category	Media Parent	New	MediaAuthor, MediaEditor
Find Media Category	Media Parent	Search	MediaAuthor, MediaEditor
New Media Definition	Media Definition	New	Designer

Name	Asset Type	Start Menu Type	Roles
Find Media Definition	Media Definition	Search	Designer
New Media Filter	Media Filter	New	Designer
Find Media Filter	Media Filter	Search	Designer
New Media Parent Definition	Media Parent Definition	New	Designer
Find Media Parent Definition	Media Parent Definition	Search	Designer
New Product	Product	New	ProductAuthor, ProductEditor
Find Product	Product	Search	ProductAuthor, ProductEditor
New Product Attribute	Product Attribute	New	Designer
Find Product Attribute	Product Attribute	Search	Designer
New Product Category	Product Parent	New	ProductAuthor, ProductEditor
Find Product Category	Product Parent	Search	ProductAuthor, ProductEditor
New Product Definition	Product Definition	New	Designer
Find Product Definition	Product Definition	Search	Designer
New Product Filter	Product Filter	New	Designer
Find Product Filter	Product Filter	Search	Designer
New Product Manufacturer	Product Parent	New	ProductAuthor, ProductEditor
Find Product Manufacturer	Product Parent	Search	ProductAuthor, ProductEditor
New Product Parent Definition	Product Parent Definition	New	Designer
Find Product Parent Definition	Product Parent Definition	Search	Designer
New Product Subcategory	Product Parent	New	ProductAuthor, ProductEditor
Find Product Subcategory	Product Parent	Search	ProductAuthor, ProductEditor

FirstSite Document Schema

Roles

The following roles are installed with the FirstSite Document Schema:

Role	Description
DocumentAuthor	Can author document assets
DocumentEditor	Can edit document assets

Users

The following describes the roles assigned to each user:

User	Roles	Description
firstsite	DocumentAuthor, DocumentEditor	

Tree Tabs

The following tree tabs are installed with the FirstSite Document Schema:

Tree Tab	Allowed Roles	Asset Types	Description
Design		Document_A, Document_PD, Document_CD, Document_F	Allows designers to work with the schema of the Document flex family
Documents	DocumentAuthor, DocumentEditor	Document_P, Document_C	Allows modification of document and document parents
Query	DocumentAuthor, DocumentEditor		
Content	DocumentAuthor, DocumentEditor		

Start Menu Configurations

Name	Asset Type	Start Menu Type	Roles
New Document	Document	New	DocumentAuthor, DocumentEditor
Find Document	Document	Search	DocumentAuthor, DocumentEditor
New Document Attribute	Document Attribute	New	Designer
Find Document Attribute	Document Attribute	Search	Designer
New Document Definition	Document Definition	New	Designer
Find Document Definition	Document Definition	Search	Designer
New Document Filter	Document Filter	New	Designer
Find Document Filter	Document Filter	Search	Designer
New Document Folder	Document Parent	New	DocumentAuthor, DocumentEditor
Find Document Folder	Document Parent	Search	DocumentAuthor, DocumentEditor

Workflow

Each category of assets has its own workflow. All workflow processes have the same general flow:

- Created (send for edit)
- Awaiting edit (send for approval)
- Awaiting approval (approve for publish OR reject back to author)

The workflow processes are configured to start with the creation of their corresponding asset from the start menu. This means that workflow processes are installed and enhanced with the modules corresponding to the assets they control.

Table A-1: FirstSite Workflow Processes

Workflow	Roles	Asset Types
Approval for Structure	Designer, Approver	AttrTypes, CSElement, Page, Query, Recommendation, SiteEntry, Template, flex asset Parent Definitions, flex asset Definitions, flex filters, and flex attributes
Approval for Documents	DocumentAuthor, DocumentEditor, Approver	Document_C, Document_P
Approval for Products	ProductAuthor, ProductEditor, Approver	Product_C, Product_P
Approval for Content	ContentAuthor, ContentEditor, Approver	Content_C, Content_P
Approval for Artwork	ArtworkAuthor, ArtworkEditor	Media_C, Media_P
Approval for Promotions	MarketingAuthor, MarketingEditor, Approver	HistoryVals, Promotions, ScalarVals, Segments

For more information the FirstSite workflow process, see Chapter 4, “Workflow & Publishing.”

Appendix B

Access Permissions

Several methods are used to control user roles and access to assets in Content Server. One such method is to configure access permissions for certain functions, such as editing or deleting an asset, based on a user's role for the site.

This appendix contains the following sections:

- Roles
- Users
- SAFE

Roles

Roles are used to manage access to sites and their components. The assignment of roles to users and interface functions on a given site determines whether the functions are enabled for the users or hidden from them. The roles assigned to a user for a site determine the following:

- Which assets the user can create on that site.
- Which assets the user can search for on that site.
- Which tabs are displayed in the tree when the user logs in to the site.
- Whether the user is eligible for participation in any workflow processes, and, if so, for which steps in those workflow processes.
- Which functions a user can or cannot perform on an asset while it is moving through a workflow process.
- Whether the user can administrate a workflow process or create or modify a workflow group on that site.

For more information on roles, see the *Content Server Administrator's Guide.*

The following roles are created for FirstSite:

Role	Module
Designer	Core
Approver	Core
ArtworkAuthor	Store Schema
ArtworkEditor	Store Schema
ContentAuthor	Store Schema
ContentEditor	Store Schema
ProductAuthor	Store Schema
ProductEditor	Store Schema
DocumentAuthor	Document Schema
DocumentEditor	Document Schema
MarketingAuthor	Engage Store Schema
MarketingEditor	Engage Store Schema
SiteAdmin	(included with Content Server)
GeneralAdmin	(included with Content Server)
WorkflowAdmin	(included with Content Server)

Users

Each Content Server user is completely defined by a user account, user profile, and, if necessary, user attributes.

- A user account is required for anyone who is to work with Content Server.
- A user profile is required for users who will be working with CS modules and products, setting a default language, and participating in workflow processes in which e-mail messages will be sent.
- User attributes, in addition to the locale and e-mail attributes in the user profile, may also be required for your operation. If so, the additional attributes can be created.

Once a use has been created, that user must be enabled for the appropriate sites by assigning roles to the user name for each site the user will work in. For more information about users, see the *Content Server Administrator's Guide*.

The following users are created for FirstSite:

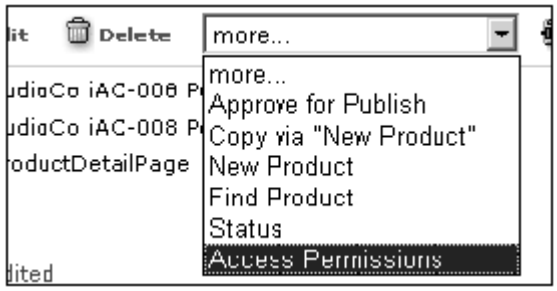
Username	Roles	Module delivered/enhanced
firstsite	all	Core, Document Schema, Store Schema, Engage Store Schema
Napoleon	Approver, SiteAdmin, WorkflowAdmin	Store Demo Data, Document Demo Data
Arthur	ArtworkAuthor	Store Demo Data,
Martha	ArtworkEditor	Store Demo Data
Conrad	ContentAuthor	Store Demo Data (ContentAuthor), Document Demo Data (DocumentAuthor)
Connie	ContentEditor, DocumentEditor	Store Demo Data (ContentEditor), Document Demo Data (DocumentEditor)
Desiree	Designer, ArtworkAuthor	Store Demo Data (Designer), Document Demo Data (Designer, ArtworkAuthor)
Mark	MarketingAuthor, ProductAuthor	Engage Store Demo Data (MarketingAuthor), Store Demo Data (ProductAuthor)
Mary	MarketingEditor, ProductEditor	Engage Store Demo Data (MarketingEditor), Store Demo Data (ProductEditor)
Rose	ProductEditor	Store Demo Data

The password for all the sample users listed above is **`firstsite`**.

SAFE

SAFE is an acronym for Secure Authorization for Enterprise. It is a Content Server interface feature where, as an administrator, you can set user access permissions for performing functions (such as creating, editing, deleting, etc) on specific assets, based on the user's role. To access this feature, you must be logged in as a user with administrative privileges (FirstSite). It is listed as Access Permissions in the **more…** menu on an asset's **Inspect** screen.

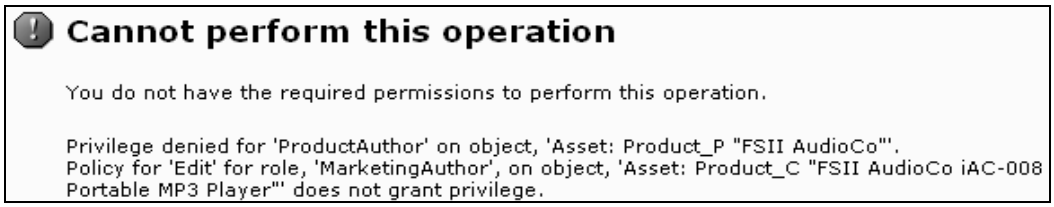
Figure 1: Accessing SAFE from the "Inspect" view of an asset.



To demonstrate a SAFE example

1. Log in to the Content Server interface as the ProductAuthor, **Mark**:
   a. Username: `Mark`
   b. Password: `firstsite`
2. Click on the **Products** tab.
3. Expand the **FSII Manufacturers** item, then expand **FSII AudioCo**.
4. Double-click on any product to load its details in your workspace.
5. Try to edit the article by clicking on Edit
   You should get the following error:

Figure 2: Error reported by SAFE.



Now login to Content Server as the ProductEditor, Mary (username: `Mary`, password: `firstsite`). Repeat steps 2-5 from above. As Mary, you should be brought to the Edit form for the product with no problems.

SAFE allows users to perform tasks on select assets. With SAFE, users can be restricted from modifying with content that does not apply to them. Using this feature in addition to other user management components (see the *Content Server Administrator's Guide* for more information) will help your team control a large team of content providers.

Glossary

The following is a list of common terms in this document and their definitions. For additional CM terms and definitions, you can refer to FatWire's Content Management Glossary located at:

`http://www.fatwire.com/products/white_papers.html`

Term	Definition
Asset	An *asset* is a general term for an individual piece of *content*. An article, image, document, an code element are examples of different types of assets.
Asset Type	The *asset type* is the structure of an asset. It indicates what type of information is contained within an asset. For example, "Article" is an asset type. An Article asset is an instance of the Article asset type.
Assetset	An *assetset* is a set of one or more flex assets or flex asset parents.
Attribute	An *attribute* refers to a characteristic or subset of a piece of content. For example, name, description, height, and width would be attributes of an image asset.
Attribute Editor	Determines how data is entered for a flex attribute.
Basic Asset	An asset with a fixed number of attributes, stored in a single table in the database. For more information on Basic Assets, see the *Content Server Developer's Guide*.
Cache	A small fast memory holding recently accessed data, designed to speed up subsequent access to the same data.
CSElement asset	An asset that contains processing logic.
Content Management (CM) Site	A CM site is the back end of the online site or sections of the online site where users work to create the online site; developers create the data models and rendering templates; admins create and manage the CM site and its users; content providers author the content for the online site.

Term	Definition
Content Provider	Anyone who creates, edits, reviews, or approves content for a site would be considered a content provider.
Core Assets	Asset types packaged with Content Server.
Delivery Site	Refers to your published site, accessible by your intended visitors.
Element	A named piece of code.
ElementCatalog table	Database table that contains elements.
Flex Asset	An asset with a flexible schema.
FlexFilter asset	An asset that processes data contained in a flex asset and sets the resultant data as attributes of the original flex asset.
Online Site	The site that visitors browse. It is the front-end of a CM site(s).
Page Assets	Structure assets that represent pages or classes of pages on a website.
Page Cache	Temporary storage of rendered pages or portions of pages (*pagelets*). Since pages are the result of executing code that displays assets, caching the result eliminates the need to execute the code and read the asset again.
Pagelet	Resultant markup fragment from an independantly renderable element.
Publishing	The process of moving a site from one system to another. For example, from the management system to the delivery system.
Recommendations	Assets that refer to other assets. Recommendations are populated in a variety of ways.
Rendered Site	Refers to the previewed version of your site.
Site Plan	A Site Plan is a hierarchal representation of the relationship between your site's page assets.
SiteCatalog table	Database table that defines the pages and pagelets that can be generated by Content Server.
SiteEntry asset	Asset that defines a pagelet.
Start Menu	The list of items that appears when you click on the **New** or **Search** button in the Content Server interface toolbar.
Template	An asset defining a pagelet and processing logic that is used to render an asset.
Typeless Template	A template that is designed to render different types of assets.
Workflow	A process in which one person passes content to another upon completing a task.

Index of Procedures