relevant ecommerce™

TECHNICAL

ZNODE MULTIFRONT 8.1.2 DEVELOPER GUIDE

Multifront is an Adaptable E-Commerce platform that has been architected from the ground up to be 100% customizable. Virtually every aspect of the storefront can be customized including the user Interface, Business Logic, Process, Workflow and Integration with other systems.

April 2016

Znode Multifront 8.1.2 Developer Guide

© Copyright 2016, Znode LLC. All Rights Reserved

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions or for damages resulting from these of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Printed: October 2016 in USA

Table of Contents

Part 1 Overview 5

Part 2 Quick Start Guide 6

Part 3 Multifront Architecture 8

Part 4 Template Management 10

Part 5 Data Access Layer (DAL) 12

Part 6 Business Logic Layer (BLL) 14

Part 7 Lucene Search Configuration 17

Part 8 Authentication and Authorization 22

Part 9 Content Management 24

Part 10 Customizing the Multifront Theme 26

Part 11 Integrating with Other Applications 27

Part 12 Activity and Exception Logging 28

Part 13 Diagnostics 31

Part 14 Zip Code Data for the Store Locator 32

Part 15 Social Media Login for Demo Site 33

Part 16 Guidelines for Customization 35

Part 17 Upgrading the Multifront 36

1 Overview

Multifront is an **Adaptable E-Commerce Platform** ™ that has been architected from the ground up to be 100% customizable. Virtually every aspect of the storefront can be customized including the User Interface, Business Logic, Process Workflow and Integration with other systems.

Philosophy

The overall goal behind the Multifront architecture is to provide you with a *standards based platform* that implements all the base e-commerce functionality. Developers can then extend this functionality to their own specific requirements. This significantly reduces the learning curve as Znode does not use proprietary technology to implement basic website architecture but instead relies on the Microsoft Prescribed Best Practices for N-Tier application design.

Installation Pre-Requisites

To install Multifront on a server you need to have a reasonably good understanding of Internet Information Services (IIS) and SQL Server 2008. The following components are required to install Multifront:

- IIS 7 or above
- .Net Framework 4.5
- SQL Server 2008 R2/2012/2014
- Requires Full Trust on the server (Check with your hosting service to ensure compatibility)
- ASP.NET 4.5

Development Pre-Requisites

Here are the pre-requisites for customizing Multifront

- Visual Studio 2013
- SQL Server 2008 R2 / 2012/2014
- IIS 7.0 with .NET 4.5 Enabled

See Also:

Quick Start Guide 5

© Copyright 2016, Znode LLC, All Rights Reserved

2 Quick Start Guide

Here are the Steps to start developing with Multifront:

Step 1: Install the EXE

After you purchase Multifront, you will be able to download an installation EXE

- Use the EXE to install the storefront on your local development machine.
- Follow the Development Installation instructions in the Installation guide carefully.
- You can now open the Multifront Solution file using Visual Studio 2012 or 2013.
- Compile the solution and run it to ensure that the storefront comes up without any errors.

Step 2: Basic Site Admin Settings

While the storefront will work with a basic install, setting the following parameters for your store using the Store Management page will ensure that all features of your storefront operate as expected.

- URL - Enter the URL that you will be using to access the store. Note that the "www" and port number of URLs are ignored.
- Profiles - A store must have at least one Default Profile and one Anonymous Profile.
- Admin, Sales, and Customer Support Email Addresses - Use an email address where you will be able to receive the messages in your in box. In this way you will be able to see orders and customer requests as they come in.
- SMTP Mail Server Settings - This setting is required for any email (orders or customer requests) to be sent out from the system.
- UPS/FedEx Settings – I
  f you intend to use the UPS Shipping Option you will first need an account. If you do not have an
- account leave these settings blank and use a Custom Shipping Options instead. In addition you should make sure that you have the following set.
  (a) At least one Shipping Option.
  (b) At least one Payment Option.

Step 3: Getting Started With Modifying the Code

Right out of the box Multifront will provide a professional and easy to use site. You can however change the code to implement just about any custom look and feel or business logic that you want (the Trial Edition only allows modification of Master Pages and CSS). Multifront is created from the ground up to be easily customizable. Here are just a few notes to get you started when looking at the code.

- You must be familiar with ASP.Net development using C# (VB is not supported).
- You can use Visual Studio 2012 or 2013 to modify your code.
- User controls for the site are in WebApp\Controls\Default. The look and feel for each page is in their own directory. For example, the Home page user controls can be found in WebApp\Controls\Default \Home. All of the code behinds for these pages are in these directories as well.
- Layout, CSS and Images can be found in WebApp\Themes\Default. The _Layout page for the Multifront is in WebApp\Views\Themes\Default\Shared_Layout.
- The top menu for the Admin is controlled by the WebApp\SiteAdmin\Web.sitemap file. To disable a menu item, such as Delete Catalog Data, just comment out the appropriate line in this file.
- _Layout pages for the remaining site admin pages can be found in WebApp\SiteAdmin\Views\Shared_Layout.

© Copyright 2016, Znode LLC, All Rights Reserved

View of the Visual Studio Solution

Step 4: Before Creating a Production Catalog

Before loading your catalog data, you should delete the existing demo catalog data as follows:

- Log into the Site Admin.
- On the top menu, click Setup > Catalogs. The Catalogs page is displayed listing all the available store catalogs in the storefront.
- Click Delete on the catalog you want to delete.
- The Catalogs page is displayed with the deleted catalog removed from the list.

Note:

1. You will not be allowed to delete the Catalog if it has associations to Departments. Delete these associations first and then delete the catalog.
2. You will not be allowed to delete the Catalog if you are logged in to a Store URL that is associated to it.

See Also:

Multifront Architecture 8

© Copyright 2016, Znode LLC, All Rights Reserved

3 Multifront Architecture

Multifront has been architected from the ground up using Microsoft-prescribed best practices to achieve the ultimate flexibility and scalability.

Multifront uses the following technologies

- Microsoft .NET Framework Version 4.5
- MVC 5
- WebApi 2
- C#
- SQL Server 2008 R2 / 2012/2014
- Microsoft Enterprise Library

Multifront utilizes the following components/ design techniques

- MVC 5 Shared Layout are used for storefront template management
- MVC 5 Membership & Role Provider for authentication and authorization
- MVC 5 Windows Communication Foundation for middleware web services (Optional Component)
- N-tier data access layer
- Clear separation of the web, business logic and data access layers using libraries
- Microsoft Enterprise Library is used for exception handling, logging, data access and encryption

© Copyright 2016, Znode LLC, All Rights Reserved

See Also:

Template Management 10
Data Access Layer (DAL) 12
Business Logic Layer (BLL) 14
Authentication and Authorization 22

© Copyright 2016, Znode LLC, All Rights Reserved

4 Template Management

Multifront uses MVC 5 Shared Layout and Cascading Style Sheets (CSS) to implement templates for the storefront. This standards based template architecture enables developers familiar with MVC to rapidly customize the storefront without having to learn a proprietary technology.

Highlights of the Multifront Templates

- The storefront themes (Share Layout, CSS and images) and related files are stored under *Znode.Engine.MvcDemo\Views\Themes\Default*
- The storefront includes one common Layout located at *Znode.Engine.MvcDemo\Views\ViewStart* - This Layout defines the common areas of the storefront including header, footer, navigation, etc.
- Each functional area has a secondary Layout that inherits from main layout. This allows each functional area to control the display specific to that functionality.

© Copyright 2016, Znode LLC, All Rights Reserved

View of the themes folder in the Multifront solution

See Also:

Customizing the Multifront Theme 28

© Copyright 2016, Znode LLC, All Rights Reserved

5 Data Access Layer (DAL)

Multifront uses a multi-tier DAL (Data Access Layer). This layer exposes strongly typed Data Sets that make it easier and more efficient to pass entities between the business logic layer and the DAL. You can either modify the DAL libraries by hand or use Code Smith templates to re-generate them based on the database schema.

Please Note

- Catalog and Customer Data is stored in a SQL Server 2008 database
- Data is accessed using stored procedures. Multifront does not use dynamic SQL
- Multifront uses XML Serialization extensively to retrieve object oriented data from the database. This reduces the number of calls back and forth from the database

The Znode DAL

- The Znode DAL includes 5 different C# libraries that can be modified independently
- The *ZNode.Libraries.DataAccess.Entities* library defines the strongly typed Data Sets. For example: The Order Entity maps all the fields from the Znode Order table to properties in the Data Set
- The *ZNode.Libraries.DataAccess.Service* library defines methods to retrieve data from the SQL Server database. The service methods return strongly typed Data Sets (Entity objects)
- The *ZNode.Libraries.DataAccess.Custom* library enables developers to add their own custom data access methods. For example, if you needed to create a new table in the database and access data from it, you would add your data retrieval functions here

View of the Znode DAL Libraries

© Copyright 2016, Znode LLC, All Rights Reserved

Example of a native data access method using the Znode DAL:

```
public TList<Manufacturer> GetAllByPortalID(int _portalID)
{
  ZNode.Libraries.DataAccess.Service.ManufacturerService _manufacturerAccess
              = new ManufacturerService();
  TList<ZNode.Libraries.DataAccess.Entities.Manufacturer> _ManufacturerList
              = _manufacturerAccess.GetByPortalID(_portalID);

  return _ManufacturerList;
}
```

Example of a custom data access method:

```
public DataSet GetDashboardItemsByPortal(int PortalID)
{
    // Create Instance of Connection Object
   string ConnectionString =
              System.Configuration.ConfigurationManager.ConnectionStrings["ZNodeECommerceDB"].Conn
   SqlConnection MyConnection = new SqlConnection(ConnectionString);

   //Create Instance of Adapter Object
   SqlDataAdapter MyDataAdapter = new SqlDataAdapter("ZNODE_GetDashboardItemsByPortal", MyConnecti

   //Mark as stored procedure
   MyDataAdapter.SelectCommand.CommandType = CommandType.StoredProcedure;

   // Add Parameters to Stored Procedure
   SqlParameter myParam = new SqlParameter("@PortalID", SqlDbType.Int);
   myParam.Value = PortalID;

   MyDataAdapter.SelectCommand.Parameters.Add(myParam);

   //Fill DataSet
   DataSet MyDataSet = new DataSet();
   MyDataAdapter.Fill(MyDataSet);

  //Return DataSet
  return MyDataSet;
}
```

See Also:

Overview 4
Business Logic Layer (BLL) 14

© Copyright 2016, Znode LLC, All Rights Reserved

6 Business Logic Layer (BLL)

Multifront uses a multi-tier BLL (Business Logic Layer) that has been organized to allow developers to easily extend the e-commerce functionality available out of the box.

Please Note

- The BLL does not include any database access functionality. You must avoid adding any data access functions and use the DAL instead to retrieve data
- Modifying certain areas of the BLL could affect PCI compliance. You should refer to the PCI Compliance Guide included with Multifront before making any changes

The Znode BLL

The Znode BLL in the Multifront Edition includes 14 different C# libraries that can be modified and deployed independently.

- The *ZNode.Libraries.ECommerce.Analytics* library contains services for managing analytics code such as Google Analytics and the affiliate tracking.
- The *ZNode.Libraries.ECommerce.Catalog* library encapsulates the functions needed for managing and displaying the product catalog.
- The *ZNode.Libraries.ECommerce.Entities* library defines classes for product, Sku and add-on.
- The *ZNode.Libraries.ECommerce.Fulfillment* library takes care of order and payment management.
- The *ZNode.Libraries.ECommerce.Promotions* library handles coupon and non-coupon based discount calculations.
- The *ZNode.Libraries.ECommerce.SEO* library provides the facilities for rewriting URLs and updating page meta tags.
- The *ZNode.Libraries.ECommerce.Service.Business* library provides the business layer for the Store Multiplier web services (optional add-on).
- The *ZNode.Libraries.ECommerce.Service.Entities* library provides the entities layer for the Store Multiplier web services (optional add-on).
- The *ZNode.Libraries.ECommerce.Shipping* library encapsulates the shipping engine that calculates shipping rates.
- The *ZNode.Libraries.ECommerce.ShoppingCart* library contains the implementation for the shopping cart.
- The *ZNode.Libraries.ECommerce.Suppliers* library provides services for sending orders to different suppliers in various formats.
- The *ZNode.Libraries.ECommerce.Tagging* library encapsulates the functions needed for product tags.
- The *ZNode.Libraries.ECommerce.Taxes* library handles tax calculations.
- The *ZNode.Libraries.ECommerce.UserAccount* library allows you to override User Account properties.
- The *ZNode.Libraries.FedEx* library provides functions specific for calculating FedEx rates.
- The *ZNode.Libraries.UPS* library provides functions specific for calculating UPS rates.
- The *ZNode.Libraries.USPS* library provides functions specific for calculating USPS shipping rates.
- The *ZNode.Libraries.Google* library provides functions specific for Google Checkout.
- The *ZNode.Libraries.Search* library provides functions specific for the Lucene Search library.

© Copyright 2016, Znode LLC, All Rights Reserved

View of the Znode **BLL** Libraries

®Copyright 2016, Znode LLC. All Rights Reserved

Extending the Znode **BLL**

The Business Logic Layer can be easily extended. New library DLLs can be copied to a production server without redeploying the entire application. Further, you can add new functionality to the Shipping, Supplier, Tax and Promotion libraries and configure them to be available as options in the storefront Site Admin. For example, you may want to add a new promotion that calculates 10% off orders on the 3rd day of every month. To do this you just need to do the following:

1. Create a new class derived from ZNodePromotionRule (in this case you could start with the ZNodePromotionAmountOffOrder.cs class as an example).
2. Compile and run your application.
3. Now go into the storefront siteadmin and choose MARKETING from the top menu. Then select PROMOTIONS AND COUPONS.
4. Next click on Add New Promotion Rule Type and fill out the form (again use ZNodePromotionAmountOffOrder as an example of how to fill this in).
5. Now your new function will show up in the Select Discount Type drop down when you add a new Promotion Rule.

The same procedure will be used for extending the Shipping, Supplier and Tax libraries.

See Also:

Multifront Architecture 8

© Copyright 2016, Znode LLC, All Rights Reserved

7 Lucene Search Configuration

Configuring the Background Service and Task

Once the Znode background service is installed, you must configure its settings, and the settings of the Lucene background task, before it will work properly.

IMPORTANT: The configuration settings for both the Znode background service and the Lucene background task are located in the configuration file of the Znode background service. This file is located in the same location as the Znode background service itself and is named Znode.BackgroundService.exe.config.

Configuration Settings for the Znode Background Service

The settings for the Znode background service are located in the <appSettings> section of the configuration file, as such:

```
<appSettings>
   <add key="EnableDebugLog" value="false" />
   <add key="DebugLogFile" value="C:\BackgroundService\Logs\_debug.log" />
   <add key="EnableWhatDoIHave" value="false" />
</appSettings>
```

EnableDebugLog – Turns on logging to the debug log file defined in the DebugLogFile setting. Set this to true if you need to enable debugging for troubleshooting purposes.

DebugLogFile – This is the actual file where the debugging statements will get logged to. You must define the full path to the file, and it should be writeable.

EnableWhatDoIHave – This setting is for internal debug purposes. If the Lucene background task isn't being executed by the Znode background service, setting this value to true will write information to the debug log file that will help determine what assemblies are being loaded by the Znode background service. Only works if the EnableDebugLog setting is set to true.

Configuration Settings for the Lucene Background Task

The settings for the Lucene background task are located in its own custom section of the configuration file for the Znode background service. The custom configuration section is named <Znode.Lucene.Tasks> and contains the following settings:

```
<Znode.Lucene.Tasks>
   <LogFolder value="C:\BackgroundService\Logs" />
   <TransactionLogFile value="C:\BackgroundService\Logs\_transaction.log" />
   <LuceneIndexLocation value="C:\LuceneData" />
   <LuceneIndexInterval value="30" />
   <LuceneIndexReaderKey value="Ajgsfdj342343kdksjsjncmflfl" />
   <LuceneIndexReaderUrl value="http://localhost/UpdateLuceneReader" />
   <OnStartWaitTime value="10" />
```

© Copyright 2016, Znode LLC, All Rights Reserved

```
  <ServerName value="Server1" />
</Znode.Lucene.Tasks>
```

LogFolder – This is the folder where any logging will occur. It should be the full path to the folder and also be writeable.

TransactionLogFile – This is the actual file where the logging will occur for the Lucene background task, and should be the full path using the same location as defined in the LogFolder setting.

LuceneIndexLocation – This is the folder where the actual Lucene search index files will reside. This should be a writeable folder.

LuceneIndexInterval – This is the interval, in seconds, that the Lucene background task is run by the Znode background service.

LuceneIndexReaderKey – This is the key value that the Lucene background task posts to the LuceneIndexReaderUrl whenever a create/rebuild index command is issued.

LuceneIndexReaderUrl – This is the fully-qualified path to the UpdateLuceneReader.aspx page, which is called by the Lucene background task whenever a create/rebuild index command is issued. This page validates that it was called with an HTTP POST and also that the key was supplied, as defined in the LuceneIndexReaderKey setting.

OnStartWaitTime – This setting, in seconds, tells the Znode background service how long to wait before executing the Lucene background task. This setting is only used when the Znode background service starts.

ServerName – This is the name of the server where the Lucene background task is running. This can be the actual server name, a fully-qualified domain name, or even an IP address (if the server is part of a web farm, this value should be unique across all servers in the web farm). This is used by the Lucene background task to determine which server should be running the search index commands found in the ZNodeLuceneIndexMonitor table.

Creating/Rebuilding a Search Index

When you first install and configure Znode Multifront, you must create the Lucene index used by navigation and search. This typically happens after population of the product catalogs, such as after a bulk import of products, categories, etc.

NOTE: It's important to understand that creating an index and rebuilding an index are effectively the same operation.

To demonstrate what happens in this scenario, first take a look at the following diagram:

© Copyright 2016, Znode LLC. All Rights Reserved

Here is a description of the events taking place in the diagram:

1. An administrator goes to the "Manage Index" section of the site admin and clicks the "Create Index" button. This will create an entry in the ZNodeLuceneIndexMonitor table that tells the Lucene background task to create/rebuild the search index.

2. At an interval specified in its config file, the Znode background windows service will run the Lucene background task to check for changes in the ZNodeLuceneIndexMonitor table.

3. If there are entries in the ZNodeLuceneIndexMonitor table since the last time the Lucene background task was run, and if in those entries is a record to create/rebuild the index, the Lucene background task will delete all other index change commands and only execute the "create/rebuild index" command.

For example, since the last time the Lucene background task was run, there were 4 product descriptions that were updated, and then an administrator clicked the "Create Index" button, while another administrator updated 3 other product descriptions. Then when the Lucene background task runs at its next interval, it will get 8 entries from the ZNodeLuceneIndexMonitor table: 7 related to the product description changes and 1 for the "create index" command. The Lucene background task will only execute the 1 "create index" command and delete the other 7 "product changed" commands.

4. The Lucene background task will then perform an HTTP POST to the UpdateLuceneReader.aspx page on the public website (the URL of which is defined in the config file). This page only accepts POST requests and will validate the key sent with the request (the value of which is also set in the config file on both the web app and the Znode background windows service).

5. If the POST and key validation is successful, the UpdateLuceneReader page simply calls the Lucene API to create a new search index.

© Copyright 2016, Znode LLC, All Rights Reserved

Updating an Existing Search Index

Once Znode Multifront is up and running, inevitably there will be changes made to products and categories. When this happens, the Lucene index must be updated so that the changes are properly reflected in the search results.

To demonstrate what happens in this scenario, take a look at the following diagram:

Here is a description of the events taking place in the diagram:

1. An administrator makes a product or category change in the site admin, such as updating a product's description.

2. There are triggers in the database that pick up the changes and write them to the ZNodeLuceneIndexMonitor table.

3. At an interval specified in its config file, the Znode background windows service will run the Lucene background task to check for changes in the ZNodeLuceneIndexMonitor table.

4. If there are entries in the ZNodeLuceneIndexMonitor table since the last time the Lucene background task was run, it processes each entry as a change to the Lucene index.

5. For each change entry in the ZNodeLuceneIndexMonitor table, the Lucene background task calls the Lucene API to update the appropriate document in the Lucene index.

Maintaining the Search Index in a MVC

For companies that will run Znode Multifront in a MVC Pages environment, there is a challenge to keep the

© Copyright 2016, Znode LLC. All Rights Reserved

search index consistent across all servers in the web farm. Znode Multifront solves this issue by requiring each web server have its own copy of the search index (note that this is not the same as having a “master” index that is replicated across servers).

To understand how Znode Multifront maintains the search index in this scenario, refer to the following example infrastructure

This picture shows MVC page environment with 4 web servers and 1 database server, where each web server has Znode Multifront installed/configured with its own copy of the search index, the background service and task, and the UpdateLuceneReader page on the public website. Each web server uses the same database, which contains the ZNodeLuceneIndexMonitor table used to track changes that need to be made in the search index.

Updating the Search Index on Multiple Servers

There are two scenarios for updating the search index:

1. When a new index needs created or an existing index needs rebuilt, each web server will go through the process as shown and described in the section titled “Creating/Rebuilding a Search Index”.

2. When product and/or category changes are made in the site admin that require updates to the search index, each web server will go through the process as shown and described in the section titled “Updating an Existing Search Index”.

However, depending on the interval for the Lucene background task and when the Znode background service was started on each web server, this process for updating the search index will occur at different times on each of the web servers in the web farm.

© Copyright 2016, Znode LLC, All Rights Reserved

Because of this, there will likely be a short period of time where the search index on each of the web servers is different. However, this will only be the case until the Lucene background task runs during its next interval on each of the web servers.

8 Authentication and Authorization

Multifront uses the MVC 5 Membership and Roles model. This allows developers to write their own Membership provider instead of using the default that is included out of the box (for example: Active Directory, LDAP, etc.)

Please Note

- The Multifront database includes schema for both ASP.NET Authentication/Authorization and Customer Accounts. Customer Accounts are linked with User IDs that are stored in the Authentication tables
- In order to enable easy separation of the Authentication/ Authorization data, the Account tables do not have foreign key constraints into the Authentication tables. As a result, you can easily separate the Authentication tables into their own database or share the database with another application for Single Sign-On

Authentication and Authorization Configuration

The Membership and Role providers used for authentication/ authorization are specified in the Web Config. You could modify these sections to use your own providers.

```
<profile enabled="true" defaultProvider="SimpleProfileProvider" lockItem="false">
  <providers>
    <add name="SimpleProfileProvider" type="WebMatrix.WebData.SimpleMembershipProvider, WebMatrix.WebData" connectionStringName="ZNodeECommerceDB"
  </providers>
</profile>
<membership defaultProvider="SimpleMembershipProvider">
  <providers>
    <add name="SimpleMembershipProvider" type="WebMatrix.WebData.SimpleMembershipProvider, WebMatrix.WebData" />
  </providers>
</membership>
<roleManager defaultProvider="SimpleRoleProvider">
  <providers>
    <add name="SimpleRoleProvider" type="WebMatrix.WebData.SimpleRoleProvider, WebMatrix.WebData" />
  </providers>
</roleManager>
```

© Copyright 2016, Znode LLC. All Rights Reserved

Database schema for user Authentication md Authorization

See Also:

Multifront Architecture 8

© Ccpyrigtt 2016, Znode LLC, All Rights Reserved

9 Content Management

Multifront includes a built-in content management system. This system is meant to allow end users to easily manage content without technical expertise. The content is stored in XML files and database tables to allow integration with other systems.

Types of content managed areas in Multifront:

1. Content Pages: These are static content pages that can be created by logging into the storefront siteadmin as a content editor. An example of this content type is the "About Us" page. Once the content page is created, it can be edited using a WYSIWYG editor.

2. Message Blocks: These are static content areas that are implemented in the MVC 5 code using a ManageMessage Block Controller. At run time the Controller gets it's messages from the database by looking up the message by ID (previous versions of Multifront stored messages in an XML file located in *Znode.Engine.MvcAdmin\Data\Default\Config\MessageConfig.xml*). You can use the storefront siteadmin to edit content in these message blocks. The advantage of the message block is that it can be referenced in several pages (for example: you could create a message block to display the customer service phone number. This block can then be placed on different types of pages - the phone number will be displayed on all of these pages)

Editing Content Pages using SiteAdmin

© Copyright 2016, Znode LLC, All Rights Reserved.

Editing Message Block Content using Site Admin

See Also:

Customizing Multifront Pages 29
Customizing the Multifront Theme 28

© Ccpyrigtt 2016, Znode LLC, All Rights Reserved

10 Customizing the Multifront Theme

It is relatively easy to customize the Multifront Theme using Znode's standardized architecture. Znode uses the MVC 5 Shared Layout template scheme to enable developers to achieve rapid customization of virtually every user interface element of the storefront.

Steps to modify your Multifront Theme:

1. In this example we will use the default theme as a starting point instead of creating all the files from scratch. Copy all the folders and files under "Znode.Engine.MvcDemo\Views\Themes\Default" to "Znode.Engine.MvcDemo\Views\Themes\MyTheme". Now we have to add the theme entry in znode themes table.

2. Edit the "Znode.Engine.MvcDemo\Views\Themes\MyTheme" to modify the overall HTML of the site including headers, footers, etc.

3. Now for CSS, we have to copy all the folders and files under "Znode.Engine.MvcDemo\Styles\Default\" to "Znode.Engine.MvcDemo\Styles\MyTheme". Now we have to add the theme entry in znode css table.

4. Edit the "Znode.Engine.MvcDemo\Styles\MyTheme" to modify the overall css of the site including headers, footers, etc.

4. Login to the Multifront Site Admin and select the newly created theme by going to Settings > Stores > Manage > Catalog. Use the drop-down list to select your theme as shown below for each Catalog:

See Also:

Customizing Multifront Pages 29

© Copyright 2016, Znode LLC, All Rights Reserved.

11 Integrating with Other Applications

Multifront has been architected from the ground up to not just be fully customizable but also to make it easier to integrate with virtually any 3rd party application. There are several ways to integrate Multifront with back-office applications such as accounting systems, ERP, Oracle, CRM systems, etc.

1. Integration with ECommerce Platform

The Znode Web API is a set of RESTFUL services that have been developed to provide system integrators with the ability to retrieve, modify, create, and delete data that is stored in the Znode E-Commerce platform. The API will essentially enable integrators to take advantage of the rich business logic and rule processing that lives at the core of the Znode E-Commerce platform.

Consult the Znode Web API user manual for complete instructions and information.

2. Integration using the Znode Admin Inventory Manager

- The Multifront includes utilities under the Inventory>Import/Export Data section of the Site Admin. These functions allow you to download and upload files to your storefront in CSV format.
- Most downloads files can be updated manually and then re-uploaded to the storefront directly. For instance, you can use the Download Inventory function to get the current inventory of your storefront. Next you can update the Quantity on Hand in the CSV file that you downloaded and then re-upload the file using the Update Inventory function.
- For uploads to work you must use the correct file format. Either get the file format from the corresponding download function or from one of the example files in the *Maintenance\Sample Data Manager Files* directory of your installation. Be aware that most of these files will use Product Num and or SKU as unique identifiers for products and must not be repeated in your upload file.
- The logic of the upload and download utilities can be modified to suite your needs in the Site Admin \Secure\Inventory\ImportExportData section of the Multifront solution.

3. Integration using SQL Server

- Since all the Multifront data is stored in the SQL Server 2008 database, you can create custom integrations that directly connect to this data using tools such as Microsoft SSIS

See Also:

Multifront Architecture 8

© Ccpyrigtt 2016, Znode LLC, All Rights Reserved

12 Activity and Exception Logging

Elmah Error Logging

Multifront provides the ability to log events programmatically. This facility can be used for debugging and also for monitoring important events on your web site.

There are two basic types of logging provided, file and database. File logging is intended to be used for debugging purposes and should not be used in a production environment. Database logging is used to record things such as user login events or payment failures. Database logging is required for PCI compliance.

Each of these logging types can be used in your code for whatever purpose you need. By default basic logging is already be defined in Multifront.

File Logging

The log file for Multifront is located at Znode.Engine.MvcDemo/Data/Default/Logs/ZnodLog.log. For file logging to work you must ensure that this file is writable by the Network Service user. File logging can be turned on or off by setting the following in the web.config:

```
<add key="EnableDebugging" value="1"/>
```

There are two basic methods that are available.

```
public static void LogObject(Type objectType, Object objectInstance)
```

This method will dump the contents of an object out to the log file.

ObjectType – The type of object that you want to log. This function will use this information to determine the structure of the object passed in.
ObjectInstance – An instance of the object that you would like to log.

```
public static void LogMessage(string Message)
```

This method writes a message to the log file.

Message – A message that you would like to write out the ZnodeLog.log file.

Database Logging

Database logs are written to the ZNodeActivityLog in the database. There is a related table, ZNodeActivityLogType that defines names for each of the logging event types. When customizing your own logging you will want to define your events in the ZNodeActivityLogType table. Events that can't be logged into the database because they have not been defined will be written out to the file log instead. To enable logging in the database make set the following in the web.config (note the database logging is required for PCI compliance):

```
<add key="EnableLogging" value="1"/>
```

© Copyright 2016, Znode LLC, All Rights Reserved.

Several error numbers are pre-defined for you in the Znode.Libraries.Framework.Business DLL. These error numbers are used internally to this DLL and must be defined in your database in the ZNodeActivityLogType table in the ActivityTypeId. Multifront ships with these entries defined. Feel free to use these error numbers for your own use.

```
public enum ErrorNum
    {
        GeneralError = 1,
        GeneralWarning = 2,
        GeneralMessage = 3,
        UpgradeSchema = 100,
        DiagnosticsPage = 101,
        DiagnosticsSent = 102,
        ActivationPage = 103,
        ActivationSuccess = 104,
        ActivationFailed = 105,
        LoginSuccess = 1000,
        LoginFailed = 1001,
        LoginCreateSuccess = 1002,
        LoginCreateFailed = 1003,
        AccountUnLock = 1104,
        AccountLock = 1105,
        PasswordChangeSuccess = 1106,
        PasswordChangeFailed = 1107,
        PasswordResetSuccess = 1108,
        PasswordResetFailed = 1109,
        UserLogout = 1110,
        ApplicationStart = 2000,
        ApplicationEnd = 2001,
        CreateObject = 3001,
        EditObject = 3002,
        DeleteObject = 3003,
        PaymentSettingsChange = 3004,
        Imported = 3005,
        Exported = 3006, ViewObject =
        3007, ActivityLogReport    = 4001,
        SubmitPaymentSuccess = 5000,
        SubmitPaymentFailed = 5001,
        OrderSubmissionSuccess = 5002,
        OrderSubmissionFailed = 5003,
        StoreKeyRotated = 9000,
        StoreSettingsChangeSuccess = 9001,
        StoreSettingsChangeFailed = 9002,
        KeywordSearch = 9500,
        Productsearch = 9501,
        SKUsearch = 9502
    }
```

The following are the basic methods for logging to the database. There are several overloaded functions to make writing your code easier but they each function in much the same way.

© Ccpyrigtt 2016, Znode LLC, All Rights Reserved

Record an Activity

This function records an event to the ZNodeActivityLog table

```
public static void LogActivity(int ActivityTypeId, string Data1, string Data2, string
Data3, strin
```

ActivityTypeId – This is your error number and must be defined in the ZNodeActivityLogType table. Data1 – This is a place for you to store any custom data that you would like to log. You can save up to
255 characters and this field is indexed in the database.
Data2 – This is a place for you to store any custom data that you would like to log. You can save up to 255 characters and this field is indexed in the database.
Data3 – This is a place for you to store any custom data that you would like to log. You can save up to 255 characters and this field is indexed in the database.
Status – A placeholder for writing any status information that you would like to save. Typically in Multifront this field is used to provide further detail about the event that has been logged. LongData – This filed is provided to allow storage for up to 4000 characters of your information. Source – Where this activity is occurring.
Target – Which resources are effected by this activity.

Start Transaction Monitoring

This function will make an entry into the ZNodeActivityLog table marking the start time of your event by setting the CreateDte entry for this log entry.

```
public void LogActivityTimerStart ()
```

End Transaction Monitoring

This function will log the end time of your logging activity as well as other data that you would like to collect. With this function and the LogActivityTimeStart function you will be able to keep metrics on how long events take. The begin and end time will be recorded in the CreateDte and EndDte columns of the ZNodeActivityLog table.

```
public void LogActivityTimerEnd(int ActivityTypeId, string Data1, string Data2,
                    string Data3, string Status, string LongData)
```

ActivityTypeId – This is your error number and must be defined in the ZNodeActivityLogType table.
Data1 – This is a place for you to store any custom data that you would like to log. You can save up to 255 characters and this field is indexed in the database.
Data2 – This is a place for you to store any custom data that you would like to log. You can save up to 255 characters and this field is indexed in the database.
Data3 – This is a place for you to store any custom data that you would like to log. You can save up to 255 characters and this field is indexed in the database.
Status – A placeholder for writing any status information that you would like to save. Typically in Multifront this field is used to provide further detail about the event that has been logged.
LongData – This filed is provided to provide storage for up to 4000 characters of information.

See Also:

Application Monitoring 33
Diagnostics 38

© Copyright 2016, Znode LLC, All Rights Reserved.

13 Diagnostics

Multifront includes a diagnostics page that you can use to diagnose environment issues such as database connectivity, SMTP settings, folder permissions, and also display detailed trace messages.

Accessing the Diagnostics Page

You can access the diagnostics page by going to *http://<your-storefront>/diagnostics*

Important!

In the production environment, you should disable the diagnostics page or you can expose critical security data in-advertently to others. You can disable this page by setting the "EnableDiagnosticsPage" property in the web.config file to "0"

See Also:

Activity and Exception Logging 34

© Ccpyrigtt 2016, Znode LLC, All Rights Reserved

14 Zip Code Data for the Store Locator

The Multifront Store Locator and County Based Taxes uses reference data for the zip code, city, state, and area code. Since this data contains over 42,000 entries and changes fairly regularly it is best to purchase it commercially.

Purchasing Data

You can purchase data that is compatible with the Multifront from a company called ZIPCodeDownload at
http://www.zipcodedownload.com/Products/Product/Z5Commercial/Standard/Overview/

Once you receive your download file do the following:

1. Unzip download file.
2. Upload this file to your FTP site.
3. Login to the Multifront administration website. The Dashboard page is displayed.
4. On the top menu, click **Inventory > Import/Export Data > Import Zip Code**.The Import Zip Code page is displayed.
5. Enter the following:
   - **Enter the Path of CSV File**: Enter the full FTP path (where you have uploaded the file in point 2) along with the file name (of the Zip Code data file you had earlier downloaded in point 1).
6. Click **Submit**. The new Zip Code Data file will replace the existing one on the storefront.
7. That's it! You can now set up County Based Taxes in the SiteAdmin and Customers can also search for store locations using the Store Locator

© Copyright 2016, Znode LLC, All Rights Reserved.

15 Social Media Login for Demo Site

Here are a few guidelines to enable social media login for Demo Site.

To enable Social Media Login you have to add new keys in API Web Config

Update Api Web.config and Mvc Demo Web.config according to your configuration settings.
Update the following Api Web.Config key settings inside the <appSettings> tag-
For Ex-

```
<appSettings>
<add key ="FacebookAppId" value="Add your Facebook App ID"/>
<add key ="FacebookAppSecret" value="Add your Facebook App Secret"/>
<add key ="GoogleClientId" value="Add your Google Client Id"/>
<add key ="GoogleClientSecret" value="Add your Google Client Secret"/>
<appSettings>
```

How to create facebook developer account?

1. Create the Facebook developer account using URL - https://developers.facebook.com.

2. Create new website app by using the "Add New App" menu inside My Apps menu.

3. Create the new Facebook APP ID, by choosing the "Apps for Pages" category.

4. In Facebook new App Dashboard view, get the "App Id" & "App Secret" values, and replace it in Znode API Web.config keys for Facebook.

<add key ="FacebookAppId" value="Add your Facebook App ID"/> <add key ="FacebookAppSecret" value="Add your Facebook App Secret"/>

5. Now go to Facebook new App Dashboard > Settings > Basic Menu, add Platform using website, and put your respective MVC Demo site hosted url in SiteUrl text box & save the changes made.

6. Now in Facebook new App Dashboard > Settings > Advanced Menu, add the respective MVC Demo site hosted URL in Valid OAuth redirect URIs and save the changes.

How to create google developer account?

1. Create the Google Developer account using URL - https://console.developers.google.com

2. Create a New Project.

3. Inside APIs & Auth menu, click on Credentials menu. Select OAuth Consent Screen.

4. Give the Product Name and save the changes.

5. Select Credentials and add Credentials by using OAuth 2 Client Id.

© Ccpyrigtt 2016, Znode LLC, All Rights Reserved

6. Select the Application Type as "Web application" and click on Create. This will create the Client Id and Client Secret.

7. Add these values in Znode API Web.config keys for Google.

<add key ="GoogleClientId" value="Add your Google Client Id"/> <add key ="GoogleClientSecret" value="Add your Google Client Secret"/>

8. Now, put your respective MVC Demo site hosted URL in Authorized redirect URL's text box & save the changes.

Note: Google support to add multiple sites URL's in one account, whereas Facebook supports only one URL per account.

© Copyright 2016, Znode LLC, All Rights Reserved.

16 Guidelines for Customization

Here are a few guidelines to follow when customizing Multifront.

Tracking Changes

- You should keep track of all custom code changes made to the Multifront code-base using a source control system like Visual Source Safe, SVN, CVS, etc. This will allow you to merge in changes when you upgrade to a newer version of Multifront.
- Znode uses Subversion (SVN) and Tortoise for our internal source control - We highly recommend this system. You can find out more about SVN at: subversion.tigris.org

Data Access

- Always add custom data access methods to the *Znode.Libraries.DataAccess.Custom* library. Do not modify the strongly typed data access layer by hand as it will potentially break compatibility for future versions.
- Add separate tables in the database instead of modifying existing tables if possible. This will ensure that you don't break the compatibility with the data access libraries.
- If you need to add columns to existing tables then set them as null-able columns - this will ensure backward compatibility with the data access libraries

Miscellaneous

- Always use the "~/" syntax to reference images and pages. This will ensure the correct reference paths when you move the framework to a hosted environment.

See Also:

Upgrading the Multifront 41

© Copyright 2016, Znode LLC, All Rights Reserved.

17 Upgrading the Multifront

Since Multifront is fully customizable, upgrading from one version to the next will require updates to the database schema and source code. Here we will outline just one strategy that you can use to upgrade your application.

Basic Concepts

- Znode does not support automatic upgrades for storefronts that have been customized or changed by the client.
- You can upgrade your storefront code-base by using a merge utility such as Beyond Compare or WinMerge
- You can upgrade your storefront database by using data compare utilities from Red Gate Software. Znode provides scripts for upgrading the database schema from one version to the next. These scripts should be considered just the starting point of your migration. Some reference tables may need to be updated according to your particular application. There is no direct upgrade path from the single store or previous editions of the storefront to Multifront.
- We do not recommend that you upgrade your storefront code simply to get bug fixes. Znode provides patch files for fixing bugs that pose a much lower risk to an existing production web site than a full upgrade.

Step 1: Back Up your Database and Source Code

This is the most important step. Znode is not responsible for any loss of data or code. When backing up your database we suggest using the Backup function that is built into the SQL Server Management studio rather than copying the database MDF files.

Step 2: Upgrade the Database

Use the scripts provided by Znode in the *Maintenance* directory to upgrade your database. You may need to apply several scripts, one for each version number between your current version and the version you want to upgrade to. Alternatively you can use a tools from Red Gate Software. This will give you finer control over what is upgraded.

Step 3: Separate out Your Code Changes

You can use a tool like WinMerge to create a zip file containing only the changes you have made to the storefront. Let's assume in this example that we will be upgrading the code from Version 6.2 to Version
6.3.

- Create a fresh install of Multifront from the version you based your current code on. In our example you will use the EXE installer to create a clean Version 6.2 installation on your development system.
- If you applied any Znode supplied patches to your current web site then also apply them to the clean v6.2 installation as well.
- Use WinMerge to compare your current web site source code with the v6.2 clean install. For this example make your current web site the first directory to compare and the clean v6.2 directory as the second to compare. This will show your code on the left pane and the clean v6.2.0 install in the right pane.

© Copyright 2016, Znode LLC, All Rights Reserved.

- In general you will want to filter out any files that do not have the aspx, ascx, asax, master or cs file extension. In most cases you can also exclude any files in the Data Access libraries (except the Custom library) since these are automatically generated. You can now right click on any files that you know should not be migrated and choose "Hide Items". You should be left with only the files that you have changed.
- From the Edit menu in WinMerge choose "Select All".
- Right click on the selected files and choose the "Zip" menu and then select "Left". This will package up only the files that you have changed into a Zip file with their directory structure intact.

Step 4: Create a New Web Site

Now it is time to create a new web site based on the latest version of the code.

- Use the EXEs from the latest version of the code (6.3 in this example) to create a clean installation on your local development system.
- Start up the demo store on your new installation to be sure that it is working correctly.
- Restore your upgraded database from Step 2 over your new development database and run the database upgrade script (or use Red Gate to upgrade). Test the storefront again. This will confirm that your database upgrade is correct.

Step 5: Merge in Your Code Changes

Now that you have the latest storefront working with your data, it is time to move over any specific template or code changes that you have made.

- Unzip the file that you created in Step 3 to an empty directory.
- Use WinMerge to compare this new directory with the upgraded development directory created in Step
  4. For this example select your Zip file directory first and your v6.3 directory second.
- In WinMerge go to the "View" menu and make sure that everything is unchecked in the first section except "Show Different Items" and "Show Left Unique Items". This will display only the files that need to be merged which should be a small subset of the total files in the project.
- Double click on each line item to see what the differences are. You may need to merge the files but in many cases you may just be able to copy your version of the file to the v6.2.0 directory.
- Once you merge these changes in you should be ready to test your upgraded storefront.

See Also:

Guidelines for Customization 40

Znode Multifront 8.1.2 Developer Guide
© Copyright 2016, Znode LLC. All Rights Reserved
Printed: October 2016 in USA.