Ucommerce Masterclass

Deep Dive Handouts

Table of Contents

Expertise 1010: Querying

Intro

LINQ to Ucommerce provides rich capabilities to query the data stores of Ucommerce, but with great power comes great responsibility: This exercise introduces the three levels of data APIs.

Relevant APIs

UCommerce.Infrastructure
    OjbectFactory.Resolve<T>

UCommerce.EntitiesV2
    IRepository<T>
        .Select()
    ISessionProvider
        .GetSession()

UCommerce.EntitiesV2
    Product
    PurchaseOrder
        .CreatedDate

NHibernate.Linq
    EagerFetchingExtensionMethods.Fetch()
    EagerFetchingExtensionMethods.FetchMany()

Quyering Excercises

Query for orders created after a certain date.

Query for products promoted to the homepage

- ProductProperties
- ProductDefinitionField
- “ShowOnHomepage”

Join product to order line on Sku and VariantSku

- Anonymous type for join { orderline.Sku, orderline.VariantSku }

N+1

- Query products
- Foreach over all products
- Access ParentProduct property in foreach
- Observe behavior in SQL Profiler

- How many queries do you see?

Eager Loading to Avoid N+1

- Use Fetch on ParentProduct property on product to tell NHibernate to initialize
- How many queries do you see?

Large Cartesian Products Avoided

- Set up two queries to load product and variants
- Use ToFuture on LINQ queries to defer execution
- Execute one query
- How many batches are executed in SQL Server (use the tuning template in profiler)

Total Control of SQL with HQL

- Use ISessionProvider to create a query
- Use keyword join, outer join, fetch to control SQL
- Watch the resulting queries in SQL Profiler (use the tuning template in profiler)

Expertise 1020: Override CatalogContext

Intro

CatalogContext determines which store, catalog, and price group the customer is using at any given time. In this exercise, we will override the active catalog based on whether the customer is logged in or not.

Relevant APIs

UCommerce.Runtime
- ICatalogContext
- CatalogContext
  - .CurrentCatalogName

Steps

New catalog "Private Catalog"

New category "My Private Category"

Add a few products to "My Private Category"

Add new class in business logic, "MyCatalogContext"

Override CurrentCatalogName property

Register MyCatalogContext in an App

Create a new folder in Apps

Remove existing components

Find existing catalog context in Configuration/Core.config

Copy in that component

Change type to use MyCatalogContext

Expertise 1030: Extending Pipelines

Intro

A typical part of ecommerce is integration. Whether it is products being synced in to the website or orders flowing back to your favorite 3rd part system, proper integration needs to be done. In this case we want to export the order. In this exercise we'll peek into one of the most core concepts of Ucommerce – Pipelines.

Relevant APIs

UCommerce.Pipelines
    IPipelineTask<T>

UCommerce.EntitiesV2
    PurchaseOrder
        OrderNumber

Steps

Create a new class for the pipeline task called "ExportOrderToErpSystem"

Register in new app called " ExportOrderToErpSystem " using a component

Hook into ToCompleted pipeline using partial-component

Use File I/O to write the OrderNumber into a text file.

Expertise 1040: Custom Data Type

Intro

When you need to be able to control how fields on products and categories are edited, a custom data type is just the thing. In this exercise you will create a new editor to set up tax groups on a per product level to support differentiated tax per product.

Relevant APIs

UCommerce.Presentation.Web.Controls
- IControlFactory
- IControlAdapter

UCommerce.EntitiesV2
- DataType
- PriceGroup
- IRepository<T>

Steps

Create a new class called "PriceGroupFactory"

Implement IControlFactory

Register component in a new app called "Differential Tax"

Set up a new data type in Ucommerce back-end in settings

Pick the Price Group component

Create a new field on a product definition using the new data type

Edit a product with the field set up

Expertise 1050: Override Tax Calculation

Intro

To support products with individual tax set we need to override ITaxService to make it take the product setting into account rather than the default tax info from the price group.

Relevant APIs

UCommerce.Catalog
- ITaxService
- TaxService

UCommerce.EntitiesV2
- Product
- PriceGroup

UCommerce.Extensions
- DynamicProperty<T>()

Steps

Set up a field with the PriceGroupControlFactory

Set a custom price group on a product

Inherit TaxService and override the CalculateTax method

Check the product for the field (take into account variants)

Load the price group based on the id stored in the field

Calculate tax based on the loaded price group