Ucommerce Master Class

Browse + Checkout Handouts

Table of Contents

Intro 010: Preparing for Master Class

Now that you've signed up for the Sitefinity + Ucommerce master class there's a few things you need to do in order to come prepared for the master class. Please read the following intro sections before the master class starts.

- Intro 020 Setup and pre-requirements
- Intro 030 Work flow
- Intro 040 Solution overview

Intro 020: Setup and pre-requirements

In the document you'll find a list of required software to attend the course. In this section please read how to get up and running. This guide assumes knowledge on how to configure your IIS, and how to install Sitefinity.

If you have any issues getting up and running please write to support@ucommerce.net and we will try to provide the help needed. Let us know that you're about to attend a Ucommerce for Sitefinity master class.

Preinstalled software

Visual Studio 2012/2013
SQL Server (Express or Standard)
SQL Server Management Studio
IIS 7
Latest version of Sitefinity (Sitefinity 10) up and running
Latest version of Ucommerce installed through NuGet

Add a website to IIS

It is highly recommended that you install and run your website under the IIS provided with Windows.

Install Ucommerce

Once you have a clean site up and running, please download the latest version of Ucommerce. The package Is installed through NuGet with the following command

Install-Package Ucommerce.Sitefinity

Once installed everything should be installed Automatically and you do not need to do anything else.

Intro 030: Work flow

Now that you've installed a clean version of Sitefinity and Ucommerce and opened the Master Class solution you're ready to build your brand new store using the solution provided.
The idea is simply that we'll push our website-components and extensions to the website that we are building using the "master class" solution provided. This can happen with the deploy script which is part of the solution (more of that later).

Intro 040: Solution overview

In this section we'll cover what's in the box before we start coding away. The solution is created so it resembles an example of how a "real life" Ucommerce project could look like, or as close as possible to what you're familiar with.

The solution projects

The solution is built up from 3 different projects:

- UCommerce.MasterClass.BusinessLogic
    - This will be used on the deep-dive part of the course primarily to create different extensions we will to deploy to the website but also on the integration project when exploring the query APIs.

UCommerce.MasterClass.Integration

- Console Application that can access the APIs and the Ucommerce data store. We will use this on the deep-dive part of the course when we explore the query APIs. This project has an assembly reference for the BusinessLogic project.

- UCommerce.MasterClass.Website
    - Website project where all resources for the site exists including:
        - Ucommerce configuration files to register custom components used for deep-dive
        - Models, Views, and Controllers for MVC components to scaffold the webshop
    - The purpose of the project is to push extensions and modifications to the site. This project has an assembly reference for UCommerce.MasterClass.BusinessLogic

The deploy tool

The deploy tool is used to push over all binaries, javascript files, style sheets, configuration files and more. The deploy tool is created as power shell files and will run a few scripts to deploy everything needed to the website.
To configure the deploy tool, follow the "Expertise 010".

Expertise 010: Setup Your New Clean MVC Site

Expexted time to complete: 15-20 minutes

Intro

For this exercise the purpose is to setup the first base template in your solution so we're ready to rock and roll right of the bat.

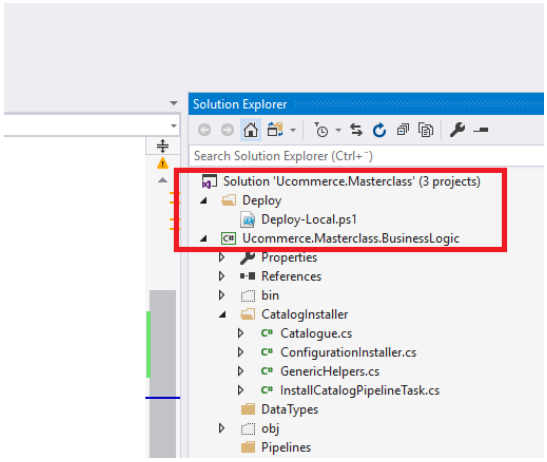
Requires

Clean installation of Sitefinity and Ucommerce.

Hands-on

The first thing we need to do is point our development environment to our running website.

PLEASE NOTE: you need to run Visual studio as administrator



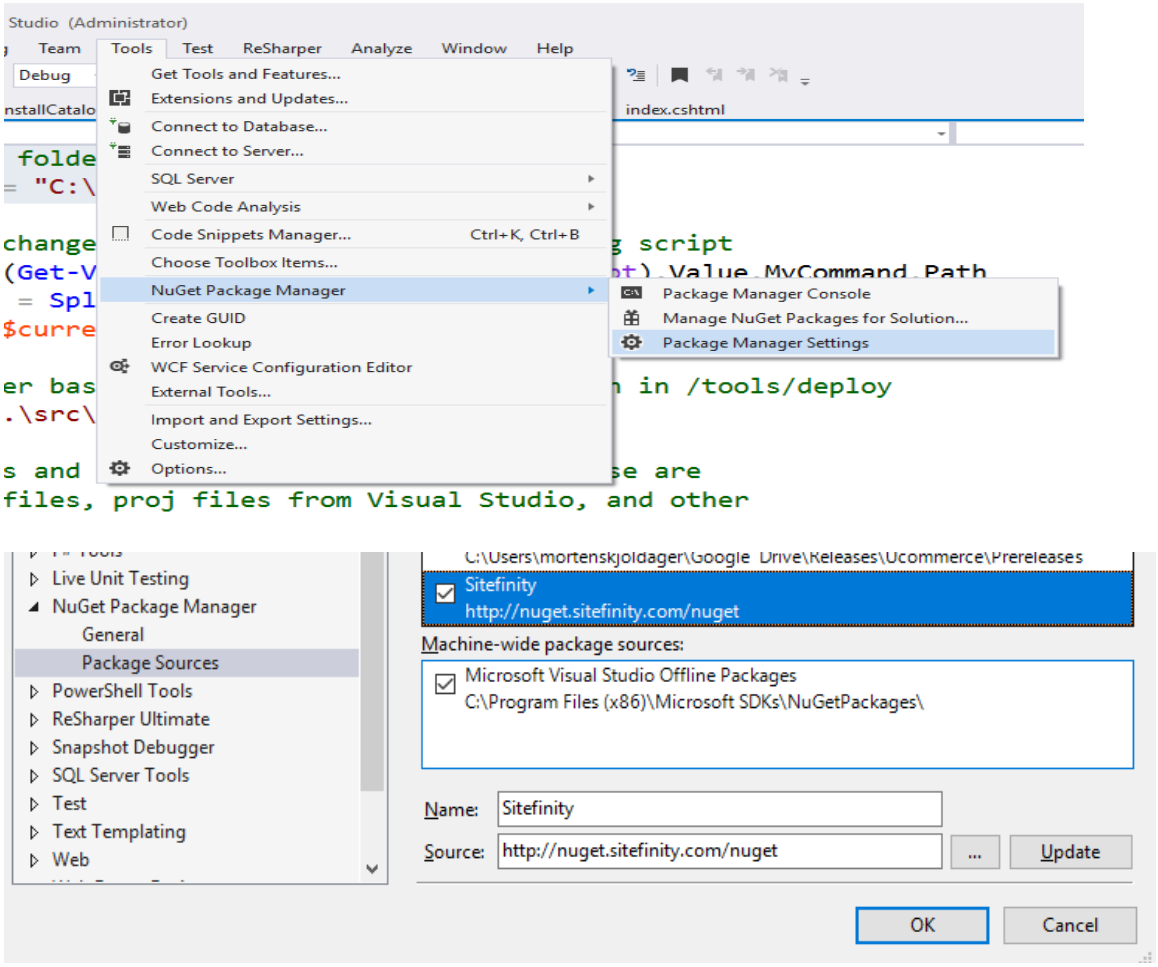
In the master class solution find the deploy-local.ps1 powershell script and open it. On line 1&2 you'll find a property we need to change. Right now it matches something, probably, different than your setup. It needs to point to the root of your Sitefinity and Ucommerce website (folder where web.config is located).

```
# Website root folder (website is deployed here)
$website_root = "C:\inetpub\Sitefinity\Website\CMS"
```

Simply modify the $website_root found in the deploy-local.ps1 file to match this and you're good to go on that part!

Next we need to be able to compile our solution. There's a bunch of NuGet packages that are referenced, including Sitefinity's packages for creating MVC widgets.

If you haven't done already, please make sure you have setup visual studio's package manager to include Sitefinitys package source. If you have already done that, you can skip this step.



In visual studio, go to tools, package manager and package manager settings.

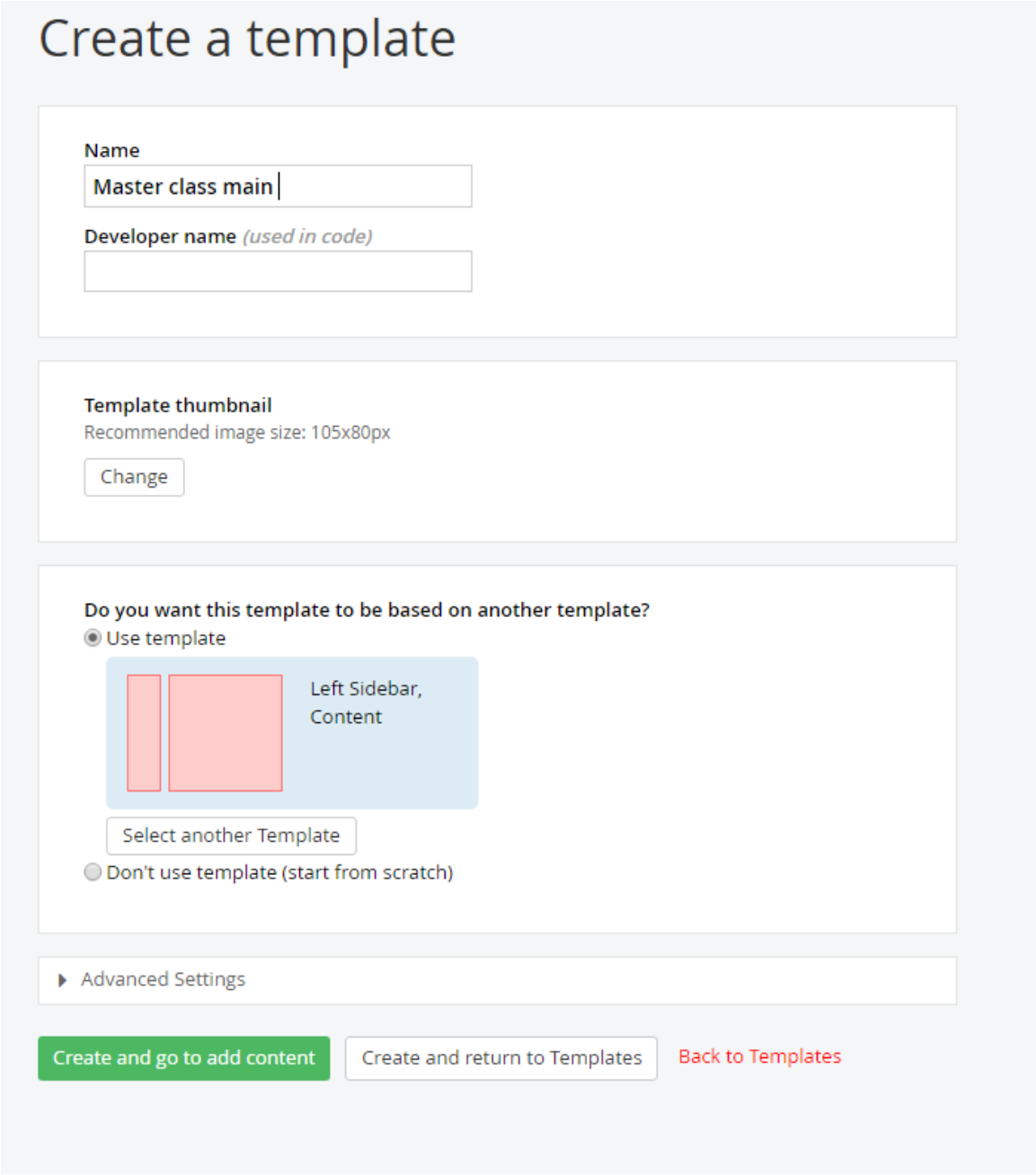
Go to package sources and add a new source for http://nuget.sitefinity.com/nuget

So far so good. Now please compile the solution and log into the Sitefinity backend.

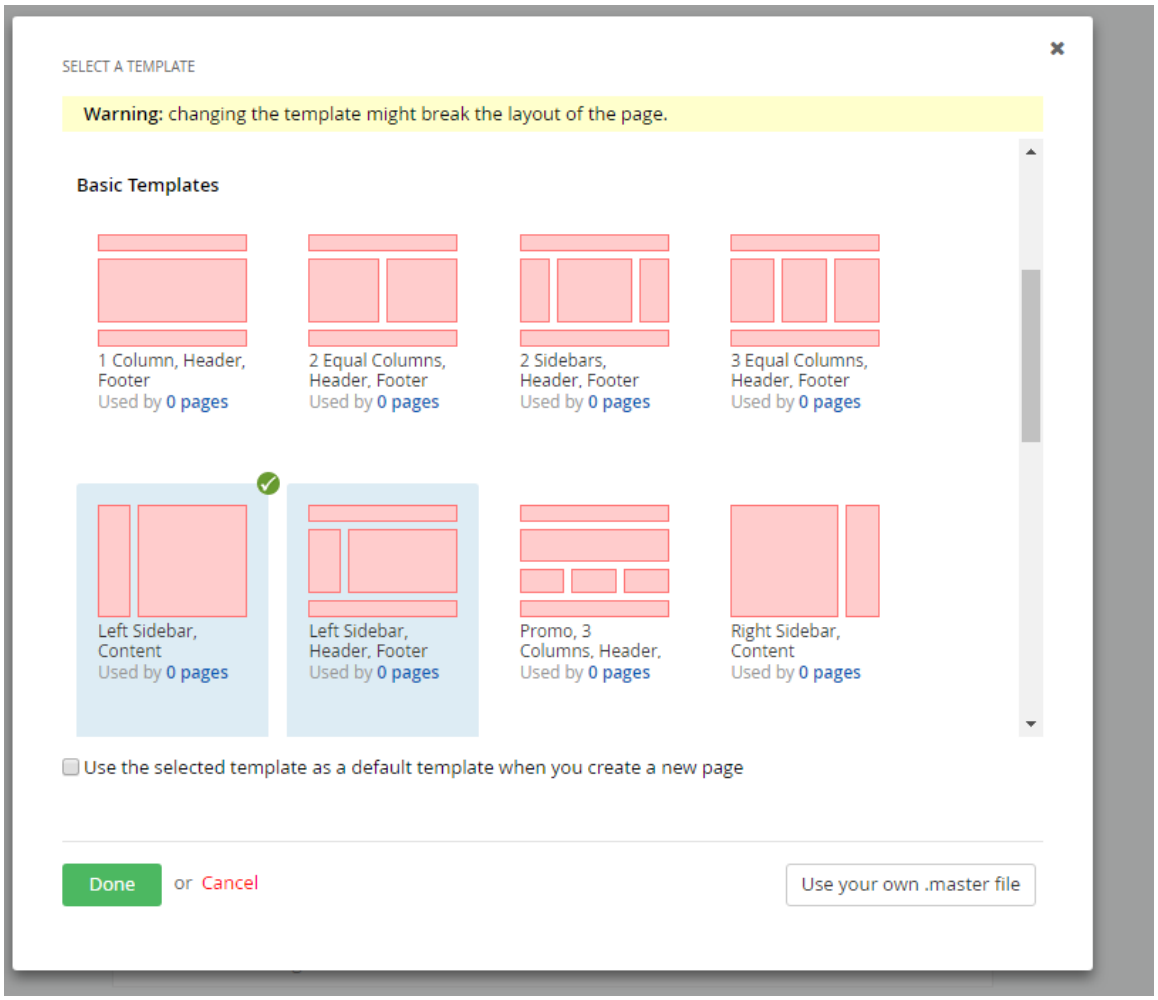
We're going to setup a template with a single widget that will be used throughout our pages.

In the top bar, in the design dropdown go to "page templates"

Click create a template and give it the name "Master class main".



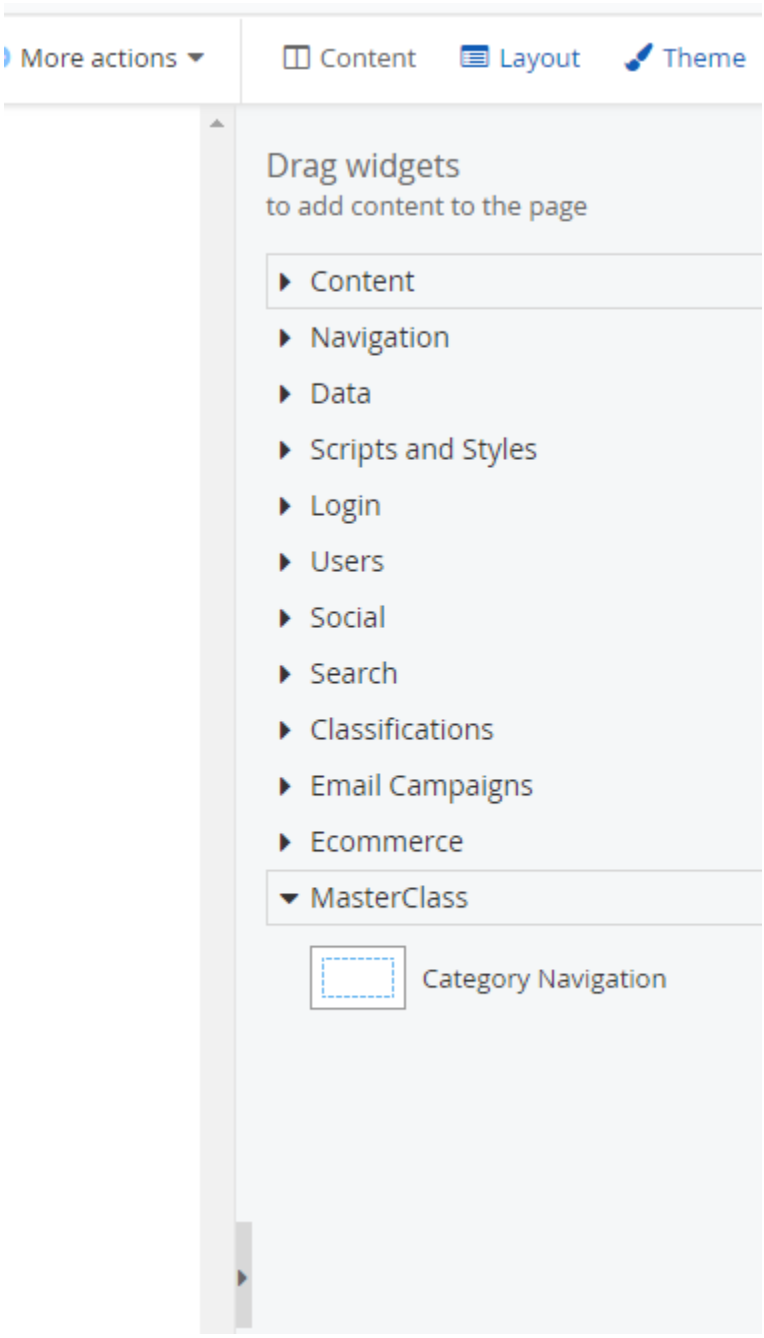
Click “select another template” and select the left sidebar content in the modal dialog.



Click on “done”

Click “create and go to add content”.

In the widget toolbox to the right, you should now see a section called “master class”.



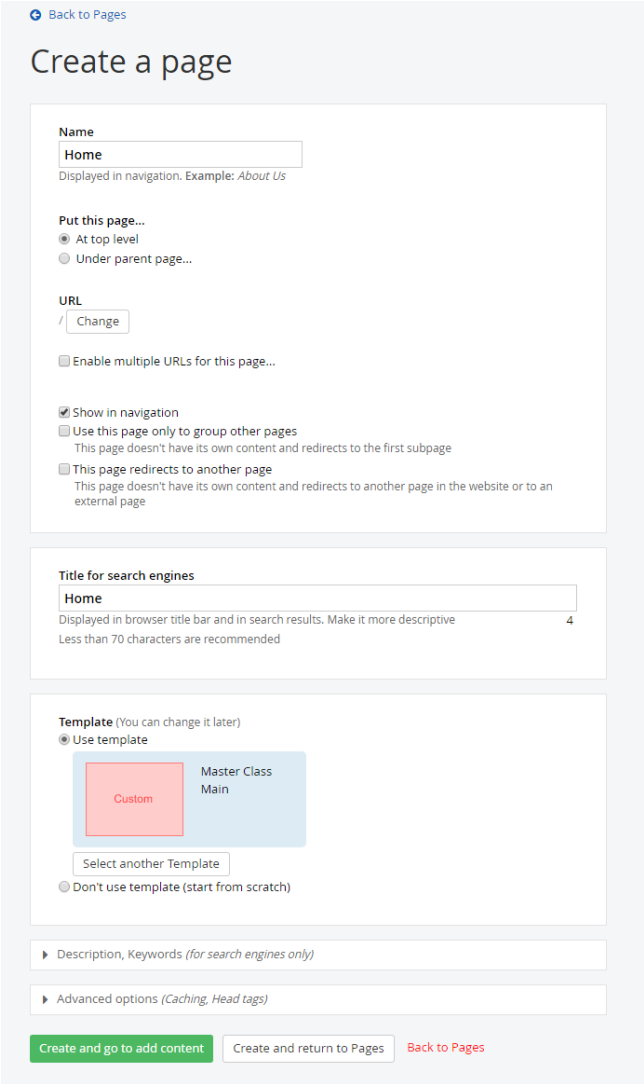
Drag in the Category Navigation in the left sidebar.

Publish the template.

Now go to “pages” in the topbar of sitefinity.

Click “Create a page”

Give it the name “Home” and select the template we created before.



Click create and go add content.

If you request the front-end on “/” you should now see a list of categories and otherwise a clean page.

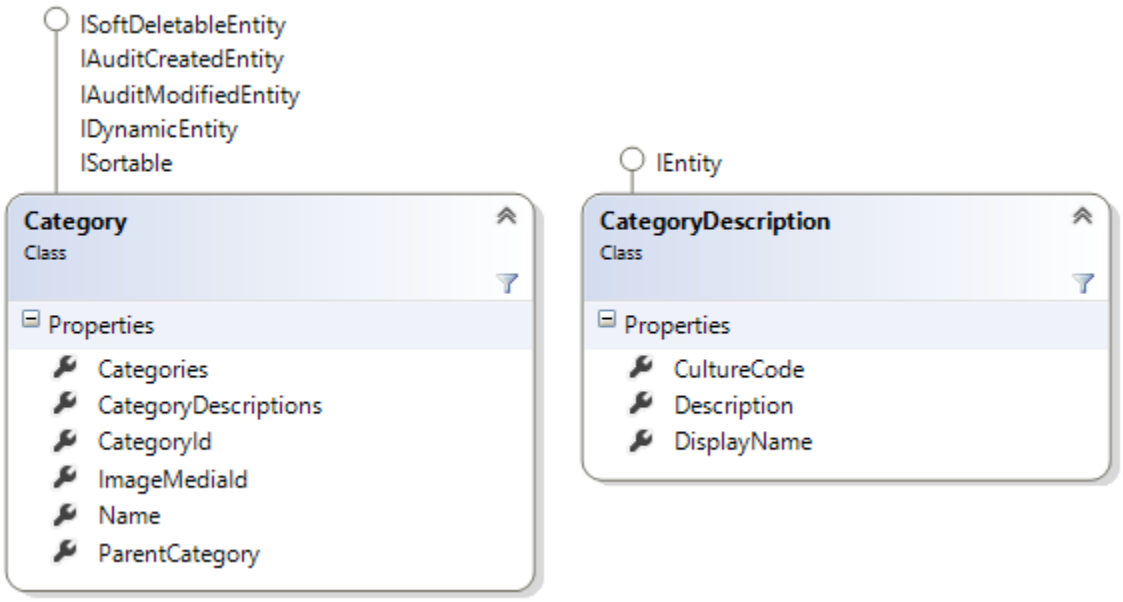
Congratulations. You’re now ready to attend the Ucommerce master class :D

Expertise 020: Browse - Category Navigation

Intro

Build a category listing for overall navigation of your store. You will gain knowledge of APIs relevant to loading categories and their related information along with an over-all understand of how to navigate the catalog structure.

Relevant APIs




```
UCommerce.Api
     CatalogLibrary.GetRootCategories(ProductCatalog)
     CatalogLibrary.GetCategories(Category)
     CatalogLibrary.GetNiceUrlForCategory() (optional)

UCommerce.Extensions
     CategoryExtensions.DisplayName()
```

Hands-on

Find the “PartialViewController” under the Controllers folder in the website project.

The Method CategoryNavigation() Renders the actionview “categoryNavigation.cshtml” as requested with the following line in “Layout.cshtml”

```
@{ Html.RenderAction("CategoryNavigation", "PartialView"); }
```

Find categories and sub categories using the CatalogLibrary and map them into the categorNavigationViewModel.Categories list

Map categories recursively

Add link to the categoryViewModel.Url that points to ‘/store/category?category=categoryId’

Bonus

Create a new store, assign domain to the new site.
Setup new catalog and new categories

Visit the new site. Does the category navigation change?

More information can be found on the documentation site:

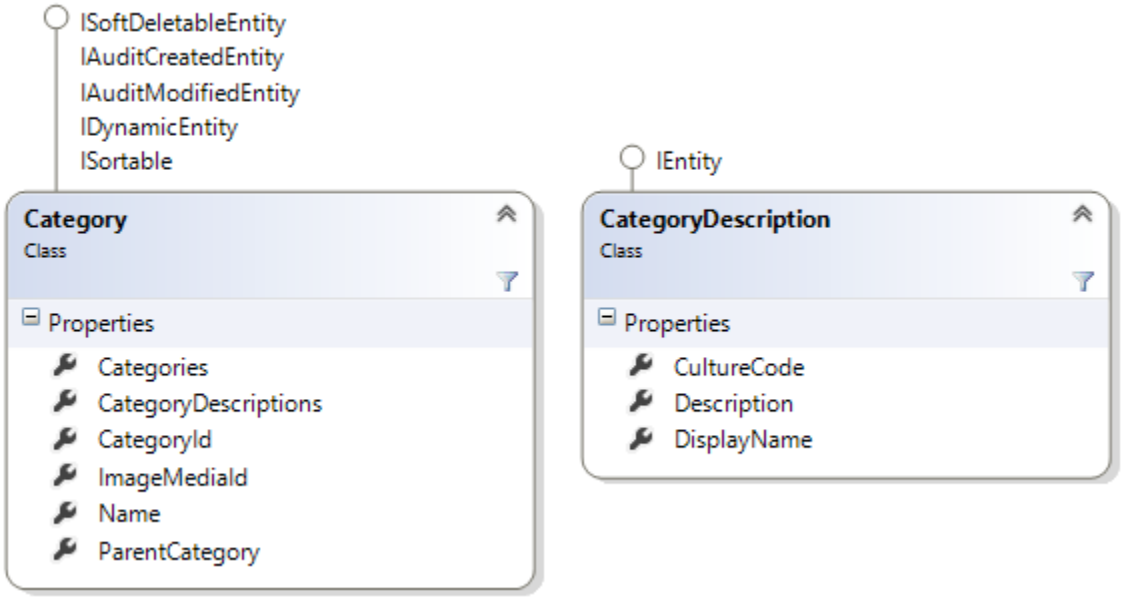
https://docs.ucommerce.net/ucommerce/v7.18/getting-started/catalog-foundation/catalog-structure.html

Expertise 030: Browse - Category Detail Page

Intro

Tease the contents of the category and get the customer excited.

Relevant APIs




```
UCommerce.Runtime
     SiteContext.Current.CatalogContext
          CurrentCategory

UCommerce.EntitiesV2.Category
      .ImageId

UCommerce.Extensions
      Category.DynamicProperty()
      Category.DisplayName()
      Category.Description()
```

Hands-on

Find the “CategoryController”.

The method “Index” renders the view “/views/category.cshtml” with the categoryViewModel.

Map “CurrentCategory” to the categoryViewModel with

- Name
- Description

Bonus

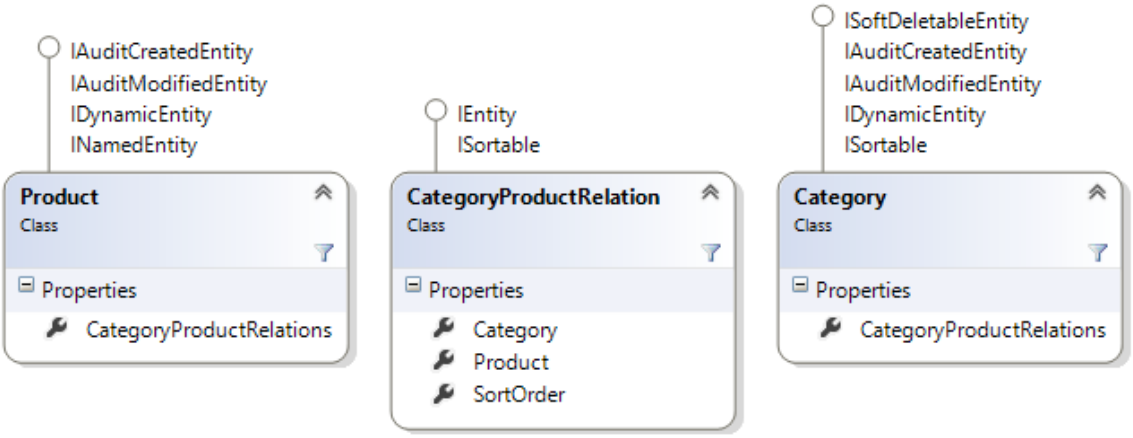
- Display category images using Sitefinity APIs.
    - Figure out if there’s a suitable property you can use, or you need to extend the CategoryViewModel with a property to hold the image

Expertise 040: Browse - Product Listing (Category Detail Page)

Intro

Build a product listing based with products in a given category. You will gain knowledge of APIs relevant to loading products and categories efficiently from Ucommerce as well as dealing with prices and simple discounts.

Relevant APIs




```
UCommerce.EntitiesV2.Product
      PrimaryImageMediaId
      ThumbnailImageMediaId

UCommerce.Runtime
     SiteContext.Current.CatalogContext
          CurrentCatalog
          CurrentCategory

UCommerce.Api
     CatalogLibrary.GetProducts(category)

UCommerce.Extensions
     Product.DynamicProperty()
     Product.DisplayName()
     Product.ShortDescription()
     Product.LongDescription()

UCommerce
     Money
```

Hands-on

In the CategoryController you need to map the Products property on the CategoryViewModel to hold the list of products in CurrentCategory.

Bonus

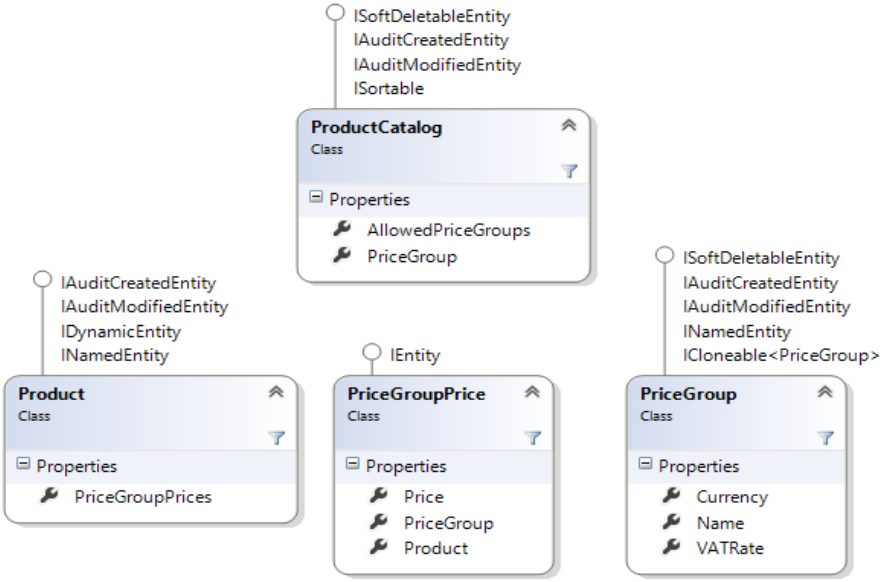
- Display product images using Sitefinity APIs.
    - Does the ProductViewModel contain enough fields to do so?

Expertise 050: Browse - Prices and Simple Discounts (Product Listing)

Intro

Add price and tax information to your product pages. Learn how Ucommerce applies price information to your products based on your catalog configuration and how products can have multiple prices.

Relevant APIs




```
UCommerce.Api
     CatalogLibrary.CalculatePrice(Product, ProductCatalog)

     PriceCalculation
           YourPrice <-- Price incl simple discounts
           ListPrice
           Discount
           YourTax
           IsDiscounted

     Price
           AmountExclTax
           Amount
           AmountInclTax

     UCommerce.EntitiesV2
           Product
           ProductCatalog
```

Hands-on

On the productViewModel you need to set the PriceCalculation coming from the API

Bonus

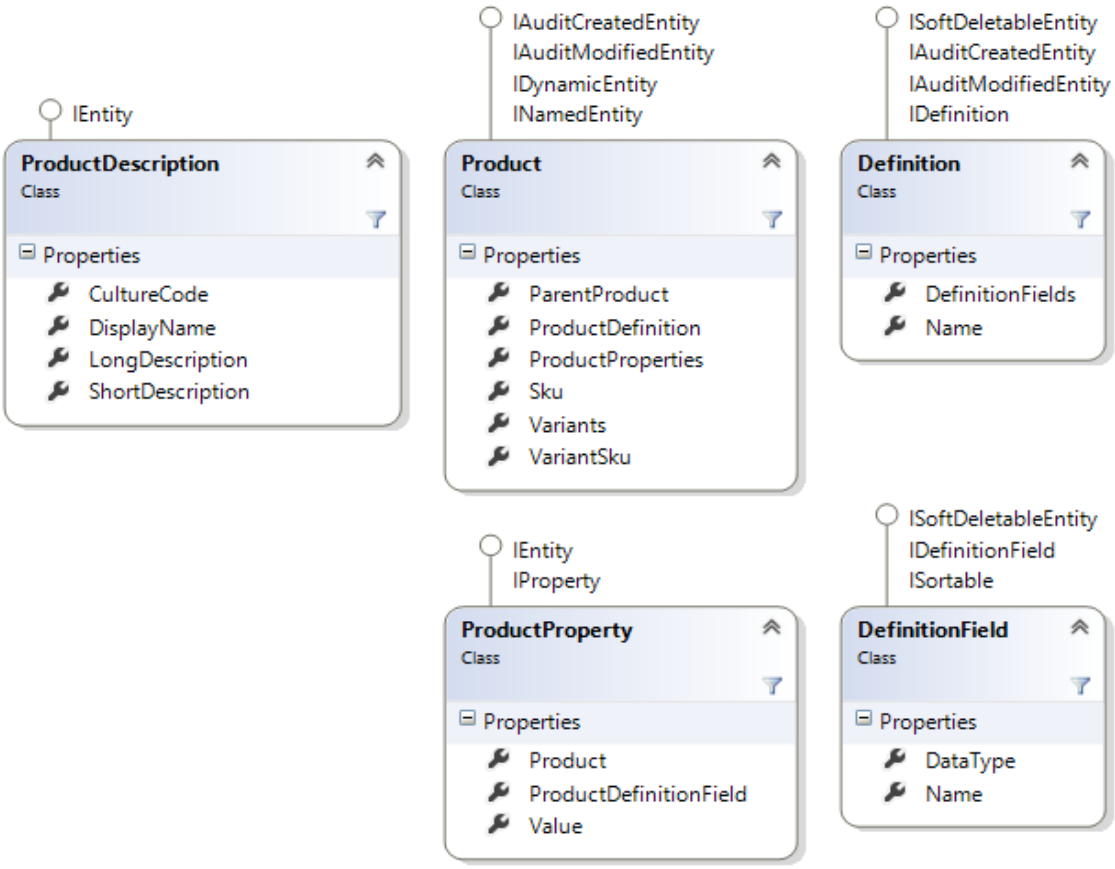
- Set up a unit discount and display the discount on the product listing.
- Add the original price with a dash through if the price is discounted

Expertise 060: Browse - Product Detail

Intro

Build a product detail page, which will delight customers using the server-side APIs. Learn how to display language specific content via dynamic properties. As a bonus we'll dive into the client-side APIs as well.

Relevant Concepts and APIs




```
UCommerce.Runtime
     SiteContext.Current.CatalogContext
          CurrentProduct
          CurrentCatalog

UCommerce.Api
     CatalogLibrary.CalculatePrice(Product, Catalog)
     CatalogLibrary.GetRelatedProducts(productId, relationType)

UCommerce.Extensions
     DynamicEntityExtensions.DynamicProperty()
     ProductExtensions.DisplayName()
     ProductExtensions.Description()
```

Hands-on

Find the “ProductController”.

The method “Index” renders the view “/views/product.cshtml” with the productViewModel.

Map “CurrentProduct” from the CatalogLibrary to the productViewModel with

- Name
- Description
- Sku
- VariantSku
- LongDescription
- Variants
- PriceCalculation

Bonus

- Display product images using Sitefinity APIs.
- Explore related products on CatalogLibrary using the following line of code:
    - CatalogLibrary.GetRelatedProducts(productId).SelectMany(x => x.Value)

Expertise 070: Browse - Add to Basket

Intro

First step towards making an honest buck is getting customers to add items to the basket. Discover the server-side APIs to get it done and sneak a peek at the client-side APIs available in the Razor Store.

Relevant Concepts and APIs

```
UCommerce.Api
     TransactionLibrary.AddToBasket(
          quantity,
          sku,
          variantSku = null,
          decimal? unitPrice = null,
          PriceGroup priceGroup = null,
          addToExistingLine = true,
          executeBasketPipeline = true
)
```

Hands-on

The sku and VariantSku from the page are posted back the POST method in your product controller. Use the API above to add the product to the basket.

Bonus

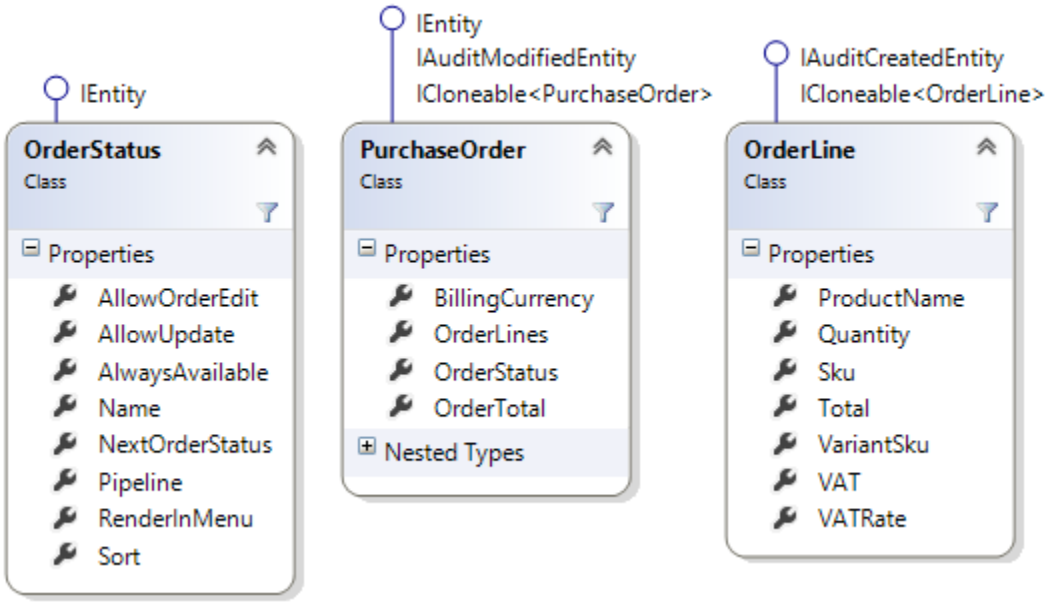
Add another field to class “AddToBasketViewModel” that takes a quantity. Also modify the form to allow the user to enter a quantity by modifying the Product.cshtml under the views folder

Expertise 210: Checkout – View Basket

Intro

Customers need an overview of what they're about to buy and also a way to manage the items they've put in their basket.

Relevant APIs




```
UCommerce.Api
     TransactionLibrary.GetBasket().PurchaseOrder

UCommerce.EntitiesV2
     PurchaseOrder
          OrderLines
          Discounts
     OrderLine
     Discount

UCommerce
     Money
```

Hands-on

Find the “BasketController”

The method “Index” renders the view “/views/Basket.cshtml” with the PurchaseOrderViewModel.

Map the customer’s basket into the viewModel by grabbing the order from the TransactionLibrary

Expertise 215: Checkout – Update Basket

Relevant APIs

```
UCommerce.Api
      TransactionLibrary.UpdateLineItem(orderLineId, quantity)
      TransactionLibrary.ExecuteBasketPipeline()
      MarketingLibrary.AddVoucherCode(voucherCode)
```

Hands-on

In the POST method in your BasketController you need to update each line item with either the new quantity or remove it if the user clicks the remove button.

You also need to execute the basket pipeline so the order is kept up-to-date. Read about pipelines here:

http://docs.ucommerce.net/ucommerce/v6.8/getting-started/transaction-foundation/pipelines-explained.html

Expertise 220: Checkout - Billing/Shipping Information

Intro

When time comes to complete the order the customer need to supply their billing and shipping details so we know where to ship their items.

Multiple shipments supported (PurchaseOrder.Shipments).
Order addresses are stored on a per order level (OrderAddress).
Shared customer addresses stored per customer (Address).

Relevant Concepts and APIs

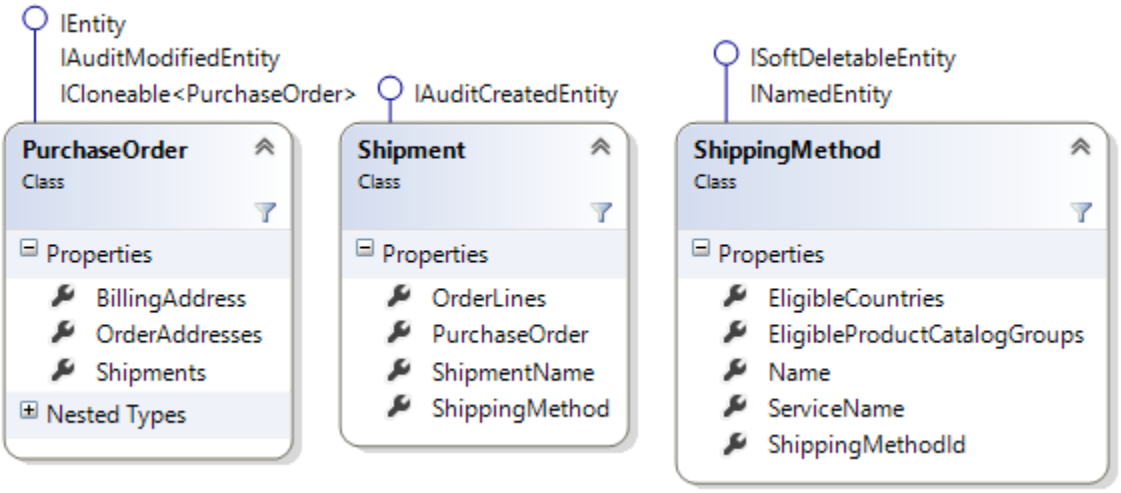
```
UCommerce.Api
     TransactionLibrary.GetBillingInformation(
          firstName,
          lastName,
          emailAddress,
          phoneNumber,
          mobilePhoneNumber,
          company,
          line1,
          line2,
          postalCode,
          city,
          state,
          attention, countryId)
     TransactionLibrary.GetShipmentInformation()
     TransactionLibrary.EditBillingInformation()
     TransactionLibrary.EditShipmentInformation()

UCommerce.EntitiesV2
     PurchaseOrder
          BillingAddress
          Shipments

     OrderLine
          Shipment

     OrderAddress

     Shipment

     Country
```

Expertise 230: Checkout - Shipping Method

Intro

Display available shipping methods for the customer to select. Shipping methods can be filtered on store level and country level.

Relevant Concepts and APIs




```
UCommerce.Api
     TransactionLibrary.GetShippingInformation()

     TransactionLibrary.GetShippingMethods(country = null)

      TransactionLibrary.GetShippingMethod(shipmentName = null)

UCommerce.EntitiesV2
     Shipment

     ShippingMethod
          GetPriceForCurrency(currency)

UCommerce
     Money(amount, currency)
```

Hands-on

Find the “ShippingController”
The method Index() renders the “Views/ShippingMethods.cshtml” view.

Map the shippingmethods available into the viewmodel.

Expertise 235: Checkout – Update Selected Shipping Method

Relevant APIs

```
UCommerce.Api
      TransactionLibrary.CreateShipment(
            shippingMehtodId,
            addressName = null,
            overwritingExisting = true)

      TransactionLibrary.ExecuteBasketPipeline()
```

Hands-on

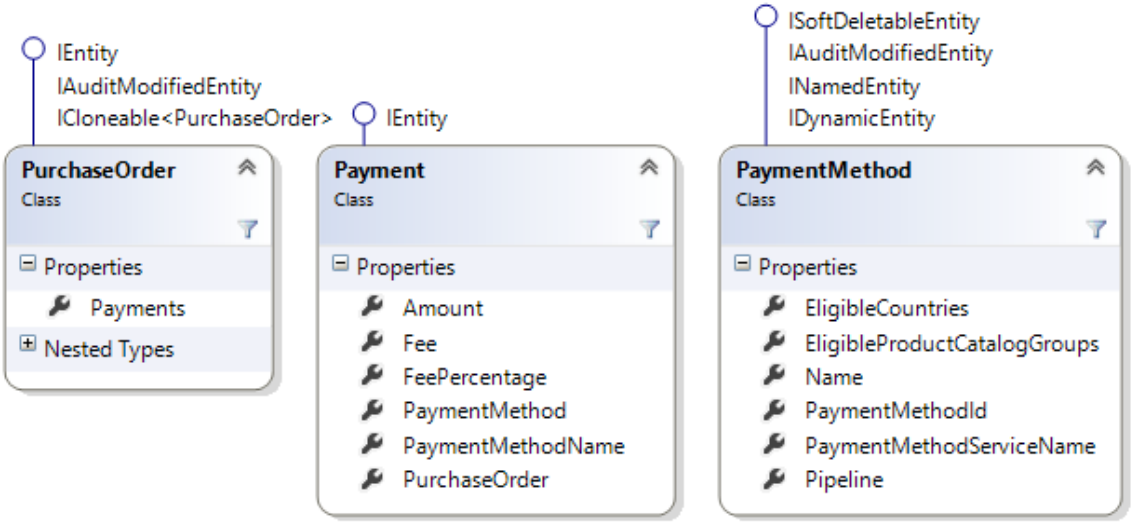
In the POST method of your ShippingController you need to create a new shipment based on selected shippingmethod.

Expertise 240: Checkout - Payment Method

Intro

Display available payment methods for the customer to select. Can be filtered based on country.

Relevant Concepts and APIs




```
UCommerce.Api
      TransactionLibrary.GetBillingInformation()

      TransactionLibrary.GetPaymentMethods(country = null)

UCommerce.EntitiesV2
      PurchaseOrder
            Payments
      Payment
      PaymentMethod

UCommerce
      Money(amount, currency)
```

Hands-on

Find the “PaymentController”.

The method Index() renders the view “/views/Payment.cshtml”” with the PaymentViewModel. Map available payment methods filtered by country into the model.

Expertise 245: Checkout – Update Selected Payment Method

Relevant APIs

```
TransactionLibrary.CreatePayment(
      paymentMethodId,
      amount = -1,
      requestPayment = true,
      overwriteExisting = true)

TransactionLibrary.ExecuteBasketPipeline()
```

Hands-on

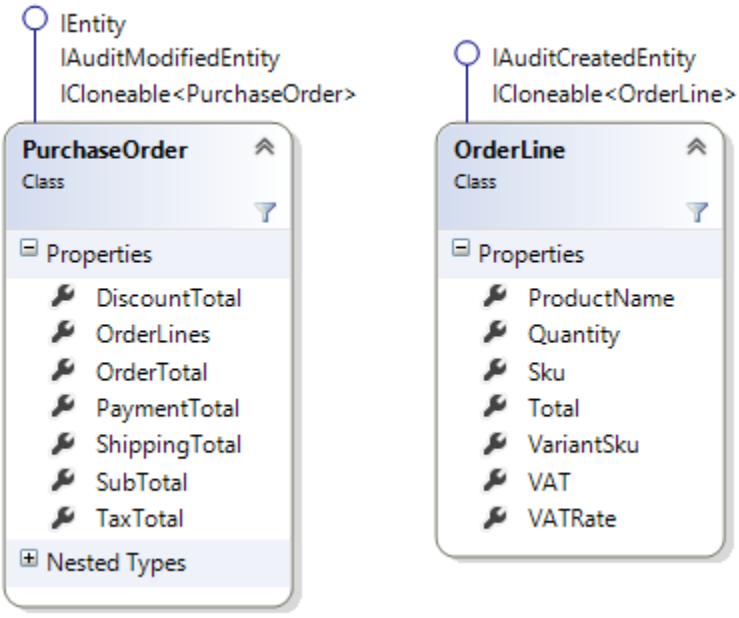
In the POST method in your PaymentController you need to update current payment by creating a new one that overrides the existing one. Use the TransactionLibrary to do so.

Expertise 250: Checkout - Order Preview

Intro

Many countries require online stores to show a complete picture of what the customer is buying and paying before completing an order. Discover the APIs and properties available to help you in this.

Relevant Concepts and APIs




```
UCommerce.Api
     TransactionLibrary.GetBasket()
     TransactionLibrary.RequestPayments()

UCommerce.EntitiesV2
     PurchaseOrder
          SubTotal
          TaxTotal
          Discount (discounts applied to the order itself)
          DiscountTotal (all discounts applied at any level)
          PaymentTotal
          ShippingTotal

          OrderLines
          Shipments
          Payments
```

```
OrderLine
          Sku
          VariantSku
          Quantity
          Price
          Discount
          VAT
          Total

     Shipment

     Payment

     Discount

UCommerce
     Money
```

Hands-on

Find the Preview controller.

Map the basket into the PurchaseOrderViewModel

When ready to purchase, the user will click checkout. In the POST method use TransactionLibrary to fulfill the payment.

Expertise 260: Checkout - Sending E-mail

Intro

Keep the customer in the loop via e-mail notifications during checkout and order processing. Discover how to use e-mail templating in Ucommerce.

Relevant APIs

```
UCommerce.Api
     TransactionLibrary.GetPurchaseOrder(Guid guid)
```

Hands-on

Modify the Preview controller to try and get the order from the OrderGuid appended in the QueryString. That way we can reuse the preview page to present the same data in the Email.