

Ucommerce Masterclass

Browse + Checkout Handouts

Table of Contents

Intro 010: Preparing for the Masterclass.....	3
Intro 020: Setup and pre-requirements.....	4
Intro 030: Work flow	5
Intro 040: Solution overview	6
Expertise 001: Modify the Deploy Script.....	8
Expertise 010: Setup Your New Clean MVC Site	8
Expertise 020: Browse - Category Navigation	9
Expertise 030: Browse - Category Detail Page.....	11
Expertise 040: Browse - Product Listing (Category Detail Page).....	13
Expertise 050: Browse - Prices and Simple Discounts (Product Listing).....	15
Expertise 060: Browse - Product Detail	17
Expertise 070: Browse - Add to Basket.....	19
Expertise 220: Checkout - Billing/Shipping Information	23
Expertise 230: Checkout - Shipping Method	24
Expertise 235: Checkout – Update Selected Shipping Method.....	26
Expertise 240: Checkout - Payment Method	27
Expertise 245: Checkout – Update Selected Payment Method.....	28
Expertise 250: Checkout - Order Preview	29

Intro 010: Preparing for the Masterclass

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

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

Intro 020: Setup and pre-requirements

In this 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, install Umbraco + Ucommerce.

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 Umbraco Masterclass.

Preinstalled software

Visual Studio 2012/2013/2014/2015

SQL Server (Express or Standard)

SQL Server Management Studio

IIS 7

Latest version of Umbraco 7 up and running

Latest version of Ucommerce installed

Latest version of Ucommerce Demo Store installed

Add a website to IIS

We recommended that you install and run your Umbraco website under the IIS provided with Windows. However if you want to install and run Umbraco and Ucommerce in visual studio using NuGet packages then that will do just fine as well.

Install Ucommerce

If you're running Umbraco out of NuGet you can install Ucommerce as well using

"Install-Package uCommerce.Umbraco7"

Otherwise you can find and download the packages for Umbraco 7 here

<http://www.ucommerce.net/en/products/download.aspx>

Once downloaded login to the backend of Umbraco and install the zip file through the developer section in the backend. Once this is done you should have Ucommerce installed and running.

Install Avenue-Clothing

Once Ucommerce is installed you need to install Avenue-Clothing as well. This can be done by following the same approach as installing Ucommerce.

Download the demo store from the same URL listed above. When that is done you should see Avenue-Clothing running on the roots for your hostnames of your website.

Intro 030: Work flow

Now that you've installed a clean version of Umbraco + Ucommerce + Demo store we're ready to start developing our brand new MVC store.

To do this we need a solid foundation to start from, download and open the visual studio solution provided from here:

<https://bitbucket.org/uCommerce/ucommerce-master-class/get/Clean%20Visual%20Studio%202013%20version.zip>

The idea is simply that we'll push our website-components and extensions to the website that we are building using the "Masterclass" solution provided above. To achieve this we'll use a little powershell script provided with the solution.

Intro 040: Solution overview

Now that you've installed a clean version of Umbraco + Ucommerce + Demo store and opened the Masterclass solution you're ready to build your brand new store using the snippets provided in the solution. In this section we'll cover what's in the box before we start coding away. The solution is created so it resembles a "real life" Ucommerce project as close as possible. Of course best practices varies and is a matter of personal preference so it may be slightly different from how you would choose to do it.

The solution will only contain the extensions we're building along with the configuration files we're modifying.

The solution projects

The solution is built up from 3 different projects:

- MyUCommerceApp.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.
- MyUCommerceApp.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.
- MyUCommerceApp.WebsiteWebsite
 - 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
 - Few javascript files and styles sheets including Bootstrap for bells and whistles 😊
 - The purpose of the project is to push extensions and modifications to the site. This project has an assembly reference for MyUCommerceApp.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". This is the first exercise of the Masterclass and is not required on beforehand.

It's all MVC

The first part of the Masterclass is to build your own version of a webshop using MVC. Here's what comes out of the box with the Masterclass solution:

- Views
 - Under the views folder we've created all the markup needed to present the e-commerce data for the visitors. If you're interested and/or want to modify it you're more than welcome to do so.
- Models
 - Under the models folder in the Masterclass we've created the view models needed for the views. There's nothing fancy to it – only classes with a few properties used in each of the views.
- Controllers
 - Under the Controllers folder you'll find all the controllers which purpose is to set the view and the model. The controllers responsibility is also to handle postbacks when the user clicks submit. The exercises will primarily be based in here.

Congratulations! You're now ready to become a Ucommerce Rockstar 😊

Expertise 001: Modify the Deploy Script

Open the provided Solution.

Modify the `$website_root` variable found in "Deploy-Local.ps1" under the Deploy folder in top of the solution explorer.

The variable needs to point to the root of your website.

When you build your solution a post-build event will push all the relevant content from the website project. **Make sure that visual studio runs in administrator mode**

Expertise 010: Setup Your New Clean MVC Site

Intro

For this exercise the purpose is to setup an additional site that runs along with the demo store installed for your clean installation of Umbraco, Ucommerce, and avenue-clothing. It is assumed that both Ucommerce and the demo store is installed on your clean version of Umbraco, so the site right now shows avenue-clothing on the front-end.

Requires

Clean installation of Umbraco, Ucommerce and Avenue-Clothing.

Hands-on

Go to the Settings section of Umbraco.

Right click the Document Types folder and click create.

Create a **Document Type without a template**.

Set the name to "**MasterClassHome**" (this is important otherwise Umbraco won't create the right link between the doc type and the controller).

Under "Permissions" allow the document type in the root.

In the content section of Umbraco, Create a new node called "Store" using the "Home" document type.

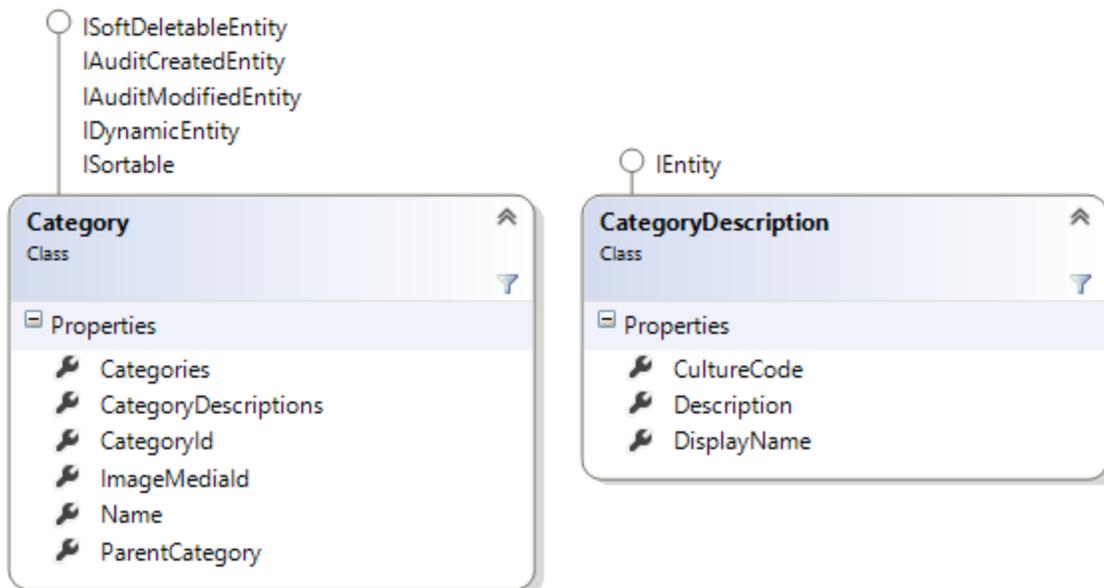
Request /store in your browser and you should see "Hello Masterclass" in another layout. This is where you wanna be right now 😊

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 “MasterClassPartialViewController” 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 `categoryNavigationViewModel.Categories` list

Map the categories recursively by adding sub categories into the model as well using `CatalogLibrary` to find the children.

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

Bonus

Try assigning domains to each of the root nodes in Umbraco.

Create a new store, assign different domains to each of the stores. Setup a new catalog and new categories under the store as well.

Visit the new site. Does the category navigation change?

More information can be found on the documentation site:

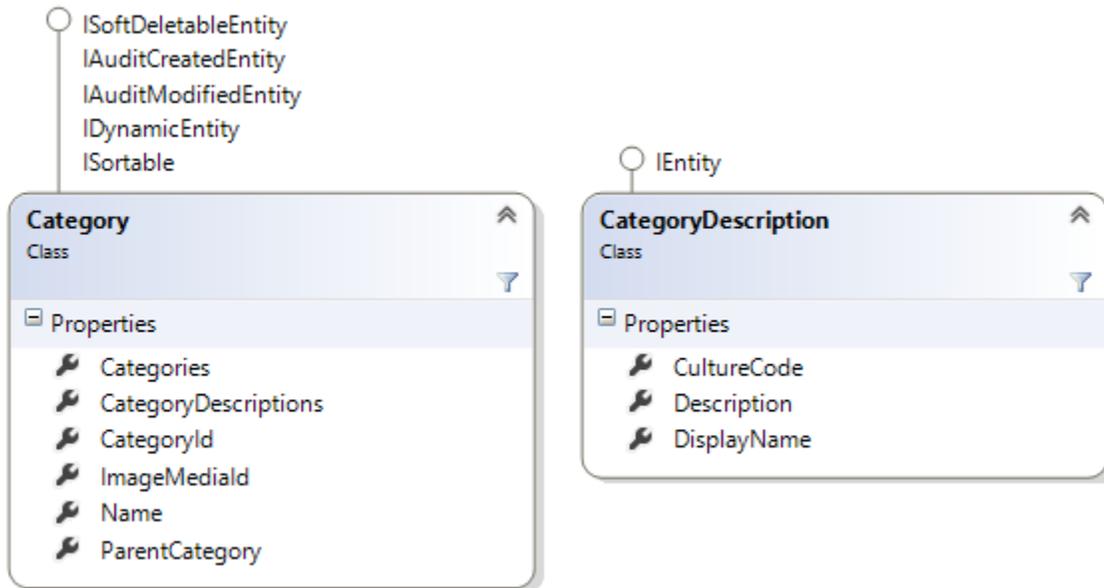
<http://docs.ucommerce.net/ucommerce/v6.8/getting-started/catalog-foundation/catalog-structure.html>

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 “MasterClassCategoryController”.

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

Map “CurrentCategory” to the categoryViewModel with

- Name
- Description

Bonus

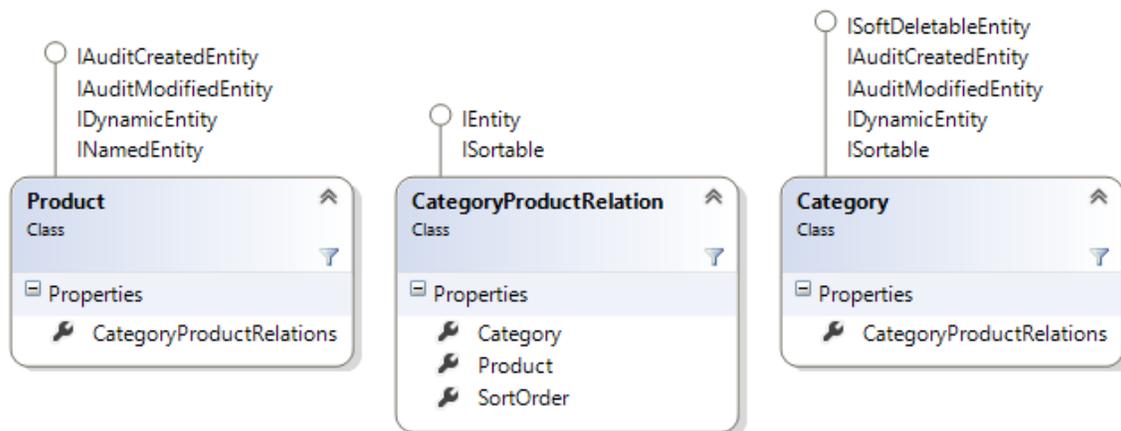
- Display category images using Umbraco APIs.
 - Figure out if there’s a suitable property you can use, or you need to extend the CategoryViewModel with a property to hold the image

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

Bonus

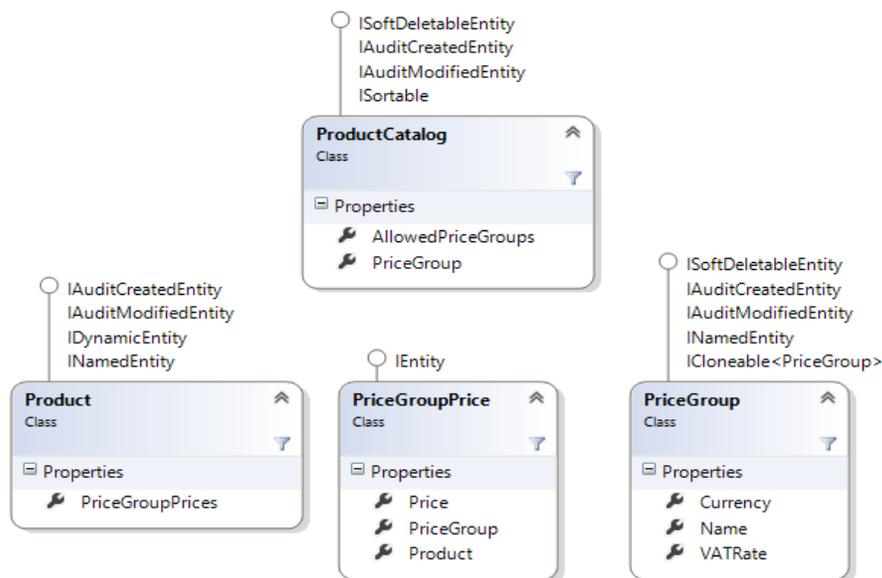
- Display product images using Umbraco APIs.
 - Does the `ProductViewModel` contain enough fields to do so or do you need to add in extra fields?

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

Amount
 AmountExclTax
 AmountInclTax

UCommerce.EntitiesV2

Product
 ProductCatalog

Hands-on

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

Bonus

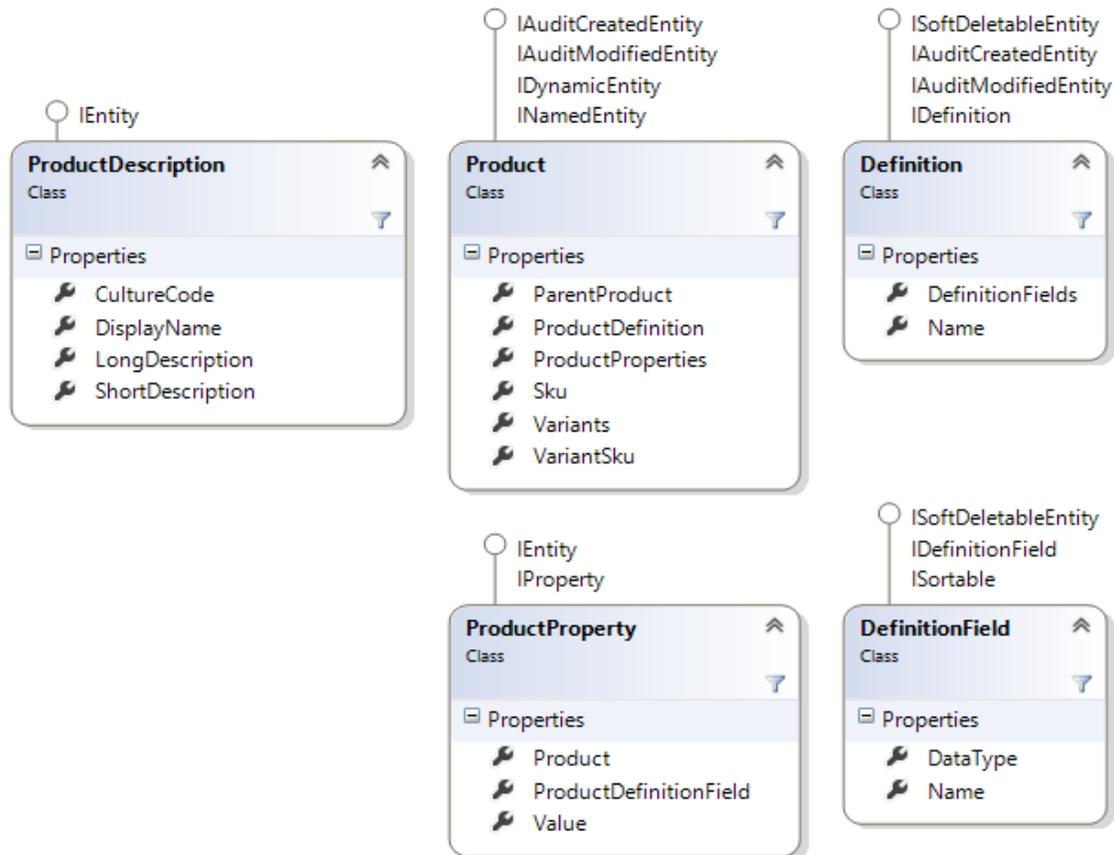
- Set up a unit discount and display the discount on the product listing.
- Add the original price with a dash through if the price is discounted

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 “MasterClassProductController”.

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 Umbraco 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,  
    addToExistingLine = true,  
    executeBasketPipeline = true,  
    catalogId = null)
```

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 1

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

Bonus 2

The Masterclass project ships with JQuery extensions that allows you to add a product async using the code snippet below.

Use your javascript ninja-skills to collect the information relevant to adding the product to basket and use that approach instead. You need to collect both Sku, VariantSku and the Quantity.

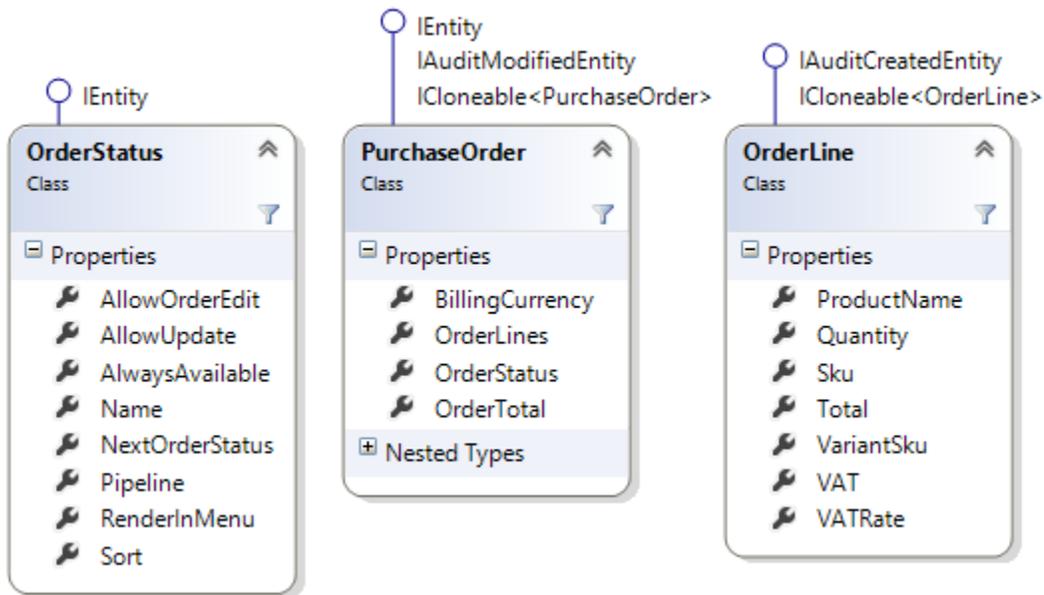
```
$.uCommerce.addToBasket(  
    {  
        sku: "",  
        variantSku: "",  
        quantity: 1,  
        addToExistingLine: true  
    },  
    onSuccess,  
    onError)
```

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 “MasterClassBasketController”

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 MasterClassBasketController 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. You can read more about pipelines here:

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

Bonus

Continue the javascript gig from before by updating the basket using the codesnippet below. You need to collect the orderLineId and the Quantity entered from the basket table.

```
$uCommerce.updateLineItem(  
  {  
    orderLineId: $(this).data("lineitemid"),  
    newQuantity: $(this).val()  
  },  
  function() {},  
  function() {}  
)
```

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

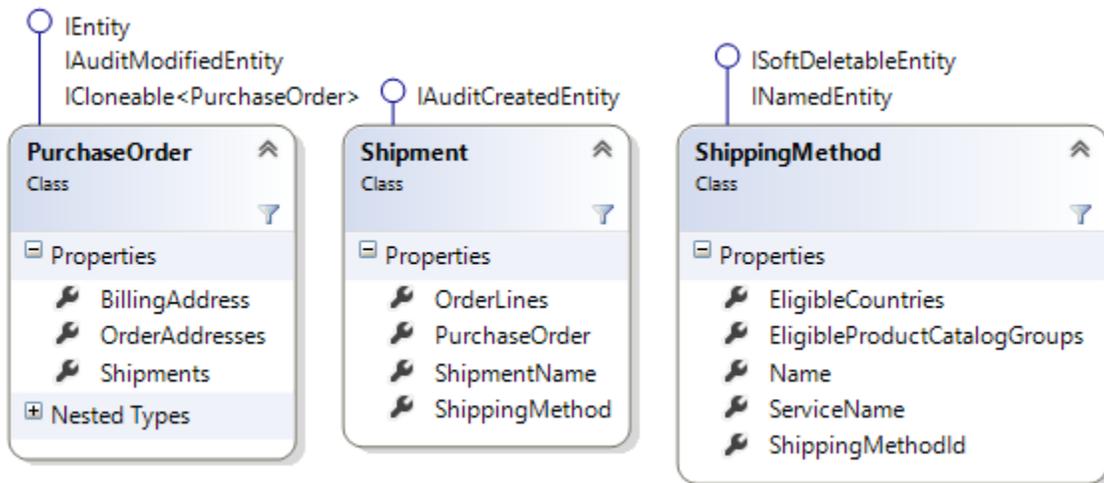
Country

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 “MasterClassShippingController”

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

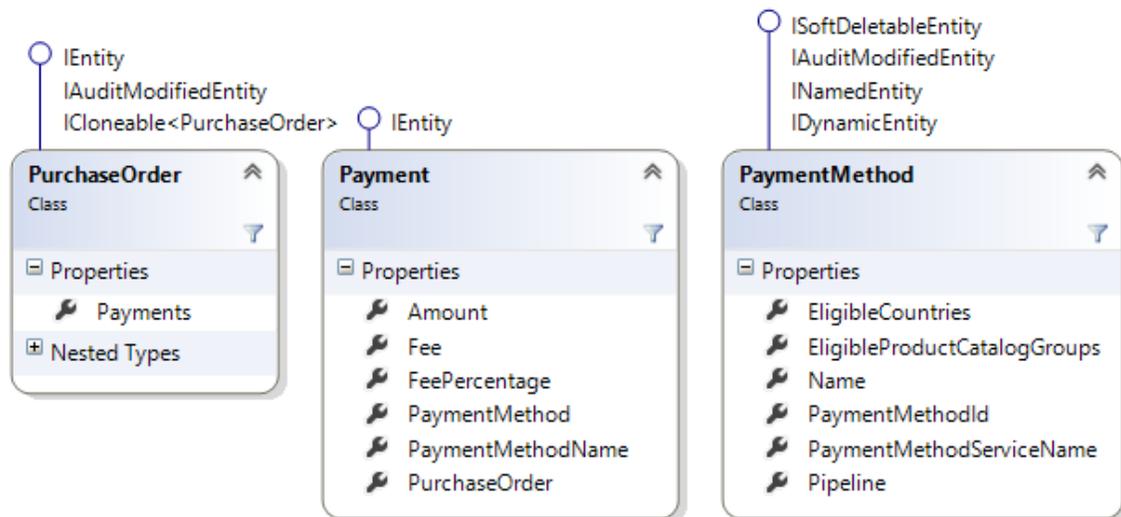
In the POST method of your ShippingController you need to create a new shipment based on selected shippingmethod.

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 "MasterClassPaymentController".

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

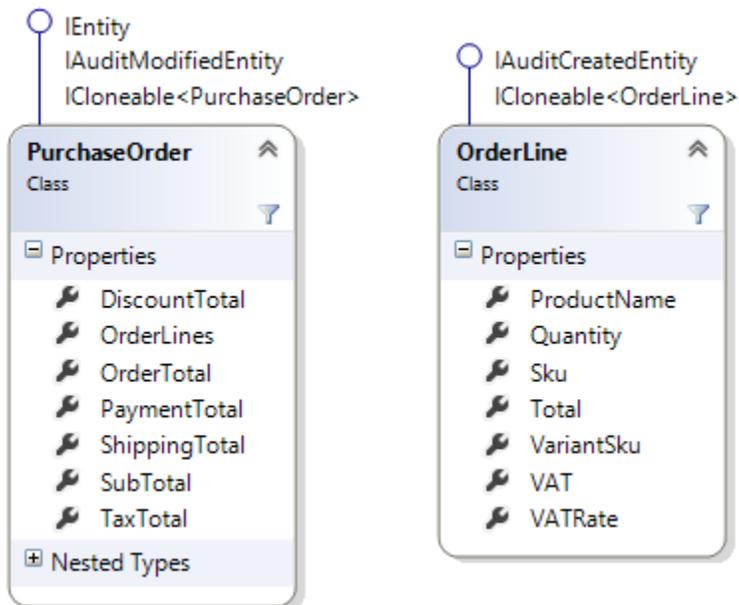
In the POST method in your `MasterClassPaymentController` you need to update current payment by creating a new one that overrides the existing one. Use the `TransactionLibrary` to do so.

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 MasterClassPreviewController.

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.