Extend Sitecore Commerce Engine

Vlad Iobagiu

Sitecore Meetup Cluj Napoca, May 2019

About me

Freelancer Solution Architect

Sitecore Technology MVP (2014-2016)

Sitecorian (2016-2018)

Sitecore Commerce MVP (2019)

Sitecore Stackexchange Addicted (https://sitecore.stackexchange.com/)

Blog: https://sitecoreclimber.wordpress.com

Twitter: https://twitter.com/SitecoreClimber

Agenda

History of Sitecore Commerce

Commerce Installation

Business Tools Overview

Commerce Views service Architecture

Extend Commerce with Plugins (when it runs my custom minions, show my environment configurations)

Demo

Sitecore Commerce History

Sitecore has a long history on Commerce:

- Sitecore E-Commerce Fundamental Edition 1.1
- Sitecore SES (Sitecore E-Commerce Services)
- Sitecore Commerce Server acquired in 2014 (Previously known as Microsoft Commerce Server-first release was in 1996)
- Sitecore Commerce powered by Commerce Server
- Sitecore Commerce powered by Microsoft Dynamics
- Sitecore Commerce 8.2.1 (April 2017)
- Sitecore Commerce 9 (January 2018)

Other commerce modules:

- UCommerce for Sitecore
- ActiveCommerce
- Sitecore Insite

Installation guide

Be prepared for

Installation guide

It took me 4 hours to install XC 9.1.

Installation guide

Instalation is made using SIF, is the most complex Sitecore installation.
Default scripts install SXA for Commerce (accelerator for Commerce).
In XC 9.1 you can skip tasks related to SXA for Commerce.
Installation Guide:

- For On-Premises - http://commercesdn.sitecore.net/SitecoreXC_9.0/Installation-Guide/Sitecore-XC-9.0_Installation_Guide(On-Prem).pdf
- for Azure App Service - http://commercesdn.sitecore.net/SitecoreXC_9.0/Installation-Guide/Sitecore-XC-9.0_Installation_Guide(Cloud).pdf

Scripts for scaled environment:

- https://github.com/asmagin/sitecore-installation-scripts
- https://community.sitecore.net/technical_blogs/b/the_commerce_experience/posts/commerce-scaled-deployment-guidance

READ PRE REQUIREMENTS VERY CAREFULLY

Sitecore Commerce SSL Certificates

A range of certificates are generated during the install process

1. Commerce Engine IIS websites is an **SSL *server certificate***
2. Identity Server Certificate: ***a code signing certificate*** and is used by Sitecore Identity Server to sign and validate identity
3. Sitecore Commerce Engine Connect Certificate: is a **SSL client certificate** required to authenticate clients to Commerce Engine using certificate authentication.

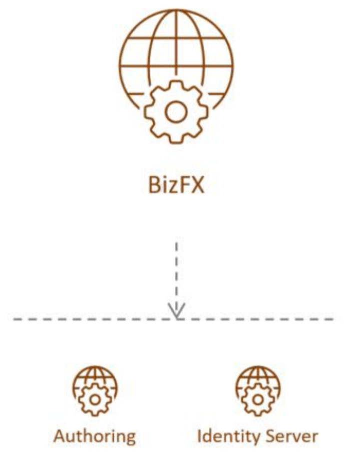
PAY ATTENTION TO CERTIFICATES

https://sitecoredude.com/the-dudes-guide-to-certificates-in-sitecore-experience-commerce-9/

In XC 9.1 I didn't have to create any commerce certificate.

Business Tools

- single page application built in Angular
- hosted as a separated website in IIS or an Azure App
- user authentication is provided by Identity Server



Business Tools

Sitecore XC Business Tools provide six key e-commerce features:

Catalog and sellable item management

Inventory management

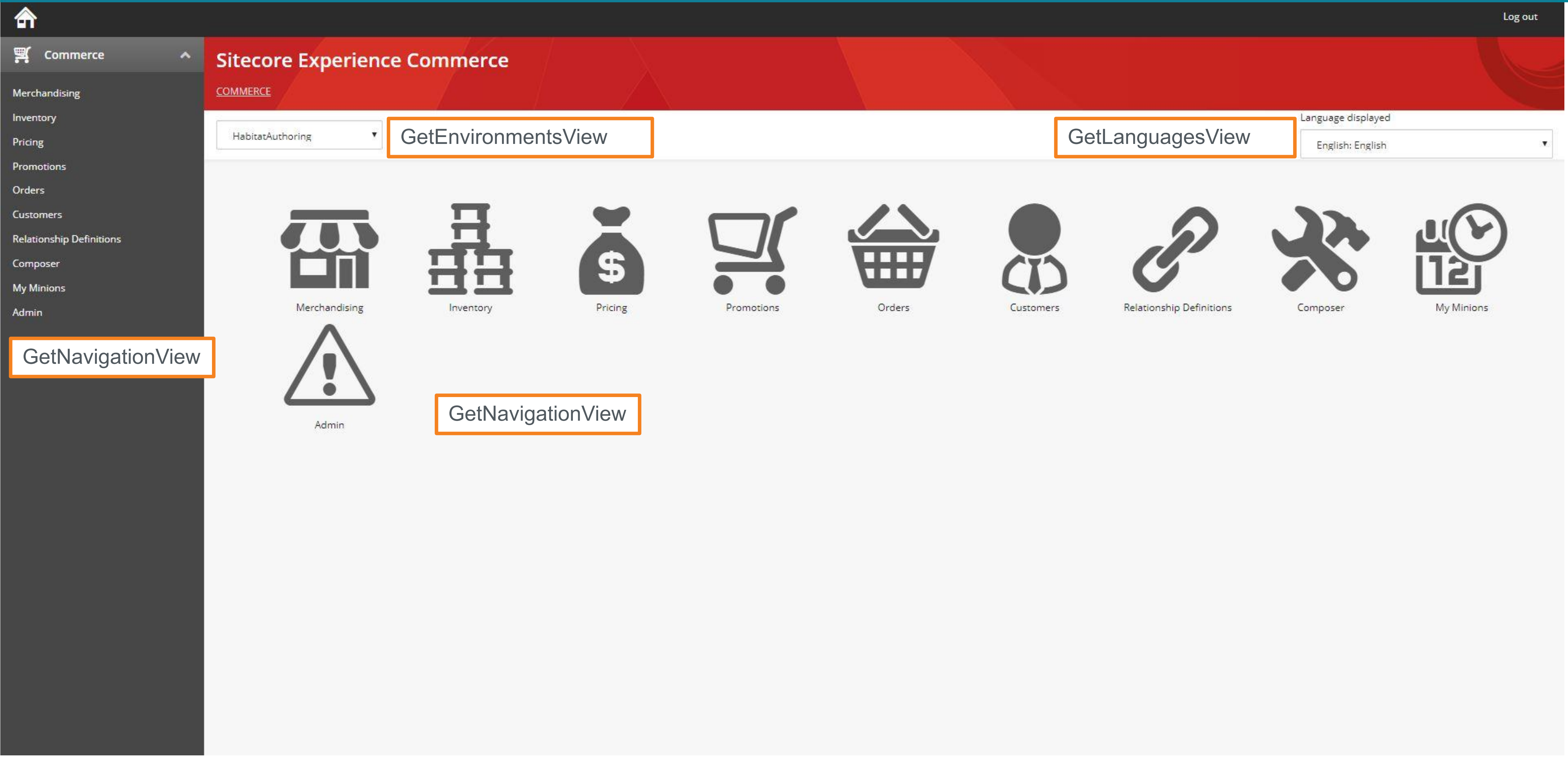
Customer management

Order management

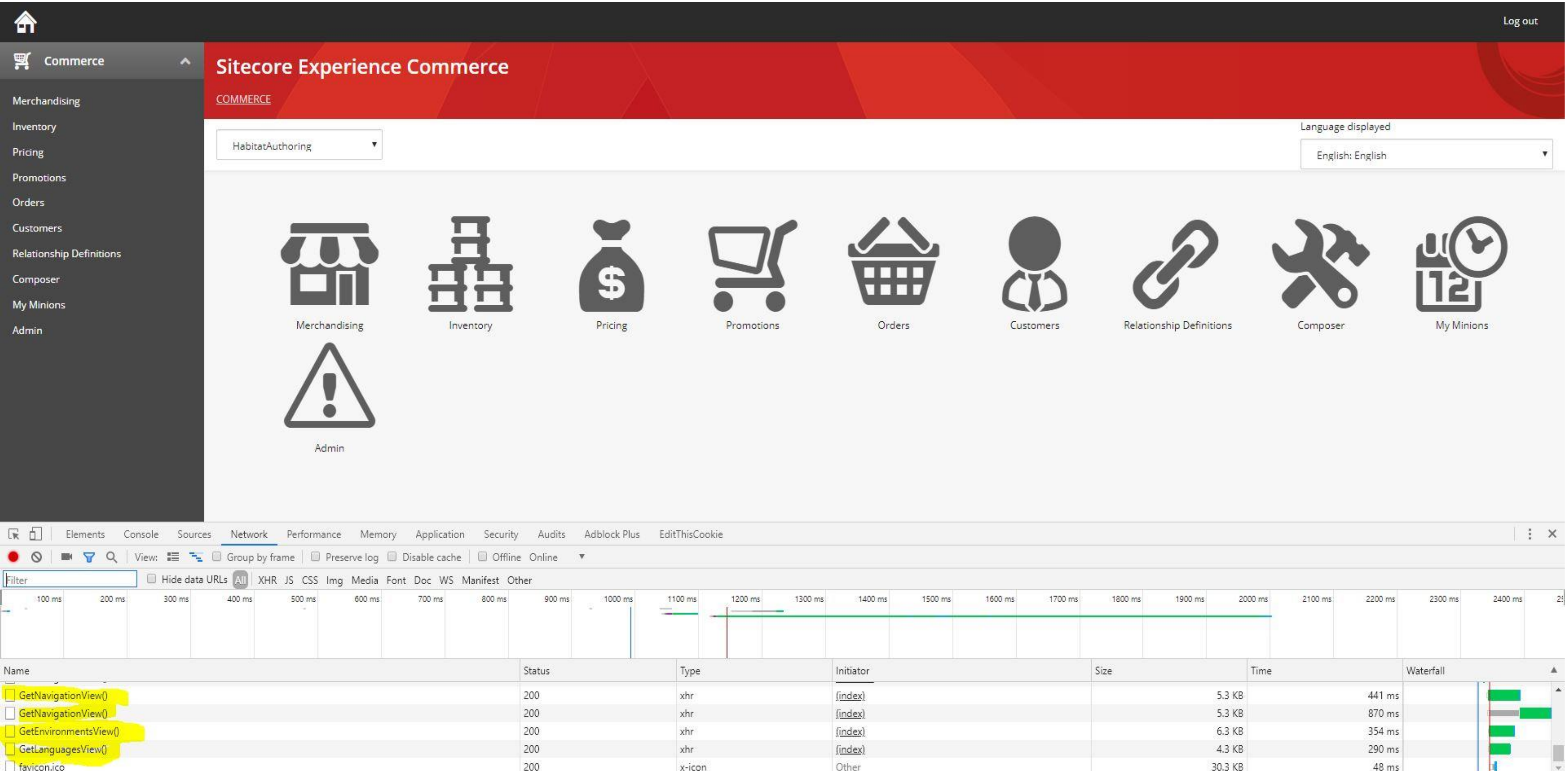
Promotion management

Pricing management

Business Tools Overview



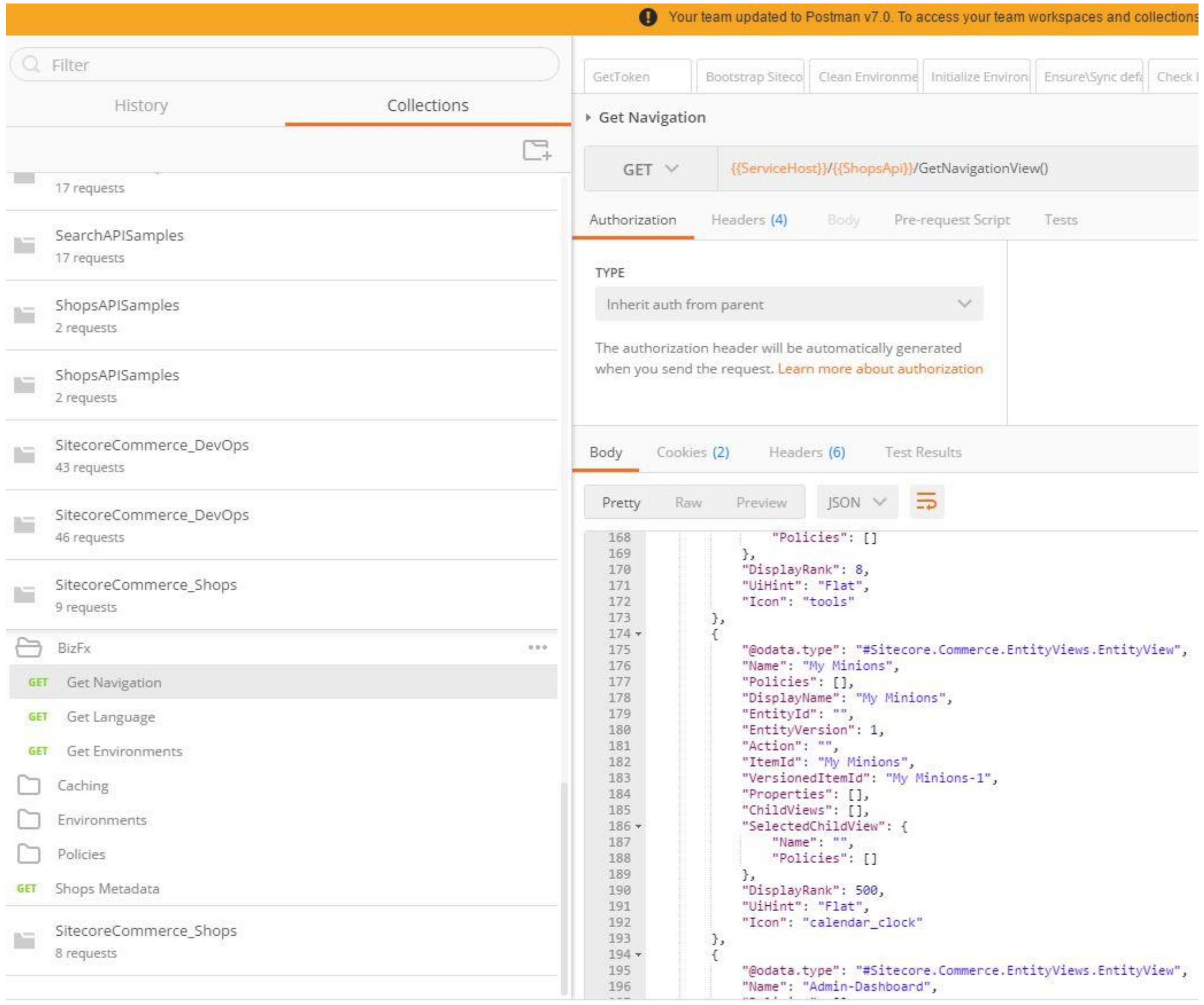
Business Tools Overview


Log out
Commerce
Merchandising
Inventory
Pricing
Promotions
Orders
Customers
Relationship Definitions
Composer
My Minions
Admin
Sitecore Experience Commerce
COMMERCE
HabitatAuthoring
Language displayed
English: English
Merchandising
Inventory
Pricing
Promotions
Orders
Customers
Relationship Definitions
Composer
My Minions
Admin
Elements Console Sources Network Performance Memory Application Security Audits Adblock Plus EditThisCookie
View: Group by frame Preserve log Disable cache Offline Online
Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other
100 ms 200 ms 300 ms 400 ms 500 ms 600 ms 700 ms 800 ms 900 ms 1000 ms 1100 ms 1200 ms 1300 ms 1400 ms 1500 ms 1600 ms 1700 ms 1800 ms 1900 ms 2000 ms 2100 ms 2200 ms 2300 ms 2400 ms

Name	Status	Type	Initiator	Size	Time	Waterfall
GetNavigationView()	200	xhr	(index)	5.3 KB	441 ms	
GetNavigationView()	200	xhr	(index)	5.3 KB	870 ms	
GetEnvironmentsView()	200	xhr	(index)	6.3 KB	354 ms	
GetLanguagesView()	200	xhr	(index)	4.3 KB	290 ms	
favicon.ico	200	x-icon	Other	30.3 KB	48 ms	

Business Tools Overview

Postman scripts to test Bizfx api.



17 requests

SearchAPISamples
17 requests

ShopsAPISamples
2 requests

ShopsAPISamples
2 requests

SitecoreCommerce_DevOps
43 requests

SitecoreCommerce_DevOps
46 requests

SitecoreCommerce_Shops
9 requests

BizFx
- GET Get Navigation
- GET Get Language
- GET Get Environments
- Caching
- Environments
- Policies
- GET Shops Metadata

SitecoreCommerce_Shops
8 requests

GET {{ServiceHost}}/{{ShopsApi}}/GetNavigationView()

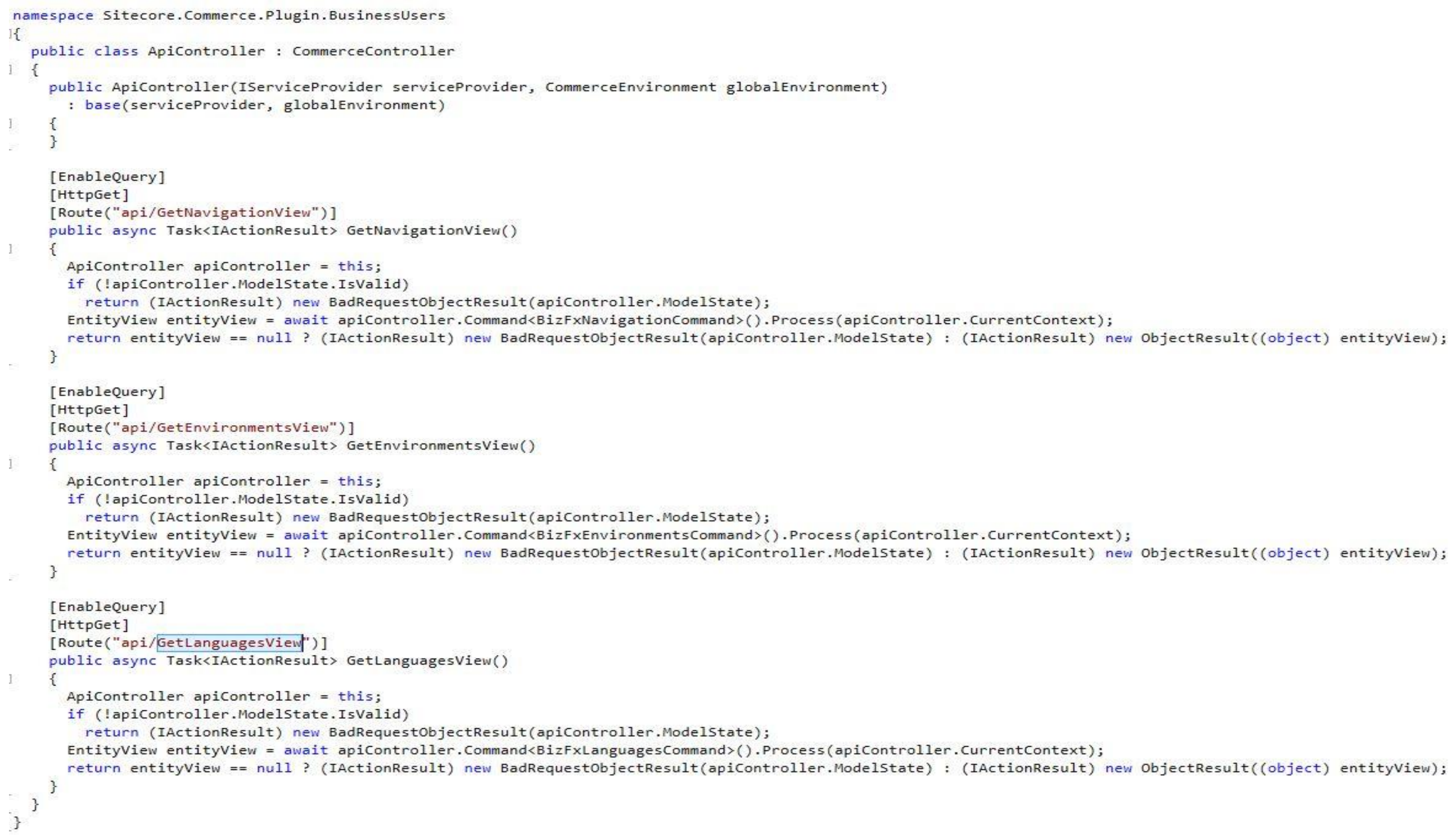
TYPE
Inherit auth from parent

The authorization header will be automatically generated when you send the request. Learn more about authorization

```
            "Policies": []
        },
        "DisplayRank": 8,
        "UiHint": "Flat",
        "Icon": "tools"
    },
    {
        "@odata.type": "#Sitecore.Commerce.EntityViews.EntityView",
        "Name": "My Minions",
        "Policies": [],
        "DisplayName": "My Minions",
        "EntityId": "",
        "EntityVersion": 1,
        "Action": "",
        "ItemId": "My Minions",
        "VersionedItemId": "My Minions-1",
        "Properties": [],
        "ChildViews": [],
        "SelectedChildView": {
            "Name": "",
            "Policies": []
        },
        "DisplayRank": 500,
        "UiHint": "Flat",
        "Icon": "calendar_clock"
    },
    {
        "@odata.type": "#Sitecore.Commerce.EntityViews.EntityView",
        "Name": "Admin-Dashboard",
```

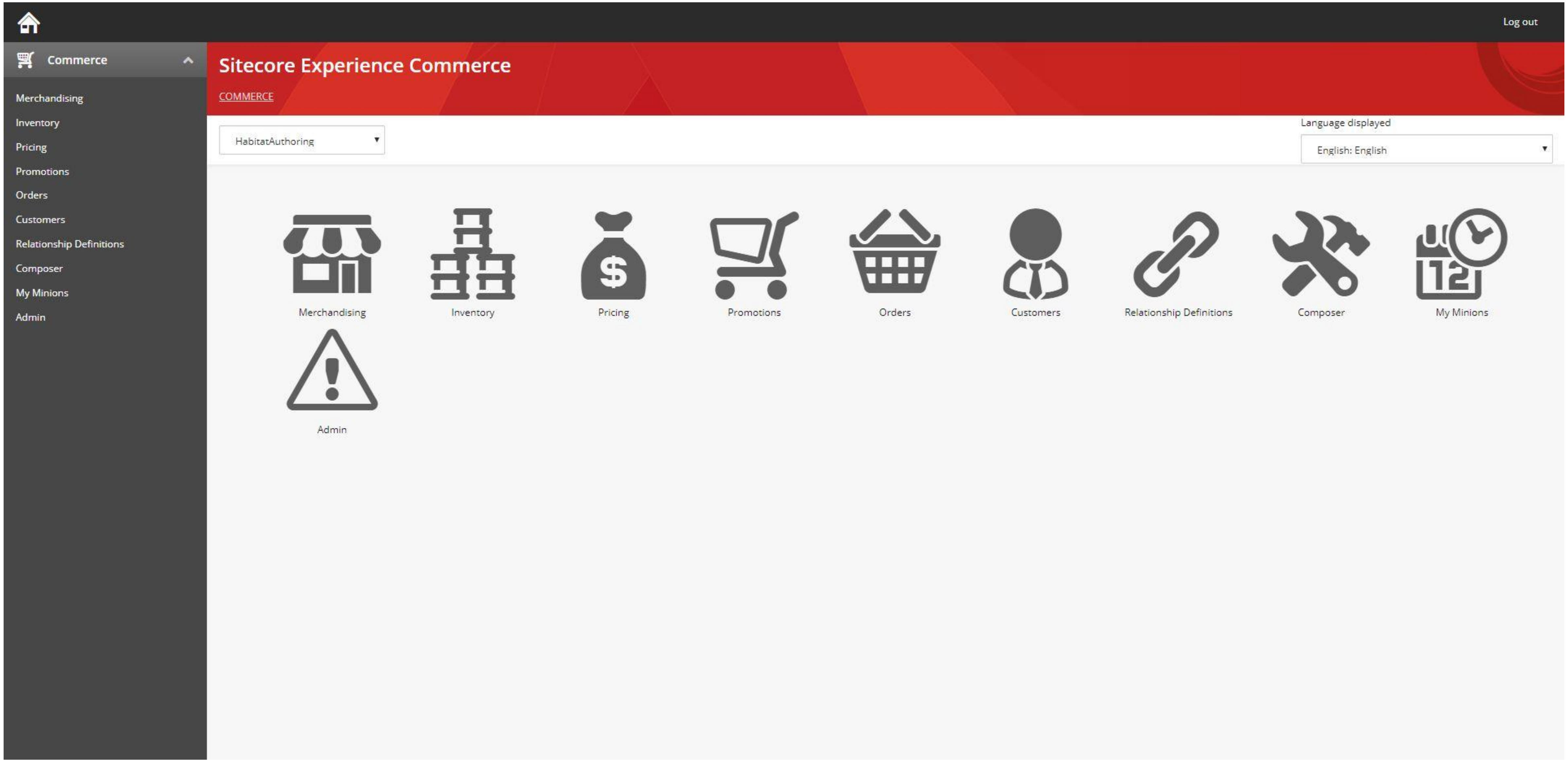
Business Tools Overview

What is behind api calls from Commerce authoring?



```
namespace Sitecore.Commerce.Plugin.BusinessUsers
{
  public class ApiController : CommerceController
  {
    public ApiController(IServiceProvider serviceProvider, CommerceEnvironment globalEnvironment)
      : base(serviceProvider, globalEnvironment)
    {
    }

    [EnableQuery]
    [HttpGet]
    [Route("api/GetNavigationView")]
    public async Task<IActionResult> GetNavigationView()
    {
      ApiController apiController = this;
      if (!apiController.ModelState.IsValid)
        return (IActionResult) new BadRequestObjectResult(apiController.ModelState);
      EntityView entityView = await apiController.Command<BizFxNavigationCommand>().Process(apiController.CurrentContext);
      return entityView == null ? (IActionResult) new BadRequestObjectResult(apiController.ModelState) : (IActionResult) new ObjectResult((object) entityView);
    }

    [EnableQuery]
    [HttpGet]
    [Route("api/GetEnvironmentsView")]
    public async Task<IActionResult> GetEnvironmentsView()
    {
      ApiController apiController = this;
      if (!apiController.ModelState.IsValid)
        return (IActionResult) new BadRequestObjectResult(apiController.ModelState);
      EntityView entityView = await apiController.Command<BizFxEnvironmentsCommand>().Process(apiController.CurrentContext);
      return entityView == null ? (IActionResult) new BadRequestObjectResult(apiController.ModelState) : (IActionResult) new ObjectResult((object) entityView);
    }

    [EnableQuery]
    [HttpGet]
    [Route("api/GetLanguagesView")]
    public async Task<IActionResult> GetLanguagesView()
    {
      ApiController apiController = this;
      if (!apiController.ModelState.IsValid)
        return (IActionResult) new BadRequestObjectResult(apiController.ModelState);
      EntityView entityView = await apiController.Command<BizFxLanguagesCommand>().Process(apiController.CurrentContext);
      return entityView == null ? (IActionResult) new BadRequestObjectResult(apiController.ModelState) : (IActionResult) new ObjectResult((object) entityView);
    }
  }
}
```

How to extend Commerce Bussiness Tool



Commerce Views service Architecture

The Commerce Views service is provided by the Sitecore.Commerce.Plugin.Views plugin.

Sitecore.Commerce.Plugin.Views provides a data-driven mechanism for servicing a dynamic business experience.

In the Sitecore XC architecture, this view layer is provided on the server side, instead of the client side. This allows standard Commerce plugins to extend existing views and add new views.

* https://doc.sitecore.com/developers/90/sitecore-experience-commerce/en/commerce-views-service.html

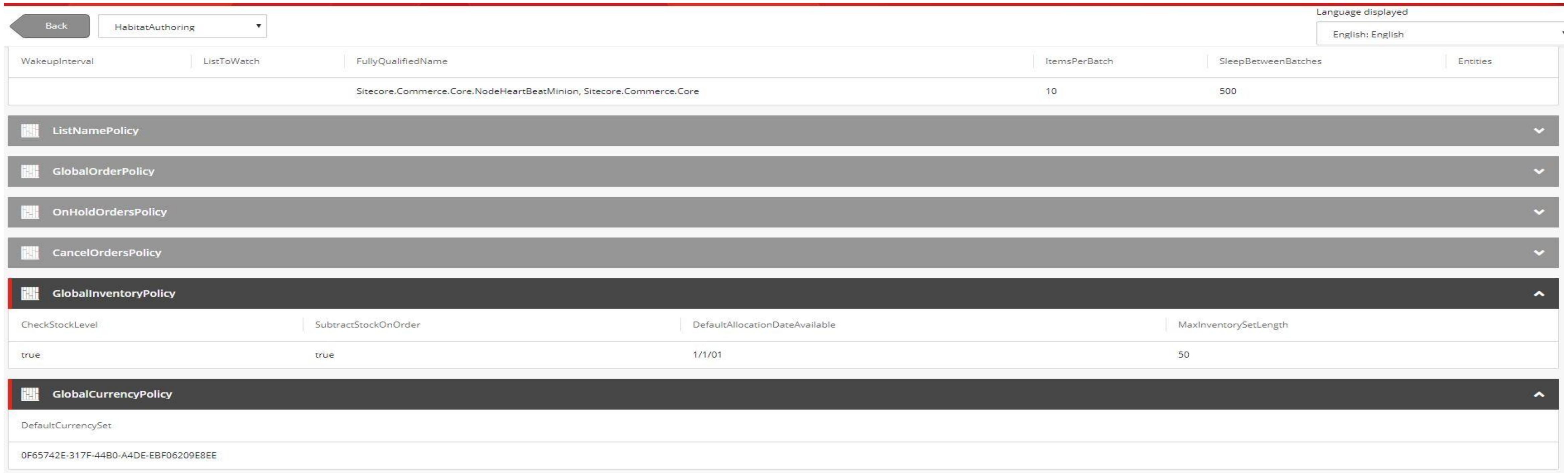
Entity Views

- The EntityView is the core Commerce artifact that supports views.
- The EntityView provides a business user experience that is completely customizable and extensible.
- The EntityView is a simple POCO class that provides a property bag.
- The EntityView can have child EntityViews. You can build out more complex views as needed.

Composite EntityViews

The Composite EntityView represents how that page is organized and what actions are available so that the user interface can simply render the view to form a usable, extensible page without having any custom logic.

A Composite EntityView includes a master view and child views.



WakeupInterval	ListToWatch	FullyQualifiedName	ItemsPerBatch	SleepBetweenBatches	Entities
		Sitecore.Commerce.Core.NodeHeartBeatMinion, Sitecore.Commerce.Core	10	500	

ListNamePolicy

GlobalOrderPolicy

OnHoldOrdersPolicy

CancelOrdersPolicy

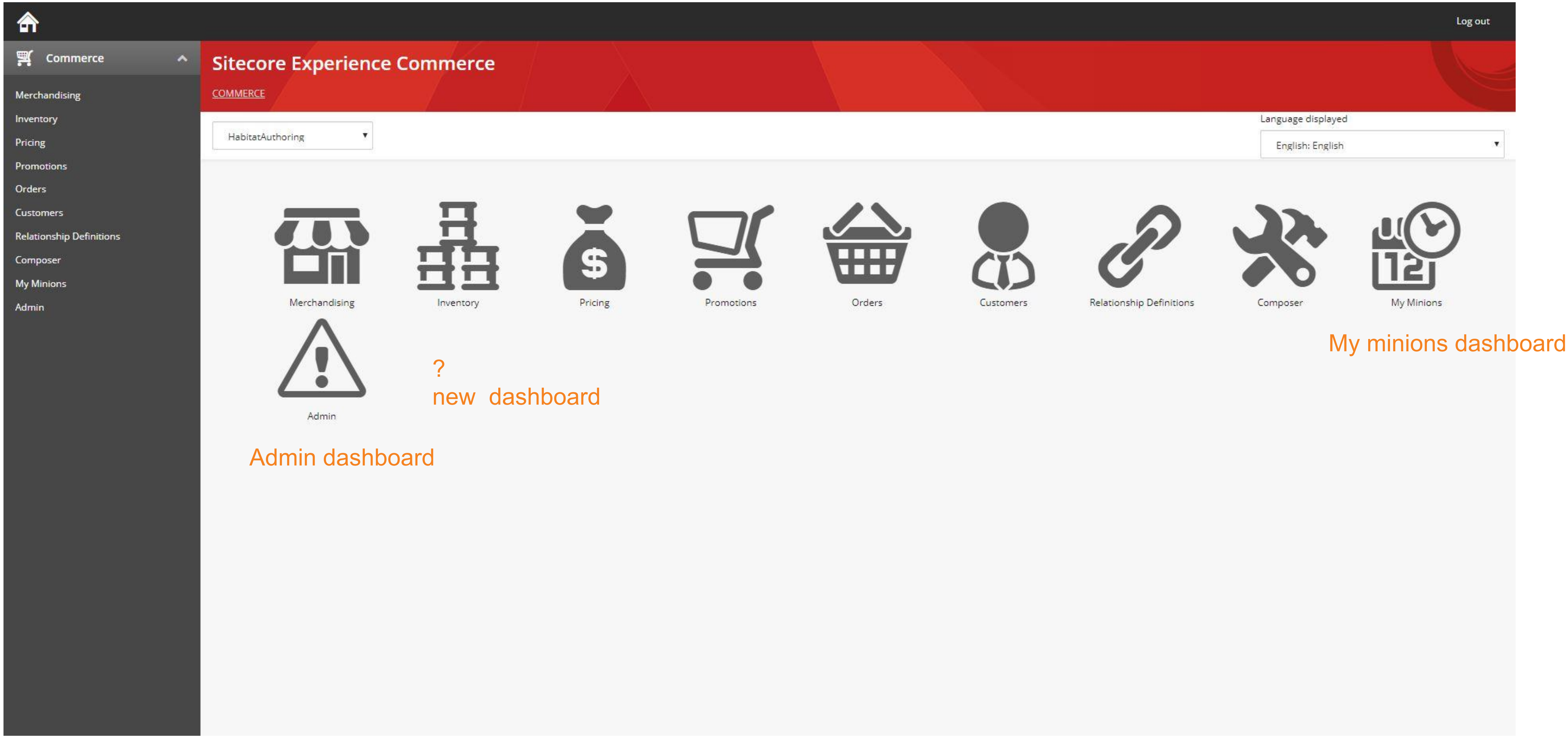
GlobalInventoryPolicy

CheckStockLevel	SubtractStockOnOrder	DefaultAllocationDateAvailable	MaxInventorySetLength
true	true	1/1/01	50

GlobalCurrencyPolicy

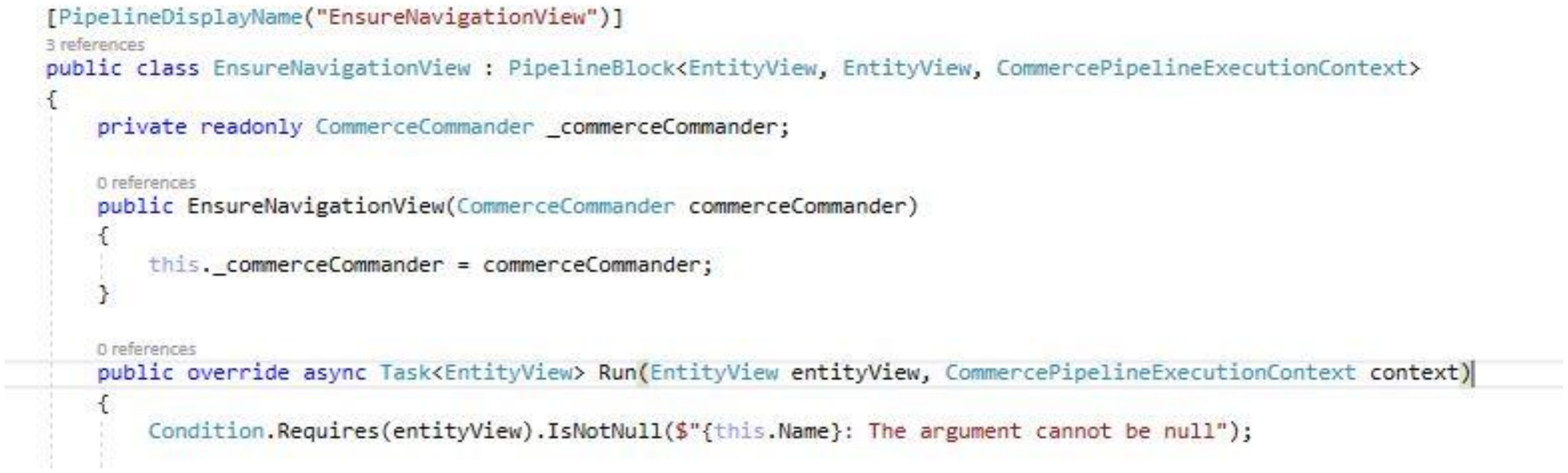
DefaultCurrencySet
0F65742E-317F-44B0-A4DE-EBF06209E8EE

How to add a new navigation dashboard



How to add a new navigation dashboard

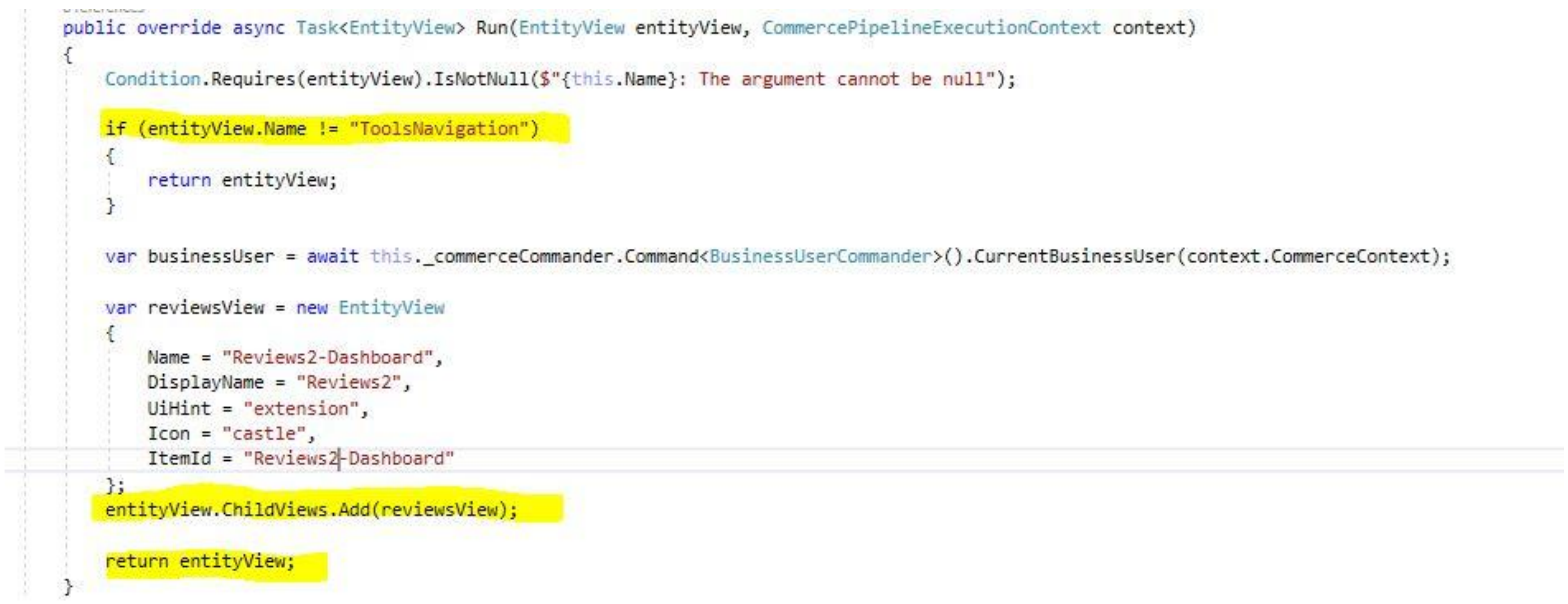
Add a class which inherits from PipelineBlock:



```
[PipelineDisplayName("EnsureNavigationView")]
3 references
public class EnsureNavigationView : PipelineBlock<EntityView, EntityView, CommercePipelineExecutionContext>
{
    private readonly CommerceCommander _commerceCommander;

    0 references
    public EnsureNavigationView(CommerceCommander commerceCommander)
    {
        this._commerceCommander = commerceCommander;
    }

    0 references
    public override async Task<EntityView> Run(EntityView entityView, CommercePipelineExecutionContext context)
    {
        Condition.Requires(entityView).IsNotNull($"{this.Name}: The argument cannot be null");
```

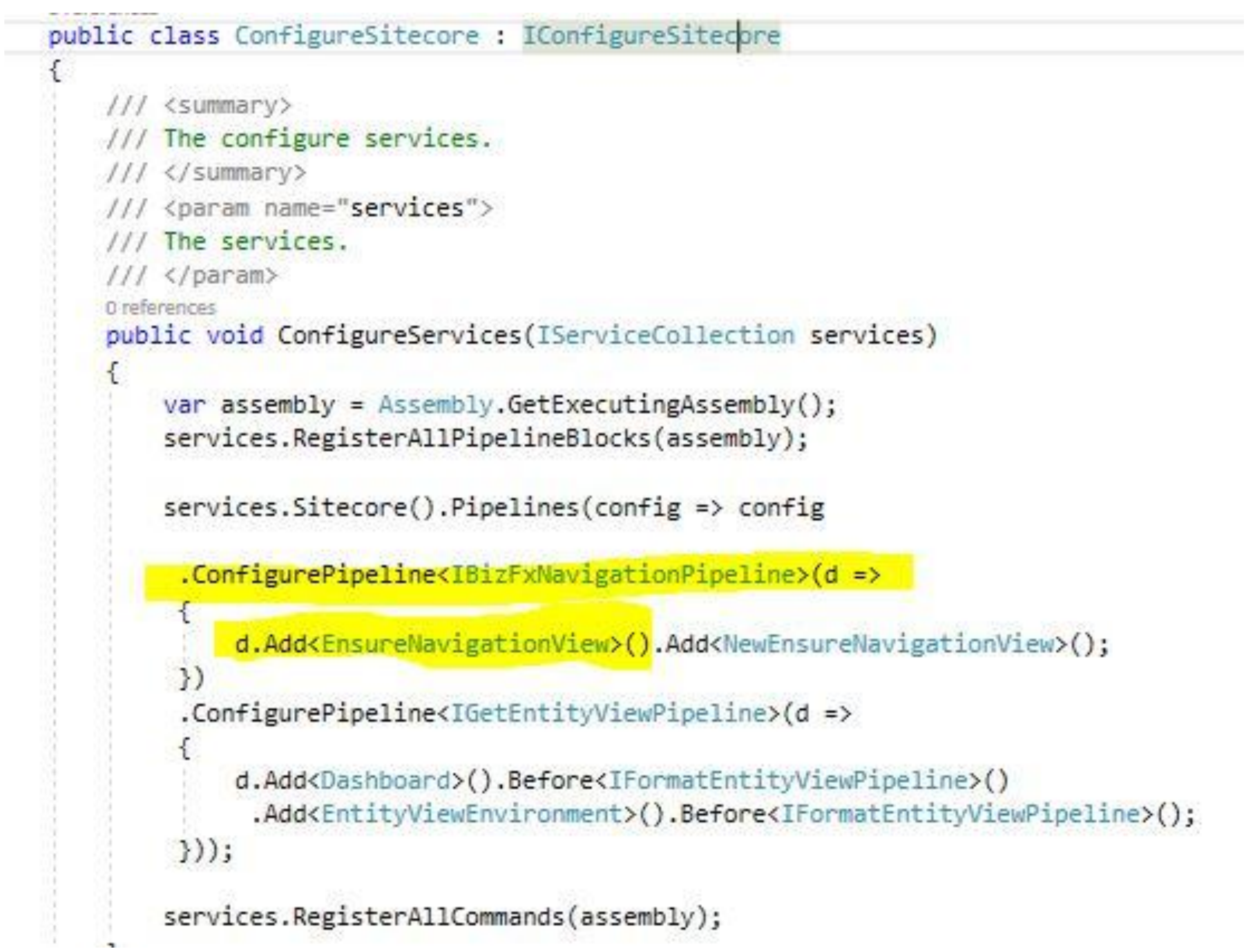
How to add a new navigation dashboard

Override Run method :



```
public override async Task<EntityView> Run(EntityView entityView, CommercePipelineExecutionContext context)
{
    Condition.Requires(entityView).IsNotNull($"{this.Name}: The argument cannot be null");

    if (entityView.Name != "ToolsNavigation")
    {
        return entityView;
    }

    var businessUser = await this._commerceCommander.Command<BusinessUserCommander>().CurrentBusinessUser(context.CommerceContext);

    var reviewsView = new EntityView
    {
        Name = "Reviews2-Dashboard",
        DisplayName = "Reviews2",
        UiHint = "extension",
        Icon = "castle",
        ItemId = "Reviews2-Dashboard"
    };
    entityView.ChildViews.Add(reviewsView);

    return entityView;
}
```

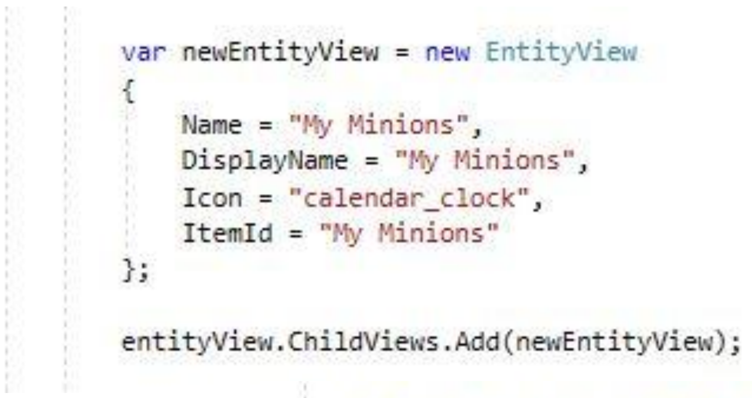
Register you pipeline



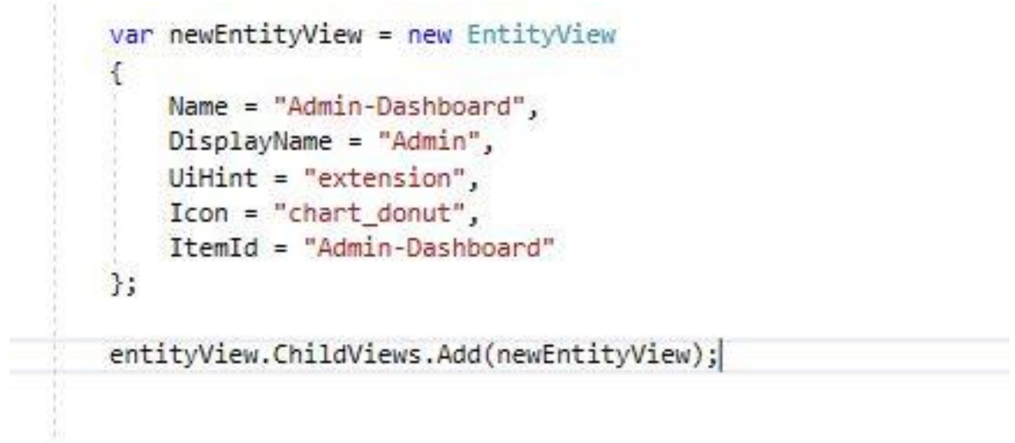
```
public class ConfigureSitecore : IConfigureSitecore
{
    /// <summary>
    /// The configure services.
    /// </summary>
    /// <param name="services">
    /// The services.
    /// </param>
    0 references
    public void ConfigureServices(IServiceCollection services)
    {
        var assembly = Assembly.GetExecutingAssembly();
        services.RegisterAllPipelineBlocks(assembly);

        services.Sitecore().Pipelines(config => config

         .ConfigurePipeline<IBizFxNavigationPipeline>(d =>
         {
             d.Add<EnsureNavigationView>().Add<NewEnsureNavigationView>();
         })
         .ConfigurePipeline<IGetEntityViewPipeline>(d =>
         {
             d.Add<Dashboard>().Before<IFormatEntityViewPipeline>()
              .Add<EntityViewEnvironment>().Before<IFormatEntityViewPipeline>();
         }));

        services.RegisterAllCommands(assembly);
```

Do not repeat yourself

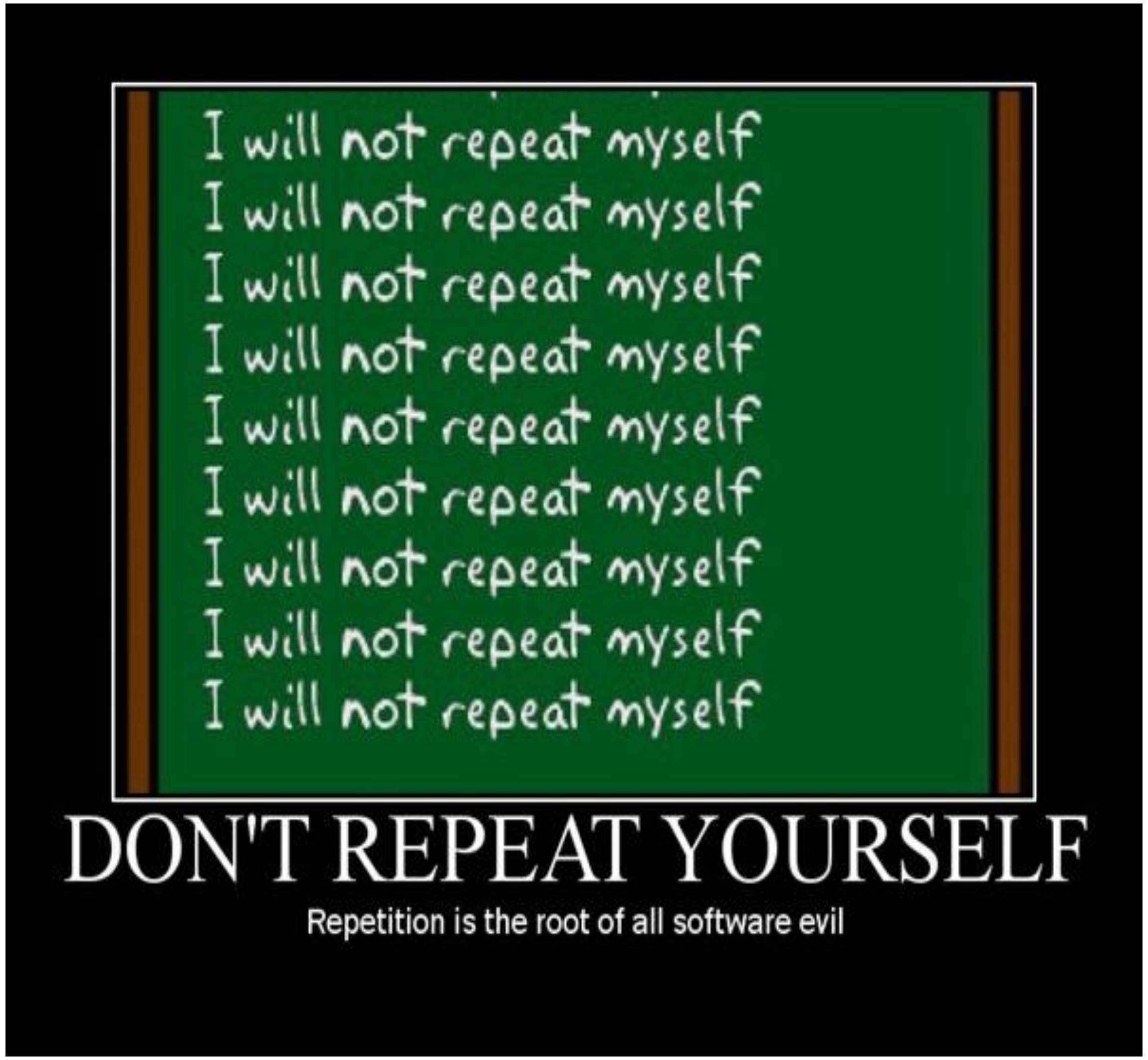
Do you like to write same code multiple times?



```
var newEntityView = new EntityView
{
    Name = "My Minions",
    DisplayName = "My Minions",
    Icon = "calendar_clock",
    ItemId = "My Minions"
};

entityView.ChildViews.Add(newEntityView);
```

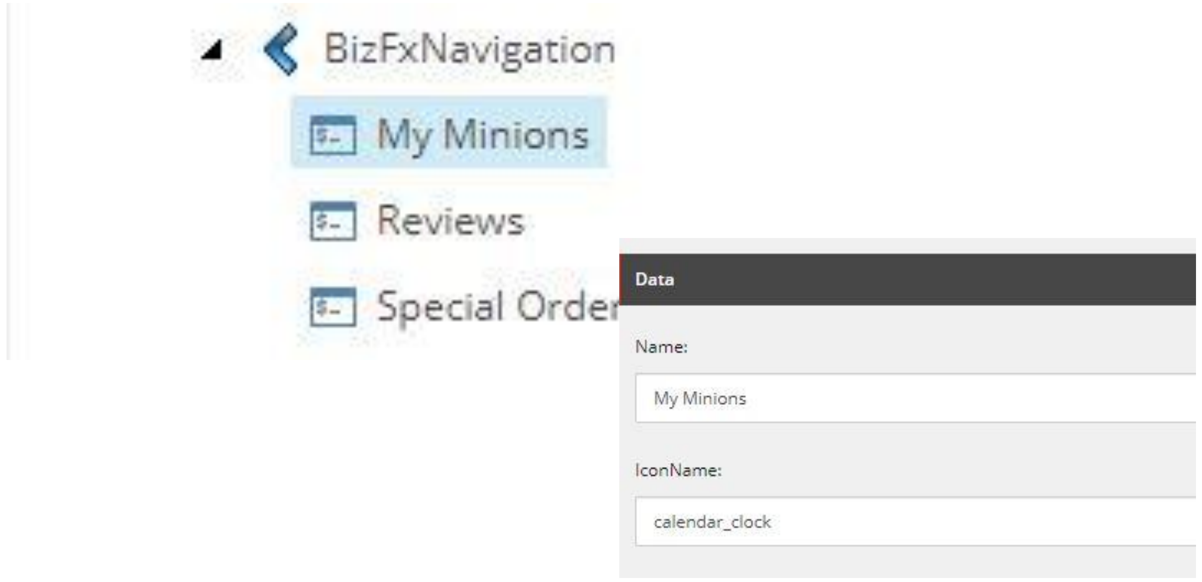


```
var newEntityView = new EntityView
{
    Name = "Admin-Dashboard",
    DisplayName = "Admin",
    UiHint = "extension",
    Icon = "chart_donut",
    ItemId = "Admin-Dashboard"
};

entityView.ChildViews.Add(newEntityView);
```



Do not repeat yourself

To not repeat yourself keep the Dashboard information in Sitecore items.

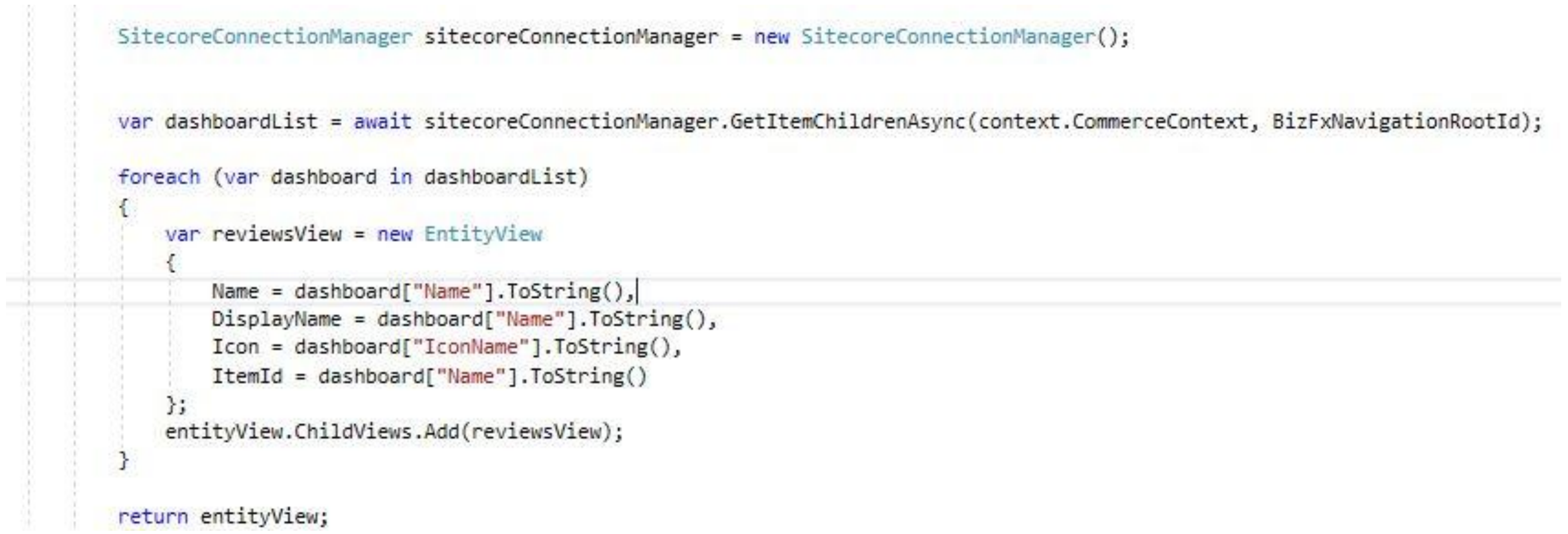


Name:
My Minions

IconName:
calendar_clock

Do not repeat yourself

Read the dashboard information using SitecoreConnectionManager



```
SitecoreConnectionManager sitecoreConnectionManager = new SitecoreConnectionManager();


var dashboardList = await sitecoreConnectionManager.GetItemChildrenAsync(context.CommerceContext, BizFxNavigationRootId);

foreach (var dashboard in dashboardList)
{
    var reviewsView = new EntityView
    {
        Name = dashboard["Name"].ToString(),
        DisplayName = dashboard["Name"].ToString(),
        Icon = dashboard["IconName"].ToString(),
        ItemId = dashboard["Name"].ToString()
    };
    entityView.ChildViews.Add(reviewsView);
}

return entityView;
```

Demo

Probably the first XC 9.1. demo in the world

Minions

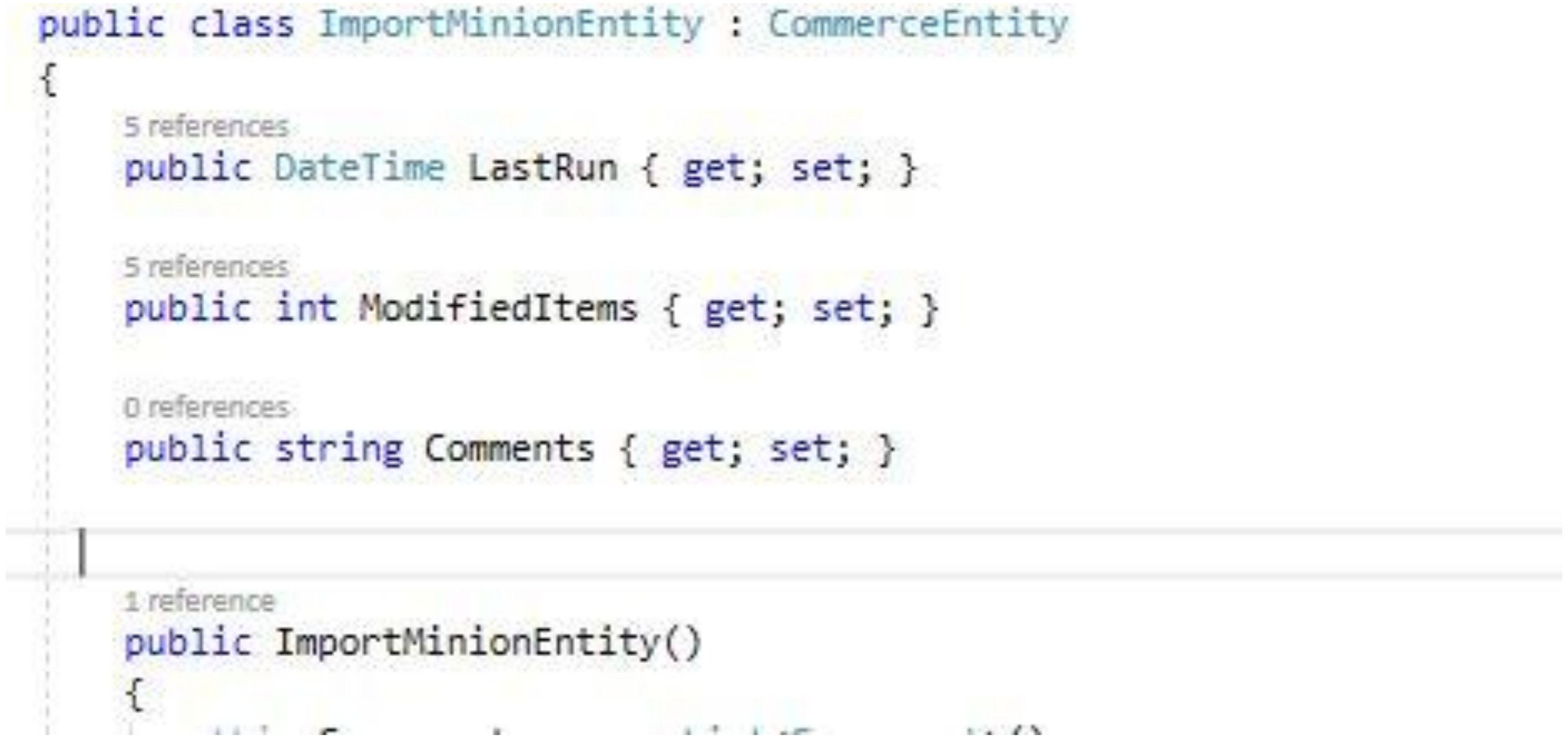
The Commerce Minions role is an instance of the Commerce Engine that runs independently and supports asynchronous processing.

When it runs last time your custom minion?

How many items did you process?

My minions

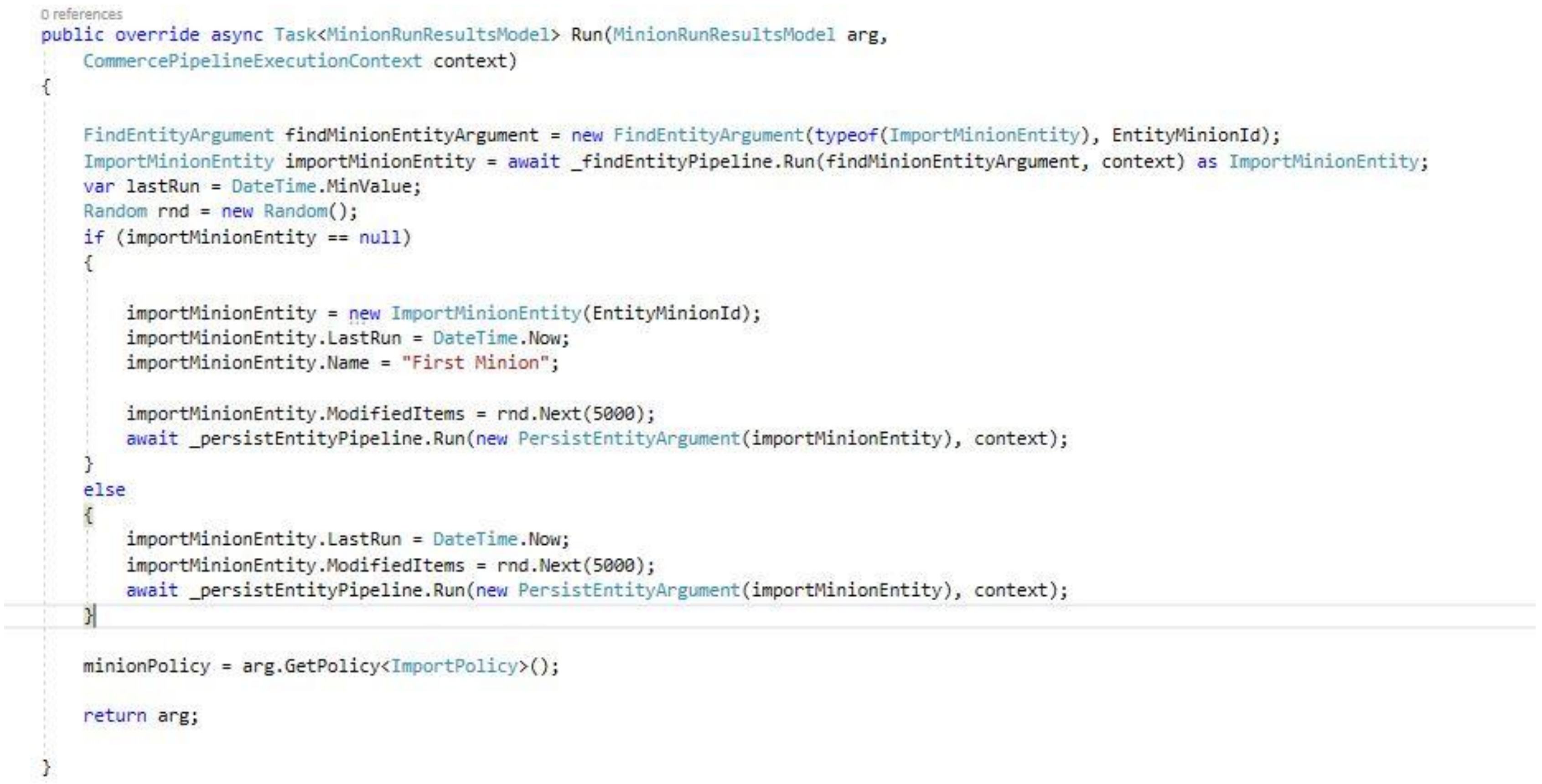
Declare Minion Entity



```
public class ImportMinionEntity : CommerceEntity
{
    5 references
    public DateTime LastRun { get; set; }

    5 references
    public int ModifiedItems { get; set; }

    0 references
    public string Comments { get; set; }


    1 reference
    public ImportMinionEntity()
    {
```

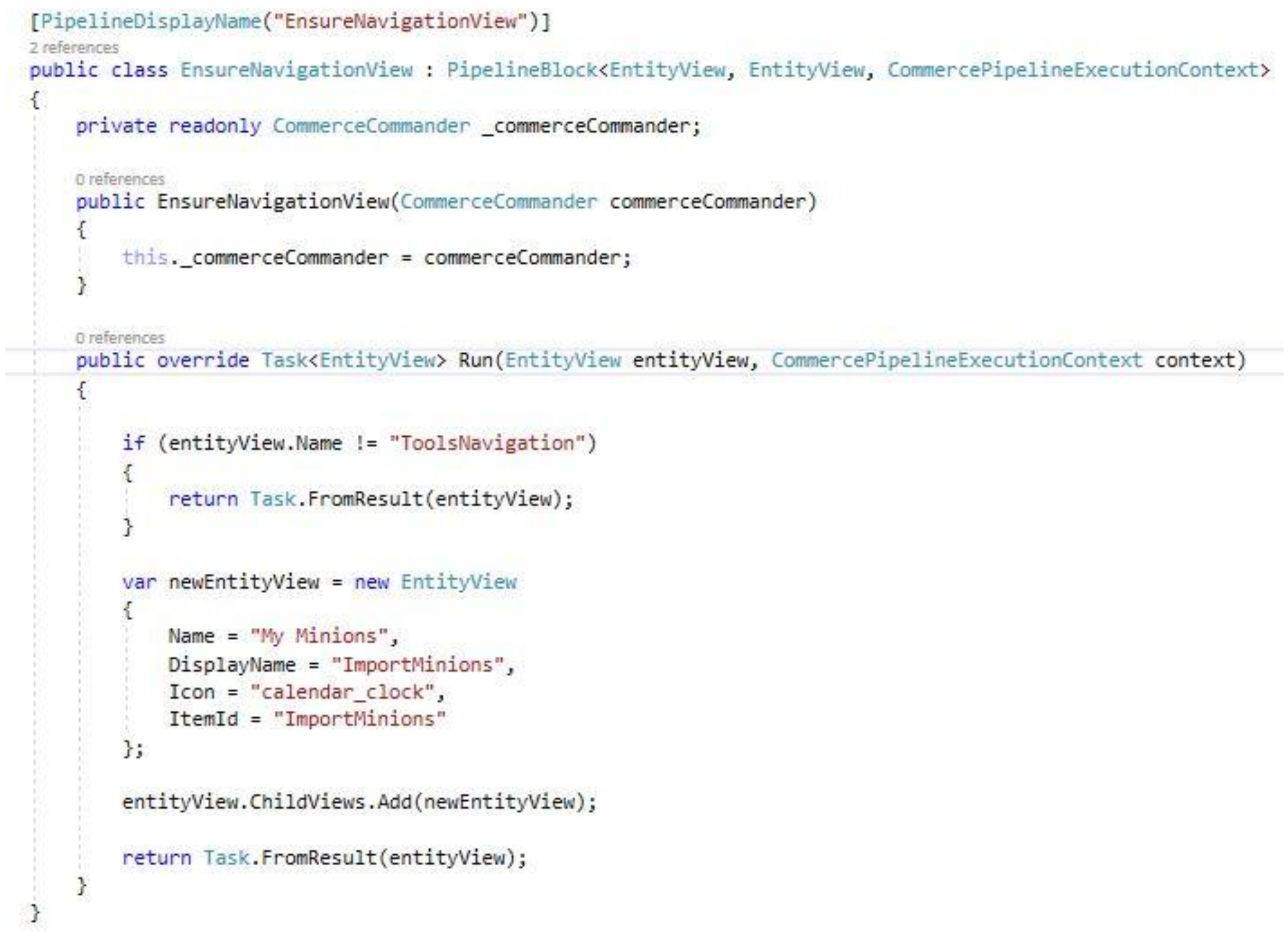
My minions

Create or update MinionEntity (dummy code)



```
0 references
public override async Task<MinionRunResultsModel> Run(MinionRunResultsModel arg,
    CommercePipelineExecutionContext context)
{

    FindEntityArgument findMinionEntityArgument = new FindEntityArgument(typeof(ImportMinionEntity), EntityMinionId);
    ImportMinionEntity importMinionEntity = await _findEntityPipeline.Run(findMinionEntityArgument, context) as ImportMinionEntity;
    var lastRun = DateTime.MinValue;
    Random rnd = new Random();
    if (importMinionEntity == null)
    {

        importMinionEntity = new ImportMinionEntity(EntityMinionId);
        importMinionEntity.LastRun = DateTime.Now;
        importMinionEntity.Name = "First Minion";

        importMinionEntity.ModifiedItems = rnd.Next(5000);
        await _persistEntityPipeline.Run(new PersistEntityArgument(importMinionEntity), context);
    }
    else
    {
        importMinionEntity.LastRun = DateTime.Now;
        importMinionEntity.ModifiedItems = rnd.Next(5000);
        await _persistEntityPipeline.Run(new PersistEntityArgument(importMinionEntity), context);
    }

    minionPolicy = arg.GetPolicy<ImportPolicy>();

    return arg;

}
```

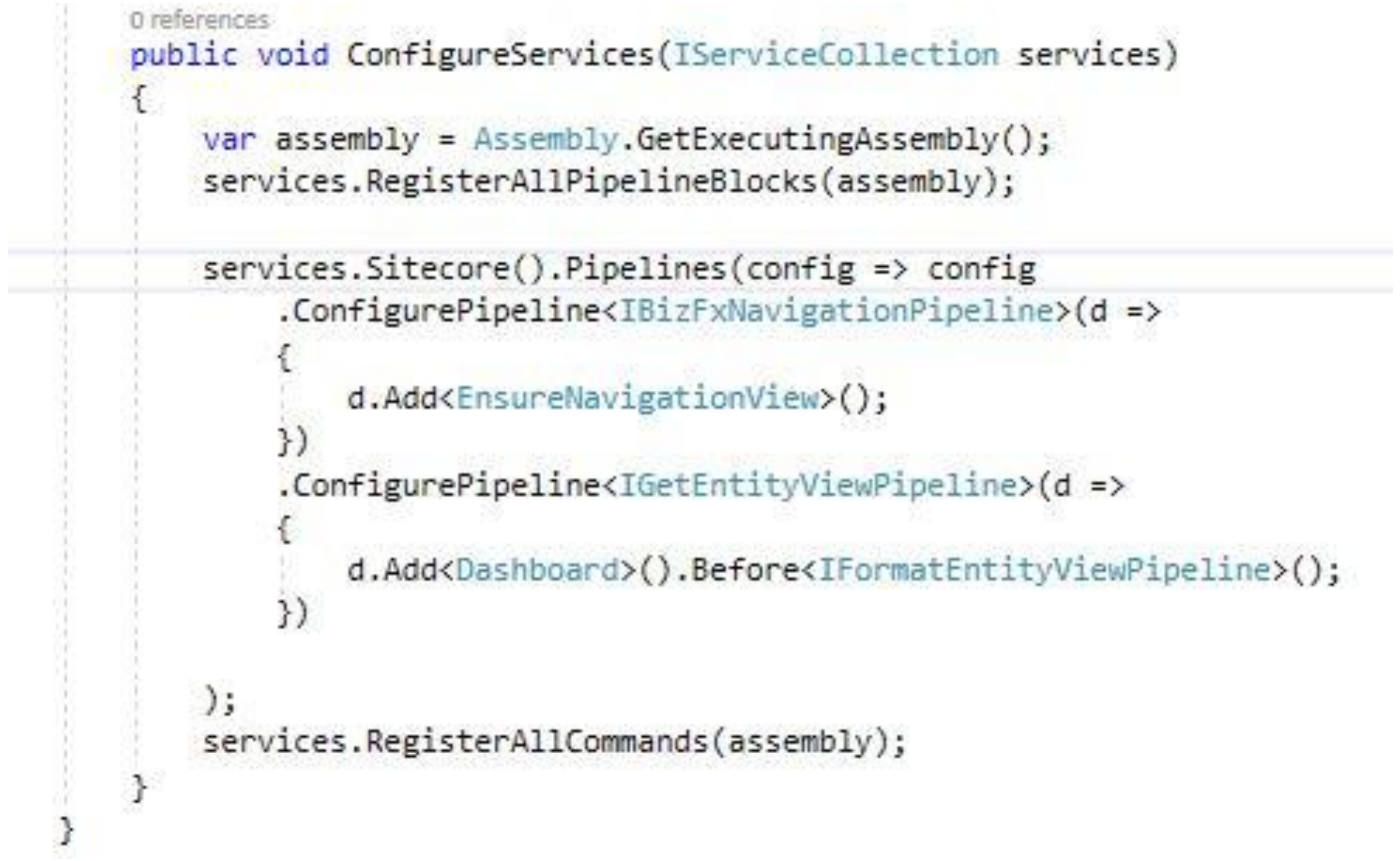
Create navigation dashboard



```
[PipelineDisplayName("EnsureNavigationView")]
2 references
public class EnsureNavigationView : PipelineBlock<EntityView, EntityView, CommercePipelineExecutionContext>
{
    private readonly CommerceCommander _commerceCommander;

    0 references
    public EnsureNavigationView(CommerceCommander commerceCommander)
    {
        this._commerceCommander = commerceCommander;
    }

    0 references
    public override Task<EntityView> Run(EntityView entityView, CommercePipelineExecutionContext context)
    {

        if (entityView.Name != "ToolsNavigation")
        {
            return Task.FromResult(entityView);
        }

        var newEntityView = new EntityView
        {
            Name = "My Minions",
            DisplayName = "ImportMinions",
            Icon = "calendar_clock",
            ItemId = "ImportMinions"
        };

        entityView.ChildViews.Add(newEntityView);

        return Task.FromResult(entityView);
    }
}
```

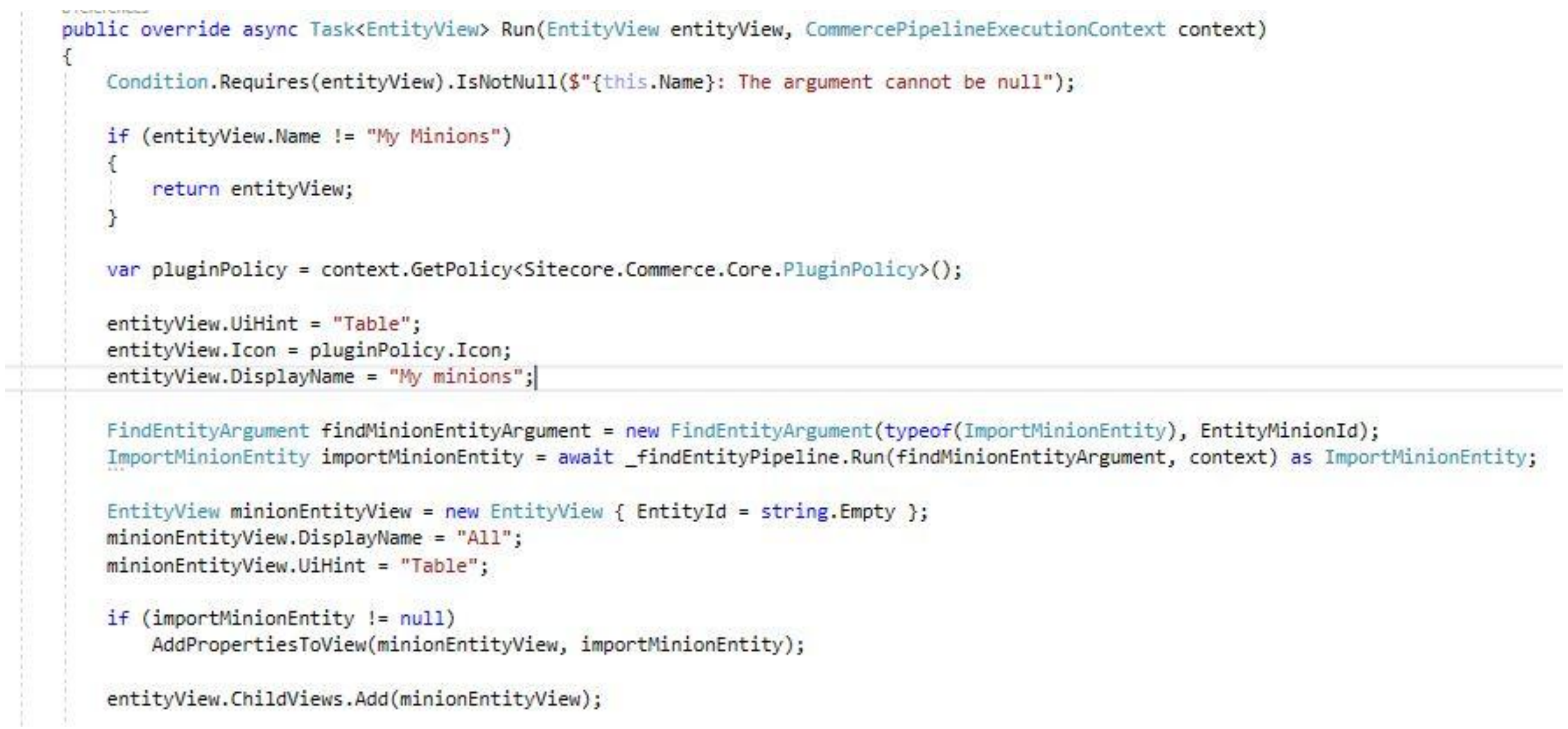
Register the pipeline

Register NavigationView inside ConfigureSitecore.cs



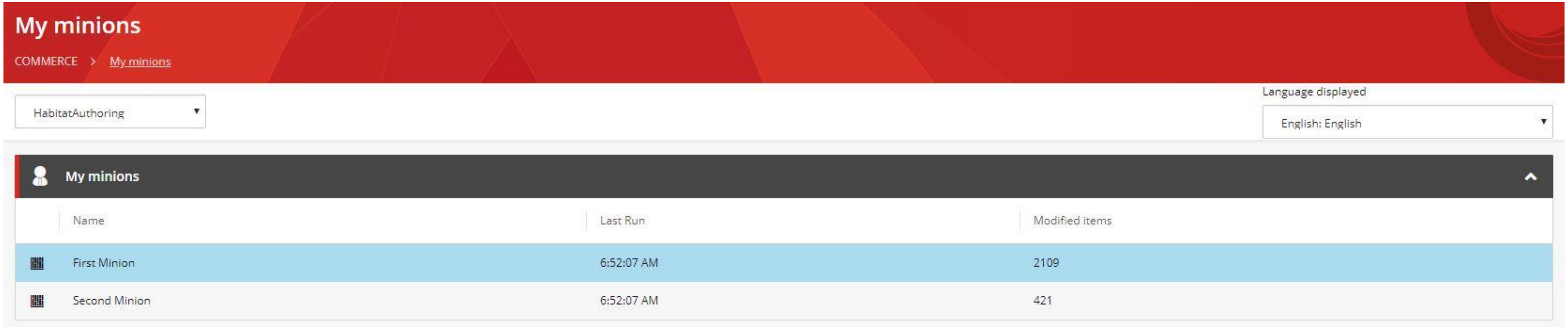
```
0 references
public void ConfigureServices(IServiceCollection services)
{
    var assembly = Assembly.GetExecutingAssembly();
    services.RegisterAllPipelineBlocks(assembly);

    services.Sitecore().Pipelines(config => config
        .ConfigurePipeline<IBizFxNavigationPipeline>(d =>
        {
            d.Add<EnsureNavigationView>();
        })
        .ConfigurePipeline<IGetEntityViewPipeline>(d =>
        {
            d.Add<Dashboard>().Before<IFormatEntityViewPipeline>();
        })

    );
    services.RegisterAllCommands(assembly);
}
}
```

Add the minion entity view



```
public override async Task<EntityView> Run(EntityView entityView, CommercePipelineExecutionContext context)
{
    Condition.Requires(entityView).IsNotNull($"{this.Name}: The argument cannot be null");

    if (entityView.Name != "My Minions")
    {
        return entityView;
    }

    var pluginPolicy = context.GetPolicy<Sitecore.Commerce.Core.PluginPolicy>();

    entityView.UiHint = "Table";
    entityView.Icon = pluginPolicy.Icon;
    entityView.DisplayName = "My minions";

    FindEntityArgument findMinionEntityArgument = new FindEntityArgument(typeof(ImportMinionEntity), EntityMinionId);
    ImportMinionEntity importMinionEntity = await _findEntityPipeline.Run(findMinionEntityArgument, context) as ImportMinionEntity;

    EntityView minionEntityView = new EntityView { EntityId = string.Empty };
    minionEntityView.DisplayName = "All";
    minionEntityView.UiHint = "Table";

    if (importMinionEntity != null)
        AddPropertiesToView(minionEntityView, importMinionEntity);

    entityView.ChildViews.Add(minionEntityView);
```

Where are my minions



My minions

COMMERCE > My minions

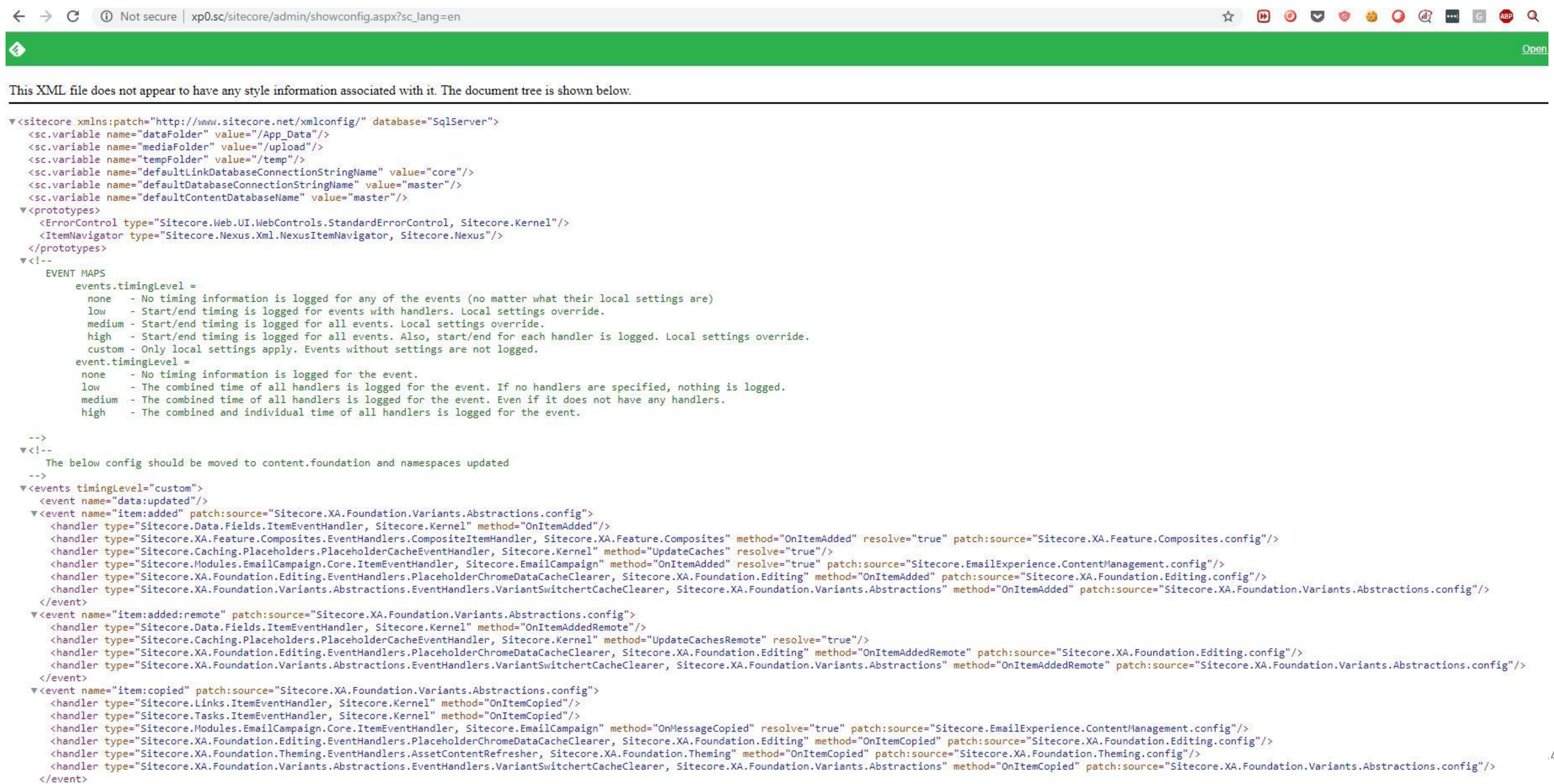
HabitatAuthoring

Language displayed: English: English

My minions

Name	Last Run	Modified items
First Minion	6:52:07 AM	2109
Second Minion	6:52:07 AM	421

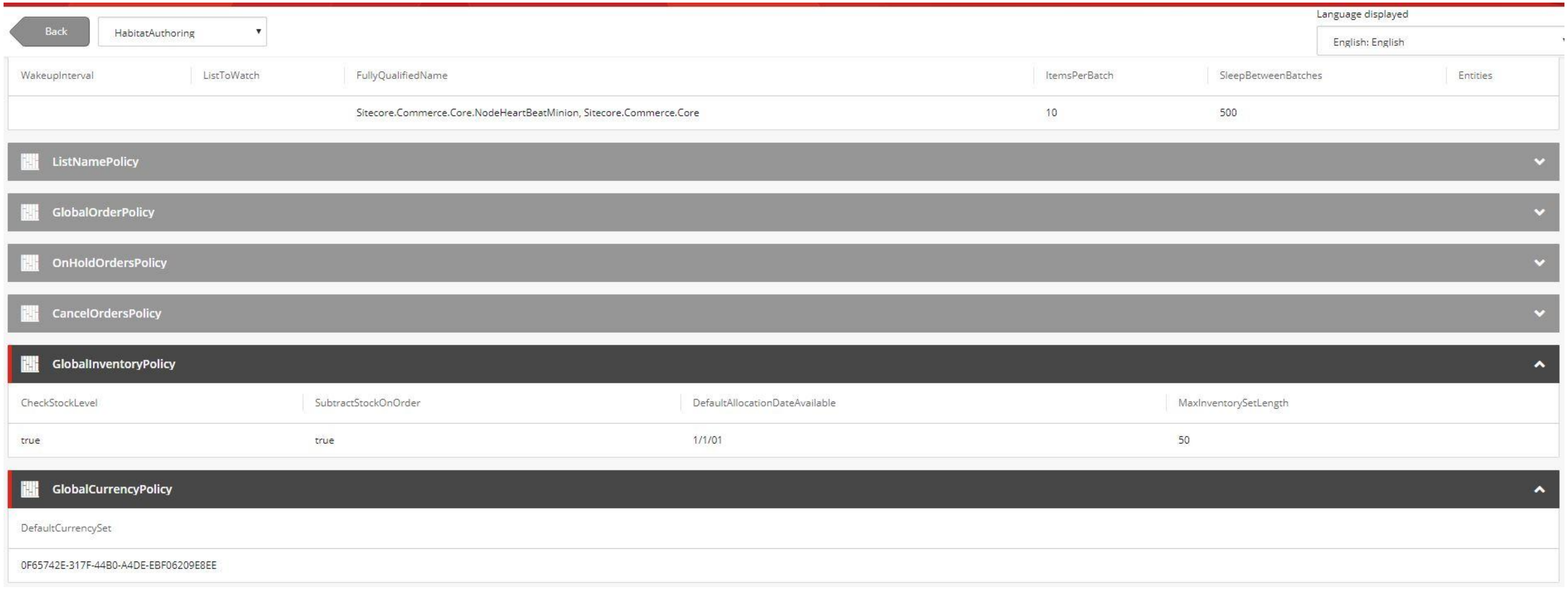
Showconfig


Not secure | xp0.sc/sitecore/admin/showconfig.aspx?sc_lang=en
Open

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<sitecore xmlns:patch="http://www.sitecore.net/xmlconfig/" database="SqlServer">
   <sc.variable name="dataFolder" value="/App_Data"/>
   <sc.variable name="mediaFolder" value="/upload"/>
   <sc.variable name="tempFolder" value="/temp"/>
   <sc.variable name="defaultLinkDatabaseConnectionStringName" value="core"/>
   <sc.variable name="defaultDatabaseConnectionStringName" value="master"/>
   <sc.variable name="defaultContentDatabaseName" value="master"/>
  ▼<prototypes>
     <ErrorControl type="Sitecore.Web.UI.WebControls.StandardErrorControl, Sitecore.Kernel"/>
     <ItemNavigator type="Sitecore.Nexus.Xml.NexusItemNavigator, Sitecore.Nexus"/>
   </prototypes>
  ▼<!--
      EVENT MAPS
           events.timingLevel =
             none   - No timing information is logged for any of the events (no matter what their local settings are)
             low    - Start/end timing is logged for events with handlers. Local settings override.
             medium - Start/end timing is logged for all events. Local settings override.
             high   - Start/end timing is logged for all events. Also, start/end for each handler is logged. Local settings override.
             custom - Only local settings apply. Events without settings are not logged.
           event.timingLevel =
            none    - No timing information is logged for the event.
            low     - The combined time of all handlers is logged for the event. If no handlers are specified, nothing is logged.
            medium  - The combined time of all handlers is logged for the event. Even if it does not have any handlers.
            high    - The combined and individual time of all handlers is logged for the event.

   -->
  ▼<!--
      The below config should be moved to content.foundation and namespaces updated
   -->
  ▼<events timingLevel="custom">
     <event name="data:updated"/>
    ▼<event name="item:added" patch:source="Sitecore.XA.Foundation.Variants.Abstractions.config">
       <handler type="Sitecore.Data.Fields.ItemEventHandler, Sitecore.Kernel" method="OnItemAdded"/>
       <handler type="Sitecore.XA.Feature.Composites.EventHandlers.CompositeItemHandler, Sitecore.XA.Feature.Composites" method="OnItemAdded" resolve="true" patch:source="Sitecore.XA.Feature.Composites.config"/>
       <handler type="Sitecore.Caching.Placeholders.PlaceholderCacheEventHandler, Sitecore.Kernel" method="UpdateCaches" resolve="true"/>
       <handler type="Sitecore.Modules.EmailCampaign.Core.ItemEventHandler, Sitecore.EmailCampaign" method="OnItemAdded" resolve="true" patch:source="Sitecore.EmailExperience.ContentManagement.config"/>
       <handler type="Sitecore.XA.Foundation.Editing.EventHandlers.PlaceholderChromeDataCacheClearer, Sitecore.XA.Foundation.Editing" method="OnItemAdded" patch:source="Sitecore.XA.Foundation.Editing.config"/>
       <handler type="Sitecore.XA.Foundation.Variants.Abstractions.EventHandlers.VariantSwitchertCacheClearer, Sitecore.XA.Foundation.Variants.Abstractions" method="OnItemAdded" patch:source="Sitecore.XA.Foundation.Variants.Abstractions.config"/>
     </event>
    ▼<event name="item:added:remote" patch:source="Sitecore.XA.Foundation.Variants.Abstractions.config">
       <handler type="Sitecore.Data.Fields.ItemEventHandler, Sitecore.Kernel" method="OnItemAddedRemote"/>
       <handler type="Sitecore.Caching.Placeholders.PlaceholderCacheEventHandler, Sitecore.Kernel" method="UpdateCachesRemote" resolve="true"/>
       <handler type="Sitecore.XA.Foundation.Editing.EventHandlers.PlaceholderChromeDataCacheClearer, Sitecore.XA.Foundation.Editing" method="OnItemAddedRemote" patch:source="Sitecore.XA.Foundation.Editing.config"/>
       <handler type="Sitecore.XA.Foundation.Variants.Abstractions.EventHandlers.VariantSwitchertCacheClearer, Sitecore.XA.Foundation.Variants.Abstractions" method="OnItemAddedRemote" patch:source="Sitecore.XA.Foundation.Variants.Abstractions.config"/>
     </event>
    ▼<event name="item:copied" patch:source="Sitecore.XA.Foundation.Variants.Abstractions.config">
       <handler type="Sitecore.Links.ItemEventHandler, Sitecore.Kernel" method="OnItemCopied"/>
       <handler type="Sitecore.Tasks.ItemEventHandler, Sitecore.Kernel" method="OnItemCopied"/>
       <handler type="Sitecore.Modules.EmailCampaign.Core.ItemEventHandler, Sitecore.EmailCampaign" method="OnMessageCopied" resolve="true" patch:source="Sitecore.EmailExperience.ContentManagement.config"/>
       <handler type="Sitecore.XA.Foundation.Editing.EventHandlers.PlaceholderChromeDataCacheClearer, Sitecore.XA.Foundation.Editing" method="OnItemCopied" patch:source="Sitecore.XA.Foundation.Editing.config"/>
       <handler type="Sitecore.XA.Foundation.Theming.EventHandlers.AssetContentRefresher, Sitecore.XA.Foundation.Theming" method="OnItemCopied" patch:source="Sitecore.XA.Foundation.Theming.config"/>
       <handler type="Sitecore.XA.Foundation.Variants.Abstractions.EventHandlers.VariantSwitchertCacheClearer, Sitecore.XA.Foundation.Variants.Abstractions" method="OnItemCopied" patch:source="Sitecore.XA.Foundation.Variants.Abstractions.config"/>
     </event>
```

How to see policies values for every environment?



WakeupInterval	ListToWatch	FullyQualifiedName	ItemsPerBatch	SleepBetweenBatches	Entities
		Sitecore.Commerce.Core.NodeHeartBeatMinion, Sitecore.Commerce.Core	10	500	

ListNamePolicy

GlobalOrderPolicy

OnHoldOrdersPolicy

CancelOrdersPolicy

GlobalInventoryPolicy

CheckStockLevel	SubtractStockOnOrder	DefaultAllocationDateAvailable	MaxInventorySetLength
true	true	1/1/01	50

GlobalCurrencyPolicy

DefaultCurrencySet
0F65742E-317F-44B0-A4DE-EBF06209E8EE

Demo Time