# XMLMappingProcessor Docs

Класс для создания отчетов Excel на основе xml маппинга.

# Требования:

1. Сборки: DocumentFormat.OpenXml (в коробку 5.4 уже включена), System.XML.LINQ (Добавление ссылки на сборку (System.XML.LINQ));
2. Корректная работа EntitySchemaQuery для объектов, используемых в отчете (+ колонка Id с типом Guid у каждого);

# Общий принцип использования

Создание отчета происходит в 2 этапа:

1. Созданиe экземпляра класса XMLMappingProcessor (при этом класс забирает из базы данные).
2. Обработка шаблона методом public void ProcessDoc(Stream ExcellDocument){}.

Пример использования:

Stream doc1 = new MemoryStream(ReportTemplateFile);

Stream doc = new MemoryStream();

doc1.CopyTo(doc);

Stream map = new MemoryStream(ReportMappingFile);

*//Создание набора параметров*

Dictionary<string, object> parameters = new Dictionary<string, object>();

parameters.Add("OpportunityId",ReadDataUserTask.ResultEntity.GetTypedColumnValue<Guid>("Id").ToString());

parameters.Add("OpportunityFormatId",ReadDataUserTask.ResultEntity

.GetTypedColumnValue<Guid>("ProductCategoriesId").ToString());

parameters.Add("OrderDataColumn",GetDetailColumnValue(

ReadDataUserTask.ResultEntity.GetTypedColumnValue<Guid>("Id")));

*//1) Создания экземпляра класса:* *map - System.IO.Stream xml файла с маппингом*

var xmlp = new XMLMappingProcessor(map, context.UserConnection, parameters);

*//2) Создание пользовательских функций*

xmlp.Functions = GetUserDefinedFunctions();

*//3) Магия*

xmlp.ProcessDoc(doc);

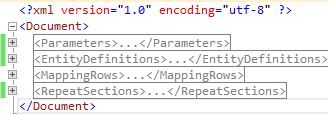
*//4) Дальнейшая обработка…*

MemoryStream currentReportStream = new MemoryStream();

((MemoryStream) doc).Close();

byte[] report = ((MemoryStream) doc).ToArray();

# XML файл с маппингом

Разделен на две (основные) части:

1. Секция определения данных.

1

1. Секция определения полей отчета.

При создании экземпляра типа XMLMappingProcessor на основании первой секции загружаются данные.

2

В результате, в поле public Dictionary<string, object> LoadedData находятся данные, определенные в секции <EntityDefinitions>. Наполняются данные по принципу:

ED name="myOpportunity" + CD name="OwnerName" = LoadedData.Add(“myOpportunity.OwnerName”,value);

Атрибут type (или его отсутсвие) узла <EntityDefinitions> со значением type="single" применяется для маппинга полей единичной сущности. Пример:

<EntityDefinitions>

<ED name="myOpportunity" path="Opportunity" type="single"> <!--определение сущности-->

<ColumnDefinitions> <!--определение колонок (обязательно, только одно)-->

<CD name="OwnerName" path="Owner.Name"/><!--определение колонки (обязательно, хотя бы 1)-->

<CD name="OwnerManagerName" path="Owner.Manager.Name"/>

<CD name="AccountManagerName" path="AccountManager.Name"/>

<CD name="FinanceDirectorName" path="FinanceDirector.Name"/>

<CD name="ECTotal" path="ECTotal"/>

</ColumnDefinitions>

<!--определение колекций фильтров, необязательное-->

<FilterDefinitions>

<!--определение колекции фильтров, значения logicalOperation - And или Or -->

<FilterCollection logicalOperation="And">

<!--определения фильтра по аналогии с EntitySchemaQueryFilter,

если rightVal начинается с @, значение берется из параметров,

comparisonType - названия типов из перечисления Terrasoft.Core.Entities.FilterComparisonType-->

<FD type="FromFD" leftVal="Id" comparisonType="Equals" rightVal="@OpportunityId"/>

<!--если type="FromParameter", EntitySchemaQueryFilter берется с колекции параметров,

тип параметра должен реализовать IEntitySchemaQueryFilterItem-->

<FD type="FromParameter" parameter="@OpportunityId"/>

</FilterCollection>

</FilterDefinitions>

</ED>

<!—и так далее-->

<ED name="myFormat"></ED>//тут аналогично

</EntityDefinitions>

\*атрибут path="FinanceDirector.Name" узла <CD/> соответствует выражению: EntitySchemaQuery.AddColumn(“FinanceDirector.Name”);

В результате в коллекцию LoadedData добавится:

[myOpportunity.OwnerName:имя ответственного]

[myOpportunity.OwnerManagerName:имя директора ответственного]

[myOpportunity.AccountManagerName:имя директора контрагента]

[myOpportunity.FinanceDirectorName:имя финансового директора]

[myOpportunity.ECTotal:сумма]

[myFormat.Name:Название значения из справочника]

[myFormat.Id:Его Id]

<ED name="Order" type="collection">

<ColumnDefinitions>

<CD name="Description" path="OrderDescription"/>

<CD name="Timestudy" path="Timestudy"/>

<CD name="PlatformName" path="Platform.Name"/>

<CD name="FormatName" path="Format.Name"/>

</ColumnDefinitions>

<FilterDefinitions>

<FilterCollection logicalOperation="And">

<FD type="FromFD" leftVal="Sale.Id" comparisonType="Equals" rightVal="@OpportunityId"/>

</FilterCollection>

</FilterDefinitions>

</ED>

В результате в коллекцию LoadedData добавится:

[myOpportunity.Order:OrderCollection]

Где OrderCollection - List<Dictionary<string, object>> по одному элементу списка на каждую сущность полученую из EntitySchemaQuery.GetEntityCollection(UserConnection), а Dictionary<string, object> аналогичен части из LoadedData для единичной сущности.

# Определение полей отчета:

Принцип маппинга полей сущностей для единичной сущности и для коллекций различны, соответственно различна и разметка xml.

Для единичной сущности используем узел <MappingRows>, который состоит из элементов <MR>.

Для коллекции – используем узел <RepeatSections>, который состоит из элементов <RS>.

Элемент <RS> имеет два атрибута:

entityColectionDefinition="Order" – соответствует значению атрибута name="Order" элемента <ED> с атрибутом type="collection".

replaceCurrentRows="false" элемента <RS> указывает на необходимость смещать строки в шаблоне excel под ячейкой с маппингом (значение false) или нет (значение true).

Пример

<RepeatSections>

<RS entityColectionDefinition="Order" replaceCurrentRows="false">

<MR cellPath="{OrderDescription}" value ="Order.Description" formatString="{0} (так как будет прописано в приложении )" sort="true"/>

<Summary cellPath="{се}" value ="Order.PriceUnitCost" summaryFunction="summ" dataType="Number"/>

</RS>

</RepeatSections>

# Элементы <MR> и <Summary>

Для сопоставления значения поля сущности (<ED>) и ячейки (множества ячеек) используется элемент <MR>.

Атрибуты элемента <MR>:

destination="Formula" Необязательный – для вывода формул в отчет.

<MR destination="Formula" cellPath="{SKIsConcidered}" formatString="IF({0},&quot;учтена&quot;,&quot;не учтена&quot;)" function="equals">

Загрузка данных с хранимок

<ED name="PlacesReport" type="collection" storedProcedure="tsc\_AdditionalAgreementReport">

<ColumnDefinitions>

<CD name="Number" path="Number"/>

<CD name="RealTitle" path="RealTitle"/>

<CD name="RealAddress" path="RealAddress"/>

<CD name="PrintFormName" path="PrintFormName"/>

<CD name="placesCount" path="placesCount"/>

<CD name="placeCost" path="placeCost"/>

</ColumnDefinitions>

<FilterDefinitions>

<StoredProcedureParameters logicalOperation="And">

<SPP name="DocumentId" value="@DocumentId"/>

</StoredProcedureParameters>

</FilterDefinitions>

</ED>

Тег ColumnDefinitions **не обязательно**

**Отчеты WORD**

<MappingRows> - аналогично

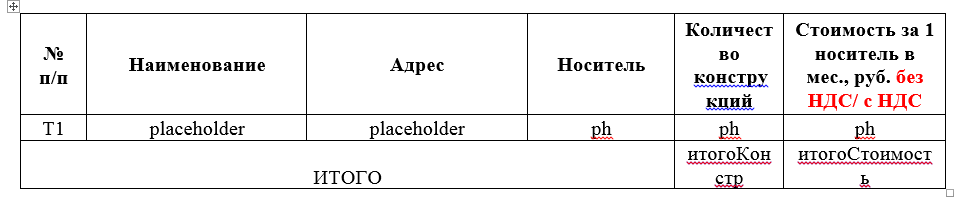
<TableMappings>

<TM tableRowPointer="T1" entityColectionDefinition="PlacesReport">

<Summary cellPath="итогоКонстр" columnName = "placesCount" summaryFunction="summ"/>

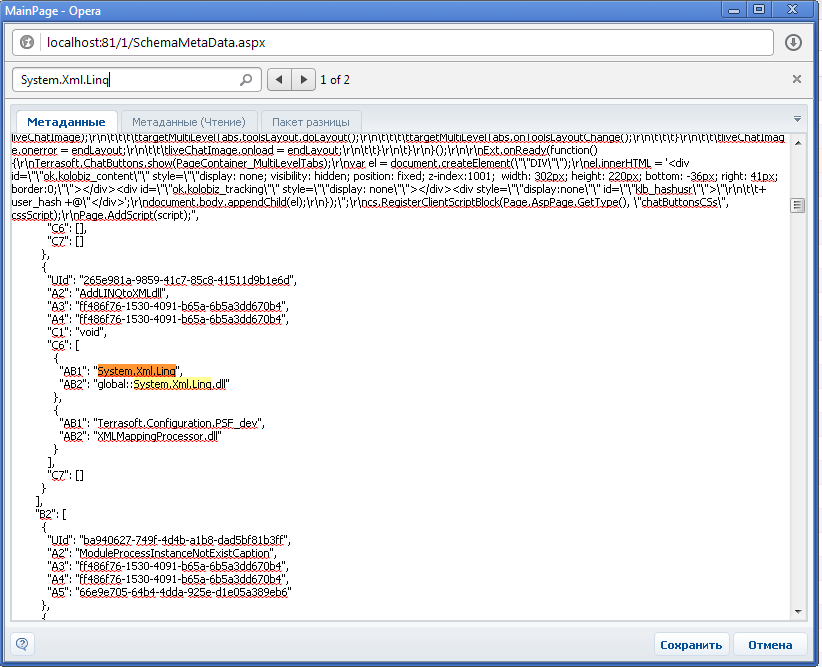
</TM>

</TableMappings>



# Добавление ссылки на сборку (System.XML.LINQ)

Можно посмотреть на <http://tswiki/pages/viewpage.action?pageId=16482795>, а можно и не смотреть:

Шаг 1: В базовой конфигурации. На любом методе (желательно создать свой) схемы MainPage в метаданных добавить свою сборку для свойства C6:

"C6": [

{

"AB1": "System.Xml.Linq",

"AB2": "global::System.Xml.Linq.dll"

}

]

Шаг 2 Удостоверяемся что в таблице SysSchemaInSolution появилась новая запись:

select top 1 Id,ModifiedOn, Name from SysSchemaInSolution order by ModifiedOn desc

Шаг 3: Выполняем скрипт:

insert into SysSchemaAssemblyReference (Id,SysSchemaInSolutionId,[Path])

values(

NEWID()

,(select top 1 Id

from SysSchemaInSolution

where Name = 'MainPage'

order by ModifiedOn desc)

,' global::System.Xml.Linq.dll'

)

Шаг 3: Публикуем MainPage.