Использование контекстного класса

Под контекстным классом понимается класс, сгенерированный механизмом сборки в LINQ-to-SQL. Этот класс наследуется от *DataContext*. Имя класса может быть задано на уровне сайта или контента, в случае отсутствия используется имя класса по умолчанию - *QPDataContext*.

Экземпляр по умолчанию

Если требуется работать только с одним контекстным классом, то можно использовать экземпляр контекстного класса по умолчанию, доступный как **LinqHelper.Context**. Он создается при первом запросе к нему и в случае вызова в Web-среде сохраняется на время выполнения текущего Web-запроса. В случае не-Web-приложений экземпляр сохраняется в статическом свойстве класса LinqHelper на все время жизни приложения.

При создании независимого от БД кода следует иметь в виду, что экземпляр контекстного класса, доступный как **LinqHelper.Context** по умолчанию использует настройки с которыми он был сгенерирован. Сменить эти настройки можно до первого обращения к **LinqHelper.Context** через изменение статических свойств контекстного класса.

Существует возможность использования нескольких экземпляров по умолчанию в Web-среде одновременно (если они находятся в разных пространствах имен). Единственное требование: чтобы имена сайтов, с которыми они работают, различались.

Явное инстанцирование контекстного класса

В случае, когда необходимо работать с несколькими контекстными классами можно использовать явный вызов конструктора, но если контекстный класс получены в режиме генерации кода, независимого от БД, то нужно использовать один из статических методов создания:

```
static QPDataContext Create(string connection, string siteName,
MappingSource mappingSource)
```

```
static QPDataContext Create(string siteName, MappingSource mappingSource)
```

```
static QPDataContext Create(string connection, string siteName)
```

```
static QPDataContext Create(string connection)
```

```
static QPDataContext Create()
```

- *connection* – строка подключения к БД
- *siteName* – имя сайта (в одной БД может быть несколько сайтов)
- *mappingSource* – источник данных для отображения на основе .map-файла, полученного в результате сборки контентов в LINQ-to-SQL классы

Если какой либо из параметров не задан, то вместо него используются соответствующие статические свойства контекстного класса, которые можно изменить перед вызовом метода создания.

Особенности использования

В обычном сценарии использования LINQ-to-SQL классов, сушностные классы используют экземпляр контекстного по умолчанию (**LinqHelper.DataContext**) в тех случаях, когда им требуется внутри себя подгрузить какие-то данные, например для реализации M2M-полей. Но в случае явного инстанцирования контекстного класса, это не будет работать, так как внешний и внутренний контексты будут отличаться. Поэтому у каждого экземпляра сущностного класса теперь есть свойство **InternalDataContext**, которое позволяет явно задать внутренний контекст для таких операций. По умолчанию оно инициализируется значением **LinqHelper.DataContext**.

Необходимость задавать внутренний контекст возникает при явном инстанцировании контекстного класса и использовании следующих операций:

1. Добавление новых статей. При этом экземпляр контекстного класса можно передавать прямо в конструкторе сущностного класса. Важно задать контекст до сохранения статьи.
2. Все операции с M2M-полями (свойствами-коллекциями внутри сущностных классов). Контекст необходимо задать у экземпляра сущностного класса перед обращением к его M2M-полю.
3. Использование дополнительных свойств полей-изображений. Контекст необходимо задать у экземпляра сущностного класса перед обращением к дополнительному полю.

```
EventsArticle event = new EventsArticle(context);
```

```
var item = ctx.TestArticles.Where(n => n.Id == 3737).Single();
var newCat = ctx.TestCategoryArticles.Where(n => n.Id == 1856).Single();
item.InternalDataContext = ctx;
item.Categories.Add(newCat);
ctx.SubmitChanges();
```

Если необходимо применение контекста сразу для коллекции объектов, то возможно использование метода, расширяющего IEnumerable<T>: **ApplyContext(DataContext context)**, определенного для каждого сущностного класса.

```
foreach (var item in ctx.PhotosArticles.Where(n => n.Photo !=
null).ApplyContext(ctx))
{
    // use item.PhotoUrl;
}
```

Использование XmlMappingSource

Источник данных для отображения обычно получают с помощью статического метода XmlMappingSource.FromXml, в который передается строка с XML-содержимым отображения.

Само XML-содержимое можно получить с помощью одного из экземплярных методов класса DBConnector:

- GetMapFileContents
- GetDefaultMapFileContents

Пример:

```
XmlMappingSource map =
XmlMappingSource.FromXml(Cnn.GetDefaultMapFileContents(Cnn.GetSiteId("Sandbo
x Net")));
XmlMappingSource map2 =
XmlMappingSource.FromXml(Cnn.GetMapFileContents(Cnn.GetSiteId("Sandbox
Net"), "EventsDataContext.map"));
```

Статические свойства контекстного класса

Позволяют настроить значения по умолчанию, которые потом будут использованы при инстанцировании контекстных классов, в том числе и для **LinqHelper.Context** перед первым обращением к нему.

- **DefaultConnectionString** – строка подключения к БД. Инициализируется строкой из секции *connectionStrings* файла *web.config*, название которой задано на уровне сайта. Если название не задано, то используется название по умолчанию - *qp_database*.
- **DefaultSiteName** – имя сайта. Инициализируется именем сайта, для которого были сгенерированы LINQ-to-SQL классы.
- **DefaultXmlMappingSource** – Инициализируется вызовом *XmlMappingSource.FromXml(Cnn.GetDefaultMapFileContents(Cnn.GetSiteId("имя сайта")))*. В качестве имени указывается имя сайта, для которого были сгенерированы LINQ-to-SQL классы.

Пример изменения статических свойств:

```
QPDataContext.DefaultXmlMappingSource =
XmlMappingSource.FromXml(Cnn.GetDefaultMapFileContents(Cnn.GetSiteId("Sandbo
x Net 7")));
QPDataContext.DefaultSiteName = "Sandbox Net 7";
```