Мультисайтовая загрузка объектов

Зачем это нужно?

Рассмотрим некоторый сайт, состоящий из нескольких логических подсайтов, например МТС. В соответствующей этому сайту базе QP7 существует один главный сайт, на котором присутствует как контент, так и шаблоны. Также в базе есть дочерние (региональные) сайты, в которых есть только контенты, относящиеся к данному региону. Вся логика работы сосредоточна на главном сайте. Главный сайт может получить доступ к контентам дочерних сайтов через механизмы Cross-Site Sharing, Dynamic Content Changing или через USER QUERY контенты. Эта схема хорошо работает, пока все региональные сайты идентичны друг другу. Как только логика их работы начинает различаться, то начинает неоправданно расти сложность объектов главного сайта, которые постепенно начинают состоять только из одних условных операторов, разделяющих функциональность по логическим сайтам.

Решение проблемы

В QP7 уже давно существует механизм перегрузки объектов на уровне страницы. Он работает следующим образом: если на уровне страницы присутствует объект с таким же названием, как и на уровне шаблона, то при загрузке соответствующего контрола предпочтение будет отдано объекту уровня страницы. Если же такого объекта на странице не существует, то будет загружен объект шаблона. Если нет и такого объекта, будет выброшено исключение.

Было логично обобщить этот алгоритм до мультисайтовой конфигурации. В этом случае наличие объекта сначала проверяется на дочернем сайте (также сначала на уровне страницы, потом на уровне шаблона), а потом и на основном. Модифицированный алгоритм загрузки объектов также поддерживает многоуровневую иерархию сайтов (например, как на МТС: главный сайт – макрорегион – регион).

Одним из требований к разработке данного функционала было то, что структуру базы данных желательно оставить неизменной, и данные об иерархии сайтов следует задавать в каком-нибудь внешнем источнике, например во внешнем XML-файле. С другой стороны, требовалось сделать достаточно гибкую архитектуру, чтобы разработчик мог модифицировать алгоритм загрузки объектов или полностью переписать его.

Поэтому была выбрана архитектура, при которой функционал мультисайтовой загрузки объектов реализуется в виде отдельной dll и была написана референсная dll. При этом разработчик может как использовать готовую референсную dll, так и написать свою dll, перекрывающую некоторые методы в Quantumart.dll и реализующую пользовательский алгоритм загрузки. Оба эти способа будут рассмотрены далее.

Как настроить?

Возможность мультисайтовой нагрузки присутствует в QP7, начиная с версии 7.6.3.0.

Рассмотрим вначале простой случай – использование готовой референсной dll.

1. Референсная MultiSitePage.dll хранится в TFS:
   $/QP7/MultiSitePage/MultiSitePage/bin/Release/MultiSitePage.dll

Ее нужно скопировать в папку bin главного сайта (или сделать ее доступной любым другим способом, например поместить в GAC).

2. Создать в корне сайта файл *sites.xml*, задающий иерархию сайтов. Главный сайт в нем должен быть в корне иерархии. Пример такого файла:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<structure>
    <site id="34">
        <site id="43">
            <site id="44"/>
        </site>
    </site>
</structure>
```

3. Передать в MulitiSitePage ID текущего дочернего сайта через глобальный параметр `HttpContext.Current.Items["MAPPED_SITE_ID"]`. Данное значение может быть получено, например, из модуля маскарадинга. Загрузка первого объекта происходит в обработчике Init, поэтому информация о текущем сайте должна уже быть доступна к этому моменту времени. Референсная dll требует, чтобы данный параметр был задан, иначе будет сгенерировано исключение. В то же время пользователь при необходимости может модифицировать референсную dll (путь в TFS: $/QP7/MultiSitePage/MultiSitePage), если, например, нужно использовать значение по умолчанию вместо генерации исключения.
4. Задать класс MultiSitePage в опции Custom Class на уровне страницы основного сайта или Custom Class for Pages на уровне шаблона основного сайта. После этого вновь собранные страницы будут унаследованы от класса MultiSitePage, а не QPage, как по умолчанию.
5. Нужно удостовериться, что физические пути дочернего сайта совпадают с виртуальными (live и stage). Настраивается в свойствах сайта.

Теперь рассмотрим более сложный случай – создание собственной логики загрузки объектов.

1. Нужно создать в Visual Studio новый проект, добавить в него ссылку на Quantumart.dll, и создать новый класс, унаследовав его от QPage. У класса желательно указать пространство имен – Quantumart.QPublishing, иначе потом в поле Custom Class придется задавать полное название класса вместе с пространством имен.
2. Нужно разрешить мультисайтовую логику. Это необходимо для того, чтобы в кэш были загружены таблицы объектов для всех сайтов, а не только для одного. Можно сделать это, например, в конструкторе класса.

```
public MultisitePage()
{
    QPageEssential.UseMultiSiteLogic = true;
}
```

3. Нужно перекрыть метод GetInternalCall. Приведенный ниже вариант используется в MultiSitePage.dll:

```
public override string GetInternalCall(string userCall)
{
    return QPageEssential.GetInternalCall(userCall, SitePath);
}
```

В качестве параметра SitePath в данном случае передается строка идентификаторов сайтов, разделенных запятыми. На этих сайтах будет проводиться поиск вызываемого объекта в том порядке, в котором эти идентификаторы встречаются в строке. Строка может формироваться

любым удобным образом. В MultiSitePage.dll это делается на основе sites.xml и параметра `HttpContext.Current.Items["MAPPED_SITE_ID"]`.

4. Перед вызовом первого объекта страницы может быть необходимо выполнить некоторую инициализацию. Это можно сделать, перекрыв метод BeforeFirstCallInitialize класса QPage.
5. Если необходима не модификация, а полное изменение логики загрузки объектов, можно перекрыть методы ShowObject и/или ShowControl класса QPage. Примеры реализации этих методов можно посмотреть в классе QPageEssential библиотеки Quantumart.dll.

Как использовать

1. Нужно создать в дочернем сайте шаблон с тем же именем, что и на главном сайте (как в случае переопределения объекта шаблона, так и объекта страницы) и страницу в этом новом шаблоне с тем же именем файла, что и на главном сайте (только в случае переопределения объекта страницы). Далее на уровне страницы или шаблона (в зависимости от того, какой объект мы переопределяем) нужно создать объект с тем же именем, что и на главном сайте.
2. после изменений, внесенных на дочернем сайте, нужно собрать объекты дочернего сайта, а затем обновить зависимости кэша главного сайта (это может быть достигнуто сборкой любой страницы основного сайта).

Замечания

1. При использовании MultiSitePage.dll режим мультисайтовой логики считается включенным, если в корне основного сайта присутствует конфигурационный файл *sites.xml* и в нем задан главный сайт.
2. Мультисайтовая логика не работает с Default Notification Template в целях оптимизации производительности.
3. Недостаток внешней реализации мультисайтовой логики в том, что невозможно инициировать сброс мультисайтового кэша при изменениях в объектах и страницах дочерних сайтов. Обычная логика сброса кэша такова, что при сборке страницы в любом случае сбрасываются все виды кэша, используемые для загрузки объектов: кэш объектов страницы, кэш объектов шаблона, кэш структуры. Но проблема мультисайтововой загрузки в том, что кэши являются специфичными для различных сайтов, то есть никакие процессы сборки на дочерних сайтах не приведут к сбросу кэша главного сайта. Для решения этой проблемы можно завести на родительском сайте специальную пустую страницу, и собирая ее, сбрасывать кэш. Страницу рекомендуется держать пустой, чтобы не росло число перекомпиляций, соответственно это позволяет избежать лишних перегрузок домена.
4. Для того, чтобы проект, использующий библиотеку MultiSitePage, работал с Visual Studio Add-In Backend Explorer, необходимо настроить иерархическую структуру для BE. В дальнейшем предполагается, что BE будет самостоятельно создавать иерархию сайтов вместо плоской структуры. Сейчас иерархию нужно создавать в самом IIS, подключая дочерние сайты как виртуальные папки. Затем через Solution Explorer нужно переключить все сайты на использование IIS вместо встроенного сервера, задавая URL каждого сайта в соответствии с настроенной иерархией.