13 Живые плитки

Есть ученые-садисты, которые бросаются искать ошибки, вместо того чтобы устанавливать истину.

Мария Кюри

Пользовательские интерфейсы Microsoft Windows 8 и Windows Phone характеризуются красочными блоками, которые напоминают многим пользователям те старомодные значки, которые сделали ранние версии Microsoft Windows столь популярными. Однако новые блоки значительно больше значков и под управлением операционной системы выводятся на экран вплотную друг к другу. Эти блоки называются *плитками* (tiles). Термин «плитки» относится главным образом к форме и размеру графического элемента. Плитки в Windows 8 (как и в Windows Phone) имеют дополнительную очень интересную способность: они могут выводить на экран специализированную информацию, генерируемую приложением в соответствии с пожеланиями пользователя, установившего приложение. Такие плитки называются *живыми* (live tiles).

Значок — это статичное изображение, позволяющее пользователю быстро идентифицировать приложение. Но он никогда не меняется, чтобы отразить своим видом текущее состояние приложения. В отличие от значка, живая плитка является своеобразным дополнением приложения, передающим некий контент операционной системе, которая затем выводит на экран содержащуюся в нем информацию пользователю, даже когда приложение отключено от сети или вообще не запущено.

С точки зрения разработчика, программирование живых плиток требует знакомства с новым прикладным программным интерфейсом и таким понятием, как *уведомление приложением* (application notification). В данной главе мы выполним

упражнение, в ходе которого добавим живые плитки к приложению TodoList, созданному в предыдущих главах.

Что собой представляет живая плитка

На рис. 13.1 показан начальный экран машины под управлением Windows 8. Каждый блок пользовательского интерфейса представляет собой установленное приложение. Хотя живая плитка может быть активной и ее информация может поддерживаться в актуальном состоянии, во многих случаях (например, когда приложение не подключено к сети) она представляет собой новую и более привлекательную версию обычного значка, знакомого всем по предыдущим версиям Windows. Все показанные плитки находятся в статичном состоянии, и каждая выводит на экран только логотип и имя приложения.

Рис. 13.1. Плитки на начальном экране машины под управлением Windows 8

Плитки в действии

Пользователи могут оперировать плитками практически так же, как значками в более ранних версиях Windows. То есть пользователь может перемещать плитки, группировать их, делать крупнее или мельче, а также включать и выключать режим

оперативных уведомлений. В конечном счете плитки представляют собой обновленные значки, обогащенные возможностью получения оперативных уведомлений от базовых приложений.

Перемещение живых плиток

Windows 8 создает новую плитку для каждого установленного приложения и закрепляет ее на начальном экране. После этого пользователь может убрать плитку с начального экрана, если не хочет ее там видеть, или даже вообще удалить вместе с самим приложением. На рис. 13.2 показано контекстное меню, которое появляется, когда пользователь щелкает на плитке правой кнопкой мыши.

Рис. 13.2. Контекстное меню плитки

Как видно на рисунке, доступными являются команды удаления приложения с начального экрана, полного удаления приложения, изменения размеров плитки, а также включения и выключения режима оперативных уведомлений от приложения.

Точно так же, как и при работе со значками в Windows 7 и более ранних версиях Windows, пользователи могут перемещать плитки и составлять из них горизонтальный прокручиваемый список. Но, в отличие от значков, плитки Windows 8 нельзя группировать в папки. Как показано на рис. 13.3, перемещение плиток осуществляется предельно просто, пользователю достаточно перетащить плитку мышью (или пальцем в случае сенсорного экрана).

Рис. 13.3. Перемещение плитки

Изменение размеров плиток

Плитки могут быть большие или малые. Малая плитка представляет собой прямоугольный блок размером 130 × 130 пикселов, большая плитка примерно в два раза шире. Как видно на рис. 13.3, перемещаемая плитка приложения Microsoft Internet Explorer является малой, а плитка приложения Mail в левом верхнем углу — большой.

Плитки для нестандартных приложений всегда создаются малыми. Только пользователь может изменить размер плитки, обратившись к контекстному меню. Из-за возможности смены размеров плитки создают сложности тем разработчикам, кто хотел бы встроить в свои приложения механизм уведомлений. Чтобы решить проблему, нужно применять разные графические шаблоны для малых и больших плиток, как мы сделаем в нашем упражнении.

Приложения без плиток

В контекстном меню плитки (см. рис. 13.2) есть команда Unpin From Start (Убрать с начального экрана), которая дает пользователю возможность убрать плитку приложения с начального экрана. Следует иметь в виду, что удаление плитки не означает удаления приложения. Это просто означает, что приложение убирается из списка приложений, видимых на начальном экране. В отличие от этой команды, команда Uninstall (Удалить) полностью удаляет приложение с устройства.

Чтобы запустить приложение, не имеющее плитки на начальном экране, нужно просто коснуться нижней части экрана (или щелкнуть правой кнопкой мыши за пределами плиток), вызвав контекстное меню с командой All Apps (Все приложения), предоставляющей доступ к полному списку установленных приложений, в том числе приложений без плиток (рис. 13.4). Можно также начать набирать имя приложения на начальном экране, и тогда пользовательский интерфейс оставит на экране только те приложения, чьи имена соответствуют набранным символам.

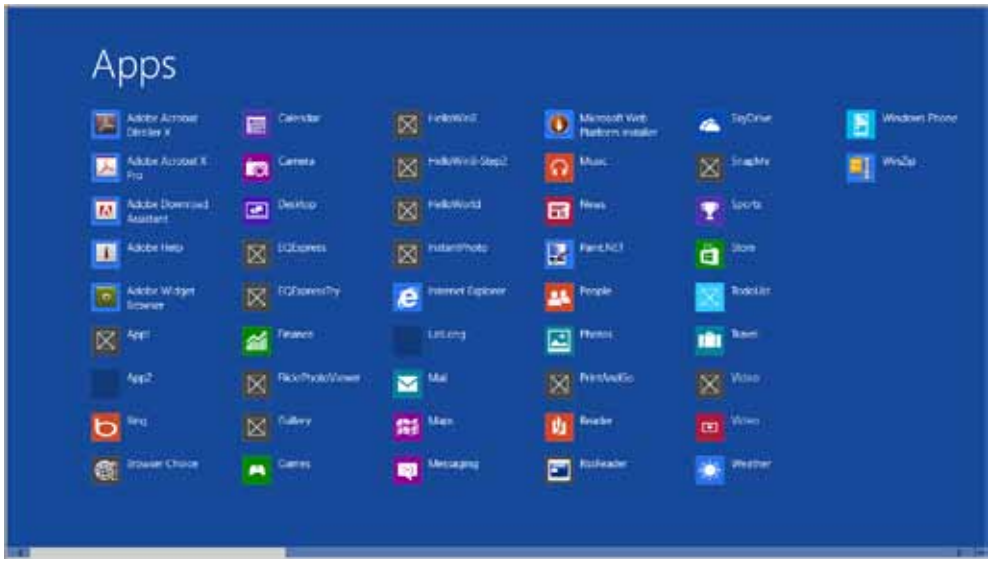

Рис. 13.4. Список всех установленных приложений

Создание живых плиток для базового приложения

Наиболее интересной стороной работы с плитками является их оживление путем программирования уведомлений. Чтобы решить задачу оживления плиток, создадим сначала новое учебное приложение, выводящее на плитке какой-нибудь статический текст. Затем мы доработаем приложение, создав более сложный пример, в котором выводимый текст будет отражать данные и состояние самого приложения.

Подготовка приложения

Создайте новое приложение Магазина Windows, используя шаблон Blank App (Пустое приложение), и добавьте в папку Pages обычные файлы header.html и footer.html. Потребуется также включить в файл default.css несколько новых стилей и добавить JavaScript-файл, названный по имени приложения tilesDemoApp.js. Кроме того, нужно включить в файл default.js следующую строку кода начальной загрузки:

```
app.onready = function (args) {
   tilesDemoApp.init();
};
```

Исходный контент файла tilesDemoApp.js выглядит так:

```
var tilesDemoApp = tilesDemoApp || {};
 tilesDemoApp.init = function () {
   // Сюда помещается остальной код
};
```

Код страницы в файле default.html должен выглядеть следующим образом:

```
<!DOCTYPE html>
 <html>
 <head>
   <meta charset="utf-8" />
   <title>Simple Tiles Demo</title>

   <!-- Ссылки WinJS -->
 <link href="//Microsoft.WinJS.1.0/css/ui-dark.css" rel="stylesheet" />
 <script src="//Microsoft.WinJS.1.0/js/base.js"></script>
 <script src="//Microsoft.WinJS.1.0/js/ui.js"></script>

   <!-- Ссылки TilesDemo -->
 <link href="/css/default.css" rel="stylesheet" />
 <script src="/js/default.js"></script>
 <script src="/js/tilesdemoapp.js"></script>
 </head>
 <body>
   <div data-win-control="WinJS.UI.HtmlControl"
     data-win-options="{uri:'/pages/header.html'}"></div>

   <h1>Simple Tiles Demo (CH13)</h1>
   <div data-win-control="WinJS.UI.HtmlControl"
     data-win-options="{uri:'/pages/footer.html'}"></div>
 </body>
</html>
```

Пока все хорошо, но еще ничего в приложении не вдохнуло жизнь в статичную по умолчанию плитку.

Объект уведомлений

Для оживления плиток нужно создать уведомление. В данном контексте уведомление является экземпляром объекта `Windows.UI.Notifications.TileNotification`. При создании экземпляра объекта уведомлений нужно передать ему какие-нибудь данные, идентифицирующие макет, который вы наметили для контента плитки. В течение жизненного цикла приложения объект уведомлений создается только один раз. Но впоследствии контент плитки можно менять столько раз, сколько требует логика работы приложения.

Объект уведомлений действует в качестве своеобразного моста между приложением и системой. После создания уведомление остается на своем месте некоторое время, даже если приложение не работает или отключено от сети.

Создание уведомления в приложении

Для добавления к приложению живой плитки нужно пройти три этапа. Сначала выбирается макет для текста плитки. Затем к макету добавляются данные приложения. И наконец, с помощью шаблона создается объект уведомлений и добавляется к системному списку.

С Windows 8 поставляется множество шаблонов плиток. Их можно найти в перечислении `Windows.UI.Notifications.TileTemplateType`. Каждый член перечисления ссылается на свой макет с несколькими местами, выделенными для заполнения текстом и (или) изображениями. Чаще всего используется следующий шаблон плитки:

```
Windows.UI.Notifications.TileTemplateType.tileSquareText02
```

Шаблон состоит из двух строк текста, автоматически стилизованных под заголовок и подзаголовок какого-нибудь приложения. Первая строка текста располагается в верхней части плитки и выводится крупным шрифтом. Вторая строка располагается в ее нижней части и выводится шрифтом меньшего размера.

В действительности шаблон является XML-строкой, но вам как разработчику не стоит слишком сильно вдаваться в детали XML. Достаточно получить содержимое шаблона и поработать над ним, заменив ряд элементов. К коду запуска вашего приложения нужно добавить следующий код в файле tilesDemoApp.js:

```
tilesDemoApp.init = function () {
    var template = Windows.UI.Notifications.TileTemplateType.tileSquareText02;
    var xml = Windows.UI.Notifications.TileUpdateManager.
      getTemplateContent(template);

    // Сюда помещается остальной код
}
```

Возвращаемый XML-контент зависит от выбранного шаблона. Что касается нашего шаблона, он состоит из двух элементов `text`, которые нужно заполнить текстом, выводимым на плитке. Для добавления контента, характерного для приложения, введите следующий код:

```
tilesDemoApp.init = function () {
    var template = Windows.UI.Notifications.TileTemplateType.tileSquareText02;
    var xml = Windows.UI.Notifications.TileUpdateManager.
    getTemplateContent(template);
    var textElements = xml.getElementsByTagName("text");

    // Заполнение текстом выделенных мест в шаблоне плитки
    textElements[0].innerText = "Title";
    textElements[1].innerText = "This is the subtitle";

    // Сюда помещается остальной код
}
```

И наконец, нужно зарегистрировать плитку в операционной системе, чтобы ее контент правильно визуализировался в начальном экране. Для этого в функцию `init` в файле tilesDemoApp.js нужно добавить несколько строк кода. В результате функция `init` должна приобрести окончательный вид:

```
tilesDemoApp.init = function () {
    var template = Windows.UI.Notifications.TileTemplateType.tileSquareText02;
    var xml = Windows.UI.Notifications.TileUpdateManager.
    getTemplateContent(template);
    var textElements = xml.getElementsByTagName("text");

    // Заполнение текстом выделенных мест в шаблоне плитки
    textElements[0].innerText = "Title";
    textElements[1].innerText = "This is the subtitle";

    // Создание и регистрация объекта уведомления
    var liveTile = new Windows.UI.Notifications.TileNotification(xml);
    Windows.UI
      .Notifications
      .TileUpdateManager.createTileUpdaterForApplication().update(liveTile);
}
```

Сначала из XML-шаблона создается новый объект уведомлений, а затем добавляется к системному списку живых плиток для установленных приложений. Для каждой активной в данный момент живой плитки система поддерживает объект, отвечающий за периодический вывод самого последнего контента на начальном экране. На рис. 13.5 показана живая плитка учебного приложения, запущенного на машине под управлением Windows 8.

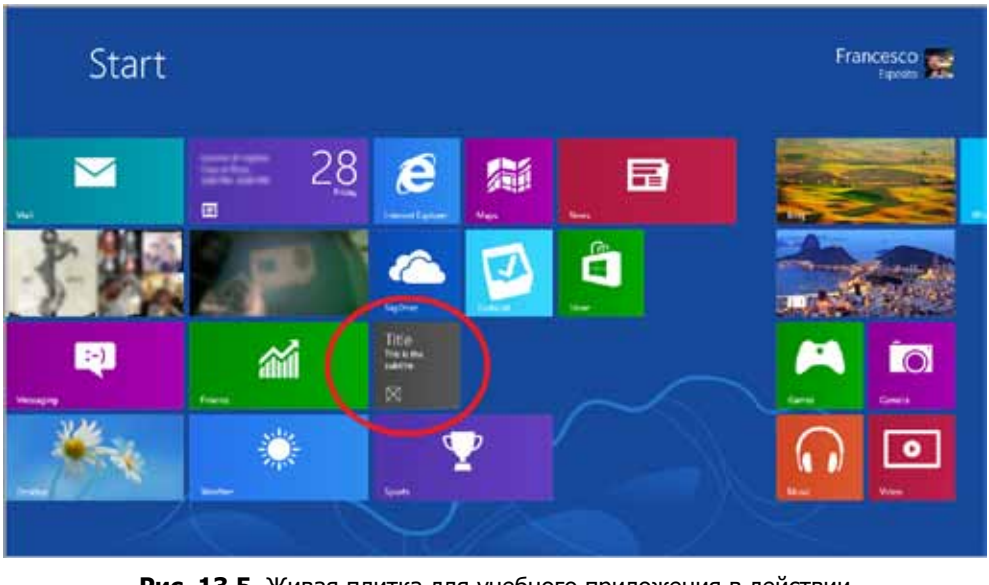

Рис. 13.5. Живая плитка для учебного приложения в действии

Независимо от выбранного шаблона, исходный логотип приложения всегда располагается в нижней части плитки. Остальная часть плитки находится в вашем распоряжении и может быть заполнена в соответствии с шаблоном.

Добавление живых плиток к существующему приложению

В конечном счете добавление живой плитки к приложению — не такая уж сложная задача. Все, что для этого требуется, сводится к нескольким вызовам предоставляемого системой прикладного программного интерфейса. Трудности возникнут, если вы попытаетесь добавить живые плитки к реально существующему приложению. В этом случае самой трудной частью работы окажется принятие решения о том, какие данные должны попасть на живую плитку, каким образом они должны извлекаться, как часто они будут обновляться и, разумеется, какой из шаблонов больше всего подойдет. Кроме того, может потребоваться поддержка как больших, так и малых плиток, чтобы у ваших пользователей создалось целостное впечатление от работы с Windows 8.

Возвращение к приложению TodoList

В главе 10 мы выполнили упражнение, в котором создали версию приложения TodoList, обеспечивающую сохранение данных. Окончательная версия приложения позволяла создавать, редактировать и удалять задания, и каждое задание сохранялась в роуминговой папке, обеспечивая тем самым попадание параметров приложения в облако и их совместное использование с другой копией того же приложения, установленной на другом персональном компьютере или устройстве.

В данном упражнении мы усовершенствуем приложение TodoList, созданное в главе 10, включив в него живые плитки.

Подготовка почвы

Сделайте копию проекта из главы 10 и назовите ее TodoList-Local. Возможно, вам захочется переименовать файлы проекта, чтобы в них отражался номер текущей главы. Возможно, вам также захочется откорректировать контент страницы в файле default.html из чисто графических соображений. Откройте файл default.html в текстовом редакторе в Microsoft Visual Studio и отредактируйте следующую строку:

```
<h1> TO-DO List (CH13) </h1>
```

Теперь все должно быть готово, чтобы вспомнить об устройстве данного приложения и составить планы относительно использования живых плиток. На рис. 13.6

показан основной пользовательский интерфейс приложения, который практически такой же, как был в главе 10. А на рис. 13.7 показана плитка, которую Windows 8 по умолчанию создает для приложения.

Рис. 13.6. Базовое приложение, дополняемое живыми плитками

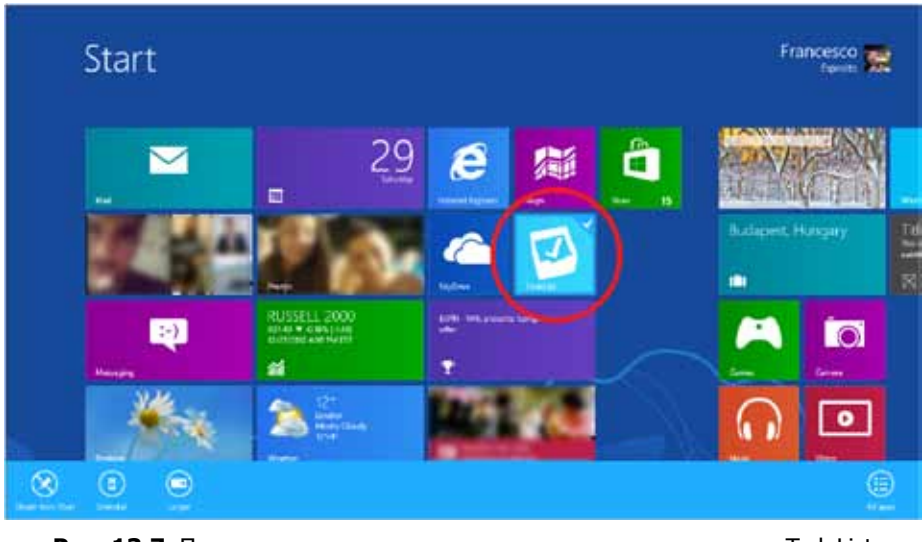

Рис. 13.7. Предлагаемая по умолчанию плитка для приложения TodoList

Когда вы в качестве пользователя щелкните правой кнопкой мыши на плитке, появится контекстное меню с доступными командами. Если живые плитки не активны,

соответствующие команды пользователю не доступны. Windows 8 определяет, поддерживает ли приложение живые плитки, и добавляет в контекстное меню еще одну команду, включающую и выключающую режим оперативного обновления.

Определение редактируемых файлов проекта

В предыдущем элементарном упражнении, как вы, наверное, заметили, добавление к приложению живых плиток требует совсем немного кода. Единственным файлом проекта, на который следует обратить внимание, является todolist.js. Фактически, живые плитки не оказывают влияния ни на общий пользовательский интерфейс, ни на манифест приложения, также они не требуют специальных разрешений.

Поскольку код управления живыми плитками работает изолированно, его нетрудно убрать в отдельный файл, который затем будет вызываться из файла основного сценария приложения. Поэтому далее при выполнении упражнения мы займемся созданием нового JavaScript-файла, а также, что, возможно, важнее, выбором более подходящего шаблона и контента, предназначенного для вывода на экран.

Реализация живых плиток

Живые плитки были задуманы как средство, дающее возможность разработчикам представить полезную информацию пользователям, не заставляя их даже открывать приложение. Живые плитки служат напоминанием о функциональности или контенте приложения, а также способствуют более частому обращению к приложению. Для успешного применения живых плиток они должны своевременно и привлекательно предоставлять полезную информацию, связанную с приложением. Поэтому выбор шаблона плитки и выводимых на ней данных является ключевым.

Подготовка почвы для живых плиток

Добавьте в проект новый JavaScript-файл. Назовем его liveTiles.js. Для начала нужно в только что созданный файл добавить следующий код:

```
var liveTilesManager = liveTilesManager || {};
 liveTilesManager.enable = function (listOfTasks) {
   // Сюда помещается остальной код
}
```

Кроме того, на файл liveTiles.js нужно сослаться из файла default.html. Поэтому откройте в редакторе Visual Studio файл default.html и добавьте к нему такую строку:

```
<script src="/js/livetiles.js"></script>
```

Теперь все готово к включению в диспетчер живых плиток более сложного кода.

Выбор шаблонов плиток

Windows 8 поставляется с большим перечнем готовых шаблонов для плиток. Здесь можно найти шаблоны как для больших, так и для малых плиток. По сути, шаблон плитки является небольшим фрагментом XML-данных, содержащим выводимую на плитке информацию. Обычно живая плитка состоит из изображений и одной или нескольких строк текста. Дополнительные сведения о доступных шаблонах плиток можно получить, обратившись по URL-адресу http://msdn.microsoft.com/en-us/library/windows/apps/hh761491.aspx.

Чаще всего плитка имеет заголовок и подзаголовок, состоящий, возможно, из нескольких строк. Но есть также ряд шаблонов, к которым добавлено изображение.

При выборе шаблона нужно также брать в расчет размер плитки и предусматривать желание пользователя изменить размер шаблона «на лету», используя для этого контекстное меню плитки. Для данного упражнения нужно выбрать следующие шаблоны, соответственно, для больших и малых плиток:

```
Windows.UI.Notifications.TileTemplateType.tileWideText01
Windows.UI.Notifications.TileTemplateType.tileSquareText02
```

Первый шаблон состоит из четырех строк текста разного стиля. Первая строка выводится крупным шрифтом, а для следующих строк используется обычный размер шрифта, они выводятся на разных строках без разрывов текста. Второй шаблон предназначен для малых плиток и состоит из одной строки заголовка, выводимой крупным шрифтом. За строкой заголовка следует подзаголовок, выводимый обычным шрифтом и занимающий максимум три строки.

Чтобы задействовать выбранные шаблоны плиток, добавьте в файл liveTiles.js следующий код:

```
liveTilesManager.enable = function (listOfTasks) {
    // Подготовка шаблона для БОЛЬШОЙ плитки
    var templateLarge =
      Windows.UI.Notifications.TileTemplateType.tileWideText01;
    var xmlLarge =
      Windows.UI.Notifications.TileUpdateManager.getTemplateContent
        (templateLarge);
    var textElementsLarge = xmlLarge.getElementsByTagName("text");

    // Подготовка шаблона для МАЛОЙ плитки
    var templateSmall =
      Windows.UI.Notifications.TileTemplateType.tileSquareText02;
    var xmlSmall =
      Windows.UI.Notifications.TileUpdateManager.getTemplateContent
        (templateSmall);
    var textElementsSmall = xmlSmall.getElementsByTagName("text");

    // Сюда помещается остальной код
}
```

Из предыдущего упражнения понятно, что шаблон плитки состоит из нескольких текстовых элементов. Имеющиеся в коде переменные `textElementsLarge`

и `textElementsSmall` являются массивами, состоящими из XML-узлов, ссылающихся на текстовые элементы в двух XML-шаблонах.

Следующий шаг состоит из заполнения этих текстовых элементов данными, принадлежащими запущенному приложению.

Выбор данных для вывода на плитках

Приложение TodoList полностью основано на списке заданий. У каждого задания есть описание, срок выполнения и приоритет. На живой плитке приложения TodoList, скорее всего, должно выводиться самое последнее задание или следующее задание, требующее завершения. Функция `liveTilesManager.enable` получает список текущих заданий и решает, какую информацию выводить.

В данном упражнении мы выберем первое задание и визуализируем его описание и срок выполнения. Для этого в файл liveTiles.js нужно добавить следующий код, точнее, этот код нужно добавить в конец функции `liveTilesManager.enable`:

```
// Получение информации из приложения
 // для плитки (плиток) с целью ее отображения
 var featuredTask = listOfTasks.getAt(0);

 // Добавление данных к плитке (плиткам)
 textElementsLarge[0].innerText = "TO DO";
 textElementsLarge[1].innerText = featuredTask.description;
 textElementsLarge[2].innerText = "";
 textElementsLarge[3].innerText = "due by</b>: " +
 featuredTask.dueDate.toLocaleDateString();

 textElementsSmall[0].innerText =
   liveTilesManager.getDueDateCompact(featuredTask);
textElementsSmall[1].innerText = featuredTask.description;
```

Также нужно добавить код к функции `liveTilesManager.getDueDateCompact`. Эта функция является вспомогательной, выполняя простое преобразование срока выполнения в формат *мм/дд/гггг*. Ее нужно поместить в конец файла liveTiles.js:

```
liveTilesManager.getDueDateCompact = function (task) {
   var date = task.dueDate;
   var day = date.getDate();
   var month = date.getMonth();
   month++;
   var year = date.getFullYear();
   var x = month + "/" + day + "/" + year;
   return x;
}
```

Объединение шаблонов малых и больших плиток

Несмотря на отсутствие на этот счет строгого требования, в любом приложении Магазина Windows нужно предусмотреть поддержку как малых, так и больших

плиток. До сих пор обе плитки настраивались независимо друг от друга, но этого недостаточно. Windows 8 требует, чтобы большие и малые плитки были объединены в один шаблон. Это можно сделать программно, дополнив функцию `liveTilesManager.enable` следующим кодом:

```
// Объединение шаблонов малых и больших плиток
 var node = xmlLarge.importNode(xmlSmall.getElementsByTagName("binding").
   item(0), true);
xmlLarge.getElementsByTagName("visual").item(0).appendChild(node);
```

В результате выполнения этого кода шаблон малой плитки добавится к шаблону большой плитки. Теперь все готово к созданию из шаблона объекта уведомлений и его регистрации в системе. Для этого нужен такой код:

```
// Создание объекта уведомлений
 var tileNotification =
   new Windows.UI.Notifications.TileNotification(xmlLarge);
 Windows.UI.Notifications
   .TileUpdateManager.createTileUpdaterForApplication().
    update(tileNotification);
```

Итак, все подготовительные операции, касающиеся создания плиток, выполнены. Теперь осталось связать приложение с плитками.

Связывание плиток и приложения

Плитки приложения обновляются, когда выполнение кода приложения доходит до инструкции обновления объекта уведомлений. Частота этих обновлений и выводимый контент зависят от приложения. Что касается приложения TodoList, объект уведомлений создается после запуска и обновляется при каждом редактировании, удалении или создании нового задания. Тем самым гарантируется, что пользователю в качестве напоминания всегда будут предоставляться самые свежие данные, даже когда приложение не выполняется.

Учитывая структуру приложения TodoList, лучше всего функцию `liveTilesManager.enable` вызывать из функции `populateTaskList`, которая, как вы знаете, определена в файле todolist.js. Найдите эту функцию и внесите в нее следующие изменения:

```
TodoList.populateTaskList = function () {
   var promise = new WinJS.Promise(function (complete) {
     var tasks = new Array();
     var localFolder = Windows.Storage.ApplicationData.current.roamingFolder;
     localFolder.getFilesAsync()
       .then(function (files) {
         var io = Windows.Storage.FileIO;
         files.forEach(function (file) {
           io.readTextAsync(file)
             .then(function (json) {
                var task = TodoList.deserializeTask(json);
                tasks.push(task);
                })
```

```
            .then(function () {
             var tasksList = new WinJS.Binding.List(tasks);
             tasksList = tasksList.createSorted(function (first,
                 second) {
                 return first.dueDate > second.dueDate;
            });
            var listview = document.getElementById("task-listview").
                 winControl;
             listview.itemDataSource = tasksList.dataSource;

            // Уведомление
            liveTilesManager.enable(tasksList);
          });
        });
    })
  });
  return promise;
}
```

По сравнению с исходной версией, в функции `populateTaskList` произошли два значимых изменения. Первое заключается в явном вызове функции `liveTilesManager.enable`, включающей живые плитки. А второе связано с объектом обязательства (`promise`), в который заключено все тело функции. Объект `promise` не является строго обязательным, но помогает в случае продолжения разработки. Когда функция `populateTaskList` возвращает объект `promise`, это дает возможность объединить поведение `populateTaskList` с другим поведением посредством методов `then` и `done`.

Большая живая плитка приложения показана на рис. 13.8.

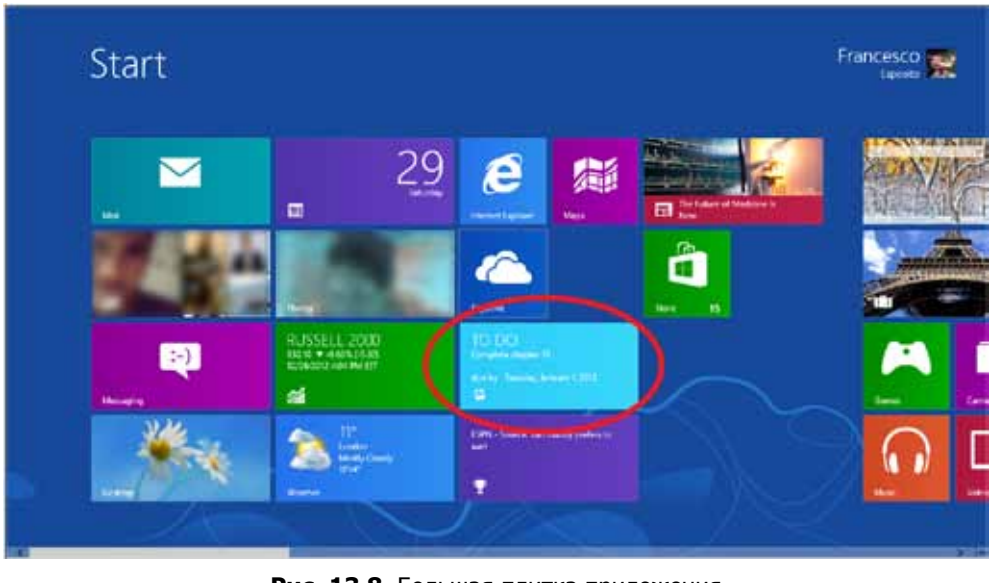

Рис. 13.8. Большая плитка приложения

Малая плитка показана на рис. 13.9. Как видите, теперь в контекстном меню плитки приложения появилась команда для включения и выключения режима оперативных уведомлений.

Рис. 13.9. Малая плитка приложения со своим контекстным меню

НЕТРИВИАЛЬНЫЕ ВОЗМОЖНОСТИ ЖИВЫХ ПЛИТОК

В двух упражнениях, рассмотренных в данной главе, представлены основные функциональные возможности живых плиток, чего вполне достаточно, чтобы разработчик мог приступить к программированию плиток. Но перечень функциональных возможностей плиток этим не исчерпывается. В частности, можно задать срок отключения режима уведомлений, чтобы живые плитки перестали обновляться, как только наступит заданное время. Это особенно полезно, когда приложению нужно выводить данные в форме напоминаний. Кроме того, содержимое плиток можно связать с фоновыми агентами, чтобы оно обновлялось в фоновом режиме, даже когда приложение не запущено. Фоновый агент представляет собой фрагмент кода, у которого нет пользовательского интерфейса, но который периодически запускается под управлением операционной системы. Благодаря фоновому агенту приложение может извлекать данные в асинхронном режиме и обновлять плитки даже без визуализации собственного пользовательского интерфейса.

Выводы

В этой главе мы узнали о живых плитках. Живые плитки — это уникальный инструмент Windows 8, которым приложения могут воспользоваться для вывода информации непосредственно на начальный экран. Плитка по умолчанию назначается любому приложению Магазина Windows во время его установки, а с помощью специального кода плитку можно оживить, заставив получать и визуализировать информацию приложения.

В первом упражнении мы сконфигурировали учебное приложение, снабжающее малую плитку заранее заготовленными данными. Затем мы доработали приложение TodoList из главы 10, включив в него малую и большую живые плитки с целью вывода деталей, касающихся последнего задания. Пользователь может управлять плитками: в любое время он может переключаться между малой и большой плитками и даже полностью отключить режим оперативных уведомлений.

Этой главой заканчивается изучение темы программирования для Windows 8, и теперь все готово для публикации приложения в Магазине Windows.