

Оцениваем технологию Adobe AIR



Александр Майоров

С бурным развитием технологий все больше и больше стирается грань между Интернетом и системами пользователей. Появляется много сервисов, заменяющих стандартные настольные программы. Идет бурное освоение десктопа интернет-приложениями, и свидетельством тому – новая технология от Adobe.

Суть проблемы

В последнее время наблюдается некоторый спад тотального доминирования ОС семейства Windows от компании Microsoft. Многие корпоративные секторы начали переходить на альтернативные системы. Также идет рост развития смартфонов и прочих мобильных персональных устройств. В связи

с этим наблюдается повышенный интерес к кроссплатформенному программированию. И это понятно. Писать для каждой платформы свою версию ПО с учетом всевозможных особенностей очень дорогое удовольствие. Многие знают о языке Java компании Sun Microsystems, девизом которого является выражение «Написано однажды –

работает везде!». Java является платформо-независимым технологическим решением. При разработке на Java нет необходимости в разработке программного обеспечения под каждую платформу: приложения можно внедрять без каких-либо изменений на любых платформах (ну почти, достаточно хотя бы попробовать написать програм-

му на J2ME, работающее одинаково хорошо в полноэкранном режиме на разных телефонах). Для работы такого ПО на каждой платформе должна быть установлена среда выполнения Java (Java Runtime Environment, JRE).

Также с бурным развитием Интернета все больше пользователей подключаются к Глобальной сети, и все больше появляется всевозможных веб-сервисов, не уступающих по своей функциональности настольным приложениям, а порой даже превосходящих их по некоторым параметрам. Достаточно привести в качестве примера Gmail и Google Documents.

Представьте себе, что сервис Gmail мог бы работать даже в оффлайн-режиме и мог бы запускаться не только через браузер, но и как самостоятельное настольное приложение. И вам не обязательно было бы все время иметь соединение с Интернетом для работы данного сервиса. Например, вы находитесь в метро с ноутбуком. У вас тем не менее открыт Gmail, оформленный в виде нативной программы, и вы пишете письмо. Нажимаете кнопку «Отправить», и Gmail сообщает вам, что письмо поставлено в очередь на отправку. Замечательно, не правда ли? Как такое возможно, спросите вы? Если бы мы писали такое приложение, то общая схема работы должна была быть следующей:

- проверка соединения с Интернетом;
- если соединение установлено – отправляем почту;
- если соединение не установлено – сохраняем почту в локальной базе данных, например SQLite. При следующем соединении с Интернетом отправляем почту.

Это примерное описание, но суть понятна. Но мало того, что веб-приложение должно синхронизироваться с локальной системой. Хотелось бы, чтобы оно было написано единожды, но работало на разных платформах, причем одинаково хорошо. Более того нужно, чтобы приложением возможно было воспользоваться на разных компьютерах, при этом не было бы необходимости синхронизировать данные. Все должно храниться на сервере, а на локальный компьютер кеширо-

вались бы только часто используемые данные. Фантастика? Ну почему же.

Средствами Java такое вполне реализуемо. Но мы сказали, что это веб-приложение, а значит оно и написано с применением таких инструментов как DHTML, XML, JS, CSS и так далее. Выходит, что нам надо бы писать отдельно веб-версию и клиентскую версию приложения, а это, в свою очередь, говорит о том, что надо применить уйму разных технологий, языков программирования и инструментов разработки. Все это накладывает свои определенные трудности и заставляет разработчиков осваивать много инструментов разработки, соответственно, тратить на это больше сил и времени.

А что, если в качестве инструмента программирования для написания такого клиентского настольного приложения мы бы использовали те же самые инструменты что и для WEB? А само приложение выполнялось бы в некоторой среде исполнения по типу Java-приложений. И весь наш сервис был бы доступен как из браузеров (естественно, с ограничениями, накладываемыми браузерами), так и в виде отдельного самостоятельного приложения, работающего независимого от того, есть ли в данный момент соединение с Интернетом. Причем одинаково вне зависимости от ОС, на которой это приложение исполняется в данный момент!

Воздушное решение от Adobe

Компания Adobe решила попытаться сделать миф реальностью и начала разработку проекта под кодовым названием Apollo, который в последствии был переименован в Adobe AIR (Adobe Integrated Runtime). Эта технология как раз и создана для веб-разработчиков, которые хотели бы не просто создавать веб-сервисы, но и писать обычные настольные приложения для десктоп-систем. Раньше это означало бы, что веб-разработчик должен переключаться в системного программиста и освоить языки программирования типа C/C++, Java, ObjectPascal или что-то подобное. Но теперь, с выходом новой платформы, это не обязательно. Adobe AIR позволяет частично решить эту проблему.

Adobe Integrated Runtime – это не просто средство для разработки многофункциональных мультимедийных сетевых RIA-приложений (Rich Internet Applications), но и среда выполнения для веб-сервисов на рабочем столе. Особенностью таких программ является обладание привлекательным интерфейсом и, конечно же, возможность работы как онлайн, так и оффлайн.

По сути AIR-приложение – это набор взаимосвязанных файлов, расширяемых в виде некоего AIR-контейнера. Среда исполнения AIR обеспечивает дополнительный функционал файлам, упакованным в AIR-архив. В обычном браузере или Flash-плеере такого функционала, естественно, нет.

Таким образом становится ясна двойственная природа AIR. С одной стороны, это программа запуска Flash-приложений, которая может показаться лишь усложненной версией старого Flash-плеера.

С другой стороны, она предназначена для запуска приложений, написанных на DHTML и JavaScript, что становится возможным благодаря механизму браузера Webkit, являющимся неотъемлемой частью Adobe AIR.

Появление такой технологии является серьезным шагом в сторону освоения настольных систем интернет-приложениями. Если раньше веб-разработчик мог работать исключительно в пределах окна браузера, при этом приходилось (да и сейчас приходится до сих пор) учитывать особенности разных версий, то теперь открываются поистине большие перспективы: разработчик может быть гораздо ближе к обычному пользователю, создавая новые продукты, более тесно интегрированные с привычной для пользователя средой. И при этом используются все те же инструменты разработки и языки.

Среди основных достоинств таких приложений компания Adobe выделяет быстроту создания и внедрения, а также простоту и удобство использования. Интерактивным приложениям будут присущи и некоторые особенности традиционных настольных программ, в частности, возможность работы с хранящимися локально файлами, функции взаимодействия

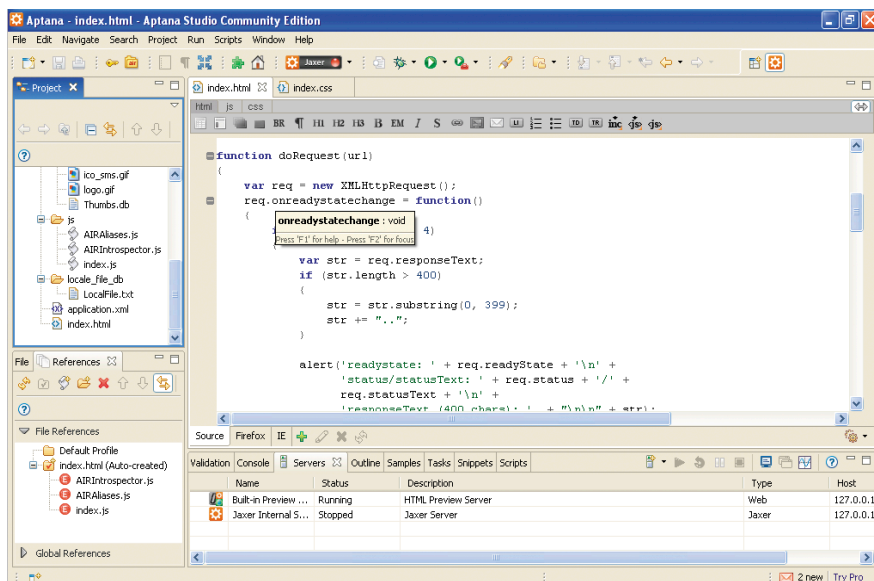


Рисунок 1. Среда разработки Aptana Studio

с другим ПО, установленным на компьютер, и прочее.

Возможности AIR

К основным возможностям, которыми обладает Adobe AIR, можно отнести встроенный браузерный движок, позволяющий использовать все возможности DHTML, CSS, JavaScript и Flash в едином рабочем пространстве.

При разработке приложений программист одновременно может использовать Javascript для доступа к возможностям Flash, а Flash-программист, в свою очередь, к JavaScript-библиотекам.

В итоге можно создавать гибридные приложения, сочетающие в себе плюсы обеих технологий. Имеется встроенное хранилище данных на базе SQLite, в котором приложение может хранить пользовательскую информацию на локальной системе. AIR-приложения имеют доступ к файловой системе на пользовательском компьютере, что позволяет осуществлять работу с файлами пользователя. Есть поддержка «Drag'n'Drop», что обеспечивает возможность «перетаскивания» данных между приложениями посредством мыши, как и во многих настольных приложениях. Поддержка прозрачности окон, возможность работы с буфером обмена и мультимедиа (звук, видео (FLV), экраном). Возможна асинхронная работа с сетевыми ресурсами, поддержка отправки и загрузки данных. Доступна работа с PDF и управление цифровыми правами на кон-

тент. Создана возможность для самостоятельного обновления приложений встроенными программными средствами.

Все основные возможности Adobe AIR по пунктам:

- File I/O API (встроенное API для работы с файловой системой пользователя);
- SQLite API (API для работы с SQLite из Adobe AIR-приложений);
- Flash Player 9;
- полнофункциональный HTML;
- связь между ActionScript3 и JavaScript;
- поддержка drag-and-drop;
- поддержка буфера обмена;
- ассоциация с типами файлов;
- иконки приложения;
- поддержка PDF.

Наряду с вышедшей бетой Adobe Flex 3 Adobe AIR является основой комплексной платформы компании, делающей возможным для разработчиков создание RIA-приложений и их доставку на устройства под управлением всех основных ОС. На сегодняшний день платформа поддерживает три операционные системы: Windows, MacOS и Linux. При разработке AIR-приложений можно использовать следующие технологии и библиотеки:

- Flash/Flex;
- DHTML;
- XML;
- CSS;
- JavaScript:
 - ✓ Yahoo User Interface (YUI);

- ✓ Prototype;
- ✓ Dojo;
- ✓ Ext JS;
- ✓ Adobe Spry;
- ✓ Aflax;
- ✓ jQuery;
- ✓ Mootools;
- ✓ Rico;
- ✓ Scriptaculous.

Инструментарий

В то время как для Windows и MacOS доступна уже 3-я бета, то для Linux доступна всего лишь альфа-версия. Поэтому приложения AIR для GNU/Linux пока не могут работать в полном объеме и использовать весь заявленный разработчиками функционал. Более того, Adobe AIR доступна не для всех GNU/Linux-платформ. Альфа-версия Linux AIR поддерживает следующие ОС: Red Hat Desktop 4, RHEL 5, Novell Desktop 9, SUSE 10, Ubuntu 6.06. Требуется рабочая среда GNOME или KDE, а также наличия композитных менеджеров Beryl, Compiz, Compiz-fusion для поддержки эффектов прозрачности. Не поддерживаются уведомления об обновлениях, регистрация типов файлов, запуск приложений из браузера, сочетания клавиш, печать, автозапуск приложений, отображение PDF и SWF внутри DHTML, стек протоколов IPv6. Нет возможности корректной работы с кириллицей.

Несомненно, все эти недостатки исправят в ближайшем будущем, и разрыв между версиями для разных платформ сократится. А пока лучше всего попробовать технологию на ОС семейства Windows или MacOS, хотя и под GNU/Linux стоит посмотреть (лично я, мягко говоря, не был в восторге от 1-й альфы, установленной под OpenSUSE 10.3).

Прежде чем приступить к разработке, следует скачать и установить необходимый набор инструментов. Альфа-версию Adobe AIR для GNU/Linux можно скачать по этой ссылке: http://labs.adobe.com/downloads/air_linux.html. Бета-версии для Windows и MacOS доступны на странице <http://labs.adobe.com/downloads/air.html>.

Помимо самой среды исполнения для разработки AIR-приложений вам необходим AIR SDK. Он состоит из следующих основных компонентов:

- AIR runtime;

- ADT;
- ADL.

ADT – AIR Developer Tool – утилита, позволяющая создавать AIR-архивы (файлы с расширением air) для распространения. По сути это утилита, создающая инсталлятор вашего AIR-приложения.

ADL – AIR Debug Launcher, является ключевым компонентом в AIR SDK. С его помощью вы тестируете AIR-приложение, перед созданием финального релиза.

AIR SDK доступна на сайте Adobe по адресу <http://www.adobe.com/products/air/tools>. Также там доступны средства разработки. AIR SDK можно скачать как отдельно, так и в виде расширения для Adobe Dreamweaver, что, несомненно, будет очень удобно разработчикам. Но лично мне показалось, что писать под AIR более удобно в специализированной студии Aptana Studio (<http://www.apтана.com>), основанной на Eclipse (см. **рис. 1**). Сам AIR SDK подключается к студии плагином. Естественно, никто не мешает просто подключить плагин в самой Eclipse. Особенно приятно, что эта среда разработки кроссплатформенная и существует почти для всех операционных систем.

Пишем приложение

После того, как вы установите среду Adobe AIR (см. **рис. 2**) и скачаете SDK, можно приступать к работе. С установкой среды проблем возникнуть не должно, так как даже для GNU/Linux-систем среда распространяется в виде бинарных пакетов. Что касается SDK, то это просто набор файлов и программ, которые можно установить куда угодно. Допустим, если это ОС Windows, то можете распаковать архив прямо в корень, в директорию C:\AirSDK, так будет удобнее работать. Тут уже вы сами решаете, как вам лучше сделать. Создайте директорию для ваших AIR-проектов, допустим это C:\AirProjects. Теперь почти все готово. Кстати, хочу сказать, что я буду показывать, как писать приложение без всевозможных дополнительных инструментов, таких как Aptana Studio. Дело в том, что таким образом вы лучше всего сможете понять архитектуру системы. А впоследствии, если вы всерьез займетесь разработкой на AIR, лучшим решением будет использовать специализированные инструментальные средства. Просто та же Aptana Studio при создании приложения за вас генерирует стартовый код, как и все студии вообще-то, что не очень годится для первого знакомства и понятия сути и идеологии разработки.

Также хочется сказать, что в статье не будет рассматриваться разработка сложного приложения, качество и возможности вашего AIR-приложения будут зависеть только от того, насколько хорошо вы знакомы с DHTML, Javascript, CSS, Flash, AJAX и прочими языками и технологиями, доступными для разработки на AIR. Скажу лишь, что хорошим учебником могут стать уже написанные приложения на AIR. Дело в том, что после инсталляции AIR-архив распаковывается в директорию с установленным приложением. И вам будут доступны некоторые файлы, из которых можно почерпнуть немало интересного. Также можно поискать в Интернете, благо статей на эту тему хватает, а с развитием технологии их будет становиться все больше и больше.

Итак, приступим. Не будем нарушать традиции, нач-



Рисунок 2. Установка среды Adobe AIR

нем с банального «Hello World», так как нам надо понять суть создания приложений на AIR. Создаем в нашей директории проектов новую папку, назовем ее HiPeople (надоело писать «хелловорлд»). В директории вашего проекта структуру вы задаете сами, как считаете нужным. Если это простое приложение, то можно все скинуть в кучу. Если вы делаете сложное приложение, то лучше разделять файлы. Минимальное разделение можно сделать таким образом: создать для каждого типа файлов свои директории. Например, для нашего проекта в директории C:\AirProjects\HiPeople создаем следующую иерархию:

- air;
- css;
- js;
- dhtml;
- xml;
- icons.

Далее условимся, что корневой директорией проекта будем называть каталог C:\AirProjects\HiPeople\ для краткости. Чтобы среда AIR могла понять, что у нас за программа и как ее запускать, нужно создать так называемый файл-дескриптор приложения. Это обычный XML-файл, в котором описываются основные свойства нашей будущей программы, например, такие как: название и автор ПО, размеры и свойства главного окна, иконки и многое другое. Без этого файла AIR-приложение работать не сможет. Итак, в корневом каталоге проекта создаем файл descriptor.xml. Содержимое файла приведено ниже:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<application xmlns="http://ns.adobe.com/air/application/1.0">
  <id>majorsoft.ru.example.HiPeople</id>
  <filename>main</filename>
  <name>HiPeople!</name>
  <version>0.1</version>
  <description>MajorSoft</description>
  <initialWindow>
    <content>index.html</content>
    <title/>
    <transparent>false</transparent>
    <visible>true</visible>
    <minimizable>true</minimizable>
    <maximizable>false</maximizable>
    <resizable>false</resizable>
    <width>600</width>
```

```

        <height>400</height>
        <x>200</x>
        <y>200</y>
        <minSize>400 200</minSize>
        <maxSize>1280 1024</maxSize>
    </initialWindow>
    <icon>
        <image16x16>icons/AIRApp_16.png</image16x16>
        <image32x32>icons/AIRApp_32.png</image32x32>
        <image48x48>icons/AIRApp_48.png</image48x48>
        <image128x128>icons/AIRApp_128.png
    </image128x128>
    </icon>
</application>

```

Некоторые записи файла нуждаются в пояснении:

- **Параметр id** – должен принимать уникальное значение и не должен совпадать с другими приложениями. Для разделения имен используются точки. Можете взять на вооружение такую схему генерации имен: адрес_вашего_сайта.ru.класс_приложения.название_данного_приложения. Тогда вы не запутаетесь в именах, и имя точно будет уникальным. Этот параметр обязательный.
- **Параметр filename** – используется для создания директории установки на системе пользователя. Параметр обязательный.
- **Параметр version** – служит для контроля версий. Обязательный параметр.
- **Параметр name** – имя, отображаемое в инсталляторе при установке вашего приложения. Устанавливается опционально по вашему желанию. То есть его можно даже не писать, если не хотите.
- **Параметры description, title, copyright** – говорят о своем предназначении сами за себя и не нуждаются в описании. Устанавливаются опционально.
- **Раздел initialWindow** – настройки главного окна приложения. Обязательный раздел. Без него приложение работать не будет. Этот раздел содержит:
 - ☑ **Параметр content** – стартовая страница приложения. Обязательный параметр. Не будем нарушать традиции веб-разработчиков и назовем наш стартовый файл index.html.
- **Раздел icon** – служит для описания доступных иконок приложения и устанавливается опционально. Если в вашем приложении нет иконок, то опустите этот раздел.

Параметров намного больше, но о них вы можете прочесть самостоятельно в документации. Могу сказать еще про пару параметров, которые могут быть интересны, но не обязательны для установки.

- **Параметр installFolder** – позволит вам установить директорию для установки по умолчанию. Этот путь высвечивается в инсталляторе при установке вашего AIR-пакета.
- **Параметр programMenuFolder** – позволяет задать директорию, которая высвечивается в «Start → Programs» в Windows. Параметр не обязателен и устанавливается только для ОС Windows.

С дескриптором закончим. Перейдем к нашему файлу стартового окна index.html.

```

<html><head>
<link href="/css/main.css" rel="stylesheet" type="text/css">
<title>My first application</title>

```

```

<script type="text/javascript"
src="/air/AIRAliases.js"></script>
</head><body>
<div class="class1">Hi, people!</div>
</body></html>

```

Для примера хватит. В файле css/main.css, как вы поняли, я создал описание стилей моего приложения. Вдаваться в подробности по поводу CSS не вижу смысла. Теперь в директорию air нашего проекта скопируйте файл AIRAliases.js из директории, куда вы устанавливали SDK. Для примера у меня этот путь выглядит так: C:\AirSDK\frameworks. Это файл для создания псевдонимов AIR API, осуществляющий быстрый доступ к классам среды.

Настал черед тестирования нашего приложения. Для этого нам понадобится утилита ADL. Она находится в папке bin в директории с SDK. Запускаете консоль и пишете команду:

```
adl descriptor.xml
```

Так как пути не прописаны, то вам придется указывать полные пути до утилит и файлов. Вы можете прописать пути в переменные окружения вашей ОС (способ зависит от ОС), можете написать простейший сценарий для запуска вашего приложения на тест, чтобы каждый раз не писать весь путь руками в консоли. Для Windows это будет файл такого типа:

```

@echo off
C:\AirSDK\bin\adl.exe descriptor.xml
pause

```

Положите этот файл в корневую директорию проекта, назовите run.cmd и запустите. Для пользователей Windows есть еще 1 способ. Достаточно перетащить мышкой ваш файл дескриптора на приложение adl.exe, и оно запустится, приняв в качестве аргумента ссылку на ваш файл дескриптора. После запуска вы увидите окно с заданными параметрами и нашей надписью.

Теперь попробуем использовать Javascript в нашем приложении. Попробуем загрузить текстовый файл и вывести его содержимое. Добавьте в файл index.html следующие строки:

```

<div>
<div id="divLocalFile">Empty...</div>
<input
type="button"
onclick="document.getElementById('divLocalFile').
innerHTML = readLocalFile('LocalFile.txt')"
value="Load local file"
>
</div>

```

Потом добавьте в файл js/main.js код следующего содержания:

```

function readLocalFile(name)
{
    name || alert('Empty file name!');

    var file = new air.File("app:" + name);
    var content = '';

    if (file.exists)
    {
        var fp = new air.FileStream();
        fp.open(file, air.FileMode.READ);
        if (fp.bytesAvailable > 0)

```

```

        {
            content = fp.readUTFBytes 1
                (fp.bytesAvailable);
        }
        fp.close();
        return content;
    }
    else
    {
        return "File " + name + " not exists!";
    }
}

```

Как видите, ничего сложного. Мы открываем файл, который должен находиться внутри нашего проекта, на что указывает `app:/` в имени файла. После чего создается объект `air.File`, с информацией о нашем файле. Например, размер файла можно узнать, обратившись к свойству объекта `size`: `alert(file.size)`. Таким образом только что было продемонстрировано, как можно работать с File/IO API, реализованном в AIR.

Кстати, для отладки не обязательно использовать `alert()` как привыкли многие веб-разработчики. В AIR есть специальная функция для этих целей. Если вы хотите вывести на консоль отладочную информацию, то используйте `air.trace()`. Вывод будет осуществляться именно в консоль, а не в виде всплывающих окон.

Также в комплекте есть специализированный инструмент для отладки приложений – интроинспектор. Эта консоль своего рода Firebug для Adobe AIR. Чтобы им воспользоваться, добавьте в директорию `air` нашего приложения файл `AIRIntrospector.js` из AIR SDK. В `index.html` добавьте строчку:

```

<script type="text/javascript" 1
src="/air/AIRIntrospector.js"></script>

```

Теперь, после запуска приложения, вы можете нажать клавишу `<F12>`, и будет запущен интроинспектор, который поможет вам отладить приложение.

Вернёмся к нашему приложению. Как вы видели, для того, чтобы обратиться к файлу, который находится в директории с нашим приложением, надо добавить префикс `app:/`. Соответственно, если вам надо обратиться к файлу из другой директории, то укажите путь до нее вместо префикса `app`. Кстати, пути можно вычислять и другим способом. Например, нашу строчку:

```
var file = new air.File("app:" + name);
```

можно заменить следующей:

```
var file = air.File.applicationDirectory.resolvePath(name);
```

Если вам надо взять файл с рабочего стола, к примеру, то можно это сделать так:

```
var file = air.File.desktopDirectory.resolvePath(name);
```

Теперь давайте посмотрим, как осуществляется навигация. Все ваши ссылки на странице будут открываться внутри приложения. Если вы создадите файл `test.html` и сделаете ссылку на него, то страница будет открыта в этом же окне приложения. Если вы хотите открыть в новом AIR-окне,

то напишите в ссылке `<target="_blank">`. То есть схема поведения понятна, здесь все то же самое, что и в обычных браузерах. Но вот если вы хотите открыть ссылку именно в браузере пользователя, который у него стоит, то для этого можно написать небольшую Javascript-функцию. В файл `js/main.js` добавьте следующий код:

```

function openUrlInStandartBrowser(url)
{
    air.navigateToURL( new air.URLRequest(url) );
}

```

В файле `index.html` для проверки функции достаточно написать:

```

<a href="#" onclick="openUrlInStandartBrowser 1
('http://majorsoft.ru/')">Open link in browser</a>

```

Рассказывать про то, как отсылать файлы на сервер посредством `XMLHttpRequest()` не буду, так как по этой теме вы точно найдете документацию. Лучше расскажу, как отправить файлы на сервер средствами AIR, и не просто, а через механизм «Drag & Drop». Итак, для начала напомним функцию инициализатор:

```

var target = null;
function init()
{
    target = document.getElementById('target');
    target.addEventListener("dragover", dragOver);
    target.addEventListener("drop", dropHandler);
}

```

Добавьте эти строки в файл `main.js`. В файле `index.html` надо добавить вызов загрузчика:

```
<body onLoad="init()">
```

и создать слой, который будет принимать «Drag&Drop» сообщения:

```

<div id="target" style="border:1px solid 1
#ccc;height:100px">Drag & Drop</div>

```

Что мы сделали? В функции `init()` мы повесили на слой с `id = target` обработчики «Drag&Drop» сообщений. Теперь их надо описать. Добавляем в `main.js`:

```

function dragOver(event) { event.preventDefault(); }
function onComplete(event) { target.innerHTML = 1
    "UploadComplete!"; }
function dropHandler(event)
{
    var request = new air.URLRequest 1
        ("http://majorsoft.ru/air_test_upload.php");
    request.method = air.URLRequestMethod.POST;
    var files = event.dataTransfer.getData 1
        ("application/x-vnd.adobe.air.file-list");
    for (i in files)
    {
        files[i].addEventListener 1
            (air.Event.COMPLETE, onComplete);
        files[i].upload(request, "uploadedFile");
    }
    // Вывод сообщений в консоль для контроля работы.
    // В финальной версии эта строка не нужна
    air.trace("File " + files[i].name + " is upload...");
}
}

```

Все. Как написать скрипт-обработчик на том же PHP, думаю, рассказывать нет смысла. Речь идет не об этом. Те-

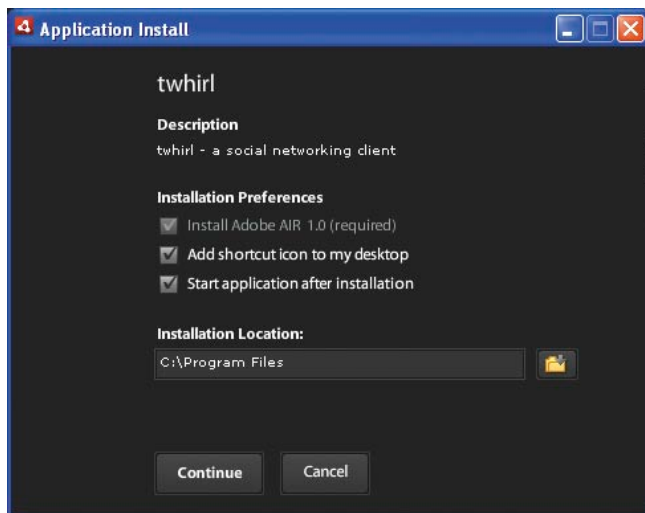


Рисунок 3. Окно инсталлятора AIR-пакета

перь перетащите любые файлы в «Drag&Drop» область нашего приложения и они будут загружены на сервер.

На этом хочу завершить вводный курс в программирование AIR-приложений. Для старта вполне хватит.

Сборка приложения

После того как было написано и протестировано приложение, его следует упаковать в AIR-контейнер, доступный для распространения. Делается это с помощью утилиты ADT из AIR SDK. Для работы ADT необходимо установить Java runtime (<http://www.java.com/ru/download/index.jsp>). AIR-контейнер должен быть подписан сертификатом. Таким образом проверяется подлинность AIR-приложения. Сертификаты известных разработчиков – это гарант качества приложения. Для тестирования можно использовать сертификат, который можно сгенерировать самостоятельно. Для продажи можно купить сертификаты, которые выдают, например, VerySign или Thawte. В нашем случае, когда пользователь устанавливает приложение, он видит, что «издатель»

не указан. Генерация сертификата осуществляется следующим образом:

```
C:\AirSDK\bin\adt -certificate -cn SelfSigned -ou Dev -o "HiPeople" -c US 2048-RSA MyCert.pfx 123
```

Последний параметр «123» в данной строке – это ваш пароль. Он пригодится при создании AIR-архива. Пароль вы задаете сами. Мне для теста проще было написать «123».

После того как выполните команду, должен быть сгенерирован файл MyCert.pfx, который и является нашим сертификатом для подписи приложения. Далее приступаем к созданию установочного пакета:

```
C:\AirSDK\bin\adt -package -storetype pkcs12 -keystore MyCert.pfx HiPeople.air descriptor.xml index.html js/main.js css/main.css air/AIRAliases.js LocalFile.txt icons/*.png
```

Мы перечисляем все файлы, которые обязательно идут в AIR-пакет. Будет запрошен ваш пароль, который вы писали при генерации сертификата. После этого на выходе получаем файл HiPeople.air весом до 40 Кб. Этот пакет мы и можем распространять и запускать на разных платформах. После установки на рабочем столе появится иконка нашего приложения, которая и позволит его запускать (см. рис. 3).

Обзор AIR-приложений

На официальном сайте Adobe предлагаются готовые AIR-приложения (http://www.adobe.com/cfusion/exchange/index.cfm?event=productHome&exc=24&loc=en_us).

По некоторым из них я сделаю краткий обзор, дабы вы имели более полное представление о рассматриваемой технологии.

Итак, первым в нашем импровизированном хит-параде идет программа WebKut (<http://toki-woki.net/p/WebKut>). Это простое, но от того не менее интересное AIR-приложение. С помощью WebKut вы сможете делать снимки веб-страниц или их частей буквально за секунду. Есть три возможности захвата скриншота: страница целиком, текущий вид или по выбору пользователя. Приложение очень удобное.

Twhirl (<http://www.twhirl.org>) – интересная реализация клиента для популярного сервиса Twitter. Twhirl переносит любимый сервис на ваш рабочий стол, и больше не нужно постоянно обновлять страницу с твитами – за вас это делает приложение (см. рис. 5).

ColorBrowser (<http://code.google.com/p/colorbrowser>) – удобный инструмент для работы с цветом. Есть возможность создавать шаблоны любимых цветов, а также наборы палитр. Полезный инструмент, пригодится дизайнерам.

Клиент для музыкальной библиотеки Finetune (<http://finetune.com>), расширяющий возможности сервиса с помощью AIR. Сервис представляет собой что-то среднее между настраиваемым радио и сервисом поиска музыкального контента. Напечатав название исполнителя, вы можете не только прослушать его композиции, но и получить список треков, которые подберет вам система рекомендаций. Есть возможность создания собственных списков проигрывания. Сервис предлагает возможность находить треки из музы-



Рисунок 4. Finetune-клиент

кальной библиотеки и на их основе создавать плей-листы (см. **рис. 4**).

AOL Top 100 Videos (<http://music.aol.com/help/syndication/desktop-widgets>) – представляет собой небольшое AIR-приложение с расширенными возможностями просмотра музыкальных клипов и текстового контента. Расширенные возможности ограничиваются просмотром видеоклипов в стандартном и полноэкранном режимах. Кроме того, можно поставить закладку понравившегося клипа и вернуться к его просмотру в любой момент.

Стоит ли овчинка выделки?

Что ж, пришла пора разобраться, что мы делали и стоит ли это того. Сказать, что технология от Adobe так уж нова, не могу. Дело в том, что в Windows что-то подобное было уже давно, только выполнялось все это не в отдельной среде исполнения и работало только на ОС Windows. Я говорю про специальный браузер, который обрабатывал файлы с расширением HTA. Этот браузер очень напоминает браузер от Adobe AIR. Правда, его не надо было дополнительно устанавливать, в отличие от AIR. Там также описывалось приложение через XML. В качестве языка программирования использовалась технология WSH (Windows Script Host), позволяющая работать с XML + JScript + VBScript. Ваш покорный слуга в далеком двухтысячном году даже писал приложение, очень напоминающее стандартный WordPad и умеющий редактировать локальные файлы пользователя. По внешнему виду приложение также очень смахивало на нативное ПО. Для его распространения использовались самораспаковывающиеся архивы SFX. Adobe AIR чем-то напоминает вышеописанный подход к созданию приложений, но более в масштабном варианте, да к тому же с использованием более развитых технологий. По этой причине я и не могу сказать, что Adobe AIR – это действительно инновационная технология.

Необходима ли данная платформа? Это спорный вопрос. Да, Adobe AIR действительно позволяет стирать грани между веб-приложениями и настольными программами. Используя возможности, предоставляемые DHTML, CSS, SWF, JS и т. д. Веб-ин-

терфейс можно оформить в стиле нативных программ. Однако следует понимать, что веб-интерфейс как клиентская часть веб-приложения будет существенно уступать по производительности и по возможностям обычным настольным приложениям, написанным с использованием компилируемых языков программирования, таких как C++ или Java. Набор инструментов, используемый при проектировании веб-интерфейсов, изначально для этого не предназначался.

Платформа AIR действительно расширяет спектр возможностей, доступных веб-разработчику. Но практически единственный аргумент, подтверждающий полезность Adobe Air, состоит в том, что не нужно разрабатывать два приложения, для веба и для десктопа, достаточно разработать одно. Однако ясно, что технологии, используемые в Adobe AIR, изначально не предназначались для проектирования десктопных приложений, поэтому создавать их будет непросто, да и по возможностям они будут уступать традиционным программам. Более того, код для клиентской настольной части и для серверной все равно будет отличаться, так что говорить, что Adobe AIR – это панацея от написания лишнего кода, никак нельзя. Я считаю, что если есть надобность в написании кроссплатформенных приложений, то надо смотреть в сторону Java или C++ с использованием кроссплатформенных библиотек. А если учесть, что сейчас идет бурное развитие мобильных устройств и все больше становится пользователей Всемирной паутины, то тогда Adobe должен начать выпускать AIR для мобильных платформ, а это, увы, с использованием текущих инструментов будет очень затруднительным. Да и скорость таких приложений будет оставлять желать лучшего. Но это мое мнение. Стоит ли осваивать эту технологию – решать вам и только вам. То, что стоит хотя бы посмотреть и опробовать, – это бесспорно! Особенно веб-разработчикам.

Также эта технология может возыметь и обратный эффект. Вроде бы делалось все для того, чтобы облегчить труд, но из-за ограниченности возможностей может возникнуть ситуация, что, чтобы решить задачу средствами AIR, надо будет писать много «косты-

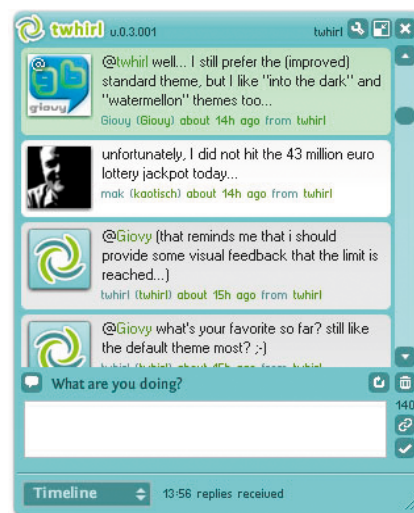


Рисунок 5. Twitter-клиент на AIR

лей», в то время как на Java или C++ это решалось бы без проблем. А вообще, конечно, неплохая идея. И эта технология имеет право на жизнь. Более того, может, через несколько лет программирование на AIR станет такой же обыденностью, как и использование AJAX в нынешних веб-сервисах. Поживем – увидим.

Уже сейчас написано достаточное количество программ для Adobe Air, позволяющее оценить технологию во всей красе (<http://airapps.pbwiki.com/FrontPage>). В Linux-версии среды AIR на данный момент работают не все приложения. Список поддерживаемого ПО в GNU/Linux можно взять здесь: <http://labs.adobe.com/technologies/air/samples>. Альфа-тестирование Adobe AIR для GNU/Linux должно закончиться 1 марта 2009 года. Выход финальной версии для всех заявленных платформ ожидается во второй половине 2009 года.

1. Статья про создание AIR-приложений в Aptana Studio – <http://extjs.com/blog/2007/06/29/building-a-desktop-application-with-ext-air-aptana-and-red-bull>.
2. Цикл статей о программировании на Adobe AIR + ExtJS – <http://dev.aol.com/blog/bricemason/adobe-air-xdrive-picture-syncing-part-1>.
3. Документация по Adobe AIR – http://www.adobe.com/devnet/air/ajax/getting_started.html.
4. Статья, демонстрирующая работу с AdobeAIR + Flex+ SQLite – http://lifeflex.shaggysmile.com/air/flex_to_air_migration.html.