Разработка Flash-проектов с использованием Apache Ant

Константин Ковалев (constantiner@narod.ru)
4 февраля 2006 года

I. Общие сведения об Apache Ant

История Apache Ant

Иструмент Apache Ant был придуман членом команды Apache Джэймсом Дунканом Дэвидсоном (James Duncan Davidson). Он работал над Tomcat 3.0 и изобрел Ant из простых побуждений – ему надо было собирать Tomcat. Согласно его версии, ANT расшифровывается как Another Neat Tool (еще один хорошо сделанный инструмент).

Ключевые факторы успеха Apache Ant

- Открытая расширяемая архитектура (extensible architecture)
- Производительность (performance)
- Наличие сообщества (community)
- Обратная совместимость (backward compatibility)

Общая информация об Apache Ant

- Бесплатный инструмент с открытым исходным кодом.
- Страница инструмента: http://ant.apache.org/
- Текущая версия – 1.6.5
- Версия 1.0 (с Tomcat 3.1) вышла в марте 2000-го года. Версия 1.1 (standalone) вышла в июле 2000-го.
- Стандарт de facto сборки приложений в настоящий момент

Зачем нужен определенный процесс сборки

- Позволяет однозначно определить составляющие сборки проекта: результатом разработки проекта является то, что, упрощенно, попадает в целевой каталог после запуска билд-файла.
- Позволяет определять версионность продукта: в системе контроля для определенной версии хранятся как исходники, так и билд-файл. В результате мы в любой момент можем собрать определенную версию.
- Наличие процесса разработки позволяет легче распределять ресурсы: каждый разработчик может самостоятельно интегрировать в систему сделанные им изменения и при этом ему не обязательно владеть всей информацией обо всех составляющих проекта.

Зачем нужен определенный процесс сборки (продолжение)

- Облегчается процесс создания сборок проекта в процессе разработки, интеграции, тестирования, продажи. Таким образом разрешаются нестыковки между этими процессами и экономятся ресурсы.
- Вселяет некоторую уверенность в заказчика: в результате разработки он получает не просто набор файлов, а способ в любой момент самостоятельно собрать из них программный продукт.
- Является одним из составляющих документирования проекта: билд-файл представляет из себя читаемый XML-файл, который может быть снабжен комментариями.
- Помогает в разворачивании серверных приложений.

Почему следует использовать Apache Ant

- Простой, основанный на XML синтаксис.
- Простота использования, которая аннулирует необходимость наличия специального специалиста по сборке (его роль может исполнять сам разработчик).
- Кроссплатформенность (написан на Java и требует лишь наличия виртуальной машины Java 1.2 и выше (для Apache Ant 1.6.5))
- Ant строит граф зависимости, что позволяет избежать исполнения ненужных целей (target). Если несколько целей зависят от данной, она все равно будет исполнена лишь однажды.
- С помощью Ant можно осуществлять сборку проектов, написанных на любых языках и технологиях, что позволяет вам собирать как клиентские приложения, так и серверные.
- Поддержка Ant встроена во многие современные IDE.

Другие известные альтернативы

IDE. Различные IDE предлагают свои возможности сборки проектов.

Недостатки:

- Зависимость от пользовательских настроек конфигурации IDE.
- Проекты, разные части которых делаются в разных IDE, не могут быть легко собраны.
- Множество IDE не являются кроссплатформенными. Таким образом, зачастую проект нельзя собрать на целевой платформе.
- Не все IDE поддерживают тестирование кода.
- Зачастую процесс сборки в IDE требует ручного вмешательства.

Другие известные альтернативы (продолжение)

Утилита Make. Эта утилита предназначена для сборки проектов на C/C++.

Недостатки:

- Плохая поддержка проверки зависимости задач.
- Более сложный и трудный для понимания синтаксис (по сравнению с XML).
- Трудность создания и понимания make-файлов со сложными зависимостями.
- Требует Cygwin при сборке под Windows.

Другие известные альтернативы (продолжение)

Jam, Amber, Cons. Менее известные и распространенные инструменты.

Основные концепции

- **Каждый билд-файл содержит в себе сборку одного проекта**. Если проект большой, он может быть разбит на подпроекты со своими билд-файлами, объединяемыми общим билд-файлом.
- **Каждый билд-файл содержит несколько целей (target)**, отражающих некие конкретные шаги сборки.
- **Каждая цель (target) может зависеть от других целей.** Вы можете определить, Какие задачи должны выполняться перед данной.
- **Каждая цель (target) состоит из задач (task).** Каждая задача представляет собой конкретное действие, совершаемое при сборке. Задача представляют собой Java-класс.
- **Вы легко можете добавить новые виды задач (task).** Для этого достаточно написать Java-класс, расширяющий один из существующих.

Установка Apache Ant под Windows

- Загрузить последнюю версию Ant с http://ant.apache.org/
- Распаковать каталоги bin и lib архива в созданный в Program Files каталог Ant.
- Указать каталог Ant в переменной окружения ANT_HOME.
- Добавить %ANT_HOME%\bin к переменной окружения PATH.

Ant и методики разработки программного обеспечения

Экстремальное программирование (XP, eXtreme Programming). В рамках XP Ant позволяет решать следующие задачи:

- **Автоматизированное тестирование**. Ant позволяет включать в билд-файл запуск тестов.
- **Постоянная интеграция**. Помимо того, что Ant позволяет разработчикам запустить билд-файл, осуществляющий запуск тестов и сборку всего проекта, существуют "обертки" Ant, позволяющий автоматизировать создание ежедневных билдов:
  - **AntHill** (http://www.urbancode.com/projects/anthill/) автоматически создает в репозитории метки для собираемого билда.
  - **CruiseControl** (http://cruisecontrol.sourceforge.net/) позволяет осуществлять весь цикл создания билда (синхронизация с репозиторием/билд/тестирование), а также выдает отчет о билдах.

Ant и методики разработки программного обеспечения (продолжение)

Унифицированный процесс (UP, RUP).

- Компиляция проекта и его разворачивание с помощью Ant.
- Тестирование в рамках RUP с помощью Ant имеет свои сложности, связанные с особенностями продуктов Rational (но данная задача также возможна).

Рекомендации по использованию Apache Ant (Best Practice)

- Начните проект с создания билд-файла.
- Назовите основной билд-файл build.xml и поместите его в корень проекта.
- Не используйте абсолютных путей локальной файловой системы в своем билд-файле. Изолируйте их в файле local.properties и поместите его в корень проекта.
- Всегда обеспечивайте в вашем билд-файле стандартный набор целей (target):
  - **init** – создает все временные каталоги
  - **clean** – удаляет все скомпилированные и промежуточные файлы, оставляя только исходники. Не стоит автоматически запускать эту цель, помещая в зависимые от других задач, разве что в случае специальной цели, генерируещей полный релиз.
  - **build** – компилирует исходники и совершает трансформации файлов.
  - **install** – копирует файлы на тестовый или публичный сервер.

Рекомендации по использованию Apache Ant (продолжение)

- Используйте элемент <description> для описания публичных целей. Список таких целей с описаниями можно посмотреть из командной строки. В качестве варианта можно предусмотреть цель help, которая выводит список публичных целей с описаниями.
- Если ваш билд-файл содержит более ста целей – следует разделить билд-файлы и вызывать из главного.
- Не стоит делить билд-файл на множество маленьких билд-файлов. Трудно понять процесс сборки при излишнем делении.
- Весь код, сгенерированный с помощью задач Ant и все файлы, порожденные в процессе билда, не должны храниться в системе контроля версий (сам билд-файл – должен).

Рекомендации по использованию Apache Ant (продолжение)

- Файл свойств, содержащий абсолютные ссылки на локальную файловую систему, не должен храниться в системе контроля версий.
- XML-файл билда должен быть хорошо отформатирован для повышения читаемости. Свойствам и целям (target) нужно давать осмысленные читаемые имена. Имя fla.dir лучше, чем fd. Стоит придерживаться единых соглашений по оформлению и именованию.
- Стоит помимо набора публичных целей, которые обеспечивают некоторый процесс сборки с зависимостями, достигающий определенной цели, обеспецивать набор экспертных целей-кирпичиков, позволяющих продвинутому разработчику осуществлять отдельные "оптимизированные" шаги.

Рекомендации по использованию Apache Ant (продолжение)

- Используйте файлы свойств (*.properties), которые позволяют хранить набор настроек в одном месте и позволяют предоставлять разные наборы для разных сред и операционных систем.
- При создании версии (метки или тэга) в вашей системе контроля версий, обязательно включайте туда билд-файл – важный артефакт разработки. При соблюдении этого правила вы всегда сможете воссоздать предыдущие релизы.
- Перед помещением файлов в систему контроля версий, запустите билд-файл и убедитесь, что ваш код компилируется.

II. Применение Apache Ant во Flash-проектах

Ant-tasks для компиляции Flash-проектов

as2ant (https://sourceforge.net/projects/as2lib/):

- Задача Ant от создателей библиотеки as2lib
- В настоящий момент доступна версия 1.0 beta
- Поддерживает задачи в рамках AMES:
  - **Mtasc Ant Task** для компиляции с помощью mtasc. Поддерживает всю функциональность данного инструмента, а также некоторые дополнительные задачи:
    - Позволяет компилировать классы в соотвествии с паттерном.
    - Позволяет компилировать классы выбранного каталога и всех его подкаталогов.
    - Позволяет использовать список компилируемых классов в формате XML.
  - **Swfmill Ant Task** для использования с XML-процессором для содания SWF-файлов Swfmill. Поддерживает все возможности swfmill.
  - **Swf Ant Task** для связи задач Mtasc и Swfmill. Позволяет создать swf, в который будут включены библиотечные символы и AS2-классы без необходимости вручную писать XML-файл, соотвествующий стандартам Swfmill.

Ant-tasks для компиляции Flash-проектов (продолжение)

as2ant (https://sourceforge.net/projects/as2lib/):

Плюсы as2ant:

- Бесплатные задачи с открытым исходным кодом.
- Полностью бесплатный процесс создания Flash-приложений (Mtasc и Swfmill – бесплатные инструменты с открытым исходным кодом)
- Наличие опенсорсного Flash-сообщества, поддерживающего данные проекты и не зависящего от рыночных механизмов.
- Поддержка Windows и MacOS (Swfmill), а также Linux для mtasc.

Ant-tasks для компиляции Flash-проектов (продолжение)

as2ant (https://sourceforge.net/projects/as2lib/):

Минусы as2ant:

- Отсутствует возможность компиляции проектов в среде разработки Flash от Macromedia/Adobe.
- Зависимость от инструментов сторонних разработчиков.
- Трудности при использовании AS2-кода от Macromedia/Adobe. Необходимость использования GUI-компонент от сторонних разработчиков.
- AS2-классы должны быть специально адаптированы для mtasc.

Ant-tasks для компиляции Flash-проектов (продолжение)

FDT (http://fdt.powerflasher.com/).

Данный инструмент поддерживает следующие задачи:

- **fdt.flashCompile** для компиляции с использованием среды разработки Flash.
- **fdt.viewDocument** для запуска swf-файла в Eclipse IDE.

Ant-tasks для компиляции Flash-проектов (продолжение)

FDT (http://fdt.powerflasher.com/).

Плюсы задач от FDT:

- Возможность работы в Eclipse IDE не переключаясь в другие среды.
- Возможность работы как с Flash MX 2004, так и с Flash 8.
- Поддержка Windows и MacOS.
- Нет необходимости задавать путь к среде Flash – запускается автоматически.

Ant-tasks для компиляции Flash-проектов (продолжение)

FDT (http://fdt.powerflasher.com/).

Минусы задач от FDT:

- Отсутствие возможности сборки вне Eclipse IDE (по крайней мере в том виде, как инструмент поставляется производителем).
- Отсутствие возможности настраивать параметры компиляции/публикации в билд-файле (все опции компиляции проекта ограничены лишь текущими настройками среды Flash и Publish Settings публикуемого файла).
- Отсутствие возможности задания classpaths (как следствие – необходимость задания во Flash IDE, либо в Publish Srttings публикуемого fla-файла. Первый вариант не исключает возможности конфликта в случае компиляции нескольких проектов, либо использования разных версий классов в разных путях. Также это требует, помимо установки необходимых приложений, их ручной настройки для компиляции под каждый проект. Второй вариант может усложнить компиляцию на других компьютерах при неверном указании путей, а также требует ручной установки для каждого fla-файла проекта – при необходимости добавить путь это чревато кропотливой работой и ошибками).
- Вследствие вышеперечисленного данные задачи пригодны лишь на этапе разработки, но малопригодня для ежедневной сборки и передачи проекта третьим лицам.

Ant-tasks для компиляции Flash-проектов (продолжение)

Flash Ant

(http://www.flashant.org/index.php?m=200312#4) от Aral Balkan.

Данная задача основана на инструменте компиляции командной строки, выпущенном Mike Williams и Ethan Malasky для Macromedia Central. Основан на использовании JSFL.

Ant-tasks для компиляции Flash-проектов (продолжение)

Flash Ant

(http://www.flashant.org/index.php?m=200312#4)

Плюсы Flash Ant:

- Возможность работы как с Flash MX 2004, так и с Flash 8.
- Поддержка Windows и MacOS.
- Возможность сборки проекта как в Eclipse IDE, так и отдельно.
- Возможность указывать множество fla-файлов.
- Возможность задавать выходной каталог для swf-файлов.

Ant-tasks для компиляции Flash-проектов (продолжение)

Flash Ant

(http://www.flashant.org/index.php?m=200312#4)

Минусы Flash Ant:

- Отсутствие возможности задания classpaths.
- Необходимость задания абсолютного пути к среде Flash (решаемо с помощью файла .properties).

Ant-tasks для компиляции Flash-проектов (продолжение)

FlashCommand от Mike Chambers (http://www.osflash.org/flashcommand) с модификацией от Constantiner'a (http://constantiner.blogspot.com/2006/01/flashcommand-091.html).

Также основан на использовании JSFL. При компиляции проекта используется задача exec.

Ant-tasks для компиляции Flash-проектов (продолжение)

FlashCommand.

Плюсы FlashCommand:

- Возможность работы как с Flash MX 2004, так и с Flash 8.
- Возможность сборки проекта как в Eclipse IDE, так и отдельно.
- Множество настроек публикации (log от Flash, выходной swf-файл итд.)
- Возможность задания classpath при публикации swf-файла.
- Возможность как автоматического поиска последней версии среды Flash, установленной на компьютере, таки и указания вручную.

Ant-tasks для компиляции Flash-проектов (продолжение)

FlashCommand.

Минусы FlashCommand:

- Работа только в Windows с установленным .Net 1.1 (для MacOS Mike Chambers создал Python-скрипт, которые не поддерживает задание classpaths).
- Необходимость при установке FlashCommand указывать переменную окружения PATH, либо задания абсолютного пути к инструменту (решаемо с помощью файла .properties).

Ant-tasks для компиляции Flash-проектов (продолжение)

Использование JSFL-скриптов в чистом виде или сформированных с помощью задачи ant.

Обладают, в общем-то, теми же плюсами и минусами, что и Flash Ant.