Основы Windows Workflow Foundation

Дхарма Шукла, Боб Шмидт

Отзывы

Если бы мне довелось писать книгу о Workflow, то я бы написал именно такую. Материал представлен очень удачно, с примерами кода и подробными объяснениями. К тому же мне нравится, как авторы обсуждают базовую архитектуру, это позволяет глубже разобраться в принципах, положенных в основу технологии, а значит, эффективнее разрабатывать собственные проекты.

– Джеффри Рихтер
(http://Wintellect.com)

Явная поддержка потоков работ в не слишком тяжеловесном каркасе – это основное достижение. К решению многих традиционно сложных задач, с которыми сталкиваются авторы приложений, например управление состоянием в случае длительных операций (речь идет о неделях и даже месяцах!), можно подойти систематически на основе потоков работ. Дхарма Шукла и Боб Шмидт представляют новую технологию, которая включена в состав .NET 3.0, причем делают это не только понятно, но и авторитетно. Читать книгу – одно удовольствие!

– Клеменс Шиперски,
архитектор программного обеспечения, корпорация Майкрософт

В технологии Windows Workflow, обогащая друг друга, соединились декларативное программирование и конечные автоматы. Это окажет заметное влияние на то, как мы будем программировать в ближайшем будущем. Следуя стилю классической книги Дона Бокса «Основы COM», Дхарма и Боб сумели описать новую технологию доходчиво для любого программиста, уже знакомого с языком C# или VB и с каркасом .NET Framework и желающего освоить декларативный подход. Не пропустите.

– Джо Даффи,
менеджер по разработке программных продуктов,
группа разработки общеязыковой среды исполнения (CLR)

Я думаю, что WF обязательно станет основной моделью Web-сервисов. Программисты, разрабатывающие такие приложения, захотят изучить новую технологию по этой книге. Она написана непосредственно авторами, а кто, как не они, знают свое детище во всех деталях?

– Кшиштоф Цвалина,
менеджер по разработке программных продуктов, корпорация Майкрософт

Эта книга, равно интересная и новичку, и ветерану, представляет собой великолепное описание новой технологии.

– Нейт Талберт,
проектировщик программного обеспечения, корпорация Майкрософт

Оглавление

Предисловие

Я пишу это предисловие в июле 2006 года и знаю, что грядет нечто важное.

Разработчики открывают для себя, насколько удобно иметь машиночитаемое представление программ, отвечающих их намерениям. Иногда эту идею называют стенографической записью кода в виде данных.

Но важнее другое: разработчики начинают осознавать, что ни архитектор среды исполнения, ни проектировщик языка не помогут им определить схему этих данных; для этого необходим специалист в предметной области, обладающий нужными знаниями. Так возникла идея предметно-ориентированных языков.

Системы типа Smalltalk, виртуальной Java-машины (JVM) и общеязыковой среды исполнения (CLR) доказали ценность машиночитаемых определений типов. Это полезно для отражения, сериализации и порождающего программирования. Но собственно определение типа (поля, методы, классы и т.п.) – это довольно замкнутый мирок, который не позволяет пользователям моделировать в виде данных такие вещи, как поток управления, параллелизм, логические конструкции или аспекты, характерные для предметной области, – правила предоставления скидок или идею ноты с точкой. Разве что в виде малопонятных команд, разбросанных по разным методам.

Возникает вопрос: «Если определение моего типа можно представить в виде данных, то почему то же самое нельзя сделать в отношении других конструкций, возникающих на этапе проектирования?»

К счастью, в 2003 году этот вопрос пришел в голову моего доброго друга Дхармы Шуклы, который тогда был членом группы по разработке Biztalk Server.

Стремясь превратить механизм взаимодействия (orchestration engine) XLANG в обобщенный встраиваемый платформенный компонент, они могли бы просто взять некоторые языковые конструкции из XLANG и обернуть их в еще один диалект XML (такая идея, конечно, выдвигалась). Это наиболее очевидный подход, и он прекрасно устроил бы всех с точки зрения построения Windows Orchestration Engine (WinOE).

Но Дхарма понимал, что не ему определять «единственно правильную схему» для всех программ, и решил перейти на «метауровень». В результате он возглавил проект создания расширяемой среды исполнения, которая позволила бы пользователям самостоятельно определять коды операций, соответствующих знакомой им предметной области. Добавьте сюда намерение определить эти коды и их взаимодействия в виде диалекта XML, и вы получите систему, которая даст возможность любому человеку определить собственный словарь и

структуру предложений для описания приложений в конкретной предметной области.

В этой книге Дхарма Шукла и Боб Шмидт умело демонстрируют нам, что Windows Workflow Foundation (WF) – отличный пример метасреды исполнения, которая позволяет разработчику управлять способами написания и исполнения программ. Разработчик описывает схему программы и ее интерпретацию, которая определяет, как программу следует развертывать и исполнять. У этой простой идеи весьма далеко идущие последствия.

Как я уже сказал, грядет нечто важное.

Дон Бокс,
июль 2006, Ярроу Пойнт, штат Вашингтон

Введение

Windows Workflow Foundation (WF) – это каркас общего назначения для создания реактивных программ, отвечающих на воздействия со стороны внешних объектов. Важнейшая характеристика реактивной программы заключается в том, что в ходе выполнения она приостанавливается на неопределенное время в ожидании входных данных.

Конечно, реактивные программы не новы. Со времен создания первых сетей, объединяющих несколько компьютеров, стала ясна польза от взаимодействия программ, работающих на разных машинах. Существуют также способы обмена данными между программами, которые исполняются на одном компьютере. С годами были разработаны самые разные методы взаимодействия – от сокетов до Web-сервисов. Хотя оперативная совместимость, масштабируемость и удобство различных программных моделей со временем значительно улучшились, вопросам совместной работы реактивных программ уделялось все же сравнительно мало внимания. Так, в популярных моделях программирования для Web ответственность за надлежащую организацию потока управления возлагается целиком на разработчика. Технология WF – попытка изменить такое положение вещей.

В модели программирования WF основным является понятие операции (activity) – предложения программы. Выполнение операции – и это принципиально важно – может быть в любой момент приостановлено, а затем возобновлено в зависимости от взаимодействий с внешними объектами. Почитав книжку на сон грядущий, вы оставляете в ней закладку. Так и операция может создать закладку, чтобы запомнить, в каком месте она была приостановлена (в ожидании входных данных), а затем продолжить выполнение с этого места (когда данные поступят).

WF – это каркас, а не фиксированный набор программных конструкций, диктуемых грамматикой языка. Понятие операции в WF расширяемо, что дает возможность писать более выразительные программы и организовывать поток управления более разнообразно, чем это позволяют такие языки, как C# или Visual Basic. С помощью WF-программы можно представить операции и специализированные потоки управления, характерные для конкретной предметной области, и тем самым описать сложные взаимодействия между людьми и программами, точно учитывая особенности решаемой задачи.

WF-программы исполняются метасредой, надстроенной над общеязыковой средой исполнения (CLR). WF-программа естественным образом приостанавливается, возобновляется и может работать сколь угодно долго в распределенной (многомашинной) среде без каких бы то ни было дополнительных усилий

со стороны программиста. WF-программа не завершается с ошибкой и не потребляет ресурсов системы, даже если неделями бездействует. Задача CLR в контексте WF – обеспечить управление объектами, которые временно представляют WF-программу, когда она находится в памяти. Задача же среды исполнения WF – управлять всем жизненным циклом WF-программы, который может включать несколько потоков и доменов приложений CLR, процессов операционной системы и даже машин.

Короче говоря, WF – это программная модель для написания и исполнения реактивных программ. WF-программы конструируются с помощью предложений, привязанных к конкретной предметной области. Эти предложения называются операциями и позволяют специалистам в предметной области выражать свои намерения на знакомом им языке.

Об этой книге

Эта книга была написана по одной простой причине. Мы оба считаем, что в основе WF лежит новый синтез заслуживающих внимания идей. Они впервые появились на одной из наиболее распространенных платформ. Поскольку программирование с ориентацией на операции базируется на иных принципах, чем современные парадигмы, то изучение технологии WF лучше всего начать с фундаментальных идей, положенных в ее основу. Чтобы эффективно разрабатывать программы для WF, недостаточно просто ознакомиться с типами в новом пространстве имен `System.Workflow`.

Кстати говоря, мы и не ставим себе целью рассмотреть все 350 с лишним типов, находящихся в трех сборках, которые в совокупности составляют WF. Мы сознательно проигнорировали некоторые внешние уровни WF и сосредоточились на сути – основах модели программирования и средствах, доступных во время исполнения. На собственном опыте мы поняли, что начинать изучение каркасов с усвоения общих принципов – самый верный способ стать квалифицированным разработчиком.

Работая над книгой, мы пересмотрели и обсудили собственные представления о природе реактивных программ и методах их разработки. Если эта книга – и WF в целом – натолкнет кого-то на свежие идеи о том, в каком направлении двигаться дальше, то мы будем считать свою цель достигнутой.

Книга организована следующим образом. В главе 1 «Составные части WF» рассматриваются ключевые идеи, лежащие в основе модели программирования WF: закладки, продолжения, независимость от процесса и потока, пассивация, возобновляемые предложения программы и среда исполнения возобновляемых программ. Все это обсуждается вне контекста WF, чтобы не затемнять суть вопроса.

В главе 2 «WF-программы» понятия, введенные в главе 1, отображаются на модель программирования в каркасе WF. Тем самым эта глава пролагает дорогу к остальной части книги. В ней мы начнем разрабатывать операции, составлять из них WF-программы и исполнять эти программы.

В главе 3 «Выполнение операций» и в главе 4 «Еще о выполнении операций» мы детально обсудим, как выполняются операции: механизм закладок, обработку ошибок, отмену и компенсацию. Сквозной темой будет конечный автомат, который описывает жизненный цикл всех операций. В главе 5 «Приложения» рассматривается создание приложений, которые загружают среду исполнения WF и пользуются ее точками расширения. Глава 6 «Транзакции» посвящена критической роли транзакций в исполнении WF-программ. В главе 7 «Дополнительные вопросы разработки» рассматривается ряд более сложных тем, относящихся к созданию операций и WF-программ, в том числе проверка и компиляция. В главе 8 «Разное» мы коснемся некоторых возможностей WF, базирующихся на ранее рассмотренных принципах.

Приложение А «Конечный автомат операций» является справочным пособием, а в Приложении Б «Образцы потоков управления» приведен код нескольких составных операций, более сложных, чем в основном тексте. На этих примерах мы иллюстрируем точки расширения, присутствующие в модели программирования WF, и способы реализации сложных потоков управления с помощью составных операций.

В этой книге мы будем заниматься только технологией Windows Workflow Foundation. Мы предполагаем, что читатель владеет языком C# 2.0 и основами CLR. Авторитетными источниками по этим темам могут служить следующие книги: Anders Hejlsberg и др. «The C# Programming Language», второе издание (Addison-Wesley, ISBN: 0321334434) и Don Box, Chris Sells «Essential .NET, Volume I: The Common Language Runtime» (Addison-Wesley, ISBN: 0201734117). Все приведенные в этой книге примеры просты, мы сознательно отсекали возникающие на практике осложнения, чтобы не отвлекаться от обсуждения основных концепций. Разобравшись, что к чему, вы сможете применить описанные идеи и методы к интересующей вас предметной области.