tion Standard. – N.Y.: Springer, 2002.

УДК 004.421.2

Рекурсивное программирование

Ковальков А.Т.
Белорусский национальный технический университет

Рекурсия, т.е. обращение некоторой подпрограммы (процедуры или функции) к самой себе, имеется практически во всех современных языках программирования. Отношение программистов к рекурсии неоднозначно – от восторженных отзывов до почти полного отрицания. В целом использование рекурсии даже опытными программистами не так активно, как она этого заслуживает. Между тем рекурсия является фундаментом таких языков программирования, как Пролог и Лисп, в которых она стала основным механизмом программирования.

Сдержанность многих программистов к рекурсии объясняется главным образом тем, что они, привыкшие при программировании на процедурном языке разрабатывать алгоритм решения задачи, с этих же позиций программируют и рекурсивную процедуру, стараясь вникнуть в достаточно непростой механизм реализации рекурсии. Такой подход непродуктивен.

Есть другой метод рекурсивного программирования, суть которого в следующем. Конец рекурсивных вызовов определяется граничным или терминальным условием, которое строится заданием таких значений входным параметрам, для которых сразу можно записать значения выходных параметров. При построении рекурсивных предложений предполагаем, что рекурсивный вызов процедуры или функции при измененных входных параметрах вычисляет промежуточный результат, который на один шаг отличается от окончательного результата. Решение получаем, используя промежуточный результат. Такой подход к программированию позволяет просто, без разработки алгоритма и не вникая во внутренний механизм работы рекурсии, логически конструировать нужные рекурсивные подпрограммы.

Таким образом, использование предлагаемой опробованной на практике методики построения рекурсивных подпрограмм позволяет просто использовать рекурсию не только в декларативных языках программирования, к которым относятся такие языки как Пролог и Лисп, но и в процедурных языках типа Паскаль, Си.

Построение рекурсивных подпрограмм по предлагаемой методике позволяет превратить программирование и на процедурных, и на декларативных языках в увлекательное занятие, равноценное решению

логических задач. В то же время предлагаемая методика поможет программистам, использующим рекурсию, быть более продуктивными при проектировании рекурсивных процедур.

УДК 683.3

Архитектура корпоративных приложений. Построение уровня доступа к данным приложений

Попова Ю.Б., Бураковский А.И.
Белорусский национальный технический университет

Объектом исследования является система управления учебным процессом – LMS (Learning Management System), разработанная студентами и магистрантами кафедры «Программное обеспечение вычислительной техники и автоматизированных систем» и широко используемая на факультете информационных технологий и робототехники. Данная автоматизированная система размещена в локальной сети БНТУ по адресу: http://Lms.fitr.bntu.by (или по IP-адресу: http://172.16.111.26).

Проект LMS состоит из нескольких модулей: управление проведением лабораторных работ, курсовым проектированием, дипломным проектированием, тестированием знаний студентов, управлением ошибками, сбором статистики, составлением учебно-методических материалов и некоторых других.

Основной целью разработки является создание единой «точки входа» к источнику данных, т.е. создание единого DAL (Data Access Layer). Отделение логики доступа к данным объекта значительно увеличивает гибкость любой системы, использующей хранилища данных. DAL помогает изолировать другие части приложения от подробностей сохраняемости объектов.

Для создания единого DAL была выбрана технология доступа к данным LinqToSql. Для работы с контекстом данных был выбран шаблон DAO (Data Access Objects – объекты, позволяющие работать с контекстом данным). Все DAO работают с контекстом данных по принципу IoC. Однако, какому провайдеру данных принадлежит этот контекст, DAO не знают. Этим достигается независимость от поставщика данных, будь то Linq2Sql, Entity Framework, XML, WCF, WebAPI или что-то иное. Контекст данных инициализируется в конструкторе DAO, но тут встает вопрос: каким образом создавать тот или иной контекст? Эту проблему решает паттерн AbstractFactory. В LMS используется двойной подход этого паттерна для создания контекстов. Сначала одна из фабрик определяет, для какого источника данных нужен контекст, далее