инженерия программного обеспечения

Нгуен Ван Доан,
Сафонов Владимир Олегович

УДК 004.432.4

ПРИМЕНЕНИЕ АСПЕКТНО-ОРИЕНТИРОВАННОГО ПОДХОДА И СИСТЕМЫ ASPECT.NET К РАЗРАБОТКЕ WEB-ПРИЛОЖЕНИЙ

Аннотация

Описаны методы применения системы Aspect.NET при разработке ASP.NET приложений и взаимодействие Aspect.NET с ASP.NET. Выделены задачи Web-программирования, которые требуют решения в виде реализации сквозной функциональности приложения: протоколирование, безопасность (authentication, authorization, impersonation), криптография строки запроса, криптография cookie-файлов, кодирование гипертекста, расширение пользовательского веб-интерфейса. Предложены методы применения аспектно-ориентированного программирования для задач разработки веб-приложений на платформе Microsoft.NET.

Ключевые слова: аспектно-ориентированное программирование, АОП, Aspect.NET, Веб-приложение, Microsoft.NET, ASP.NET.

ВВЕДЕНИЕ

Аспектно-ориентированное программирование (*АОП*) [1] – один из новых подходов к программированию, предназначенный для модуляризации сквозной функциональности (*cross-cutting concerns*) и ее автоматизированного, безопасного и надежного добавления в целевую программу, а также поиска (локализации) и модификации в целевой программе некоторой уже реализованной сквозной функциональности. Сквозная функциональность – это новая функциональность, реализация которой рассредоточена по коду приложения. Тем самым, АОП позволяет систематически добавлять и модифицировать новую функциональность, в том числе и относящуюся к Web-программированию. Подробное описание АОП дано в работах [1–3].

В настоящее время Web-программирование играет важную роль в сфере разработки программного обеспечения (ПО). В данной статье рассмотрено применение АОП в Web-программировании для платформы Microsoft.NET. Web-программирование для данной платформы реализуется с помощью *ASP.NET* [5], а АОП для платформы .NET реализовано в системе *Aspect.NET* [4], разработанной в лаборатории Java-технологий математико-механического факультета СПбГУ под руководством профессора В.О. Сафонова. Цель статьи – анализ методов применения АОП, реализованного в системе Aspect.NET, для разработки и модификации ASP.NET-приложений, а также выделение некоторых типичных задач Web-программирования,

к которым можно применить АОП: *протоколирование* (*logging*), *безопасность* (*authentication*, *authorization*, *impersonation*), *криптография строки запроса* (*Query String*), *криптография cookie-файлов*, *кодирование гипертекста* (*Html Encoding*), *расширение пользовательского Web-интерфейса*.

1. ОСОБЕННОСТИ ASP.NET

ASP.NET – технология создания Web-приложений и Web-сервисов, разработанная компанией Microsoft. Она является составной частью платформы Microsoft.NET и развитием более старой технологии Microsoft ASP. Хотя технология ASP.NET берёт своё название от Microsoft ASP, она значительно отличается от последней. Microsoft полностью перестроила ASP.NET, основываясь на общей системе поддержки выполнения программ – *Common Language Runtime* (*CLR*), которая является основой Microsoft.NET. Преимущество технологии ASP.NET – в более высокой степени абстракции, построенной над стандартным HTML-кодом: использование объектно-ориентированной парадигмы, поддержка нескольких языков программирования, динамическая компиляция страницы при первом запросе к странице, наличие универсальной библиотеки классов, содержащей значительное число готовых для использования в проектах решений – *Microsoft .NET Framework*.

Важная особенность ASP.NET – технология разделения кода. ASP.NET позволяет разработчику отделить код программной логики от кода представления, то есть поместить код программной логики страницы в файл .cs или .vb отдельно от кода собственно страницы, размещаемого в .aspx-файле. Эта технология называется *Code-Behind*. Таким образом, дизайн страницы может быть изменен без модификации кода страницы, что позволяет распределить работу по дизайну внешнего вида страницы и реализации ее функционирования между дизайнером и программистом.

В Microsoft Visual Studio.NET существуют два типа ASP.NET-проектов [6]: ASP.NET Web-сайт (*ASP.NET Web Site*) и ASP.NET Web-приложение (*ASP.NET Web Application*). Основное различие между ними в следующем. Для ASP.NET Web-приложения все файлы классов программной логики страниц (*code-behind classes*) и все другие отдельные классы для проекта компилируются в единую сборку, которая помещается в папку *bin*. При запросе страниц ASP.NET динамически компилируется только код представления страниц, классы которых являются подклассами классов программной логики. Для ASP.NET Web-сайта исходные файлы развертываются (*deployed*), и при запросе страниц ASP.NET динамически компилируется код программной логики и код представления страниц. При этом скомпилированная сборка помещается во временную директорию ASP.NET (*C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\Temporary ASP.NET Files* – в Windows XP). Более детальное сравнение рассмотренных типов ASP.NET – проектов дано в [6].

Microsoft Visual Studio.NET 2005 по умолчанию не включает в себя шаблон ASP.NET Web-приложения. Для работы с этим шаблоном необходимо установить Service Pack 1 [7]. В отличие от версии 2005, Microsoft Visual Studio.NET 2008 по умолчанию поддерживает оба этих типа ASP.NET проектов. Так как разработка ASP.NET Web-сайта связана с динамической компиляцией кода программной логики и кода представления страниц, в текущей версии Aspect.NET не поддерживает внедрение аспектов в проекты вида «ASP.NET Web-сайт». При работе с ASP.NET Web-приложениями Aspect.NET использует скомпилированную сборку кода программной логики страниц, находящуюся в папке *bin*, для выполнения своей функциональности – сканирования точек присоединения (*join points*) и внедрения аспектов (*weaving*). Таким образом, в текущей версии (2.1) Aspect.NET применим к ASP.NET Web-приложениям и к коду программной логики страниц.

Все аспекты Web-программирования разработаны в среде Microsoft Visual

Studio.NET 2008 с использованием новой версии Aspect.NET, которая будет выпущена в ближайшее время, и внедрены в ASP.NET Web-приложения.

2. ВЗАИМОДЕЙСТВИЕ ASPECT.NET С ASP.NET

Aspect.NET – это языково-независимая визуальная среда разработки АОП-приложений для Microsoft.NET, реализованная как расширение (*add-in*) Microsoft Visual Studio.NET 2005. С помощью Aspect.NET пользователь может определять, внедрять и оценивать результаты внедрения аспектов в своих проектах.

Наиболее разумным решением при описании аспектов была бы возможность использовать как метаязык, так и атрибутные спецификации, чтобы скомбинировать преимущества этих подходов. Преимуществом метаязыка является лаконичное описание сложных аспектов, а пользовательских атрибутов – совместимость с существующими инструментами для разработки на платформе .NET. Система Aspect.NET использует оба этих подхода. Определению аспекта на метаязыке Aspect.NET.ML соответствует специальный тип проекта, который затем может быть сконвертирован в класс аспекта с соответствующими атрибутными аннотациями или напрямую в сборку аспекта. Такая скомпилированная сборка аспекта затем используется для внедрения.

Ниже представлен более детальный сценарий взаимодействия Aspect.NET с ASP.NET:

1. Пользователь описывает аспекты на метаязыке (или на языке C# с использованием специализированных атрибутов АОП), загружает исходные коды целевого Web-приложения (в виде обычного «*ASP.NET Web Application*» проекта MS Visual Studio) в редактор и компилирует коды программной логики страниц (*code-behind*) в целевую сборку.

2. Редактор конвертирует описания аспектов (если аспекты описаны на метаязыке) в классы и пользовательские АОП-атрибуты, комбинируя их затем в общую аспектную сборку.

3. Для того чтобы получить полный список точек внедрения выбранного аспекта в пределах целевой сборки, редактор вызывает подсистему внедрения аспектов, которая снабжается отладочной информацией (*.pdb*-файл) для поддержания взаимно-однозначного соответствия между расположением инструкций целевой сборки и целевого исходного текста.

4. Подсистема внедрения аспектов формирует и возвращает обратно редактору XML-файл с информацией о координатах точек внедрения в целевой сборке и исходном тексте.

5. Редактор отображает найденные альтернативы применения аспектов в исходном коде пользователя и приглашает пользователя подтвердить или отфильтровать список точек для применения аспектов. Итоговый список передается подсистеме внедрения.

6. На основе спецификаций из данного списка аспектная и целевая сборки интегрируются в новую результирующую сборку, которая и будет представлять собой итоговый результат применения аспектов (рис. 1).

На рис. 1 представлена организация среды АОП-разработки Aspect.NET для ASP.NET приложения.

Процесс работы пользователя с Aspect.NET реализуется следующим образом. Создаются сборки с аспектами как результат компиляции классов аспектов с атрибутными аннотациями или компиляции специальных проектов (template projects) на метаязыке Aspect.NET. После загрузки пользовательского ASP.NET Web-приложения в Visual Studio в Aspect.NET Editor можно выбрать эти сборки с аспектами для внедрения в данный проект и запустить режим сканирования для получения списка доступных точек внедрения (рис. 2).

Щелчок мыши на какой-либо точке внедрения выделяет это место в исходном тексте. Отменить внедрение в заданную

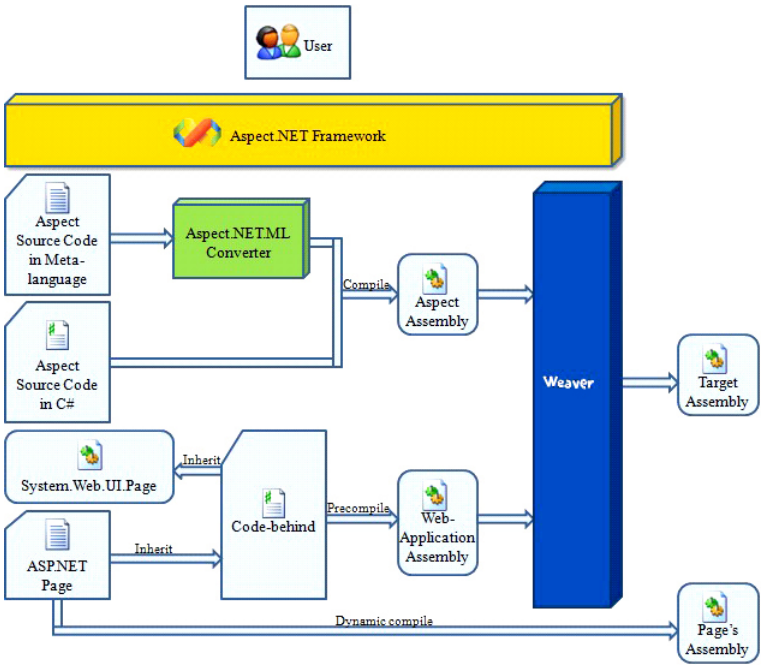

Рис. 1. Организация среды АОП-разработки Aspect.NET для ASP.NET приложения

точку внедрения можно, сняв пометку в этом дереве. После согласия пользователя на внедрение действий аспекта в предложенные места запускается собственно процесс внедрения, и пользователь получает результирующую сборку с вплетенными в нее аспектами.

3. НЕКОТОРЫЕ ЗАДАЧИ WEB-ПРОГРАММИРОВАНИЯ, ДОПУСКАЮЩИЕ РЕШЕНИЕ С ПОМОЩЬЮ АОП

При разработке ASP.NET приложений разработчики сталкиваются с задачами, решение которых может потребоваться во многих модулях кода приложения. Эти задачи для Web-приложений являются примерами сквозной функциональности. Приведем список некоторых типичных задач подобного рода, для которых были разработаны аспекты в системе Aspect.NET: протоколирование, безопасность, криптография строки запроса, криптография cookie-файлов, кодирование гипертекста и расширение пользовательского Web-интерфейса. Отмеченные нами задачи были выбраны, так как, во-первых, их решение необходимо в реальном процессе разработки Web-приложений, во-вторых, они требуют решения в виде реализации сквозной функциональности приложения (за счет возможного повторения их реализации в разных местах кода Web-приложения), а АОП – наиболее подходящая технология для разработки подобных реализаций. В дальнейшем рассмотрим также другие задачи, например: расширение событий мониторинга (*Custom Health Monitoring Events*), проверка безопасности (*Security Checking*), валидация XML-документов (*XML Validation*) и т. д.

Протоколирование (или ведение протокола) – хронологическая запись с различной (настраиваемой) степенью подробности сведений о происходящих в системе событиях обычно в некоторый файл. Эта задача является типичной задачей, которая может быть решена с помощью АОП.

Безопасность Web-приложений является важной задачей Web-программирования. Безопасность ASP.NET – приложений основана на трех операциях: а*утентификация*, *авторизация* и *олицетворение* (*impersonation*) – предоставление серверному процессу ASP.NET прав доступа клиента. ASP.NET предоставляет удобные механизмы для реализации этих операций [5, 8]. При разработке Web-приложений при необходимости осуществляются программно действия аутентификации и авторизации пользователей или действия олицетворения некоторого фрагмента кода в Web-приложениях с Windows-аутентификацией. Эти действия могут повторяться в различных модулях приложения.

Один из способов передачи информации между страницами в Web-приложениях – через *строки HTTP-запросов*

(*Query Strings*). Строка HTTP-запроса – часть унифицированного указателя ресурса (*URL*), который содержит данные для передачи в Web-приложения. Например, если URL-адрес запроса: http://vnn.vn/news.aspx?name=asp.net&author=abc, то значение строки запроса равно *name=asp.net&author=abc*. Во многих ситуациях информация URL-запроса, соответствующая переданным пользователем данным, не содержит проблем, связанных с тем, что пользователь может видеть или модифицировать ее. Но в других ситуациях строка запроса содержит информацию, которую пользователям запрещено визуализировать. Поэтому необходимо решить задачу криптографии строки запроса.

Один из способов хранения данных на стороне пользователя в Web-приложениях – с помощью *cookie-файлов*. *Cookie* – небольшой фрагмент данных, созданный Web-сервером и хранимый на компьютере пользователя в виде файла, который Web-клиент (обычно Web-браузер) каждый раз пересылает Web-серверу в HTTP-запросе при попытке открыть страницу соответствующего сайта. Поскольку cookie-файлы могут содержать конфиденциальную информацию (имя пользователя, условия доступа и т. д.), их содержимое не должно быть доступно другим, следовательно, необходимо решить задачу криптографии их содержимого.

В Web-приложениях часто необходимо представлять статическую информацию на страницах, то есть информацию, которую пользователи не могут изменить. К этим Web-приложениям относятся, главным образом, Web-приложения, публикующие новости, например http://gazeta.ru, http://www.news.com.au. Проблема, касающаяся этой информации, в следующем: она может содержать в себе символы, которые браузеры неправильно отображают, например пробел, меньше (<), больше (>), амперсанд (&) и кавычки ("). Поэтому необходимо кодировать их в *Html-эквиваленты*, например, символ "<" кодируется в "<". Таким образом, необходимо реализовать функциональность кодирования гипертекста при задании текста для элементов управления вида *Label*, *LinkButton*, *HyperLink*, *CheckBox*, *ListItem*, *RadioButton*.

При сопровождении Web-приложений разработчики сталкиваются с проблемой расширения пользовательского интерфейса разработанного Web-приложения, например, следующего содержания: добавить карту Google в Web-страницах; добавить

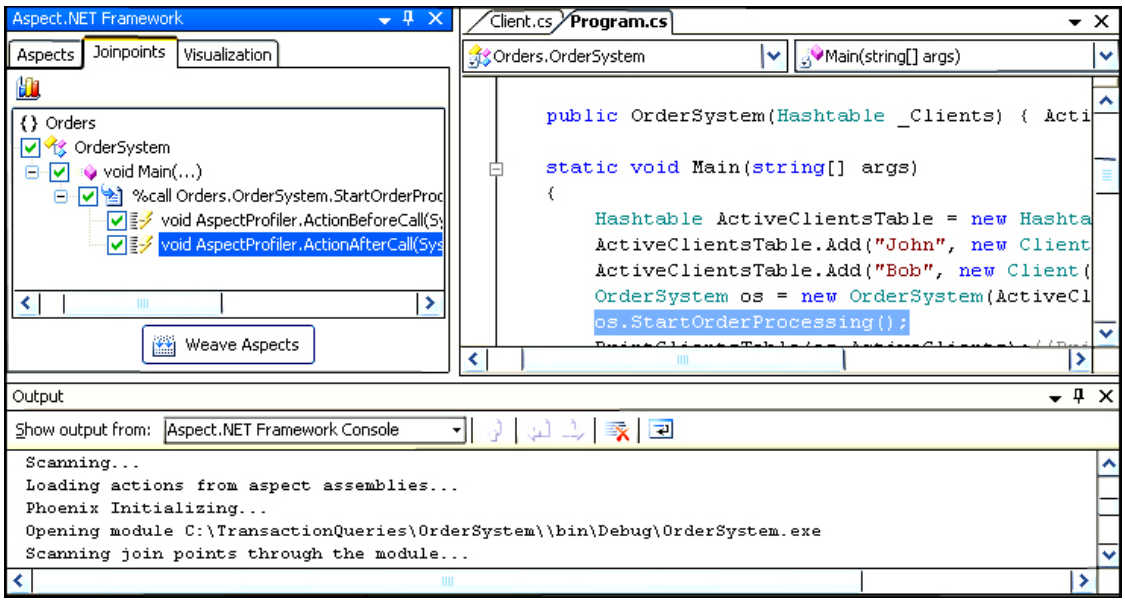

Рис. 2. Получение списка доступных точек внедрения

ссылку на некоторый сайт; добавить рекламные элементы; добавить элементы типа сообщения после некоторого действия. При этом разработчикам приходится изменять код приложения, например, если необходимо добавить элементы типа сообщения после действий какой-либо бизнес-логики, разработчики сначала разрабатывают элемент сообщения (например, в виде окна), затем после каждого действия (или в конце их выполнения) создают динамически этот элемент сообщения и добавляют его в нужный контейнер на странице.

4. СУЩЕСТВУЮЩИЕ РЕШЕНИЯ ЗАДАЧ WEB-ПРОГРАММИРОВАНИЯ

В настоящее время существует значительное число реализаций АОП [2] для платформы Microsoft.NET. Однако пока они не были использованы для решения задач Web-программирования. В данном разделе рассмотрим более традиционные, основанные на ООП, решения некоторых задач из вышеперечисленных задач Web-программирования, а затем покажем, как они могут быть решены с помощью АОП.

Рассмотрим задачу криптографии строки запроса. В [5] (раздел *Cryptography*) авторы демонстрируют интересный пример шифрования строки запроса. Создается класс *EncryptedQueryString*, который выполняет функциональность криптографии строки запроса. При шифровании, создается объект этого класса с помощью конструктора без аргументов, потом добавляются необходимые данные для строки запроса и, наконец, вызывается метод *ToString()* для шифрования введенных данных (листинг 1).

При расшифровке создается объект класса *EncryptedQueryString*, аргументом конструктора которого являются зашифрованные данные строки запроса:

```
EncryptedQueryString QueryString =
new EncryptedQueryString
(Request.QueryString["data"]);
```

и для получения расшифрованных данных вызывается код

```
myDataText = QueryString["MyData"].
```

Теперь рассмотрим, как реализуется безопасность в ASP.NET приложениях. Безопасность ASP.NET реализуется двумя подходами: реконфигурируемая безопасность (*configurable security*) и программируемая безопасность (*programmatic security*). Реконфигурируемая безопасность связана с доступом к URL страницы Web-сайта, то есть конфигурируется URL безопасность в файле *web.config* Web-приложения. При этом действия аутентификации, авторизации и олицетворения выполняются по странице. Более подробно о реконфигурируемой безопасности сказано в [5, 8]. Программируемая безопасность связана с проверкой права доступа в коде приложения. Возможны следующие опции для программной безопасности (рис. 3).

В зависимости от характеристики конкретного Web-приложения, используется либо реконфигурируемая безопасность, либо программируемая безопасность, либо оба этих подхода.

Другие задачи обычно реализуются в отдельном модуле или в наборе модулей; при необходимости разработчик вызывает эти модули в нужных ему точках выполнения кода.

Листинг 1

```
EncryptedQueryString QueryString = new EncryptedQueryString();
    QueryString.Add("MyData", MyData.Text);
    QueryString.Add("MyTime", DateTime.Now.ToLongTimeString());
    QueryString.Add("MyDate", DateTime.Now.ToLongDateString());
    Response.Redirect("QueryStringRecipient.aspx?data=" +
                                          QueryString.ToString());
```

5. АСПЕКТНО-ОРИЕНТИРОВАННЫЙ ПОДХОД К РЕШЕНИЮ ЗАДАЧ WEB-ПРОГРАММИРОВАНИЯ

Проведенный анализ показывает, что традиционные, основанные на ООП подходы могут решать задачи Web-программирования: решение каждой задачи обычно реализуются в отдельном модуле или в наборе модулей; при необходимости разработчик вызывает эти модули в нужных ему точках выполнения кода. Проблема возникает, когда нам необходимо применить наши модули во многих точках выполнения приложения, например при вызове метода шифрования и метода расшифровки строки запроса или при использовании программируемой безопасности многократно в разных точках выполнения приложения. При этом размер кода увеличивается за счет повторения и рассредоточения кода вызова (реализации) задачи (или функциональности). Возможна также ситуация модификации разработанного Web-приложения для добавления новой сквозной функциональности, такая, как рассмотренные нами задачи, например кодирование гипертекста (информация отображена на страницах неправильно, из-за того что данные в базе данных содержат специальные символы, например *пробел*, *меньше* и т. д.), криптография *cookie*-файлов или выдача сообщений в браузере при выполнении какой-либо бизнес-логики. При этом для реализации новой функциональности приходится изменять код приложения вручную, реализуя эту функциональность в модулях и добавляя код вызова этих модулей вручную в нужные точки выполнения приложения.

a. Явная проверка безопасности с использованием *IPrincipal*-интерфейса (объект *User* – объект класса, реализующего интерфейс *System.Security.Principal.IPrincipal*; является свойством объекта страницы *System.Web.UI.Page*):

Аутентификация:

```
if (User.Identity.IsAuthenticated) { ... }
```

Авторизация (проверка роля):

```
if (User.Identity.IsAuthenticated && User.IsInRole("Admin")){ ... }
```

b. Использование *PrincipalPermission* класса для проверки безопасности:

Аутентификация:

```
PrincipalPermission pp = new PrincipalPermission(null, null, true);
pp.Demand();
//equivalent with the check: if (User.Identity.IsAuthenticated)
//Do business action
```

Авторизация:

```
PrincipalPermission pp = new PrincipalPermission(null, "Admin", true);
pp.Demand();
//Do business action
```

c. Олицетворение кода с правами доступа аутентифицированного пользователя:

```
WindowsIdentity windowsID = User.Identity as WindowsIdentity;
if (windowsID != null)
{
   WindowsImpersonationContext wiContext =
   windowsID.Impersonate();

   //Do something here with permission of authenticated user

    wiContext.Undo();
  }
```

Рис. 3

Исходя из данных предпосылок, возникает идея применения АОП для решения указанных задач при разработке ASP.NET приложений. С помощью АОП каждая задача (или функциональность) реализуется в аспекте в виде набора действий (*action*s), затем определяются условия внедрения для присоединения этих действий к нужным нам точкам выполнения Web-приложения, после этого запускается *подсистема внедрения* (*weaver*) аспектов системы Aspect.NET. Действия аспекта будут автоматически добавляться подсистемой внедрения в точки присоединения (то есть в нужные нам точки выполнения), определенные условиями внедрения аспекта. При этом изменения целевого Web-приложения выполняются на уровне *MSIL* кода. Такой подход имеет следующие преимущества, по сравнению с реализацией задач без применения АОП:

- Использование синтаксиса определения условия внедрения аспекта [3] позволяет описать множество точек присоединения, в которые необходимо добавить функциональность наших задач.
- Действия аспекта будут автоматически добавляться подсистемой внедрения (weaver) в точки присоединения (то есть в нужные нам точки выполнения), определенные условиями внедрения аспекта. Таким образом, не требуется «ручных» вставок кода реализации функциональности в этих точках выполнения. Благодаря этому, уменьшаются объем кода, вероятность программных ошибок, время и стоимость разработки.
- Никаких изменений кода в целевом Web-приложении не требуется.
- Упрощается сопровождение и расширение Web-приложения. Новые требования (новая функциональность) реализуются в аспектах, затем внедряются в целевое Web-приложение. Изменение функциональности осуществляется также в коде реализации аспектов.
- Web-приложения полностью работоспособны без применения аспектов. При этом функциональность, реализованная в аспектах, отсутствует в целевых Web-приложениях.

Для практического подтверждения описанной идеи было принято решение разработать аспекты, поддерживающие рассмотренные выше задачи, с использованием системы Aspect.NET, преимущества которой по сравнению с другими инструментами реализации АОП для .NET описаны в [2].

Разработка ASP.NET-приложений с применением АОП не требует существенных изменений системы Aspect.NET, так как она применена к коду программной логики страниц (code-behind), который написан либо на Csharp, либо на Visual basic.

Изменение кода в аспекте действительно влияет на поведение результирующего Web-приложения, однако это изменение касается только той сквозной функциональности, для которой разрабатывался аспект. Добавление действий аспекта выполняется в точках присоединения, которые находятся вокруг вызовов целевых методов (перед ними – %before, после них – %after, вместо них – %instead), поэтому оно не влияет на архитектуру программы, то есть не изменяет ее. Добавление новой функциональности методом, отличным от АОП, тоже выполняется, отталкиваясь от целевых методов. Предположим, что реализация некоторой новой функциональности требует изменения архитектуры приложения. При этом добавление данной функциональности без применения АОП тоже изменит архитектуру программы.

На сайте Aspect.NET [4] опубликована библиотека Web-аспектов с примером Web-приложения.

ЗАКЛЮЧЕНИЕ

В статье описаны условия применения Aspect.NET при разработке ASP.NET приложений, и также взаимодействие Aspect.NET с ASP.NET. Выделены задачи Web-программирования, которые требуют решения в виде реализации сквозной функциональности приложения (за счет возможного повторения их реализации в разных местах кода Web-приложения): *протоколирование* (*logging*), *безопасность* (*authentication*, *authorization*, *impersonation*),

криптография строки запроса (*Query String*), *криптография cookie-файлов*, *кодирование гипертекста* (*Html Encoding*), *расширение пользовательского Web-интерфейса*. Предложены методы применения аспектно-ориентированного программирования для задач разработки Web-приложений на платформе *Microsoft.NET*. На основании рассмотренных и использованных идей выполняется дальнейшее исследование применения АОП и развитие библиотеки аспектов для разработки Web-приложений и Web-сервисов: разработка аспектов для решения других задач Web-программирования, применение АОП для разработки WindowCommunication Foundation-сервисов.

Литература

1. Web-сайт по аспектно-ориентированной разработке программ. [Электронный ресурс], режим доступа: http://aosd.net свободный, язык английский, последнее обращение 02/09/2010.

2. Safonov V.O. Using aspect-oriented programming for trustworthy software development. Hoboken, New Jersey: Wiley Interscience, John Wiley & Sons. 2008. 338p.

3. Сафонов В.О. Практическое руководство по системе аспектно-ориентированного программирования Aspect.NET // Компьютерные инструменты в образовании. 2008. № 2. С. 12–20.

4. Web-сайт проекта Aspect.NET. [Электронный ресурс], режим доступа: http://www.aspectdotnet.org свободный, язык английский, последнее обращение 02/09/2010.

5. Matthew MacDonald, Mario Szpuszta. Pro ASP.NET 3.5 in C# 2008. Apress. 2008. New York: Apress. 2008. 1498 p.

6. Introduction to Web Application Projects. [Электронный ресурс], режим доступа: http://msdn.microsoft.com/en-us/library/aa730880(VS.80).aspx свободный, язык английский, последнее обращение 02/09/2010.

7. Microsoft Visual Studio 2005 Team Suite Service Pack 1. [Электронный ресурс], режим доступа: http://www.microsoft.com/downloads/details.aspx?familyid=bb4a75ab-e2d4-4c96-b39d-37baf6b5b1dc&displaylang=en свободный, язык английский, последнее обращение 02/09/2010.

8. Building Secure ASP.NET Applications: Authentication, Authorization, and Secure Communication. [Электронный ресурс], режим доступа: http://msdn.microsoft.com/en-us/library/aa302388.aspx свободный, язык английский, последнее обращение 02/09/2010.

Abstract

Methods of application of the Aspect.NET system for ASP.NET application development and interaction of Aspect.NET system with ASP.NET are covered. Typical tasks of Web-programming that require solution in the form of cross-cutting concern implementation are selected, such as: logging, security (authentication, authorization, impersonation), query string cryptography, cookie cryptography, html encoding, user Web interface extension. Methods of application of aspect-oriented programming for Web application development tasks on Microsoft.NET platform are suggested.

Keywords: aspect-oriented programming, AOP, Aspect.NET, Web application, Microsoft.NET, ASP.NET.

Нгуен Ван Доан,
аспирант кафедры информатики
математико-механического
факультета СПбГУ,
ngvdoanbk@yahoo.co.uk,

Сафонов Владимир Олегович,
доктор технических наук,
профессор кафедры информатики
математико-механического
факультета СПбГУ,
vosafonov@gmail.com