УДК 004.4'22

СИСТЕМА ПОРОЖДЕНИЯ ДОКУМЕНТОВ В ФОРМАТАХ HTML, MHT И WordProcessingML

Д.В. Деев, Ю.С. Окуловский

Рассмотрена проблема автоматической генерации документов и приложений, производящих эти документы. Приведено общее описание языка разметки Thorn, разработанного специально для задач порождающего программирования. Изложено использование языка Thorn для упрощения набора HTML и MHT документов. Изложены основные проблемы создания документов в формате WordProcessingML и методы их решения с помощью Thorn. Приведена технология порождения по документу исходного кода программы, создающей этот документ.
Ключевые слова: порождающее программирование, шаблоны документов.

Введение

В работе рассматривается проблема порождения документов. Данная проблема актуальна для большинства приложений, из которых необходимо получать данные в печатном виде. Реализация печати в рамках приложений является трудоемкой и часто не оправдывает себя. Вместо этого предлагается экспорт документов в формате, поддерживаемом текстовыми процессорами, с последующим открытием в текстовом процессоре, возможной ручной правкой и печатью. Существующие решения в этой области часто разрабатываются производителями программного обеспечения самостоятельно либо интегрированы в состав программных пакетов (например, 1С). Необходима разработка кросс-платформенной системы, не привязанной к конкретной операционной системе (ОС) или программному пакету.

Предлагается решение, основывающееся на языке Thorn и системе Thornado. Thorn – это новый язык разметки, который воплощает в себе новый подход к преобразованию текстовых документов, комбинируя идеологии языков разметки и скриптовых языков. Документ Thorn состоит из команд, каждая команда ассоциирована с программой на скриптовом языке (например, Perl). Thorn осуществляет разбор документа, получает всю информацию из входного документа, которая касается данной команды, запускает программу на скриптовом языке и передает ей эту информацию. Выходное значение программы интерпретируется как результат выполнения команды и подставляется в документ вместо команды. Об-

мен информацией между командами осуществляется через глобальные переменные [1, 2]. Thornado – это набор инструментов кодогенерации на языке Thorn, одной из главных областей применения которой является генерация бизнес-приложений. В рамках этой системы разработаны кодогенераторы подсистем ввода и хранения данных, подсистемы вывода документов и подсистемы пользовательского меню [3].

Целью данной работы является разработка подсистемы Thornado, которая предназначена для порождения документов в форматах HTML, MHT и WordProcessingML, а также вставку в них переменных. Это решение основывается на введении разметочной информации с клавиатуры, а не на WYSIWYG-подходе. Это обусловлено следующими причинами.

– Внести необходимое форматирование руками в большой массив текста гораздо проще, чем исправлять его в WYSIWYG. Например, дана таблица в формате Comma separated value. Требуется оформить ее для буклета с различными нюансами оформления: подсветкой четных и нечетных колонок и/или рядов, с указанием границ определенной толщины и шрифта, отличающегося от стандартного. Такое оформление в WYSIWYG-редакторе весьма трудоемко, в то время как разметка с клавиатуры решает эту проблему значительно быстрее.
– При изменении кода вручную в текстовом виде гарантируется внесение только необходимой разметочной информации. Таким образом, обеспечивается кросс-платформенность решения. Базовая разметочная информация поддерживается всеми редакторами и браузерами. Расширенное же форматирование, вносимое WYSIWYG-редакторами, поддерживается не полностью.
– В WYSIWYG-редакторах отсутствуют полноценные способы введения в документы ссылок на переменные, которые позволяют превратить индивидуальный документ в шаблон семейства документов.

Генерация HTML и MHT документов

Рассмотрим преимущества и недостатки форматов HTML и MHT при использовании для разметки документов, порождаемых бизнес-приложениями. HTML является широко распространенным стандартом, его формат одинаково хорошо отображается в большинстве текстовых редакторов, кроме того, в нем есть большинство средств форматирования текста. С другой стороны, HTML не имеет некоторых средств форматирования, таких как колонтитулы, автоматическая нумерация страниц, в тело HTML документа нельзя встроить изображения и другие файлы.

Формат MHT – это развитие формата HTML (MIME HTML). Он создан для того, чтобы сохранять веб-страницы в едином файле, с изображениями, java-апплетами и флеш-анимацией, которые обычно вставляются с помощью внешних ссылок. В данном формате можно описывать колонтитулы документа и вставлять номера страниц. Главным недостатком MHT является то, что читать и редактировать документы в данном формате можно лишь в некоторых текстовых процессорах, которые доступны только в ОС Windows. В других ОС средства редактирования отсутствуют.

Рассмотрим применение языка Thorn для создания HTML документов. Документ на языке Thorn состоит из команд, вложенных одна в другую:

```
\table[border=1 align=center] {
\tr {
       \td {a11} \td {a12}
       }
       \tr {
         \td {a21} \td {a22}
         }}
```

Можно сравнить этот код с соответствующим HTML кодом, который порождается из вышеуказанного:

```
<table border=1 align=center>
<tr>
  <td>a11</td> <td>a12</td>
</tr>
<tr>
  <td>a21</td> <td>a22</td>
</tr>
</table>
```

Видно, что код на языке Thorn гораздо компактнее. Он может быть приведен к более компактному виду. Можно опустить названия параметров, зафиксировав их порядок. Кроме того, поскольку точно известно, что `tr` всегда внутри `table`, а `td` внутри `tr`, то фигурные скобки можно опустить. В результате тот же код будет записан как:

```
\table[1 border]
\tr \td a11 \td a12
\tr \td a21 \td a22
```

Порядок имен параметров, правила наследования и логика команд определены в файлах команд. Библиотека таких файлов должна быть загружена перед началом компиляции. Ниже приведен пример команды `\table`:

```
#Keys=border,align;Blocks=entry;
#Parents=body;Type=Perl;Free=yes;
$STRING="<table
border=$PARAM{border}
align=$PARAM{align}>
$TEXT{entry}
</table>";
```

В первых строчках файла команды определяются ее общие свойства: порядок следования параметров (предложение Keys), список команд, которые могут содержать данную (Type), факт, что для данной команды могут быть опущены фигурные скобки (Free), язык, на котором запрограммирована логика команды (Type). Далее следует программа на выбранном языке программирования, которая использует специальные переменные: PARAM (параметры команды), TEXT (текстовые блоки) и STRING (возвращаемое программой значение). Программа является произвольным кодом на выбранном языке: она может открывать файлы, подключать модули и т.д. За счет этого достигается высокая гибкость языка Thorn, поскольку команда может содержать в себе неограниченную функциональность. Именно за счет этой особенности языка Thorn были реализованы возможности, описанные ниже. В дальнейшем мы будем избегать технических деталей реализации тех или иных возможностей, приводя вместо этого общее описание. Технические детали обычно весьма объемны и связаны более с программированием на языке Perl, чем с кодогенерацией.

Созданная библиотека позволяет упростить следующие моменты создания HTML документов.

1. Парные теги закрываются автоматически: в каждой команде указывается, какие команды могут содержать данную команду (parents), и компилятор Thorn самостоятельно закрывает парные теги.
2. Атрибуты тегов можно опустить и указать только их значения в определенном порядке (`\table[1 center]`). В описании команды, порождающей тег `table`, заранее указывается, что на первой позиции ряда значений пишется ширина границы, а на второй – значение атрибута `align` (выравнивание текста).
3. Для совместимости с XHTML использование некоторых атрибутов тегов не рекомендуется, поэтому можно передать значения в стиле, указав восклицательный знак перед названием атрибута команды. `\table[!border-color=red]`

Приведенные средства позволяют создавать удобные и компактные Thorn-файлы документов. Также была разработана команда `\page`, которая содержит в себе стили, соответствующие государственному стандарту, в котором описаны требования к оформлению документов [4].

Для порождения документов с колонтитулами в формате MHT разработана команда `\mhtpage`, которая позволяет вставить данные и переменные в колонтитулы. Примерный вид команды такой: `\mhtpage{верхний колонтитул} {содержание}{нижний колонтитул}`.

Язык Thorn позволяет создавать команды, которые порождают тег с заданными атрибутами и/или стилями. Например, код разрыва страницы в HTML: `<br style="page-break-after:always;" />` следует заменить командой `\pbr` (сокращенно от Page Break). Аналогичным способом заменяются и другие устойчивые последовательности HTML.

WordProcessingML

Формат WordProcessingML хорошо поддерживается большинством визуальных текстовых редакторов, в том числе кроссплатформенных, а также имеет широкие возможности для форматирования не только текста, но и его представления пользователю. Кроме того, текстовым редакторам для правильного разбора документа не обязательно буквальное следование документа спецификации [5].

С другой стороны, у данного формата есть существенный недостаток. Объем описаний форматирования избыточен и сложен. Логика форматирования в документах WordProcessingML гораздо больше приближена к логике текстового редактора, чем к более понятной и ожидаемой логике HTML разметки. Приведем лишь несколько примеров.

- Если в потоке текста изменить шрифт нескольких слов на полужирный, то в XML коде мы увидим, что поток текста прерывается в том месте, где изменяются свойства шрифта: происходит закрытие блока с «обычным текстом», далее начинается блок с полужирным текстом, после чего этот блок закрывается и снова начинается блок с обычным текстом. В HTML, TeX и других языках разметки мы привыкли к другому подходу: блок полужирного текста вложен в блок обычного текста.
- Особое внимание в WordProcessingML следует уделить созданию списков. В данном формате нет понятия List item, как в HTML. Пунктом списка является абзац, к которому применены заранее определенные стили списков. Эти стили определяют уровень пункта списка, его оформление и формат нумерации.

– Для корректного отображения ячеек таблиц необходимо указание столбца в атрибутах каждой из них.
– Еще одним недостатком, который снижает доступность WordProcessingML, является использование непривычных единиц измерения. Так, например, все отступы указываются в твипах, а шрифты в полупунктах.

Таким образом, создание WordProcessingML-документов путем ввода с клавиатуры неоправданно трудоемко. Создание их в WYSIWYG-редакторе нежелательно по причинам, указанным во введении. Нами разработана библиотека команд языка Thorn, решающая указанные проблемы следующим образом.

– Разрывы текста в местах, где применяется форматирование, оформляются инверсным способом. Команда на языке Thorn создает следующую последовательность тегов: 1) закрывает текущие теги на месте прерывания; 2) открывает теги, необходимые для форматирования текста, и перечисляет стили форматирования; 3) вставляет форматируемый текст; 4) закрывает теги, открытые в пункте (2); 5) заново открывает теги, закрытые в пункте (1). Например, команда, обозначающая полужирный шрифт, будет выглядеть как:
```
$STRING = '</w:t></w:r>'.'<w:r><w:rPr><w:b/></w:rPr><w:t>'.
$TEXT{entry}. '</w:t></w:r>'.'<w:r><w:t>'.
```
– Многоуровневые списки вкладываются один в другой, как в HTML или TeX. Каждая команда `\ul` сохраняет в глобальных переменных номер уровня, а каждая команда `\li` генерирует абзац с соответствующим этому уровню стилем.
– Также через глобальные переменные решается проблема нумерации столбцов: каждая команда новой ячейки `\td` увеличивает счетчик в глобальных переменных, а каждая команда новой строки `\tr` обнуляет его.
– Все коэффициенты в пунктах, дюймах и других единицах измерения пересчитываются в твипы в скриптовом языке.

На данный момент далеко не все команды WordProcessingML поддержаны в нашей системе. Однако созданная библиотека позволяет задавать базовое форматирование: списки, таблицы, стили оформления абзацев, колонтитулы, параметры страницы и т.д. Thorn является открытым программным продуктом, поэтому возможна доработка этой библиотеки пользователями под конкретный проект. Также будут предприняты дальнейшие усилия по расширению функциональности библиотеки.

Вставка переменных в документ

В 1 и 2 разделах описано создание документов. Однако для использования в бизнес-приложении необходимо создание шаблона документа, в который на указанные места можно подставить переменные. Данный шаблон реализуется в каком-либо языке программирования (был выбран язык C#). Переменные заполняются значениями из базы данных, в результате чего система производит конкретные экземпляры документов. Разработана библиотека языка Thorn, которая позволяет из любого текстового документа создать его шаблон. Для этого в тело документа вводятся дополнительные команды, как показано в следующем листинге.

```
\document{
\html \body \p
   Количество: \variable[name=Var type=Int default=5]}
```

Команды `\document` и `\variable` определены в двух библиотеках: lib.programmer и lib.makerup. В последней `\document` ничего не делает, кроме того, что возвращает свое значение. Команда `\variable` возвращает ее значение по умолчанию. Таким образом, верстальщик видит примерный документ, который будет сгенерирован бизнес-приложением.

В библиотеке lib.programmer эти команды определены по-другому. Команда `\variable` превращена в маркер, который разделяет текст на переменную и постоянную части. Команда `\document` собирает части и создает методы, которые принимают все переменные как аргументы и хранит получившийся документ в потоке. Существует две версии библиотеки lib.programmer: одна порождает PHP код для использования на веб-сайтах, и другая – C# код для локальных приложений.

Разработаны варианты команды `\variable`, предназначенные для вставки конкретных типов данных. Так, например, с помощью команды `\fio` вставляется фамилия, имя и отчество. В зависимости от параметров эта команда вставляет строку вида «Иванов Иван Иванович» в различных сокращениях («Иванов И.И.», «Иван Иванов» и т.д.) и различных падежах. Команда `\date` тоже параметризуется: дата 11.04.2005 может быть отображена как «11 апреля 2005 года», «11.04.2005 г.» и т.д. В lib.makerup значения по умолчанию для этих команд выбираются случайным образом из списка. В lib.programmer вставляются специальные типы аргументов-параметров. Эти типы выполняют соответствующие преобразования данных из БД. Кроме того мы можем вставлять в документ арифметические формулы.

Приведенные выше построения позволяют создавать шаблоны для произвольных документов. Разработано интеграционное решение, упрощающее создание документов с переменными из баз данных, разработанных при помощи Thornado. Описание базы данных на Thornado имеет следующий вид:

```
\field[string Fio]
  \desc Фамилия Имя Отчество
\field[int Age]
  \desc Возраст
\field[string Email]
  \desc Электронная почта
…
```

На основании этих данных могут быть порождены база данных, средства сохранения данных в файлы, пользовательский интерфейс и т.д. [3]. Была решена задача прямой связи переменных внутри документа с полями базы данных. В документ вставляется команда вида `\link`, которая открывает файл с описанием Thornado и извлекает из этого файла информацию, необходимую для помещения переменной в документ, т.е. команда использует тип переменной и ее комментарий. Таким образом, устраняется дублирование информации описания данных, которая была однажды введена: комментариев, типа полей и т.д. Кроме того, существенно упрощается передача данных из базы данных в методы, создающие документы: вместо ручного перечисления полей объекта в качестве аргументов метода передается объект в метод целиком, и далее он самостоятельно выбирает нужные ему поля. Фактически, порождение документа сводится к одной строчке программного кода вида PrintDocument(myObject), где myObject – объект в базе данных. Программисту требуется лишь определить, когда вызвать этот код; все остальное генерируется системой автоматически.

Заключение

В работе описана подсистема Thornado, предназначенная для порождения документов в форматах HTML, MHT и WordProcessingML и вставки в них переменных. Данная подсистема была разработана и объединена с общей системой кодогенерации Thornado. С использованием вышеуказанной подсистемы были разработаны и внедрены программные решения в области автоматизированного документооборота.

Литература

1. Окуловский Ю.С. Язык ТН / Ю.С. Окуловский // Проблемы теорет. и прикл. математики: молодежн. конф. – Екатеринбург: УрО РАН,2006. – С. 496–500.
2. Окуловский Ю.С. Спецификация языка Thorn [Электронный ресурс]. – Режим доступа: http://ai.math.usu.ru/wiki/doku.php/нир/продукты/thorn, свободный. – Яз. рус. (дата обращения 18.04.2010).
3. Деев Д.В. Система кодогенерации Thornado и ее использование для создания бизнес-приложений /Деев Д.В., Окуловский Ю.С., Попов В.Ю., Часовских В.П. // Научно-технический вестник СПбГУ ИТМО. – 2008. – № 57. – С. 80–87.
4. ГОСТ Р 6.30–2003. Унифицированная система организационно-распорядительной документации. Требования к оформлению документов. – Введ. 03.03.2003. – М.: Изд-во стандартов, 2003. – 17с.
5. Ecma-international. Standard ECMA-376 [Электронный ресурс]. – Режим доступа: http://www.ecma-international.org/publications/standards/Ecma-376.htm, свободный. – Яз. англ. (дата обращения 11.05.2010).

Деев Дмитрий Васильевич – Уральский государственный лесотехнический университет, ассистент, deyeff@gmail.com

Окуловский Юрий Сергеевич – Уральский государственный университет им. А. М. Горького, кандидат физ.-мат. наук, зав. лабораторией, yuri.okulovsky@gmail.com