Краткие сообщения

УДК 004.043 **DOI: 10.14529/ctcr160413**

ПРИМЕНЕНИЕ ОБЪЕКТНОГО ОТОБРАЖЕНИЯ XML В ОБРАБОТКЕ ДАННЫХ С ПОВЫШЕННЫМИ ТРЕБОВАНИЯМИ К ЦЕЛОСТНОСТИ

А.В. Родионов, В.Е. Драч

Московский государственный технический университет им. Н.Э. Баумана, Калужский филиал, г. Калуга

Рассмотрена роль схемы XML-документа и ее отображения на структуры данных программы в приложениях с повышенными требованиями к достоверности данных. Актуальность темы обусловлена широким применением XML как формата представления данных в задачах обмена электронными документами и распространенность XML Schema в задачах обеспечения форматно-логического контроля. Представлен оригинальный пакет для языка программирования Python, предназначенный для отображения схем документов XML на иерархию классов в приложении. Применение пакета в разработке программного обеспечения позволит существенно сократить повторяющийся код, связанный с сериализацией и десериализацией данных в XML-контейнерах, и, тем самым, повысить надежность и поддерживаемость кода в приложениях электронного документооборота. Результаты работы нашли практическое применение в разработках специализированного ПО для юридически значимого документооборота.

Ключевые слова: XML, XML Schema, Python, объектно-ориентированный подход, объектное отображение документов.

Введение

Современные задачи обработки данных, такие как сдача налоговой отчетности в электронном виде, обмен юридически значимыми документами и др., имеют повышенные требования к достоверности информации и надежности алгоритмов ее обработки. Одна из важнейших процедур при проведении входного и выходного форматно-логического контроля обрабатываемых данных – проверка типов данных и верности структур, которые ими образованы. Применение XML как формата обмена данными и XML Schema для обеспечения контроля их целостности, зарекомендовало себя как проверенное и широко распространенное решение [1]. К преимуществам XML относятся высокая степень стандартизации, разнообразие программных компонентов для обработки, человеко-читаемый формат данных. Возможность создать и опубликовать схему XML-документа позволяет контролирующим органам формализовать обмен данными на уровне нормативных документов. Например, Федеральная налоговая служба России регламентирует оборот электронной налоговой и бухгалтерской отчетностью на веб-ресурсе [2], где можно ознакомиться с различными версиями схем документов для различных видов обмена данными. Таким образом, эффективная работа со схемой документа актуальна при разработке надежных систем обмена и обработки данных.

Взаимосвязь схемы документа и ее отображения на объектную модель данных

Исторически одной из самых первых идеологий работы с XML-документами является DOM (англ. Document object model) [3], в которой данные представляются в виде дерева из узлов XML различных типов, в частности текстовых блоков, элементов с атрибутами, комментариев и инструкций обработки. Этот подход предполагает последующее преобразование узлов дерева в экземпляры конкретных структур данных приложения и обратную операцию при сериализации этих структур в XML. Введение схемы документа в этот алгоритм предполагает проверку схемы

при первоначальном построении дерева DOM, т. е. входной контроль целостности данных, однако, в ответственных приложениях необходим и выходной контроль при сериализации данных перед их передачей. Так как схема документа имеет конкретную структуру, специфическую для приложения, можно рассматривать внутренние структуры данных и схему документа, им соответствующую, во взаимосвязи. Для минимизации повторяющегося кода целесообразным представляется отталкиваться от схемы документа как при реализации считывания XML во внутренние объекты программы, так и при разработке кода для обратной операции. Для разработчиков на языке Python [4] авторами предлагается пакет для отображения объектной модели документов на иерархию классов Python, получивший название XML_ORM [5]. Пакет позволяет структурно проектировать схемы XML-документов, аналогично моделям таблиц БД фреймворка Django [6]. Объектно-ориентированный подход позволяет повторно использовать элементы схем путем наследования и переопределения полей XML-элементов. При задании класса-схемы автоматически генерируется код для загрузки и сохранения XML-документов, удовлетворяющих схеме. Также в составе пакета имеется класс-примесь, помещающий XML-контейнер внутрь архива Zip. Установка пакета производится из центрального репозитория пакетов для Python [7], с помощью менеджера пакетов Pip или из исходного кода, путем запуска в командной строке команды:

```
python setup.py install
```

Рекомендуется предварительно создать и активировать виртуальное окружение с помощью пакета `virtualenv` или `venv`, если используется Python версии 3. Основным модулем пакета является модуль `xml_orm.core`, в котором объявлены классы для создания классов-отображений схем XML и для задания полей данных в документах.

Проектирование схем документов с помощью XML_ORM

Класс `xml_orm.core.Schema` аналогично классу `django.db.models.Model` [8] задает объектно-ориентированную модель XML-узла. Схема должна наследовать непосредственно класс `Schema` или один из его потомков. Множественное наследование схем на данный момент не разрешено, но число предков схемы, не относящихся к XML_ORM, а также глубина дерева наследования, не ограничены. Следующий пример определяет схему XML-узла с двумя простыми субэлементами `ИД` и `ИмяФайла`, которые будут транслированы в поля `uid` и `name` соответственно:

```
from xml_orm.core import Schema
from xml_orm.fields import *

class Document(Schema):
    uid = SimpleField(u'ИД')
    name = SimpleField(u'ИмяФайла')

    class Meta:
        root = u'Документ'
        namespace = u'http://www.example.com/ns2'
```

Простые поля задаются с помощью классов `SimpleField`, `BooleanField` и др. Для них характерно отсутствие дочерних элементов и атрибутов. С такими полями связываются атомарные значения или списки атомарных значений. Параметры `minOccurs` и `maxOccurs` определяют, какой вид поле примет в составе объекта. Если поле имеет `maxOccurs`=1, то оно транслируется в обычное значение, причем, если задать `minOccurs`=0, то оно будет необязательным к заполнению. Поле с `maxOccurs` отличным от 1 (это значение не может быть равным 0), транслируется в список значений. При `minOccurs`=0 такой список по умолчанию инициализируется в пустое значение.

Класс `Meta` в составе схемы задает неизменные для всех подобных узлов параметры. В частности `root` задает имя элемента, а `namespace` – идентификатор пространства имен.

Простое поле может соответствовать как простому субэлементу XML-узла, так и его атрибуту. Его вид в XML-документе определяется наличием или отсутствием префикса `@` в его имени. В следующем примере идентификатор документа из субэлемента преобразован в атрибут:

```
class NewDocument(Document):
    uid = SimpleField(u'@ИД')
```

Схема NewDocument получена из исходной наследованием. При наследовании все поля схемы предка, имена которых совпадают с именами полей потомка, перекрываются. Новые поля добавляются в конец последовательности полей предка. Изменить позицию полей потомка в последовательности полей предка можно с помощью параметров insert_before и insert_after. В следующем примере поле uid перемещается в конец документа, а новое поле author добавляется в его начало.

```
class AnotherDocument(Document):
    uid = SimpleField(u'ИД', insert_after='name')
    author = SimpleField(u'Автор', insert_before='name')
```

Мета-информация при наследовании копируется и, при совпадении имен, заменяется:

```
class Article(Document):
    class Meta:
        root = u'Статья'
        encoding = 'windows-1251'
```

Новые экземпляры объектов схемы создаются, как обычные объекты Python, вызовом конструктора. Конструктору можно передавать именованные параметры, которые будут записаны в одноименные поля объекта. Неинициализированные таким образом поля можно присвоить позже, как обычные поля объекта. В следующем примере создается экземпляр класса Article.

```
art = Article(uid=1, author=u'Лев Толстой', name=u'Война и мир')
```

Преобразование в XML

Экземпляр схемы может быть преобразован в объект класса etree.Element [9] с помощью встроенного метода xml(). Преобразование будет выполнено, если все поля объекта, обязательные к заполнению, инициализированы, в противном случае будет поднято исключение.

```
tree = art.xml()
```

Также экземпляр схемы может быть преобразован в байтовую строку или в строку unicode. Для преобразования в строку можно использовать стандартные функции str() и repr(). К сформированному XML-документу добавляется декларация <?xml ?> с кодировкой, которая берется из мета-параметра encoding. Его значение по умолчанию равно utf-8. При отсутствии нужных полей, преобразование в строку поднимает исключение SerializationError, поэтому желательно заключать его в блоки try-except или with, например:

```
from xml_orm.core import SerializationError
try:
    print str(art)
except SerializatonError as err:
    print err
```

Также экземпляр может быть преобразован в строку unicode с помощью стандартной функции unicode(). При этом декларация <?xml ?> к строке не добавляется.

Загрузка из файла

Новый экземпляр класса Schema можно создать путем загрузки из строки, файла или файлоподобного объекта, содержащих данные XML. Для этого служит универсальный метод класса load(), принимающий в качестве параметра строку или произвольный объект, обладающий методом read(). В следующем примере создаются три новых объекта Article из трех разных объектов. Предполагается, что на диске присутствует файл data.xml.

```
fn = 'data.xml'
fp = open(fn)
st = open(fn).read()

art1 = Article.load(fn)
art2 = Article.load(fp)
art3 = Article.load(st)
```

Загрузка из файла или строки может поднять исключение ValidationError, если загружаемый XML не соответствует схеме.

Сохранение в файл

Потомки класса `Schema` могут переопределять метод `save()`, который по умолчанию ничего не делает. Класс реализует протокол менеджера контекстов, что позволяет использовать оператор `with`. При выходе из контекста автоматически вызывается метод `save()`. В следующем фрагменте, класс потомок `Article` добавляет простейшую функциональность сохранения данных в файл.

```
class SavedArticle(Article):
    def save(self):
        fn = geattr(self, 'xmlfile', None)
        if fn:
            open(fn, 'w').write(str(self))

with SavedArticle() as art:
    art.uid = 1
    art.author = u'Лев Толстой'
    art.name = u'Война и мир'
    art.xmlfile = 'article.xml'
    # при выходе из блока файл 'article.xml' будет сохранен
```

Выводы

Применение объектного отображения документов, предлагаемого пакетом `xml_orm`, позволяет отказаться от большого количества повторяющегося кода, связанного с загрузкой, сохранением и проверкой целостности типизированных документов. Это положительно сказывается на надежности и поддерживаемости кода, что особо важно в ответственных приложениях, зависящих от периодически изменяющейся нормативной базы. Переход на `xml_orm` позволяет разработчикам на Python, использующим XML, сосредоточиться на обрабатываемых данных и их структуре, что положительно сказывается на времени выхода приложений в релиз. Пакет успешно применяется при разработке приложений, обрабатывающих XML, как в open-source сообществе, так и в коммерческих приложениях. В частности, пакет играет ключевую роль в реализации шлюза межоператорского документооборота [9] в ЗАО Калуга Астрал. Дальнейшее развитие пакета `xml_orm` авторы видят в более тесной интеграции с различными СУБД, с целью обеспечения прозрачного цикла приема, контроля и сохранения документов в базе.

Литература

1. Extensible Markup Language (XML). Официальный сайт консорциума w3. – http://www.w3.org/XML/

2. Справочник налоговой и бухгалтерской отчетности. ФНС России. – http://format.nalog.ru/

3. Document Object Model (DOM). – http://www.w3.org/DOM/

4. Python software foundation. Официальный сайт. – https://www.python.org/

5. Официальный сайт проекта Django. – https://www.djangoproject.com/

6. Репозиторий пакетов для Python. Пакет xml_orm. – https://pypi.python.org/pypi/xml_orm

7. Django ORM. Официальная документация. – https://docs.djangoproject.com/en/dev/topics/db/models/

8. Пакет Etree. Страница документации. – https://docs.python.org/2/library/xml.etree.elementtree.html

9. Технология обмена юридически значимыми электронными документами между операторами электронного документооборота. НП РОСЭУ. Роуминг. – http://www.roseu.org/roaming/

Родионов Андрей Викторович, канд. техн. наук, доцент, доцент кафедры конструирования и производства электронной аппаратуры (ЭИУ1-КФ), Московский государственный технический университет им. Н.Э. Баумана, Калужский филиал, г. Калуга; andviro@gmail.com.

Драч Владимир Евгеньевич, канд. техн. наук, доцент кафедры компьютерных систем и сетей (ЭИУ2-КФ), Московский государственный технический университет им. Н.Э. Баумана, Калужский филиал, г. Калуга; drach@bmstu-kaluga.ru.

Поступила в редакцию 15 июня 2016 г.

DOI: 10.14529/ctcr160413

USE OF OBJECT TO XML MAPPING IN PROCESSING OF DATA WITH HIGH INTEGRITY REQUIREMENTS

***A.V. Rodionov**, andviro@gmail.com,*
***V.E. Drach**, drach@bmstu-kaluga.ru*
Bauman Moscow State Technical University, Kaluga Branch, Kaluga, Russian Federation

The role of XML document Schema and its mapping to program data structures is discussed, in relation to applications with high data reliability requirements. The actuality of work is defined by the wide application of XML as a data representation format in the tasks of electronic document exchange and popularity of XML Schema as a document integrity control tool. Original package for Python programming language is introduced, suitable for defining XML document Schemas as class hierarchies, which will allow to use the package in software development will allow to substantially reduce the code repetition in serialization and deserialization of XML containers. The results of work are applied to development of specialized software for document exchange.

Keywords: XML, XML Schema, Python, Object-oriented design, Object-document mapping.

References

1. Extensible Markup Language (XML). Official site. Available at: http://www.w3.org/XML/

2. Spravochnik nalogovoy i bukhgalterskoy otchetnosti FNS Rossii [Reference Book of Tax and Accounting Records. FTS of Russia]. Available at: http://format.nalog.ru/

3. Document Object Model (DOM). Available at: http://www.w3.org/DOM/

4. Python software foundation. Available at: https://www.python.org/

5. Official site of Django Project. Available at: https://www.djangoproject.com/

6. Repository of packages for Python. Package xml_orm. Available at: https://pypi.python.org/pypi/xml_orm

7. Django ORM. Official documentation. Available at: https://docs.djangoproject.com/en/dev/topics/db/models/

8. Package Etree. Page of documentation. Available at: https://docs.python.org/2/library/xml.etree.elementtree.html

9. Tekhnologiya obmena yuridicheski znachimymi elektronnymi dokymentami mezhdu operatorami elektronnogo dokumentooborota [Technology of Exchange of Legally Significant Electronic Documents between Operators of Electronic Document Flow]. Available at: http://www.roseu.org/roaming/

Received 15 June 2016

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Родионов, А.В. Применение объектного отображения XML в обработке данных с повышенными требованиями к целостности / А.В. Родионов, В.Е. Драч // Вестник ЮУрГУ. Серия «Компьютерные технологии, управление, радиоэлектроника». – 2016. – Т. 16, № 4. – С. 122–126. DOI: 10.14529/ctcr160413

FOR CITATION

Rodionov A.V., Drach V.E. Use of Object to XML Mapping in Processing of Data with High Integrity Requirements. *Bulletin of the South Ural State University. Ser. Computer Technologies, Automatic Control, Radio Electronics*, 2016, vol. 16, no. 4, pp. 122–126. (in Russ.) DOI: 10.14529/ctcr160413