Софийски университет „Св. Кл. Охридски”

Факултет по математика и информатика

Бакалавърска програма
„Софтуерно инженерство”

Предмет: XML технологии за семантичен Уеб

Зимен семестър, 2021/2022 год.

Тема №59: „. XML to/from relational DB applications“

Курсов проект

Автори:

Боян Богданов, фак. номер 62383

Христо Белев, фак. номер 62408

януари, 2022 г.

София

Съдържание

1 Въведение

Темата ни засяга проблема с преобразуването на данни (основополагащ в програмирането) и по-конкретно: превръщане от записи в база данни към XML файл и обратно. Това е важно, тъй като базите данни (били те релационни или не, в случая това не е от значение) са фактически единствения избор за продължително съхранение на дигитална информация, а XML е силен инструмент, за представяне на данни с определена структура, така че да могат да се манипулират и преизползват лесно както и от хора, напр. програмисти, които да правят промени по структурата на даден XML файл, така и от компютри, напр. когато изпълняват скриптове върху тях.

И XML и базите данни са безспорни стандарти в програмирането и от това следва, че възможността да преобразуваме данни от едната структура в друга може да е полезно за множество цели- напр. извличаме данните от базата, превръщаме ги в HTML/XHTML и ги рендерираме в уебсайт, без да трябва да създаваме персонализирана логика за тяхното преобразуване- нещо, за което във всеки Уеб проект се отделя значително време.

Предлагаме решение разчитащо на езика Java, заради неговата гъвкавост и мощност в решаването на всякакви проблеми, Java библиотеката JDOM, която представлява алтернатива на DOM на W3C с предимството, че се възползва максимално от силните страни на Джава, за да постигне същото, което бихме постигнали с DOM, но по-елегантно. JDOM използва външни XML парсъри и се интегрира лесно с DOM и SAX, и MSSQL, тъй като е доказано една от най-добрите системи за управление на бази данни и се интегрира лесно с Java.

Останалата част от този документ илюстрира по-подробно целите и начините, по които решаваме проблема.

2 Анализ на решението

2.1 Работен процес

В изходно положение, приемаме, че имаме работеща база данни с 2 таблици с попълнени данни в нея: vendors и products, които имат релация 1:many помежду си: 1 продавач (vendor) притежава много продукти (products). При стартиране на приложението в първи режим на работа, извличаме данните от базата, обработваме ги и резултата запазваме в XML файл, запазващ оригиналната им йерархия. При стартиране на втори режим на работа, приемаме, че имаме валиден XML файл, с аналогична структура на файла получен при режим 1- извличаме данните от файла и ги запазваме обратно в базата.

2.2 Структура на съдържанието

Съдържанието бива структурирано по два начина според това кое от двете възможни превръщания използваме:
1. При превръщане DB->XML започваме със съдържание под формата на записи в база данни.

Записите спадат към 1 от 2-те гореспоменати таблици (products & vendors) (фиг.1 и 2). Връзката между двете таблици се осъществява чрез колоната vendor_id (фиг.2): всеки продукт пази информация относно обектът си родител- продавач.

	id	name	phone
1	1	Ivan Ivanov	081258155
2	2	Georgi Ivanov	086235268155

Фиг. 1: Данни от таблицата `vendors`

	id	name	price	vendor_id
1	1	TV	155	1
2	2	PC	1550	1
3	3	Tomatoes	5	2

Фиг. 2: Данни от таблицата `products`

2. При превръщане XML->DB започваме със съдържание под формата на XML файл. Имаме коренен елемент vendors, за да спазим изискванията за well-formed XML. Запазваме йерархията установена по условие в записите в БД като представяме продуктите като списък от деца на елемент vendor. Колоните от таблиците тук са представени като атрибути на елементите `vendor` и `product`. (фиг. 3)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<vendors>
  <vendor id="1" name="Ivan Ivanov" phone="081258155">
    <product id="1" name="TV" price="155" />
    <product id="2" name="PC" price="1550" />
  </vendor>
  <vendor id="2" name="Georgi Ivanov" phone="086235268155">
    <product id="3" name="Tomatoes" price="5" />
  </vendor>
</vendors>
```

Фиг. 3: Същите данни, представени под формата на XML файл.

Гореописаните данни са чисто илюстративни- бихме могли да имаме много повече търговци и продукти, както и например corner cases като търговец без продукти или продукт без търговец. Засега решението не поддържа данни, в които продукт няма търговец, но това е напълно реалистична и постижима цел за в бъдеще- макар, че е спорно каква би била ползата от такива продукти- смятаме, че може би най-полезно би било, да се извежда предупреждение за тези продукти и те да не биват включвани в крайния резултат от даденото превръщане.

2.3 Тип и представяне на съдържанието

Съдържанието, както стана ясно от описанието на вида му, е изцяло текстово- било то под формата на SQL записи, или XML. За превръщане от DB->XML са необходими записи в БД, отговарящи на гореописаната структура и в резултат се получава XML файл с гореописания формат. При обратното превръщане XML->DB входните данни и крайния резултат са разменени. Размерите на изходните/крайните данни са почти незначителни- 8KB за всяка от простите SQL таблици (като този размер идва най-вече от добавената тежест на MSSQL- при по-минималистична система за управление на БД размерът би бил дори по-малъб) и под килобайт за примерния XML файл.

3 Дизайн

Решението е структурирано под формата на конзолно Java приложение. При стартиране, потребителят е подканен да избере в коя посока желае да трансформира данните си. При избор, се извиква съответната функция- dbToXml() или xmlToDB(). Те си приличат по структура и се различават по ред на действията и посока на създаване и запазване на данни. Кратко резюме на стъпките, които се изпълняват и при двете:

dbToXml():

1. Отваря се връзка към базата данни
2. Извличат се необходимите данни чрез SQL заявка
3. Те се преобразуват към списък от обекти VendorModel като всеки такъв обект има собствен списък от обекти ProductModel
4. Чрез JDOM създаваме йерархията от елементи и атрибути
5. Записваме резултата в XML файл

xmlToDb():

1. Чрез вградения в JDOM SAX parser отваряме изходния XML файл
2. Минавайки през йерархията на файла, запазваме данните му във VendorModel и ProductModel обекти
3. Отваря се връзка към базата данни
4. В базата се записват данните запазени във VendorModel и ProductModel

Засега решението разчита да се спазва зададения формат на данните (в схемата на SQL таблиците за данните в БД и в DTD файла за XML файла). При невалидни данни, SQL схемата и DTD ще се погрижат да хвърлят грешка и да прекратят изпълнението, но не е писана персонализирана логика за error handling и за ситуации като конфликти на данни (множество записи с идентични id ст-ти, запис на продукт с невалиден vendor_id и т.н.), но това лесно може да се надгради.

4 Тестване

При тестването, първото, което направихме бе да създадем таблиците в БД и да ги попълним с примерни данни. С тях тествахме първата половина от решението (dbToXml()). Функционалността работи според очакванията, когато данните в БД са валидни (което е винаги, тъй като MSSQL не позволява невалидни INSERT заявки). За обратното превръщане използвахме, както и получения XML файл от първата трансформация, така и ръчно написани XML файлове- при неспазване на изискванията, описани в DTD файла приложението хвърля грешка (но засега не при пряко валидиране с DTD файла, а при откриване на несъответствие в очаквания формат при обработката). Резултатът от успешно изпълнение съвпада с илюстрираното във фиг. от 1 до 3.

5 Заключение и възможно бъдещо развитие

Решението ни предлага основната функционалност за двупосочно превръщане на съдържание от БД към XML. Сред предимствата на решението са удобството, при нужда данни да се превърнат от единия вид в другия, надеждна валидация на указания формат и бързодействие на решението. Ограниченията включват използването на hardcoded данни- за имената на таблици/елементи и колони/атрибути, липса на персонализирани съобщения при грешка в обработката, както и нуждата от конфигурация с конкретна инстанция на база от данни.
Сред алтернативните решения на този проблем са употребата на MSSQL синтаксиса `FOR XML`, който генерира XML от базата автоматично, но с големи ограничения и впоследствие затруднява обработката на данните, тъй като те са представени като резултат на SQL заявка, но имат особеностите и структурата на XML израз. Това бе и една от първите идеи за подход към проблема, но не я предпочетохме поради споменатите сериозни недостатъци. Друга алтернатива би била да изградим XML файла с други помощни средства (вместо JDOM), или ръчно (чрез конкатениране/парсване на XML под формата на стринг, за изграждането/обхождането на XML файла например). Спряхме се точно върху JDOM, вместо върху други инструменти, тъй като е мощен, опростен и много добре се интегрира с Java и не избрахме да изграждаме XML ръчно, тъй като това биха били излишни усилия, когато има толкова инструменти, създадени да улесняват решаването на подобни проблеми.
В бъдеще кратък списък от аспектите, в които решението може да се подобри:

1. Изнасяне на имената на таблиците/елементите и колоните/атрибутите им в отделен модел за тях- напр. class Entity, с член-данни име и списък от колони/атрибути, които пък са шаблонен клас Column- така решението би се адаптирало по-лесно към релации 1:many, в които данните и структурата им са различни.

2. Програматична DTD валидация.

3. Персонализиран error handling.

6 Разпределение на работата

Заедно: проучване на проблема и потенциални методи за решение. Технически дискусии и тестване. Създаване на тази документация.
Боян: Връзка с базата данни и функцията dbToXml()
Христо: функцията xmlToDb() и рефакториране на кода.

7 Използвани литературни източници и Уеб сайтове

1. Документация на IntelliJ за свързване с MSSQL Server: https://www.jetbrains.com/help/idea/db-tutorial-connecting-to-ms-sql-server.html
2. JDOM документация: http://www.jdom.org/docs/apidocs/
3. Онлайн ресурс с примери за употреба на JDOM за създаване на XML йерархия и запис във файл: https://mkyong.com/java/how-to-create-xml-file-in-java-jdom-parser/
4. Онлайн ресурс с пример за употреба на JDOM за извличане на данни от XML файл: https://www.tutorialspoint.com/java_xml/java_jdom_parse_document.htm