АНАЛИЗ МОДЕЛЕЙ SAX И DOM ДЛЯ ОБРАБОТКИ XML ФАЙЛОВ

Кристина ОНИШОРУ,
Факультет Математики и Информатики

CZU: 004.6 *cristina.onishoru@gmail.com*

В мире, который включает в себя сотни миллионов потребителей и миллионы пунктов оказания медицинской помощи, необходимо развивать единое решение для подключения ко всем партнерам по обмену информацией, а также для сбора, сопоставления и оптимизации цифровых данных о здоровье. В данной исследовательской работе подобное решение рассматривается с точки зрения создания возможного *desktop* приложения, которое будет анализировать уведомления вышестоящих в иерархической структуре медицинских органов о принятии или отклонении тех или иных медицинских заявок.

Для разработки проекта следует определить следующие технические аспекты:

- Выделить основную идею приложения;
- Определить функциональную составляющую;
- Спроектировать структуру приложения;
- Выбрать платформу и язык программирования;
- Составить подробный план реализации.

Успешная реализация этих требований подразумевает решение двух проблем. Первая из них - это правильный выбор платформы, под которую будет создан проект. Для удобства создания программного модуля под разные ОС нами была выбрана кроссплатформенная разработка на языках Java и Scala [1, 2]. Вторая проблема заключается в выборе наиболее подходящего API-интерфейса для XML файлов большого размера. Существует два типа синтаксических анализаторов XML, а именно: простой API для XML (SAX) и объектная модель документа (DOM) [3]. В случае XML файлов больших размеров базовым аспектом становится анализ преимуществ и недостатков между SAX и DOM на некотором практическом примере. В качестве такого примера используются файлы *excel* формата.

Таким образом, основной задачей создаваемого программного модуля является извлечение данных из *excel* файлов больших размеров для последующего мапирования и сохранения в базе данных. В папку коннектор помещаются один или более .xlsx файлов, которые в последствии обрабатываются программой. Каждая *excel* таблица может содержать до 40 вкладок, каждая вкладка содержит около 65 тысяч строк и 27 колонок. Размер такого файла достигает 200 – 300 мегабайт. Данные из таблицы должны быть извлечены корректным образом (без потерь и повреждений) и помещены во временные хранилища (ассоциативные массивы) для валидации, обработки и последующего анализа.

Преимущества **SAX** обработки очень похожи на преимущества потокового мультимедиа. Анализ может начаться немедленно вместо ожидания обработки всех данных. Более того, поскольку приложение проверяет данные только тогда, когда они читаются, нет необходимости хранить данные в памяти. Это огромное преимущество для больших документов. Фактически, приложению даже не нужно анализировать весь документ, оно может прекратить синтаксический анализ при выполнении определенного условия. Вообще говоря, SAX намного быстрее, чем его DOM. С другой стороны, поскольку приложение не хранит данные каким-либо образом, невозможно использовать SAX для изменения данных или перемещения их обратно в поток данных.

DOM и обработка на основе дерева в широком смысле имеют несколько преимуществ. Во-первых, поскольку дерево является постоянным в памяти, его можно изменять с тем, чтобы приложение могло вносить изменения в данные и структуру. Возможно перемещение вверх и вниз по дереву в любое время, вместо одноразового процесса, как в случае SAX. DOM также намного проще в использовании. С другой стороны, построение такого дерева в памяти требует больших затрат. Нередко большие файлы занимают весь объем системной памяти. Кроме того, создание дерева DOM может быть медленным процессом.

Разработанное приложение подтвердило ожидаемую разницу между SAX и DOM. Хотя оба являются синтаксическими анализаторами XML, первый обладает замечательной эффективностью памяти, в то время как второй отстает в этом аспекте. SAX загру-

жает в свою память существенно ограниченный объем файла. Напротив, DOM становится гораздо более полезным, поскольку он может полностью загрузить документ в свою память. Кроме того, SAX имеет примечательную скорость, DOM работает относительно медленнее.

Библиография:

1. Что такое нативные и кроссплатформенные приложения? Плюсы и минусы [Электронный ресурс] https://itvdn.com/ru/blog/article/native-cross-platform
2. Руководство по Scala. Введение [Электронный ресурс] https://proselyte.net/tutorials/scala/introduction/
3. Difference Between SAX Parser and DOM Parser in Java [Электронный ресурс] https://www.geeksforgeeks.org/difference-between-sax-parser-and-dom-parser-in-java/

Рекомендовано:
Ион АНДРИЕШ, др., конф.унив.