

Конфигурация JAVA приложения с помощью XML

Андрей Романенко

1. Введение

Очень часто Java разработчику необходимо вынести настройку программы за пределы исходного текста, чтобы конфигурирование можно было проводить без перекомпиляции всей системы. К подобным настройкам можно отнести сетевые порты, адреса серверов, пути на жестком диске и даже внешний вид приложения. Если разработчик использует какой-либо framework, то, возможно, данные задачи уже решены. Но такое решение будет иметь небольшой недостаток: пользователь, проводящий настройку, должен разбираться в framework, что не всегда возможно.

Исходя из этого многие разработчики придумывают свою систему: текстовые файлы, xml файлы, механизм properties, предоставляемый самой платформой java, и тп.

На мой взгляд, для самого широкого применения больше всего подходит xml. Как правило, большинство администраторов и программистов с xml уже знакомы, и редактирование настроек будет весьма тривиальным процессом. Да и иерархическая модель xml позволяет лучше скомпоновать настройки в группы.

Примером подобного xml файла с настройками может служить фрагмент:

```
<config>
<listen>
<ip>192.168.5.2</ip>
<port>5677</port>
</listen>
<mail>
<smtp>mail.acme.com</smtp>
```

```
<login>username</login>
<password>12345</password>
</mail>
<performance>
<threads>10</threads>
<openfiles>50</openfiles>
<sessions>100</sessions>
<sessionTimeout>1h</sessionTimeout>
</performance>
</config>
```

Далее приведен пример приложения, которое прочитает подобный файл, и позволит читать все параметры.

При этом реализация используют только бесплатные open-source библиотеки.

Для исполнения примеров Вам понадобится JDK 1.5 (хотя примеры сработают и в среде 1.4).

Основная библиотека доступна для скачивания с сайта <http://configloader.sourceforge.net>

где Вы можете скачать и исходные текста, и уже скомпилированную библиотеку в виде JAR.

ConfigLoader поддерживает загрузку конфигураций из файлов и текстовых переменных. Загрузка из текстовых переменных удобна при получении настроек с сервера, по сети.

Сами примеры были написаны в IDE Eclipse, но аналогичные действия можно произвести в любой другой среде разработки, или при работе с командной строки.

Итак.

2. Скачаем все необходимые библиотеки

ConfigLoader – доступен на сайте по адресу: <http://configloader.sourceforge.net/index.php/Downloads>

файл configloader_*.jar

Конфигурация JAVA приложения с помощью XML

Дополнительно нам понадобится библиотека, реализующая SAX – simple api for XML. В мире java представлено множество подобных разработок, но для определенности воспользуемся xerces – решением от сообщества apache.

Актуальную версию можно скачать здесь:

<http://xerces.apache.org/xerces2-j/download.cgi>

понадобится файл - Xerces-J-bin.*.zip

3. Создадим проект в Eclipse

В IDE Eclipse выбираем пункт создания нового проекта, имя задайте на ваш вкус, и укажите JDK версию 1.5

Остальные настройки проекта на ваше усмотрение.

В самом проекте создайте папку lib и скопируйте в нее 3 файла:

- скачанный configloader.jar
- xercesImpl.jar (из скачанного zip архива)
- xml-apis.jar (так же из архива)

И добавьте их в build path, путем правого клика – Build Path – Add to Build Path

4. Создадим тестовый файл с конфигурацией и заготовку класса

В Eclipse создадим папку xml (в корне нашего проекта) и в нее запишем файл example1.xml, в который скопируем ранее представленный xml.

Теперь создадим пакет (File – New – Package) и зададим имя “configarticle”. В этом пакете мы и будем держать наши тестовые классы.

5. Наш первый класс

Для определенности мы хотим, чтобы наш первый класс читал секцию mail из файла конфигурации: smtp, login и password; и выдавал эти данные на консоль.

Конфигурация JAVA приложения с помощью XML

Создадим новый класс “Example1” в пакете “configartice”.

Для начала нам необходимо импортировать нужные классы, точнее класс:

```
import com.romanenco.configloader.ConfigLoader;
```

В самом теле класса создадим метод main

```
public static void main(String[] args) {  
    ...  
    ...  
}
```

наши дальнейшие строки кода мы будем добавлять в тело этого метода.

Первая наша строка – создание объекта, который и позволит нам удобно работать с конфигурациями.

```
ConfigLoader config = new ConfigLoader();
```

Этот момент опишем чуть подробнее. Как уже говорилось ранее для работы с xml нам нужен так называемый sax parser. Т.е. java класс поддерживающий определенные интерфейсы.

Библиотека ConfigLoader по умолчанию работает с Xerces parser, а раз мы его уже скачали и добавили в CLASSPATH, то конструктора без параметров нам достаточно.

В случае если вы выбрали другой parser, то вам надо сделать несколько дополнительных шагов.

Если вы сконфигурировали default xml parser для вашей JRE (через файл jaxr.properties), то создать ConfigLoader можно так:

```
ConfigLoader config = new ConfigLoader("");
```

(параметр – пустая строка)

Конфигурация JAVA приложения с помощью XML

Если вы не уверены, что у клиента на его JRE подобная настройка будет, то вам надо выяснить имя класса parser'a для sax1 (из документации).

Для Xerces это будет: `org.apache.xerces.parsers.SAXParser`

Для Piccolo (другой популярный parser): `com.bluecast.xml.Piccolo`

И создание ConfigLoader будет таким:

```
ConfigLoader config =
    new
    ConfigLoader("org.apache.xerces.parsers.SAXParser;
```

Экземпляр создан.

Следующим шагом будет загрузка самой конфигурации, сделать это тоже весьма просто:

```
config.LoadFromFile("xml/example1.xml");
```

метод может вызвать RuntimeException – если файл не найден, либо содержит не правильный xml. Поэтому в реальной программе правильно будет поместить операцию загрузки в try-catch блок и в случае RuntimeException сообщать пользователю об ошибке.

Теперь самое интересное, нам надо прочитать значения трех тегов smtp, login и password, которые вложены в тэги mail и config.

Код при этом прост:

```
System.out.println("Host = " +
config.getTagValue("config.mail.smtp"));

System.out.println("Username = " +
config.getTagValue("config.mail.login"));

System.out.println("Password = " +
config.getTagValue("config.mail.password"));
```

Не трудно заметить, как нам обратиться нужному тегу: надо просто задать цепочку

имен тегов, начиная от корневого и так до него нужного нам.

Т.е. если мы хотим получить количество потоков(threads) из секции производительность(performance), то нам надо задать имя: "config.performance.threads"

Полный текст нашего примера будет таким:

```
package configarticle;
import com.romanenco.configloader.ConfigLoader;

public class Example1 {

    public static void main(String[] args) {

        ConfigLoader config = new ConfigLoader();
        config.LoadFromFile("xml/example1.xml");
        System.out.println("Host = " +
            config.getTagValue("config.mail.smtp"));
        System.out.println("Username = " +
            config.getTagValue("config.mail.login"));
        System.out.println("Password = " +
            config.getTagValue("config.mail.password"));
    }

}
```

Как мы видим, подобный подход к оформлению конфигурационных файлов создает удобства и для разработчика, и для конечного пользователя.

Разработчик может использовать простой и надежный подход во всех проектах, а пользователь (даже не высококвалифицированный) будет иметь возможность ясно настраивать приложение.

6. Дополнительные возможности работы с параметрами конфигурации

Ранее был приведен пример загрузки настроек без повторяющихся тегов, т.е. цепочка

Конфигурация JAVA приложения с помощью XML

“имя_тега.имя_тега....имя_тега” однозначно идентифицировала нужное нам значение.

Но параметры могут иметь и подобную структуру:

```
<config>
  <listen>
    <ip>192.168.5.2</ip>
    <ip mustpresent="true">192.168.5.3</ip>
    <ip mastpresent="true">192.168.5.4</ip>
    <port>5677</port>
  </listen>
</config>
```

Т.е. цепочка config.listen.ip уже соответствует не одному, а нескольким (трем в данном случае) значениям.

Попробуем решить эту задачу....

Приведенный xml скопируем в новый файл: xml/example2.xml

И создадим новый класс Example2, с содержимым:

```
package configarticle;

import com.romanenco.configloader.ConfigLoader;

public class Example2 {
    public static void main(String[] args) {

        ConfigLoader config = new ConfigLoader();

        config.LoadFromFile("xml/example2.xml");

        //new lines insert here

    }
}
```

Для начала попробуем добавить след. строку после комментария и посмотрим, что она вернет:

```
System.out.println("Value1 = " +  
config.getTagValue("config.listen.ip"));
```

Результат будет: Value1 = 192.168.5.2

Что было предсказуемо: путь "config.listen.ip" означает кратчайшее расстояние по тегам, т.е. в случае если у тега А есть несколько вложенных тегов В, то путь "А.В" будет означать первый дочерний тег В.

Для получения значения второго вложенного тега можно применить такой путь: "А.В", "1"

Или для нашего файла example2:

```
System.out.println("Value2 = " +  
config.getTagValue("config.listen.ip", "1"));
```

что нам вернет: Value2 = 192.168.5.3

Таким образом мы видим, что можем получить значение любого тега из группы, зная его номер (считая с 0).

Примерами путей могут быть:

```
config.server", "1", "ip", "2"  
тег config имеет несколько вложенных тегов server, которые,  
в свою очередь, содержат несколько тегов ip  
  
config", "0", "server", "0", "ip", "0"  
аналог записи "config.server.ip"
```

7. Последние штрихи

Напоследок опробуем несколько дополнительных операций...

Теги могут иметь параметры, как например некоторые теги ip из нашего примера. Получить эти значения можно так:

```
System.out.println("Arg = "
```

Конфигурация JAVA приложения с помощью XML

```
+ config.getTagAttributeValue("mustpresent",  
"config.listen.ip", "1");
```

Подобным образом можно получить список имен всех параметров:

```
System.out.println("All args = "  
  
+ config.getTagAttributeNames("config.listen.ip",  
"1"));
```

Допустим мы заранее не знаем количество вложенных тегов ip. Для того, чтобы выяснит их кол-во мы можем применить метод:

```
System.out.println("Tag count (ip) = "  
  
+ config.getTagChildrenCountWithName("ip",  
"config.listen"));
```

что вернет нам кол-во вложенных тегов с именем "id".

Получить общее кол-во вложенных тегов можно так:

```
System.out.println("Tag count = "  
  
+ config.getTagChildrenCount("config.listen"));
```

Полный текст класса Example2

```
package configarticle;  
  
import com.romanenco.configloader.ConfigLoader;  
  
public class Example2 {  
  
    public static void main(String[] args) {  
  
        ConfigLoader config = new ConfigLoader();  
        config.LoadFromFile("xml/example2.xml");  
  
    }  
  
}
```

Конфигурация JAVA приложения с помощью XML

```
System.out.println("Value1 = " +
config.getTagValue("config.listen.ip"));

System.out.println("Value2 = " +
config.getTagValue("config.listen.ip","1"));

System.out.println("Arg = "
+ config.getTagAttributeValue("mustpresent",
"config.listen.ip", "1"));

System.out.println("All args = "
+ config.getTagAttributeNames("config.listen.ip",
"1"));

System.out.println("Tag count = "
+ config.getTagChildrenCount("config.listen"));

System.out.println("Tag count (ip) = "
+ config.getTagChildrenCountWithName("ip",
"config.listen"));
}
}
```