Sax Parser

<https://dev64.wordpress.com/2012/04/28/learning-sax/>

# Осваиваем SAX парсер: Quick start

### [Галерея](https://dev64.wordpress.com/type/gallery/)

Posted on [Апрель 28, 2012](https://dev64.wordpress.com/2012/04/28/learning-sax/)

Работая с JAXB и XPath API я столкнулся с необходимостью манипуляции namespace-ами XML-документов. Проблема решена и описана в посте [Чтение XML-документа с помощью JAXB с заменой namespace](https://dev64.wordpress.com/2012/04/26/namespace-filter-example/). Решение этой проблемы было найдено в интернете. Ещё, небольшие поиски по интернету показали, что в большинстве случаев, когда речь идёт о SAX API, везде приводят один и тот же пример замены namespace-ов. Я решил разобраться с ситуацией и отправился на официальный сайт <http://www.saxproject.org/>. На официальном сайте я перечитал документацию и нашёл, почему везде выложен один и тот же пример. Этот же пример присутствует и в официальной документации…

Тем не менее, SAX API весьма важен и лежит в основе многих других технологий, поэтому я решил изучить SAX API немного подробнее. Ниже перевод материалов с официального сайта. Я нашёл их весьма интересными и заслуживающими более подробного изучения.

**Оглавление**

* [SAX парсер: Quick Start](https://dev64.wordpress.com/2012/04/28/learning-sax/)
* [SAX парсер: Features и Properties](https://dev64.wordpress.com/2012/04/28/sax-parser-features-and-properties/)
* SAX парсер: Namespaces
* [JavaDocs](http://www.saxproject.org/apidoc/overview-summary.html)

**Требования**

SAX — общий интерфейс реализованный для многими XML парсерами (и штуками, позиционирующими себя как XML парсеры), он практически является на столько же общим API для работы с XML, насколько JDBC является общим интерфейсом, реализованным для различных реляционных баз данных (и штруковинами, позиционирующими себя как реляционные базы данных). Если вы хотите использовать SAX, вам потребуются всё перечисленное ниже:

* Java 1.1 или современнее.
* SAX2-совместимый XML парсер, указанный в вашем Java classpath. Если вам нужен подобный парсер, смотрите ссылки.
* SAX2 дистрибутив, указанный в вашем Java classpath. (Наиболее вероятно, вы получите его вместе с вашим парсером).
* Большинство Java/XML дистрибутивов включают SAX2 парсер. Большинство Web application серверов используют его для работы с XML. Все дистрибутивы с поддержкой JAXP 1.1 включают поддержку SAX2.

**Парсинг XML документа**

Начните созданием класса, который наследует (extends) DefaultHandler:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | import org.xml.sax.helpers.DefaultHandler;    public class MySAXApp extends DefaultHandler  {        public MySAXApp ()      {      super();      }    } |

Так как это Java — приложение, мы создадим статический метод main, который использует метод createXMLReader класса XMLReaderFactory для выбора SAX драйвера динамически (программно). Обработка исключений удалена из примера для облегчения понимания. Реальное приложение должно обрабатывать исключения:

|  |  |
| --- | --- |
| 1  2  3  4  5 | public static void main (String args[])  throws Exception  {  XMLReader xr = XMLReaderFactory.createXMLReader();  } |

В случае, если ваша Java среда разработки автоматически не добавляет compiled-in default (или не настраивает META-INF/services/org.xl.sax.driver system resource), вам вероятно потребуется установить в Java system property оrg.xml.sax.driver значение, указывающее полный путь к SAX driver, как в примере:

java -Dorg.xml.sax.driver=com.example.xml.SAXDriver MySAXApp sample.xml

Множество разных SAX2 драйверов распространены на сегодняшний день Поэтому в качестве имени класса вы можете использовать, например:

| **Class Name** | **Notes** |
| --- | --- |
| gnu.xml.aelfred2.SAXDriver | Lightweight non-validating parser; Free Software |
| gnu.xml.aelfred2.XmlReader | Optionally validates; Free Software |
| oracle.xml.parser.v2.SAXParser | Optionally validates; proprietary |
| org.apache.crimson.parser.XMLReaderImpl | Optionally validates; used in JDK 1.4; Open Source |
| org.apache.xerces.parsers.SAXParser | Optionally validates; Open Source |

Иначе, если вас не беспокоит привязка вашего приложения к определённому SAX драйверу, вы можете использовать его конструктор напрямую. Мы предполагаем, что SAX driver для вашего XML парсера называется com.example.xml.SAXDriver, однако, в реальности такого класса не существует. Вы должны знать имя настоящего драйвера для вашего парсера, чтобы использовать такой подход.

|  |  |
| --- | --- |
| 1  2  3  4  5 | public static void main (String args[])  throws Exception  {  XMLReader xr = new com.example.xml.SAXDriver();  } |

Созданный объект может быть использован для парсинга XML-документов, но сначала, нужно зарегистрировать обработчики событий, которые парсер может использовать для передачи информации. Это делается с помощью методов setContentHandler и setErrorHandler интерфейса XMLReader. В настоящем приложении, эти обработчики будут обычно разными объектами, но для приведённого простого примера, мы помещаем эти обработчики в Top-level class. Таким образом нам нужно создать наш класс и зарегистрировать его в XMLReader:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | public static void main (String args[])  throws Exception  {  XMLReader xr = XMLReaderFactory.createXMLReader();  MySAXApp handler = new MySAXApp();  xr.setContentHandler(handler);  xr.setErrorHandler(handler);  } |

Приведённый пример создаёт экземпляр MySAXApp для получения XML parsing events и регистрирует его в XML reader для получения regular content событий и error событий (есть другие виды событий, но они редко используются). Теперь давайте предположим, что все аргументы командной строки — имена файлов и мы будем пытаться парсить их по-одному используя метод parse из интерфейса XMLReader:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | public static void main (String args[])  throws Exception  {  XMLReader xr = XMLReaderFactory.createXMLReader();  MySAXApp handler = new MySAXApp();  xr.setContentHandler(handler);  xr.setErrorHandler(handler);                // Parse each file provided on the              // command line.  for (int i = 0; i < args.length; i++) {      FileReader r = new FileReader(args[i]);      xr.parse(new InputSource(r));  }  } |

Заметьте, что каждый reader должен быть представлен InputSource объектом, чтобы быть «распарсенным». Ниже полный пример описанного demo класса:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | import java.io.FileReader;    import org.xml.sax.XMLReader;  import org.xml.sax.InputSource;  import org.xml.sax.helpers.XMLReaderFactory;  import org.xml.sax.helpers.DefaultHandler;      public class MySAXApp extends DefaultHandler  {        public static void main (String args[])      throws Exception      {      XMLReader xr = XMLReaderFactory.createXMLReader();      MySAXApp handler = new MySAXApp();      xr.setContentHandler(handler);      xr.setErrorHandler(handler);                    // Parse each file provided on the                  // command line.      for (int i = 0; i < args.length; i++) {          FileReader r = new FileReader(args[i]);          xr.parse(new InputSource(r));      }      }          public MySAXApp ()      {      super();      }  } |

Вы можете откомпилировать этот код и запустить его (Не забудьте указать SAX driver класс в свойстве org.xml.sax.driver), однако ничего особенного не возникнет, если только ваш документ не содержит «плохой» XML, поскольку вы пока не установили обработчиков SAX событий для вашего приложения.

**Обработка событий**

Всё становится интересней когда вы начинаете реализовывать методы для реакции на XML parsing events ( помните мы зарегистрировали наш класс для получения XML parsing events в предыдущем разделе). Наиболее важные события — начало и завершение документа (start and end of the document), начало и завершение элемента (start and end of elements) и символьных данных.

Чтобы узнать про начало и завершение документа, клиентское приложение реализует методы startDocument и endDocument.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public void startDocument ()  {  System.out.println("Start document");  }    public void endDocument ()  {  System.out.println("End document");  } |

Обработчики событий start/endDocument не имеют аргументов. Когда SAX driver находит начало документа, он вызовет функцию startDocument один раз, когда найдёт завершение документа — вызовет endDocuemnt (даже если были обнаружены ошибки).

Приведённые примеры печатаю сообщения в стандартный output, но ваше приложение может содержать любой код для этих обработчиков, наиболее часто, этот код будет создавать какой то тип дерева в памяти, делать вывод, производить сохранение в базу данных или извлекать информацию из XML stream.

SAX driver будет сообщать о стартовых и конечных элементах практически также за исключением того, что он будет передавать несколько параметров в методы startElement и endElement.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | public void startElement (String uri, String name,                String qName, Attributes atts)  {  if ("".equals (uri))      System.out.println("Start element: " + qName);  else      System.out.println("Start element: {" + uri + "}" + name);  }    public void endElement (String uri, String name, String qName)  {  if ("".equals (uri))      System.out.println("End element: " + qName);  else      System.out.println("End element:   {" + uri + "}" + name);  } |

Приведенные методы будут печатать в начале и конце каждого элемента. Перед именем элемента (local name) будет печататься namespace URI в фигурных скобках. Параметр qName содержит имя без namespace в формате XML 1.0, которое вы должны использовать для всех элементов не имеющих namespace URI. В этом Quick Introduction, мы не будем рассматривать как обрабатываются атрибуты. Атрибуты могут быть получены по имени или итерацией по ним как будто они хранятся в массиве.

Наконец, SAX2 передает (reports) обычные символьные данные через метод characters. Следующий пример напечатает все символьные данные на экран. Output будет возможно слегка длиннее потому что он производит escaping специальных символов (pretty-prints the output by escaping special characters):

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | public void characters (char ch[], int start, int length)  {  System.out.print("Characters:    \"");  for (int i = start; i < start + length; i++) {      switch (ch[i]) {      case '\\':      System.out.print("\\\\");      break;      case '"':      System.out.print("\\\"");      break;      case '\n':      System.out.print("\\n");      break;      case '\r':      System.out.print("\\r");      break;      case '\t':      System.out.print("\\t");      break;      default:      System.out.print(ch[i]);      break;      }  }  System.out.print("\"\n");  } |

Заметьте, что SAX driver может разбивать символьные данные как угодно, так что вы не можете рассчитывать на то, что все символьные данные придут в одном событии (single characters event).

**Пример SAX2 приложения**

Ниже пример готового приложения (конечно в настоящем приложении обработчики событий будут скорее всего реализованы в отдельном классе, а не в классе приложения).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100 | import java.io.FileReader;    import org.xml.sax.XMLReader;  import org.xml.sax.Attributes;  import org.xml.sax.InputSource;  import org.xml.sax.helpers.XMLReaderFactory;  import org.xml.sax.helpers.DefaultHandler;      public class MySAXApp extends DefaultHandler  {        public static void main (String args[])      throws Exception      {      XMLReader xr = XMLReaderFactory.createXMLReader();      MySAXApp handler = new MySAXApp();      xr.setContentHandler(handler);      xr.setErrorHandler(handler);                    // Parse each file provided on the                  // command line.      for (int i = 0; i < args.length; i++) {          FileReader r = new FileReader(args[i]);          xr.parse(new InputSource(r));      }      }          public MySAXApp ()      {      super();      }          ////////////////////////////////////////////////////////////////////      // Event handlers.      ////////////////////////////////////////////////////////////////////          public void startDocument ()      {      System.out.println("Start document");      }          public void endDocument ()      {      System.out.println("End document");      }          public void startElement (String uri, String name,                    String qName, Attributes atts)      {      if ("".equals (uri))          System.out.println("Start element: " + qName);      else          System.out.println("Start element: {" + uri + "}" + name);      }          public void endElement (String uri, String name, String qName)      {      if ("".equals (uri))          System.out.println("End element: " + qName);      else          System.out.println("End element:   {" + uri + "}" + name);      }          public void characters (char ch[], int start, int length)      {      System.out.print("Characters:    \"");      for (int i = start; i < start + length; i++) {          switch (ch[i]) {          case '\\':          System.out.print("\\\\");          break;          case '"':          System.out.print("\\\"");          break;          case '\n':          System.out.print("\\n");          break;          case '\r':          System.out.print("\\r");          break;          case '\t':          System.out.print("\\t");          break;          default:          System.out.print(ch[i]);          break;          }      }      System.out.print("\"\n");      }    } |

**Sample Output**

Предположим обрабатывается следующий XML документ:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <?xml version="1.0"?>    <poem xmlns="<http://www.megginson.com/ns/exp/poetry>">  <title>Roses are Red</title>  &lt;l>Roses are red,&lt;/l>  &lt;l>Violets are blue;&lt;/l>  &lt;l>Sugar is sweet,&lt;/l>  &lt;l>And I love you.&lt;/l>  </poem> |

Если этот документ назван roses.xml и у вас есть SAX2 driver в classpath с именем com.example.xml.SAXDriver (приведённый в примере на деле не существует), в можете вызвать пример приложения примерно так:

java -Dorg.xml.sax.driver=com.example.xml.SAXDriver MySAXApp roses.xml

При запуске, вы получить output подобный этому:

Start document

Start element: {http://www.megginson.com/ns/exp/poetry}poem

Characters: "\n"

Start element: {http://www.megginson.com/ns/exp/poetry}title

Characters: "Roses are Red"

End element: {http://www.megginson.com/ns/exp/poetry}title

Characters: "\n"

Start element: {http://www.megginson.com/ns/exp/poetry}l

Characters: "Roses are red,"

End element: {http://www.megginson.com/ns/exp/poetry}l

Characters: "\n"

Start element: {http://www.megginson.com/ns/exp/poetry}l

Characters: "Violets are blue;"

End element: {http://www.megginson.com/ns/exp/poetry}l

Characters: "\n"

Start element: {http://www.megginson.com/ns/exp/poetry}l

Characters: "Sugar is sweet,"

End element: {http://www.megginson.com/ns/exp/poetry}l

Characters: "\n"

Start element: {http://www.megginson.com/ns/exp/poetry}l

Characters: "And I love you."

End element: {http://www.megginson.com/ns/exp/poetry}l

Characters: "\n"

End element: {http://www.megginson.com/ns/exp/poetry}poem

End document

Заметьте, что даже этот короткий документ генерирует (по крайней мере) 25 событий: один на старт и завершение каждого из шести элементов (тегов), один на каждый из одиннадцати фрагментов символьных данных и один на старт и завершение документа.

Если у входного документа нет namespace нет атрибута xmlns=»<http://www.megginson.com/ns/exp/poetry&#187>; указывающего, что элементы находятся в указанном namespace, outoput будет выглядеть подобно этому:

Start document

Start element: poem

Characters: "\n"

Start element: title

Characters: "Roses are Red"

End element: title

Characters: "\n"

Start element: l

Characters: "Roses are red,"

End element: l

Characters: "\n"

Start element: l

Characters: "Violets are blue;"

End element: l

Characters: "\n"

Start element: l

Characters: "Sugar is sweet,"

End element: l

Characters: "\n"

Start element: l

Characters: "And I love you."

End element: l

Characters: "\n"

End element: poem

End document

Наиболее вероятно, вам придется работать с обеими типами документов: c документами использующими XML namespace-ы и не использующими их. Документы могут быть также с разными элементами (часть элементов и атрибутов с namespace, часть нет). В любом случае убедитесь, что ваш код проверяет namespace URI элементов, а не просто рассчитывает на то, что он всегда есть или его всегда нет.

[Продолжение здесь…](https://dev64.wordpress.com/2012/04/28/sax-parser-features-and-properties/)

# Осваиваем SAX парсер: Features и Properties

### [Галерея](https://dev64.wordpress.com/type/gallery/)

Posted on [Апрель 28, 2012](https://dev64.wordpress.com/2012/04/28/sax-parser-features-and-properties/)

Продолжаю перевод документации с официального сайта: <http://www.saxproject.org>

**Оглавление**

* [SAX парсер: Quick Start](https://dev64.wordpress.com/2012/04/28/learning-sax/)
* [SAX парсер: Features и Properties](https://dev64.wordpress.com/2012/04/28/sax-parser-features-and-properties/)
* [SAX парсер: Namespaces](https://dev64.wordpress.com/2012/04/28/study-sax-parser-namespaces/)
* [JavaDocs](http://www.saxproject.org/apidoc/overview-summary.html)

Обратите внимание: свойства (properties), перечисленные на этой странице не последней версии. (out of date). Используйте Javadoc для текущей документации.

SAX2 определяет стандартные методы для получения и установки feature flags и значений свойств в XMLReader. Возможно изменение поведения парсера, например можно указать, чтобы XML reader валидировал или не валидировал документ, можно зарегистрировать новые типы обработчиков событий используя getFeature, setFeature, getProperty и setProperty методы:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | try {    String id = "<http://xml.org/sax/features/validation>";    if (xmlReader.getFeature(id)) {      System.out.println("Parser is validating.");    } else {      System.out.println("Parser is not validating.");    }  } catch (SAXNotRecognizedException e) {    System.out.println("Can't tell.");  } catch (SAXNotSupportedException e) {    System.out.println("Wrong time to ask.");  } |

Тогда как feature flags — всегда boolean, значения свойств могут принимать любые объекты. Все имена свойств и feature — полностью указанные (fully-qualified) URI, например «<http://www.acme.com/features/foo&#187>;. Также как и с namespace-ами, значения свойств и feature указываются с помощью URI, которые они управляют.

**Исключения для Features и свойств**

Если приложение пытается запросить или установить feature flag, который XML reader не знает (или не может получить или установить для него значение), тогда XML reader throws SAXNotRecognizedException. Это должно указать вашему приложению, что вы используете неверный URI. Если приложение пытается установить feature или значение свойства, которое XML не поддерживает или пытается модифицировать feature или свойство, являющееся read-only, XML reader бросает исключение SAXNotSupportedException.

One important application for properties is getting and setting extension event handlers, for event types not supported by the four core SAX2 handlers, EntityResolver, DTDHandler, ContentHandler, and ErrorHandler. Outside parties are free to define handlers for any kinds of events, and to create properties for setting and querying them.

**URI идентифицирующие Feature Flags и свойства**

# Нет фиксированного набора feature флагов и свойств доступных для SAX2, кроме двух feature флагов, Осваиваем SAX парсер: Namespaces

### [Галерея](https://dev64.wordpress.com/type/gallery/)

Posted on [Апрель 28, 2012](https://dev64.wordpress.com/2012/04/28/study-sax-parser-namespaces/)

Продолжаю перевод документации с официального сайта: <http://www.saxproject.org>

**Оглавление**

* [SAX парсер: Quick Start](https://dev64.wordpress.com/2012/04/28/learning-sax/)
* [SAX парсер: Features и Properties](https://dev64.wordpress.com/2012/04/28/sax-parser-features-and-properties/)
* [SAX парсер: Namespaces](https://dev64.wordpress.com/2012/04/28/study-sax-parser-namespaces/)
* [JavaDocs](http://www.saxproject.org/apidoc/overview-summary.html)

В SAX2 реализована поддержка XML Namespace, требующаяся для высокоуровневых стандартов таких как XPath (используется XSLT, XPointer, XLink и другими), XML Schema, RDF и другими. Каждая реализация SAX2 XMLReader интерфейса должна поддерживать обработку Namespace-ов по-умолчанию. Дополнительно, многие XML reader-ы позволяют модифицировать обработку Namespace-ов или отключить её.

Обработка Namespace-ов изменяет только именование элементов и атрибутов, а также вносить ограничения на «некоторые другие имена». В XML 1.0, каждый XML элемент имеет единственное имя называемое qName (которое может содержать двоеточия). С Namespace-ами, имена элементов и атрибутов — состаные, состоящие из 2-х частей, такое составное имя иногда называется «Universal» или «Expanded» имя. Имя состоит из URI (подобно именам пактов в Java или Perl) и localName (никогда не содержащее двоеточий).

SAX2 способен поддерживать оба из подходов одновременно. XML документы также могут поддерживать оба подхода. SAX2 XMLReader реализации должны поддерживать Namespaces, когда документы используют их.

**Namespaces в SAX Events**

Поддержка Namespace влияет на ContentHandler и Attributes интерфейсы. Вам нужно обратить основное внимение на поддержку Namespace в startElement() callbacks.

**Имена элементов**

В SAX2 startElement и endElement callback в обработчике контента выглядят так:

public void startElement (String uri, String localName,

String qName, Attributes atts)

throws SAXException;

public void endElement (String uri, String localName, String qName)

throws SAXException;

По-умолчанию, XML reader будет сообщать Namespace URI и localName для каждого элемента, принадлежащего namespace и в start и в end handler-e (обработчике).

Пример:

|  |  |
| --- | --- |
| 1 | <html:hr xmlns:html="<http://www.w3.org/1999/xhtml>"/> |

С дефолтной обработкой SAX2 Namespace, XML reader сообщит о начале и завершении элемента событиями с Namespace URI: <http://www.w3.org/1999/xhtml> и localName hr. Большинство XMLReader реализаций также сообщат оригинальное имя в qName html:hr, однако этот параметр может быть и просто пустой строкой (за исколючением элементов не находящихся в описанном namespace).

**Имена атрибутов**

Для атрибутов, значение именованного атрибута может быть найдено использованием метода getValue или вы можете искать Namespace URI и localName атрибута по его индексу используя getURI и getLocalName методы (обычно когда вы итерируетесь через весь список атрибутов).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | String rdfns = "[http://www.w3.org/1999/02/22-rdf-syntax-ns#](http://www.w3.org/1999/02/22-rdf-syntax-ns)";  String resource = atts.getValue(rdfns, "resource");    for (int i = 0; i < atts.getLength(); i++) {    String uri = atts.getURI(i);    String localName = atts.getLocalName(i);    String value = atts.getValue(i);    /\* ... \*/  } |

Мэппинг префиксов

В дополнения к описанным событиям, SAX2 сообщает границы использования Namespace деклараций используя startPrefixMapping и endPrefixMapping методы, так, что приложения могут определять префиксы в именах атрибутов и символьных данных по необходимости. Callback-и выглядят так:

public void startPrefixMapping (String prefix, String uri)

throws SAXException;

public void endPrefixMapping (String prefix)

throws SAXException;

Для примера выше, XML reader сделает следующий callback вызов сразу перед start-element событием:

startPrefixMapping("html", "http://www.w3.org/1999/xhtml")

XML reader сделает следующий вызов сразу за end-element событием:

endPrefixMapping("html")

Для элементов с множественными описаниями namespace, после startPrefixMapping не обязательно следует соответствующий endPrefixMapping, поскольку эти вызовы могут идти в любом порядке.

**Конфигурация**

Этот раздел применим к SAX2 приложениям с требованиями по сохранению префиксов и их деклараций (preserving prefixes and their declarations). Приложения, которые пишут XML-текст часто попадают в эту категорию, включая множество XML-редакторов.

**Конфигурирование поддержки Namespace-ов**

Feature flag c именем <http://xml.org/sax/features/namespaces> управляет Namespace processing. Когда флаг установлен (по-умолчанию), любой применяемый Namespace URI и localName (для элементов и namespace-ов) должен быть доступен в startElement и endElement callbacks в ContentHandler интерфейсе и в различных методах  
Attributes интерфейса. Кроме того start/endPrefixMapping() события должны выдаваться. Для элементов и атрибутов за пределами namespace-ов, связанный с ними namespace URI будет пустой строкой и параметр qName гарантированно должен быть непустой строкой.

Feature flag с именем <http://xml.org/sax/features/namespace-prefixes> управляет сообщением qNames и Namespace деклараций (xmns атрибутов). Если значение этого флага не изменено на true (его умолчание false), qNames могут быть опционально (!) быть пустыми строками для элементов и атрибутов имеющих связанное с ними namespace URI, xmlns атрибуты не будут рапортоваться. При установке флага в true, эта информация всегда будет выставлена.

Приведенная таблица суммирует взаимодействие упомянутых 2-х features.

| **namespaces** | **namespace-prefixes** | **Namespace names** | **start/end PrefixMapping** | **qNames** | **xmlns\* attributes** |
| --- | --- | --- | --- | --- | --- |
| true | false | YES | YES | unknown | NO |
| true | true | YES | YES | YES | YES |
| false | false | **ILLEGAL MODE** | | | |
| false | true | unknown | unknown | YES | YES |

SAX2 по-умолчанию — это первая запись приведенной таблицы. Заметьте, для документов где каждый элемент и атрибут находится в namespace, значения qName (включая namespace префиксы) могут не сообщаться, и xmlns атрибут (декларирующие префиксы) тоже не будут сообщаться, до тех пор пока namespace-prefixes feature не изменится из значения по-умолчанию. Единственный способ избежать этого processor-specific поведения — сделать это изменение (установить namespace-prefixes feature flag в true). Для имен, не находящихся в namespace, только qName значения будут предоставляться парсерами, никакой разницы между парсерами в этой части быть не должно). Также помните, что вы не можете выключить namespace processing выключением только namespaces feature. Так как это переводит процессор в неверное бессмысленное (illegal mode) состояние.

**Пример конфигурации**

Предположим есть документ:

|  |  |
| --- | --- |
| 1  2  3 | <?xml version="1.0"?>  <h:hello xmlns:h="<http://www.greeting.com/ns/>"      id="a1" h:person="David"/> |

Если namespaces — true и namespace-prefixes — false (по-умолчанию), тогда SAX2 XML reader выдаст следующее:

* элемент с Namespace URI «<http://www.greeting.com/ns/&#187>; и localName «hello»
* атрибут без Namespace URI (пустая строка) и qName (и обычно localName) «id»
* атрибут с Namespace URI «<http://www.greeting.com/ns/&#187>; и localName «person».

Если namespaces — true и namespace-prefixes — true, тогда SAX2 XML reader выдаст следующее:

* элемент с Namespace URI «<http://www.greeting.com/ns/&#187>;, localName «hello» и qName «h:hello»
* атрибут без Namespace URI (пустая строка), localName (пустая строка) и qName «xmlns:h»
* атрибут без Namespace URI (пустая строка), и qName (и обычно localName) «id»
* атрибут с Namespace URI «<http://www.greeting.com/ns/&#187>;, localName «person» и qName «h:person».

Если namespaces — false, namespace-prefixes — true, тогда SAX2 XML reader выдаст следующее:

* элемент с qName «h:hello»
* атрибут с qName «xmlns:h»
* атрибут с qName «id»
* атрибут с qName «h:person»

Заметьте, что когда SAX2 сообщает namespace declaration атрибут (типа «xmlns:h»), он делает это в соответствии с Namespaces в рекомендации от W3C: эти атрибуты не находятся ни в каком Namespace. Некоторые W3C спецификации нарушают эту спецификацию и помещают эти описания в <http://www.w3.org/2000/xmlns/> namespace. Если вы работаете с декларациями Namespace, не забывайте про это issue.

которые должны поддерживаться всеми XML парсерами. Разработчики имеют право определять новые feature flags и свойства по мере необходимости, используя URI идентифицирующий их.

Все XML reader-ы должны понимать «<http://xml.org/sax/features/namespaces&#187>; и «<http://xml.org/sax/features/namespace-prefixes&#187>; features (по крайней мере для получения значений, если нет возможности установить их) и поддерживать значение tru для свойства namespaces и false для namespace-prefixes свойства. Требования к реализациям требуют что все SAX2 XML reader-ы должны предоставлять минимальную поддержку Namespace-ов для более высокоуровневых спецификаций, таких как RDF, XSL, XML Schema и XLink. XML reader-ы не обязаны поддерживать или распознавать любые другие feature или свойства.

Для полного списка стандартных SAX2 feature и свойств, смотрите [описание org.xml.sax package.](http://www.saxproject.org/apidoc/org/xml/sax/package-summary.html#package_description)

\*\*\*\*\*\*\*\*\*\*\*\*\*

<http://www.saxproject.org/apidoc/overview-summary.html>

# Осваиваем SAX парсер: Namespaces

### [Галерея](https://dev64.wordpress.com/type/gallery/)

Posted on [Апрель 28, 2012](https://dev64.wordpress.com/2012/04/28/study-sax-parser-namespaces/)

Продолжаю перевод документации с официального сайта: <http://www.saxproject.org>

**Оглавление**

* [SAX парсер: Quick Start](https://dev64.wordpress.com/2012/04/28/learning-sax/)
* [SAX парсер: Features и Properties](https://dev64.wordpress.com/2012/04/28/sax-parser-features-and-properties/)
* [SAX парсер: Namespaces](https://dev64.wordpress.com/2012/04/28/study-sax-parser-namespaces/)
* [JavaDocs](http://www.saxproject.org/apidoc/overview-summary.html)

В SAX2 реализована поддержка XML Namespace, требующаяся для высокоуровневых стандартов таких как XPath (используется XSLT, XPointer, XLink и другими), XML Schema, RDF и другими. Каждая реализация SAX2 XMLReader интерфейса должна поддерживать обработку Namespace-ов по-умолчанию. Дополнительно, многие XML reader-ы позволяют модифицировать обработку Namespace-ов или отключить её.

Обработка Namespace-ов изменяет только именование элементов и атрибутов, а также вносить ограничения на «некоторые другие имена». В XML 1.0, каждый XML элемент имеет единственное имя называемое qName (которое может содержать двоеточия). С Namespace-ами, имена элементов и атрибутов — состаные, состоящие из 2-х частей, такое составное имя иногда называется «Universal» или «Expanded» имя. Имя состоит из URI (подобно именам пактов в Java или Perl) и localName (никогда не содержащее двоеточий).

SAX2 способен поддерживать оба из подходов одновременно. XML документы также могут поддерживать оба подхода. SAX2 XMLReader реализации должны поддерживать Namespaces, когда документы используют их.

**Namespaces в SAX Events**

Поддержка Namespace влияет на ContentHandler и Attributes интерфейсы. Вам нужно обратить основное внимение на поддержку Namespace в startElement() callbacks.

**Имена элементов**

В SAX2 startElement и endElement callback в обработчике контента выглядят так:

public void startElement (String uri, String localName,

String qName, Attributes atts)

throws SAXException;

public void endElement (String uri, String localName, String qName)

throws SAXException;

По-умолчанию, XML reader будет сообщать Namespace URI и localName для каждого элемента, принадлежащего namespace и в start и в end handler-e (обработчике).

Пример:

|  |  |
| --- | --- |
| 1 | <html:hr xmlns:html="<http://www.w3.org/1999/xhtml>"/> |

С дефолтной обработкой SAX2 Namespace, XML reader сообщит о начале и завершении элемента событиями с Namespace URI: <http://www.w3.org/1999/xhtml> и localName hr. Большинство XMLReader реализаций также сообщат оригинальное имя в qName html:hr, однако этот параметр может быть и просто пустой строкой (за исколючением элементов не находящихся в описанном namespace).

**Имена атрибутов**

Для атрибутов, значение именованного атрибута может быть найдено использованием метода getValue или вы можете искать Namespace URI и localName атрибута по его индексу используя getURI и getLocalName методы (обычно когда вы итерируетесь через весь список атрибутов).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | String rdfns = "[http://www.w3.org/1999/02/22-rdf-syntax-ns#](http://www.w3.org/1999/02/22-rdf-syntax-ns)";  String resource = atts.getValue(rdfns, "resource");    for (int i = 0; i < atts.getLength(); i++) {    String uri = atts.getURI(i);    String localName = atts.getLocalName(i);    String value = atts.getValue(i);    /\* ... \*/  } |

Мэппинг префиксов

В дополнения к описанным событиям, SAX2 сообщает границы использования Namespace деклараций используя startPrefixMapping и endPrefixMapping методы, так, что приложения могут определять префиксы в именах атрибутов и символьных данных по необходимости. Callback-и выглядят так:

public void startPrefixMapping (String prefix, String uri)

throws SAXException;

public void endPrefixMapping (String prefix)

throws SAXException;

Для примера выше, XML reader сделает следующий callback вызов сразу перед start-element событием:

startPrefixMapping("html", "http://www.w3.org/1999/xhtml")

XML reader сделает следующий вызов сразу за end-element событием:

endPrefixMapping("html")

Для элементов с множественными описаниями namespace, после startPrefixMapping не обязательно следует соответствующий endPrefixMapping, поскольку эти вызовы могут идти в любом порядке.

**Конфигурация**

Этот раздел применим к SAX2 приложениям с требованиями по сохранению префиксов и их деклараций (preserving prefixes and their declarations). Приложения, которые пишут XML-текст часто попадают в эту категорию, включая множество XML-редакторов.

**Конфигурирование поддержки Namespace-ов**

Feature flag c именем <http://xml.org/sax/features/namespaces> управляет Namespace processing. Когда флаг установлен (по-умолчанию), любой применяемый Namespace URI и localName (для элементов и namespace-ов) должен быть доступен в startElement и endElement callbacks в ContentHandler интерфейсе и в различных методах  
Attributes интерфейса. Кроме того start/endPrefixMapping() события должны выдаваться. Для элементов и атрибутов за пределами namespace-ов, связанный с ними namespace URI будет пустой строкой и параметр qName гарантированно должен быть непустой строкой.

Feature flag с именем <http://xml.org/sax/features/namespace-prefixes> управляет сообщением qNames и Namespace деклараций (xmns атрибутов). Если значение этого флага не изменено на true (его умолчание false), qNames могут быть опционально (!) быть пустыми строками для элементов и атрибутов имеющих связанное с ними namespace URI, xmlns атрибуты не будут рапортоваться. При установке флага в true, эта информация всегда будет выставлена.

Приведенная таблица суммирует взаимодействие упомянутых 2-х features.

| **namespaces** | **namespace-prefixes** | **Namespace names** | **start/end PrefixMapping** | **qNames** | **xmlns\* attributes** |
| --- | --- | --- | --- | --- | --- |
| true | false | YES | YES | unknown | NO |
| true | true | YES | YES | YES | YES |
| false | false | **ILLEGAL MODE** | | | |
| false | true | unknown | unknown | YES | YES |

SAX2 по-умолчанию — это первая запись приведенной таблицы. Заметьте, для документов где каждый элемент и атрибут находится в namespace, значения qName (включая namespace префиксы) могут не сообщаться, и xmlns атрибут (декларирующие префиксы) тоже не будут сообщаться, до тех пор пока namespace-prefixes feature не изменится из значения по-умолчанию. Единственный способ избежать этого processor-specific поведения — сделать это изменение (установить namespace-prefixes feature flag в true). Для имен, не находящихся в namespace, только qName значения будут предоставляться парсерами, никакой разницы между парсерами в этой части быть не должно). Также помните, что вы не можете выключить namespace processing выключением только namespaces feature. Так как это переводит процессор в неверное бессмысленное (illegal mode) состояние.

**Пример конфигурации**

Предположим есть документ:

|  |  |
| --- | --- |
| 1  2  3 | <?xml version="1.0"?>  <h:hello xmlns:h="<http://www.greeting.com/ns/>"      id="a1" h:person="David"/> |

Если namespaces — true и namespace-prefixes — false (по-умолчанию), тогда SAX2 XML reader выдаст следующее:

* элемент с Namespace URI «<http://www.greeting.com/ns/&#187>; и localName «hello»
* атрибут без Namespace URI (пустая строка) и qName (и обычно localName) «id»
* атрибут с Namespace URI «<http://www.greeting.com/ns/&#187>; и localName «person».

Если namespaces — true и namespace-prefixes — true, тогда SAX2 XML reader выдаст следующее:

* элемент с Namespace URI «<http://www.greeting.com/ns/&#187>;, localName «hello» и qName «h:hello»
* атрибут без Namespace URI (пустая строка), localName (пустая строка) и qName «xmlns:h»
* атрибут без Namespace URI (пустая строка), и qName (и обычно localName) «id»
* атрибут с Namespace URI «<http://www.greeting.com/ns/&#187>;, localName «person» и qName «h:person».

Если namespaces — false, namespace-prefixes — true, тогда SAX2 XML reader выдаст следующее:

* элемент с qName «h:hello»
* атрибут с qName «xmlns:h»
* атрибут с qName «id»
* атрибут с qName «h:person»

Заметьте, что когда SAX2 сообщает namespace declaration атрибут (типа «xmlns:h»), он делает это в соответствии с Namespaces в рекомендации от W3C: эти атрибуты не находятся ни в каком Namespace. Некоторые W3C спецификации нарушают эту спецификацию и помещают эти описания в <http://www.w3.org/2000/xmlns/> namespace. Если вы работаете с декларациями Namespace, не забывайте про это issue.