

RELAX NG

A Simpler, Easier XML Schema Language

Ben Galbraith
ben@galbraiths.org

Presentation Outline

- Introduction
- Schema Languages
 - DTD, XML Schema, RELAX NG, Schematron
- RELAX NG in Detail
- Java Support for RELAX NG
 - Jing, MSV, JARV, JAXP

Introduction

- Do you feel that XML and its army of related specifications have grown too complicated?
 - You're not alone!
- RELAX NG and JDOM are alternatives to the W3C's whole-hearted embracing of specification obfuscation and functional confusion

What is a “Schema Language”?

- Defines a set of constraints that an XML document must conform to in order to be **valid**
 - Not to be confused with **well-formed** (i.e., compliant with XML spec.)
- **W3C XML Schema** is one of many XML schema languages
 - Name has caused confusion

Why a Schema Language?

- Allows for interchange of XML documents
- Ensures data integrity
 - Detects bugs in XML generators
- Enables binding of XML to typed objects
 - JAXB
- Eases XML authoring when used with schema-aware editing tools

DTD

- **DTD** (Document Type Definition) was the original XML schema language
 - Subset of SGML's DTD
- Has fallen out of favor
 - Not sufficiently expressive
 - Limited to one namespace
 - Non-XML syntax
 - Etcetera

XML Namespaces

- W3C **Namespaces in XML** standard is a foundation for better schema languages
- XML Namespaces extends XML:
 - To be grammar language agnostic
 - To support multiple grammars in the same document
- XML Namespaces introduces the concept of using URIs to uniquely identify grammars

XML Namespaces

Example:

```
<widgets xmlns="http://unique.url.com"
  xmlns:other="http://other.url.com">
  <widget attribute="value" />
  <other:nugget />
  <bludger xmlns="http://bludgers.com">
    <price>$10.00</price>
  </bludger>
</widgets>
```

W3C XML Schema

- W3C XML Schema is the W3C's blessed successor to DTD
 - About 3 years in the making
 - Finished May 2001
- Inspired by many other schema languages
- Endorsed by many major companies upon release

W3C XML Schema

- Attempt at creating an object-oriented schema language
- Introduces fairly extensive typing system
- Supports XML Namespaces

W3C XML Schema

- Despite widespread endorsement and adoption, critics abound
- James Clark (of XPath and XSLT fame; co-editor of RELAX NG spec.):
 - “Considerable expertise [is required] to be able to understand a W3C XML Schema correctly. There are many cases where you cannot guess what a construct means or where you might guess wrong.”

W3C XML Schema

- James Clark, cont'd:
 - “There are many things about XML Schema that are just plain bizarre.”
 - “[W3C XML Schema] is without doubt the hardest to understand specification that I have ever read ... It is extraordinarily hard for a reader to determine from the Recommendation the meaning of some particular construct they are not sure of.”
 - “There seems to be a tendency for people to suspend their technical judgment when it comes to W3C XML Schema. The attitude seems to be ‘It's a W3C Recommendation; everybody is using it, so we should too, regardless of its technical merits.’”

W3C XML Schema

- Simon St. Laurent (XML luminary):
 - It seems that XML Schema is chock-full of "features" that were seen as "maximizing market acceptance", but painfully short of features which might have produced a flexible, nimble, and coherent specification.

W3C XML Schema

- “XML documents *shall* be easy to create ... XML documents should be human-legible and reasonably clear.”
 - *from the XML Specification (emphasis added)*
 - The W3C XML Schema working group must not have read that part of the spec
- *Using W3C XML Schema is not a fait accompli*

Other Schema Languages

- RELAX NG
- Schematron
- XML-Data / XML-Data-Reduced (XDR)
- SOX

RELAX NG

- RELAX NG is short for “Regular Language for XML Next Generation”
 - Tagline: “Tired of complicated specifications? RELAX!”
 - Pronounced "relaxing"
- James Clark:
 - “RELAX NG was designed to be simple and easy to understand. RELAX NG is simple enough that without even reading the RELAX NG spec, somebody familiar with XML can read a RELAX NG grammar and understand what it means. You can learn to write RELAX NG in 30 minutes by reading the tutorial. RELAX NG is fairly free of surprises. Constructs mean what you would guess they mean.”

RELAX NG: Example

- Given an example XML document:

```
<contents>
  <section name="Section 1">
    <content>
      Hello, world!
    </content>
  </section>
</contents>
```

RELAX NG: Example

- Here's a RELAX NG Schema:

```
<element name="contents"  
  xmlns="http://relaxng.org/ns/structure/1.0">  
  <oneOrMore>  
    <element name="section">  
      <element name="content">  
        <text/>  
      </element>  
      <attribute name="name"/>  
    </element>  
  </oneOrMore>  
</element>
```

RELAX NG: Tutorial

- All RELAX NG schemas must use the RELAX NG namespace:
 - <http://relaxng.org/ns/structure/1.0>
- Omitted for future examples

RELAX NG: Tutorial

- Elements and attributes are defined with `<element>` and `<attribute>`:

```
<element name="contents">
  <element>
    <name>section</name>
    <attribute name="number"/>
  </element>
</element>
```

RELAX NG: Tutorial

- Order of <element>'s is significant; order of <attribute>'s is not
- <interleave> **allows** <element> order to be insignificant:

```
<element name="widget">
  <interleave>
    <element name="cost">
      <text/>
    </element>
    <element name="weight">
      <text/>
    </element>
  </interleave>
</element>
```

RELAX NG: Tutorial

- `<text>` used to indicate any text content (including no text)

```
<element name="contents">
  <element name="section">
    <attribute name="number"/>
    <element name="content">
      <text/>
    </element>
  </element>
</element>
```

- `<attribute>` **implies** `<text>`; `<element>` **does not**

RELAX NG: Tutorial

- Because of `<attribute>`'s default `<text>` behavior, `<element/>` is ambiguous
 - For consistency, it should default to contain `<text>`, but it is logical that it indicates an empty element
- Thus, `<element/>` is disallowed
 - Use `<empty/>` for an empty tag

```
<element name="contents">  
  <empty/>  
</element>
```

RELAX NG: Tutorial

- Often times, choosing between elements and attributes can be difficult:

```
<widget>  
  <manufacturer>  
    Kohsuke  
  </manufacturer>  
</widget>
```

versus:

```
<widget manufacturer="Kohsuke"/>
```

RELAX NG: Tutorial

- With RELAX NG, changing between the two is trivial:

```
<element name="widget">
  <element name="manufacturer">
    <text/>
  </element>
</element>
```

can be changed to:

```
<element name="widget">
  <attribute name="manufacturer">
    <text/>
  </attribute>
</element>
```

RELAX NG: Tutorial

- `<zeroOrMore>`, `<oneOrMore>`, and `<optional>` control occurrence; otherwise, exactly one instance of an `<element>` and `<attribute>` are required where specified

```
<element name="widget">
  <optional>
    <attribute name="manufacturer"/>
  </optional>
  <oneOrMore>
    <element name="warehouse">
      <text/>
    </element>
  </oneOrMore>
  <zeroOrMore>
    <element name="defect">
      <text/>
    </element>
  </zeroOrMore>
</element>
```

RELAX NG: Tutorial

- `<choice>` enables choosing between multiple patterns:

```
<element name="widget">
  <choice>
    <attribute name="manufacturer"/>
    <element name="manufacturer">
      <text/>
    </element>
  </choice>
</element>
```

RELAX NG: Tutorial

- `<group>` composites multiple patterns into one:

```
<element name="contact">
  <choice>
    <element name="name">
      <text/>
    </element>
    <group>
      <element name="firstName">
        <text/>
      </element>
      <element name="lastName">
        <text/>
      </element>
    </group>
  </choice>
</element>
```

RELAX NG: Tutorial

- `<value>` is used to restrict text to a particular value:

```
<element name="friend">
  <attribute name="name">
    <value>Nick</value>
  </attribute>
</element>
```

- Default XML whitespace rules can be disabled with:

```
<value type="string">no whitespace!</value>
```

RELAX NG: Tutorial

- `<value>` is most useful when combined with `<choice>`, creating an enumeration:

```
<element name="swallow">
  <element name="type">
    <choice>
      <value>African</value>
      <value>European</value>
    </choice>
  </element>
</element>
```

RELAX NG: Tutorial

- **Named patterns** and **references** allow for reuse and recursion
 - `<define>` and `<ref>`, respectively
- To use references, schema must be given a starting point with `<grammar>` and `<start>`

RELAX NG: Tutorial

```
<contents>
  <section name="Section 1">
    <section name="Subsection 1">
      <content>
        Hello, world!
      </content>
    </section>
  </section>
  <section name="Section 2">
    <section name="Subsection 2">
      <section name="Subliminalsection 3">
        <content>
          W3C XML Schema got you down? Relax!
        </content>
      </section>
    </section>
  </section>
</contents>
```

RELAX NG: Tutorial

```
<grammar>

  <start>
    <element name="contents">
      <oneOrMore>
        <ref name="Section"/>
      </oneOrMore>
    </element>
  </start>

  <define name="Section">
    <element name="section">
      <zeroOrMore>
        <ref name="Section"/>
      </zeroOrMore>
      <zeroOrMore>
        <element name="content">
          <text/>
        </element>
      </zeroOrMore>
      <attribute name="name"/>
    </element>
  </define>

</grammar>
```

RELAX NG: Tutorial

- **External references** may also be used:

```
<externalRef href="module.rng"/>
```

RELAX NG: Tutorial

- RELAX NG designed to integrate with multiple type systems
 - e.g., XML Schema Datatypes
- The type system must be supported by the RELAX NG validator

RELAX NG: Tutorial

- To enable typing, the `datatypeLibrary` attribute of the schema's root element should be set
 - `<grammar datatypeLibrary="url">`
 - Can be set on any element; value is inherited

RELAX NG: Tutorial

- `<data>` element is used to require a datatype
- `<param>` element can be used to pass parameters to the datatype

```
<element name="age">
  <data type="positiveInteger">
    <param name="maxInclusive">120</param>
  </data>
</element>
```

RELAX vs. W3C Schema

- Demo

RELAX NG and Java

- JAXP API leaves XML validation up to the parser implementation
- No JAXP implementation supports RELAX NG validation
 - Xerces had RELAX NG support under development at one time; no one has stepped up to finish the work
- Many third-party tools exist for validation
 - JING
 - MSV

RELAX NG and Java

- A third-party API, **JARV**, was created to separate validation from parsing

JING

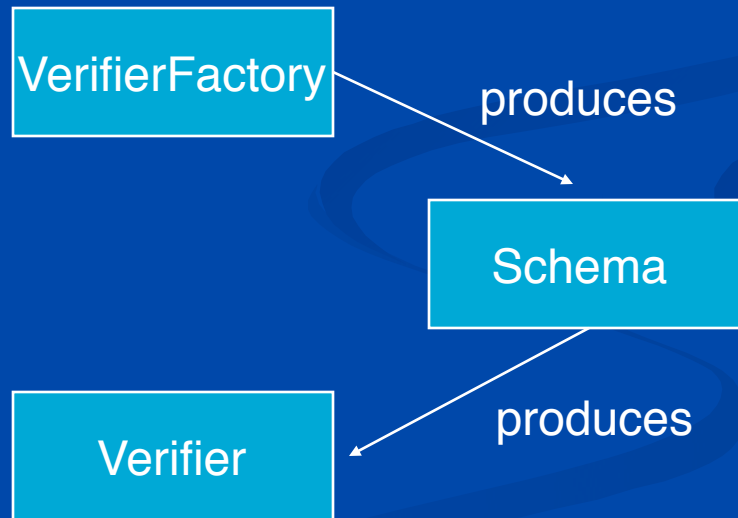
- JING is James Clark's XML validator
- Supports RELAX NG validation
 - Supports W3C XML Schema Part 2
- Wants to support W3C XML Schema and Schematron
- Includes Java API and stand-alone validator
 - Bundles JARV

MSV

- MSV (Multi-Schema Validator) is Sun's validator
- Supports RELAX NG, W3C XML Schema, and DTD
 - Supports W3C XML Schema Part 2
- Includes Java API and stand-alone validator
 - Bundles JARV

JARV

- Separates verification from parser
- Simple API:



JARV

- Supports the schema languages of its implementations (e.g., JING and MSV)
- Validation can be performed in the absence of a parser
- If used with a parser, the parser's built-in validation should be disabled
 - Default behavior with JAXP

JARV Example

```
VerifierFactory factory = VerifierFactory.newInstance(  
    "http://relaxng.org/ns/structure/1.0"  
);
```

```
Schema schema = factory.compileSchema(  
    new File("/source/xml/rng/schema.rng")  
);
```

```
Verifier verifier = schema.newVerifier();
```

```
File xmlFile = new File("/source/xml/rng/example.xml");  
if (verifier.verify(xmlFile)) {  
    System.out.println("The file is valid!");  
} else {  
    System.out.println("The file is invalid!");  
}
```

JAXP

- JAXP 1.3 includes a new Validation API
 - Inspired by JARV
- Demo

Validation Demo



RELAX NG and the World

- emacs RELAX NG extensions
- Oxygen XML editor
- Eclipse plug-in XML Buddy to have forthcoming support
- Official schema language of OpenOffice and KDE Office
- RELAX NG book in the works
 - On-line preview
- More folks starting to give it a look