An Extensible API for Documents with Multiple Annotations

Nils Diewald & Maik Stührenberg

Universität Bielefeld

INSTITUT FÜR DEUTSCHE SPRACHE

Balisage 2013
The Markup Conference

Montréal, Canada, 8/8/13

Overview

Motivation ...

... for an Extensible API ...

... for Documents with Multiple Annotations

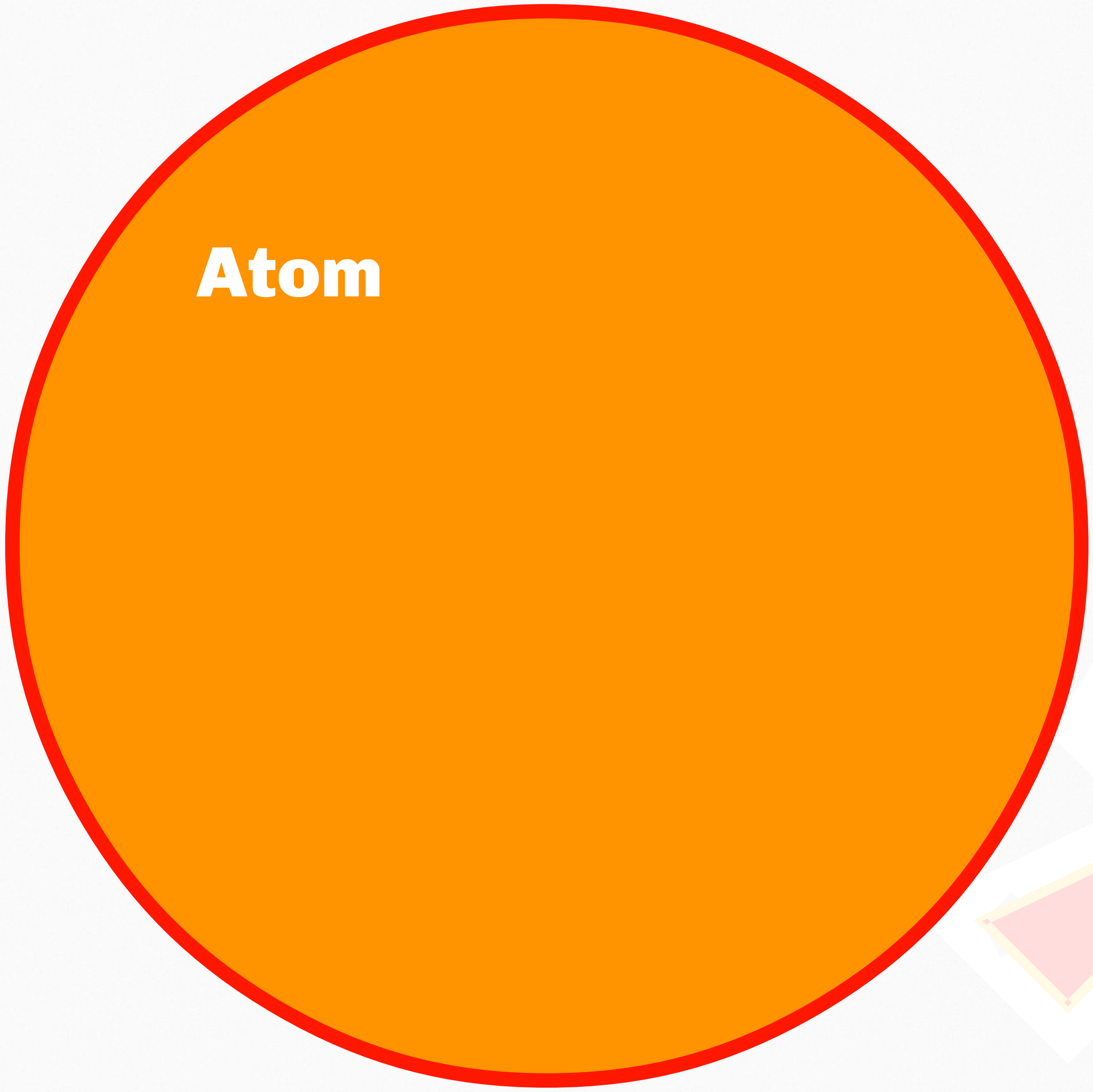
Sojolicious

Initial Motivation » Sojolicious

- A Toolkit for the Federated Social Web
- Support for Ostatus
- Based on Mojolicious (Perl)

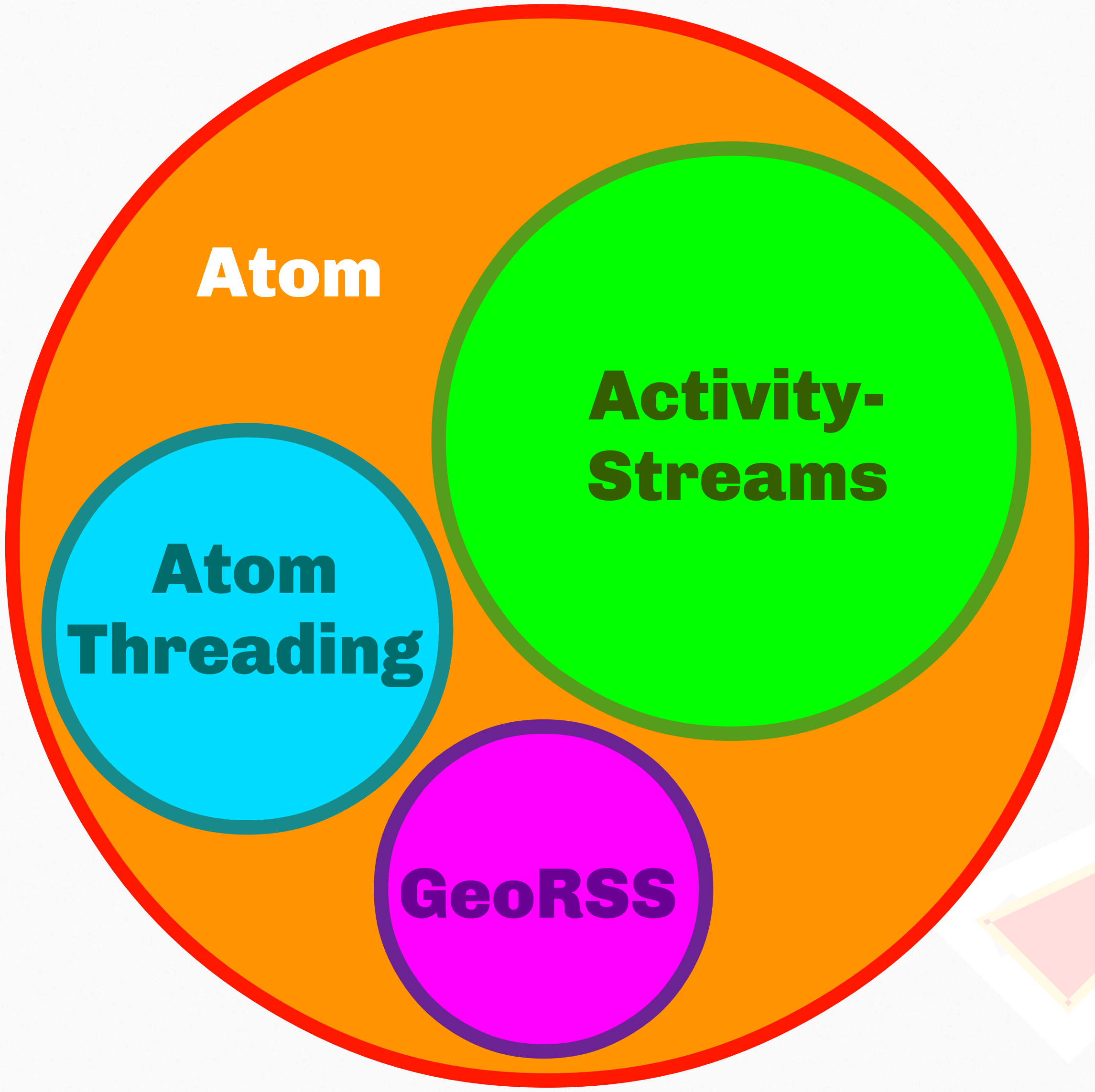
OStatus XML

Initial Motivation » Sojolicious



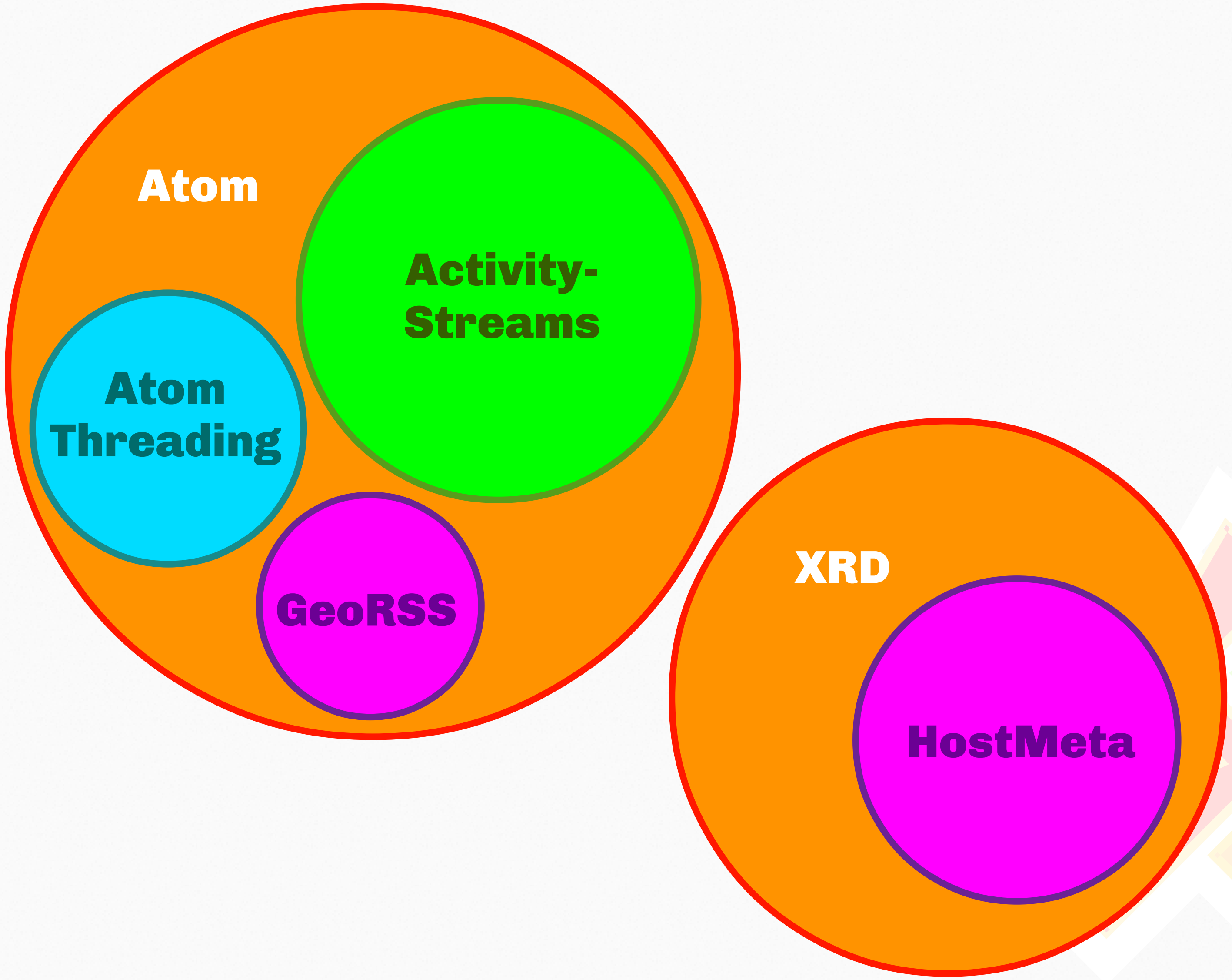
OStatus XML

Initial Motivation » Sojolicious



OStatus XML

Initial Motivation » Sojolicious



ActivityStreams

Initial Motivation » Sojolicious

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:activity="http://activitystrea.ms/schema/1.0/"
      xmlns:thr="http://purl.org/syndication/thread/1.0">
  <entry xml:id="answer-1">
    <id>answer-1</id>
    <title type="xhtml">
      <div xmlns="http://www.w3.org/1999/xhtml">Nils answers to Maik</div>
    </title>
    <published>2013-07-06T13:56:49Z</published>
    <author>
      <name>Nils</name>
      <activity:object-type>
        http://activitystrea.ms/schema/1.0/person
      </activity:object-type>
    </author>
    <activity:verb>
      http://activitystrea.ms/schema/1.0/answers
    </activity:verb>
    <activity:object>
      <activity:object-type>
        http://activitystrea.ms/schema/1.0/person
      </activity:object-type>
      <name>Maik</name>
    </activity:object>
    <thr:in-reply-to ref="http://sojolicio.us/blog/2" />
    <link href="http://sojolicio.us/blog/1/replies"
          rel="replies"
          thr:count="7"
          thr:updated="2013-07-06T13:56:49Z"
          type="application/atom+xml" />
  </entry>
</feed>
```

ActivityStreams

Initial Motivation » Sojolicious

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:activity="http://activitystrea.ms/schema/1.0/"
      xmlns:thr="http://purl.org/syndication/thread/1.0">
  <entry xml:id="answer-1">
    <id>answer-1</id>
    <title type="xhtml">
```

```
<feed xmlns="http://www.w3.org/2005/Atom"

xmlns:activity="http://activitystrea.ms/schema/1.0/"

xmlns:thr="http://purl.org/syndication/thread/1.0">
  <entry xml:id="answer-1">
    <title type="xhtml">
      <div xmlns="http://www.w3.org/1999/xhtml">
        Nils answers to Maik</div>
    </title>
    <!-- ... -->
    <activity:verb>
      http://activitystrea.ms/schema/1.0/answers
    </activity:verb>
    <!-- ... -->
    <thr:in-reply-to ref="http://sojolicio.us/blog/2" />
```

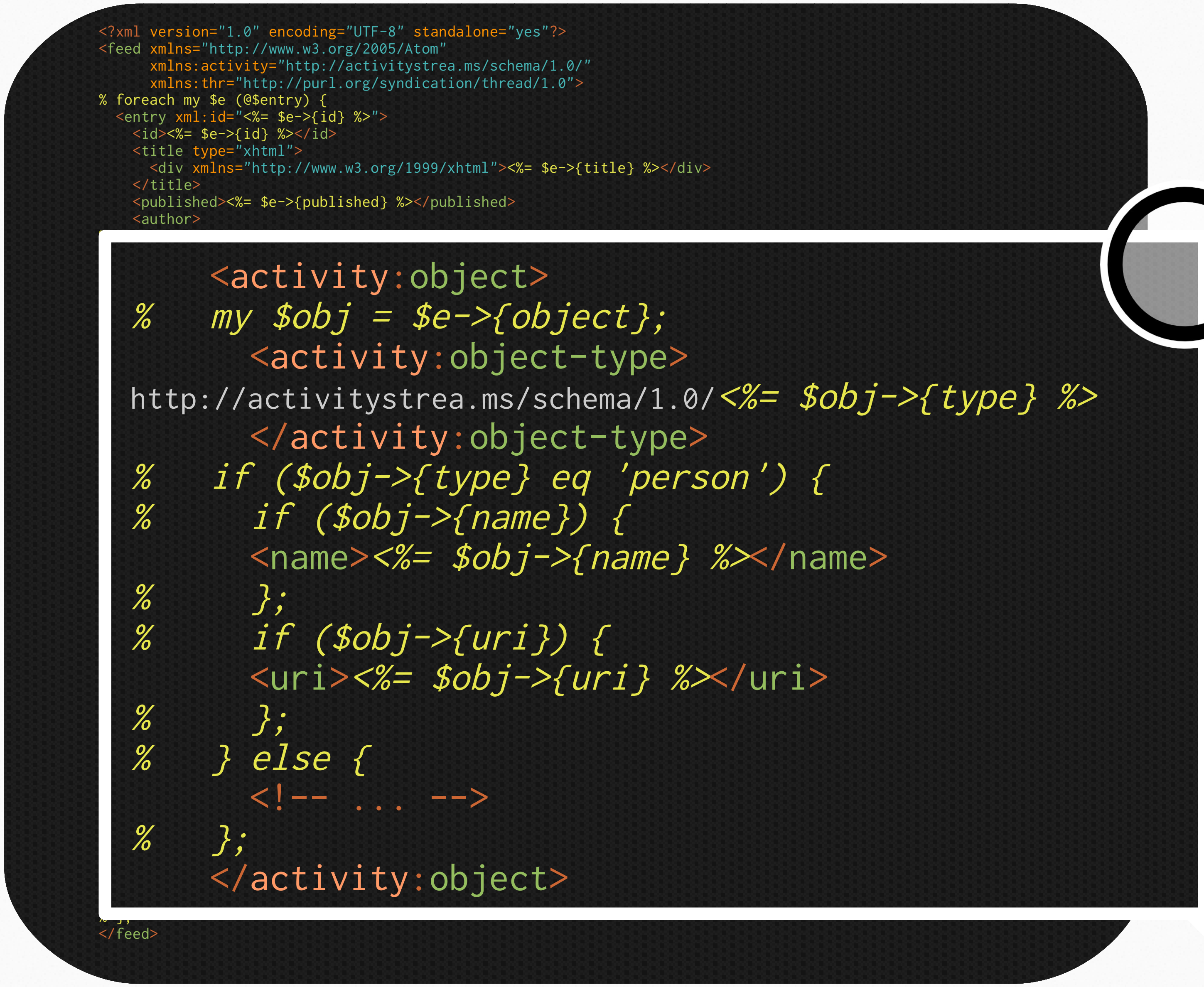
Preferred Way … ?

Initial Motivation » Templating Languages

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:activity="http://activitystrea.ms/schema/1.0/"
      xmlns:thr="http://purl.org/syndication/thread/1.0">
% foreach my $e (@$entry) {
  <entry xml:id="<%= $e->{id} %>">
    <id><%= $e->{id} %></id>
    <title type="xhtml">
      <div xmlns="http://www.w3.org/1999/xhtml"><%= $e->{title} %></div>
    </title>
    <published><%= $e->{published} %></published>
    <author>
%   my $author = $e->{author};
%   if ($author->{name}) {
      <name><%= $author->{name} %></name>
%   };
%   if ($author->{uri}) {
      <uri><%= $author->{uri} %></uri>
%   }
      <activity:object-type>person</activity:object-type>
    </author>
    <activity:verb>http://activitystrea.ms/schema/1.0/<%= $e->{verb} %></activity:verb>
    <activity:object>
%   my $obj = $e->{object};
      <activity:object-type>http://activitystrea.ms/schema/1.0/<%= $obj->{type} %></activity:object-type>
%   if ($obj->{type} eq 'person') {
%     if ($obj->{name}) {
      <name><%= $obj->{name} %></name>
%     };
%     if ($obj->{uri}) {
      <uri><%= $obj->{uri} %></uri>
%     };
%   } else {
      <!-- ... -->
%   };
    </activity:object>
%   if ($e->{'in-reply-to'}) {
    <thr:in-reply-to ref="<%= $e->{'in-reply-to'} %>" />
%   };
%   if ($e->{replies}) {
%   my $replies = $e->{replies};
    <link href="<%= $replies->{uri} %>"
          rel="replies"
%     if ($replies->{count}) {
          thr:count="<%= $replies->{count} %>"
%     };
%     if ($replies->{updated}) {
          thr:updated="<%= $replies->{updated} %>"
%     };
          type="application/atom+xml" />
%   };
  </entry>
% };
</feed>
```

Preferred Way … ?

Initial Motivation » Templating Languages



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:activity="http://activitystrea.ms/schema/1.0/"
      xmlns:thr="http://purl.org/syndication/thread/1.0">
% foreach my $e (@$entry) {
  <entry xml:id="<%= $e->{id} %>">
    <id><%= $e->{id} %></id>
    <title type="xhtml">
      <div xmlns="http://www.w3.org/1999/xhtml"><%= $e->{title} %></div>
    </title>
    <published><%= $e->{published} %></published>
    <author>
```

```
    <activity:object>
%   my $obj = $e->{object};
      <activity:object-type>
http://activitystrea.ms/schema/1.0/<%= $obj->{type} %>
      </activity:object-type>
%   if ($obj->{type} eq 'person') {
%     if ($obj->{name}) {
      <name><%= $obj->{name} %></name>
%     };
%     if ($obj->{uri}) {
      <uri><%= $obj->{uri} %></uri>
%     };
%   } else {
      <!-- ... -->
%   };
    </activity:object>
```

```
% };
</feed>
```

Preferred Way … ?

Initial Motivation » Drawbacks

- Verbose
- Not easily extensible
- Not easily reusable
- No Fun!

Requirements

Initial Motivation » Requirements

- Simple to use

Requirements

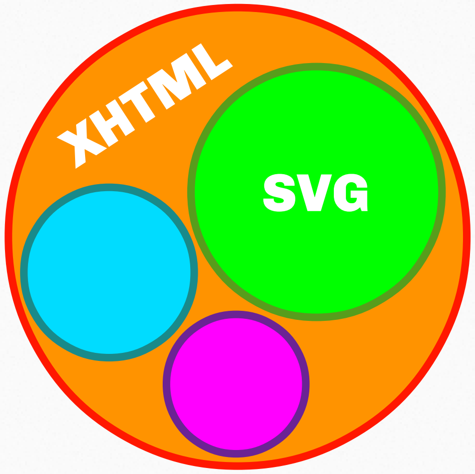
Initial Motivation » Requirements

- Simple to use
- Extensible

Requirements

Initial Motivation » Requirements

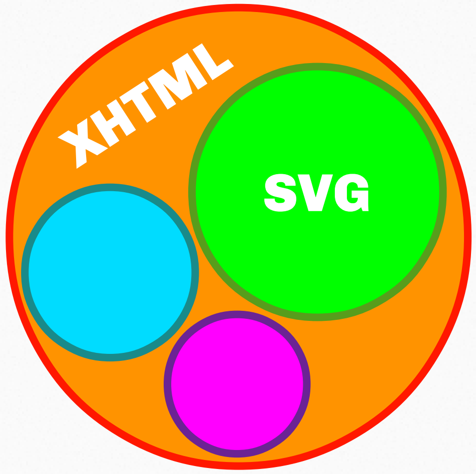
- Simple to use
- Extensible



Requirements

Initial Motivation » Requirements

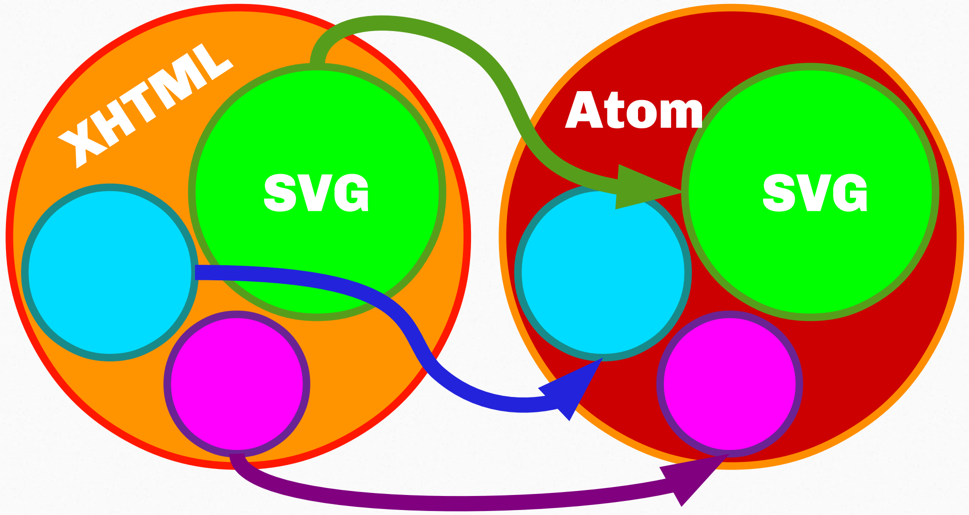
- Simple to use
- Extensible
- Reusable



Requirements

Initial Motivation » Requirements

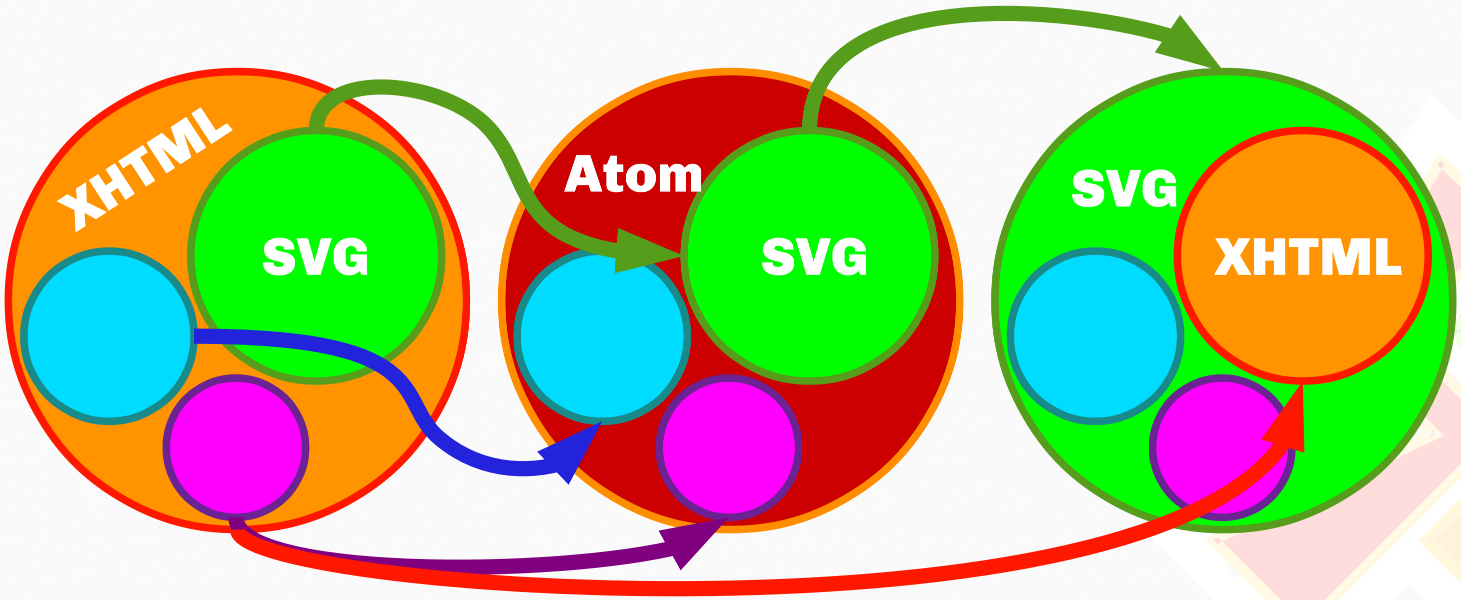
- Simple to use
- Extensible
- Reusable



Requirements

Initial Motivation » Requirements

- Simple to use
- Extensible
- Reusable



Requirements

Initial Motivation » Requirements

- Simple to use
- Extensible
- Reusable
- Fun!

Requirements

Initial Motivation » Requirements

- Simple to use
- Extensible
- Reusable
- Fun!

```
use XML::Loy;
```

Mojo::DOM

XML::Loy » Foundation

- Based on Mojo::DOM
    - Minimalistic HTML5/XML DOM parser
    - Support for CSS Selectors

Mojo::DOM

XML::Loy » Foundation

- Based on Mojo::DOM
  - Minimalistic HTML5/XML DOM parser
  - Support for CSS Selectors

```perl
#!/usr/bin/env perl
use feature ':5.10';
use Mojo::DOM;

my $dom = Mojo::DOM->new(<<XML);
<div id="section">
  <p id="para1">Hello</p>
  <p id="para2">World</p>
</div>
XML

say $dom->at('#para1')->text;
# Hello
```

Mojo::DOM

XML::Loy » Foundation

- Based on Mojo::DOM
  - Minimalistic HTML5/XML DOM parser
  - Support for CSS Selectors

```
$dom->find('p[id]')->each(
  sub {
    say $_->text;
  });
# Hello
# World

$dom->at('p:nth-child(2)')->remove;

say $dom->to_xml;
# <div id="section">
#   <p id="para1">Hello</p>
#
# </div>
```

```
<div id="section">
  <p id="para1">Hello</p>
  <p id="para2">World</p>
</div>
```

XML::Loy

XML::Loy » Document Creation

```
use XML::Loy;

my $doc = XML::Loy->new('document');
```

XML::Loy

XML::Loy » Document Creation

```
use XML::Loy;

my $doc = XML::Loy->new('document');
$doc->set(title => 'My Title');
$doc->set(title => 'My New Title');
```

XML::Loy

XML::Loy » Document Creation

```perl
use XML::Loy;

my $doc = XML::Loy->new('document');
$doc->set(title => 'My Title');
$doc->set(title => 'My New Title');
$doc->add(paragraph =>
  { id => 'p-1' } =>
    'First Paragraph');
$doc->add(paragraph =>
  { id => 'p-2' } =>
    'Second Paragraph');
```

XML::Loy

XML::Loy » Document Creation

```perl
use XML::Loy;

my $doc = XML::Loy->new('document');
$doc->set(title => 'My Title');
$doc->set(title => 'My New Title');
$doc->add(paragraph =>
  { id => 'p-1' } =>
    'First Paragraph');
$doc->add(paragraph =>
  { id => 'p-2' } =>
    'Second Paragraph');

print $doc->to_pretty_xml;
```

XML::Loy

XML::Loy » Document Result

```
<?xml version="1.0"
      encoding="UTF-8"
      standalone="yes"?>
<document>

  <title>My New Title</title>

  <paragraph id="p-1">
    First Paragraph
  </paragraph>

  <paragraph id="p-2">
    Second Paragraph
  </paragraph>
</document>
```

XML::Loy

XML::Loy » Document Result

```
<?xml version="1.0"
      encoding="UTF-8"
      standalone="yes"?>
<document>

  <title>My New Title</title>

  <paragraph id="p-1">
    First Paragraph
  </paragraph>

  <paragraph id="p-2">
    Second Paragraph
  </paragraph>
</document>
```

Extensions

XML::Loy » Extension Creation

```
package XML::Loy::Example::Morphemes;

use XML::Loy with => (
  namespace =>
    'http://www.xstandoff.net/morphemes',
  prefix => 'morph'
);
```

Extensions

XML::Loy » Extension Creation

```perl
package XML::Loy::Example::Morphemes;

use XML::Loy with => (
  namespace =>
    'http://www.xstandoff.net/morphemes',
  prefix => 'morph'
);

# Add morphemes root
sub morphemes {
  my $self = shift;
  return $self->add(morphemes => @_);
};
```

Extensions

XML::Loy » Extension Creation

```
package XML::Loy::Example::Morphemes;

use XML::Loy with => ( ... );

sub morphemes { ... };

# Add morphemes
sub morpheme {
  my $self = shift;
  return unless $self->type =~
    /^(?:morph:)?morphemes$/;
  return $self->add(morpheme => @_);
};
```

Extensions

XML::Loy » Extension Use

```
use XML::Loy::Example::Morphemes;
```

Extensions

XML::Loy » Extension Use

```
use XML::Loy::Example::Morphemes;

my $doc = XML::Loy::Example::Morphemes
  ->new('document');
```

Extensions

XML::Loy » Extension Use

```
use XML::Loy::Example::Morphemes;

my $doc = XML::Loy::Example::Morphemes
  ->new('document');

my $m = $doc->morphemes;
```

Extensions

XML::Loy » Extension Use

```
use XML::Loy::Example::Morphemes;

my $doc = XML::Loy::Example::Morphemes
  ->new('document');

my $m = $doc->morphemes;

$m->morpheme('The');
$m->morpheme('sun');
$m->morpheme('shine');
$m->morpheme('s');
$m->morpheme('bright');
$m->morpheme('er');
```

Extensions

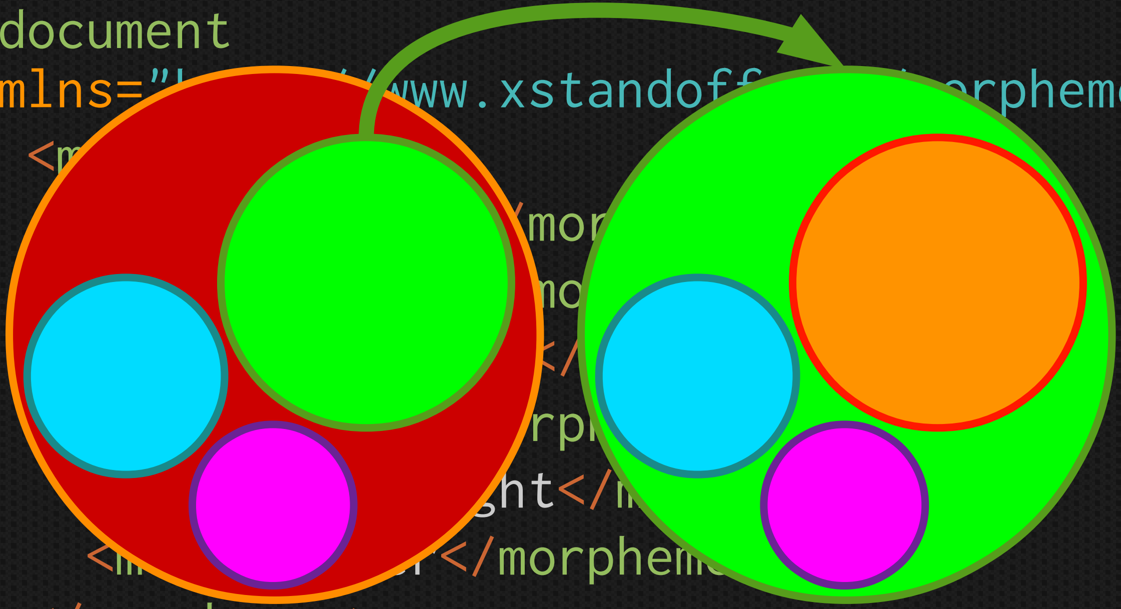
XML::Loy » Extension Use Result

```
<?xml version="1.0"
      encoding="UTF-8"
      standalone="yes"?>
<document
xmlns="http://www.xstandoff.net/morphemes">
  <morphemes>
    <morpheme>The</morpheme>
    <morpheme>sun</morpheme>
    <morpheme>shine</morpheme>
    <morpheme>s</morpheme>
    <morpheme>bright</morpheme>
    <morpheme>er</morpheme>
  </morphemes>
</document>
```

Extensions

XML::Loy » Extension Use Result

```
<?xml version="1.0"
      encoding="UTF-8"
      standalone="yes"?>
<document
xmlns="h        //www.xstandoff        orphemes">
  <m
                  /mor
                   mo
                   </
                   rph
                ght</m
    <m         r</morpheme
  </morphemes>
</document>
```



Extensions

XML::Loy » Extension Use as Extension

```
use XML::Loy;

my $doc = XML::Loy->new(<<'XML');
<?xml version="1.0" encoding="UTF-8"
      standalone="yes"?>
<html>
  <head>
    <title>The sun</title>
  </head>
  <body />
</html>
XML
```

Extensions

XML::Loy » Extension Use as Extension

```
use XML::Loy;

my $doc = XML::Loy->new(<<'XML');
<?xml version="1.0" encoding="UTF-8"
      standalone="yes"?>
<html>
  <head>
    <title>The sun</title>
  </head>
  <body />
</html>
XML

$doc->extension(-Example::Morphemes);
```

Extensions

XML::Loy » Extension Use as Extension

```
my $p = $doc->at('body')
        ->add(p => 'The sun shines');
my $m = $p->morphemes;
$m->morpheme('bright');
$m->morpheme('er');
```

Extensions

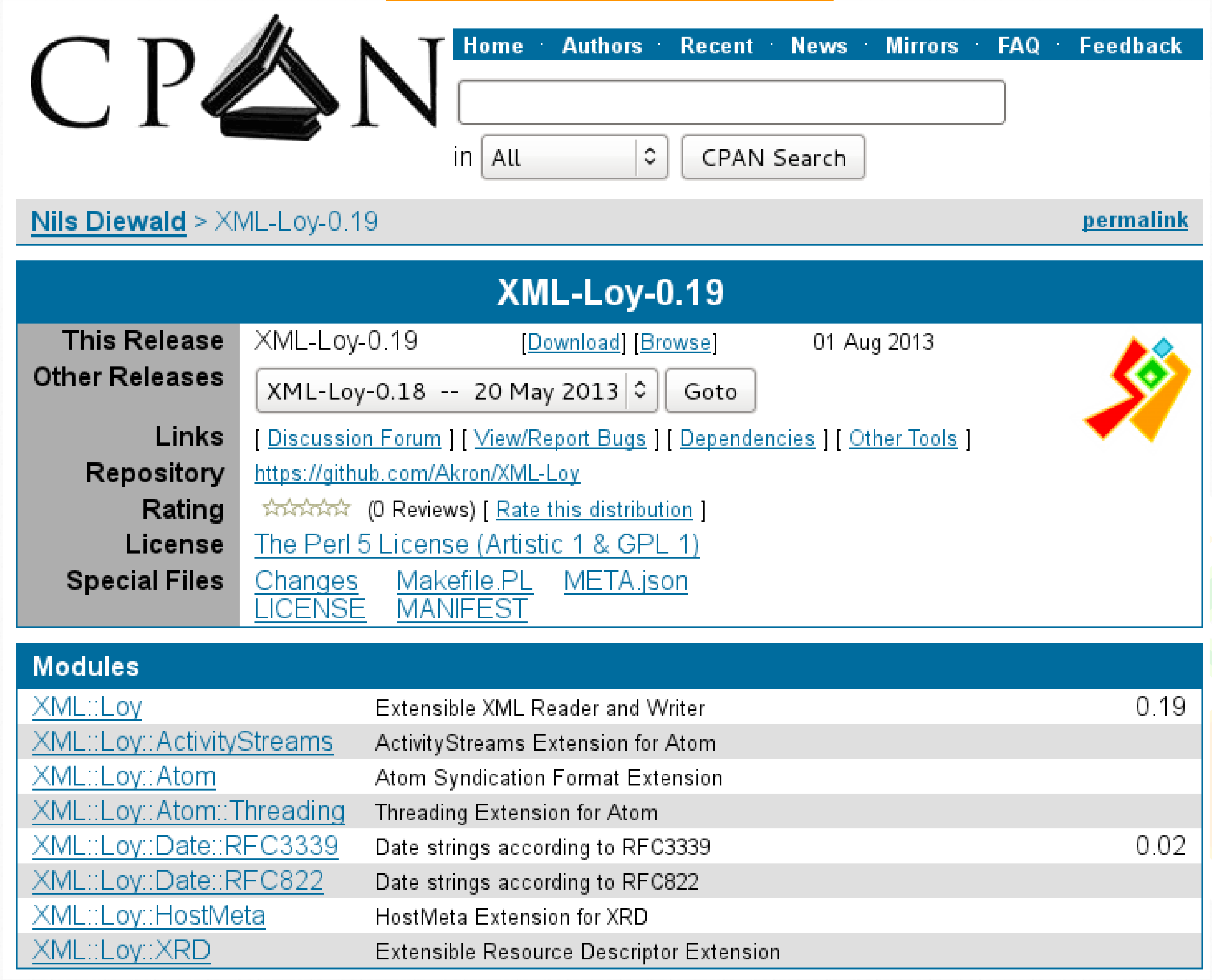
XML::Loy » Extension Use as Extension

```
my $p = $doc->at('body')
        ->add(p => 'The sun shines');
my $m = $p->morphemes;
$m->morpheme('bright');
$m->morpheme('er');
```

```
<?xml version="1.0" ... ?>
<html xmlns:morph="http://.../morphemes">
  <head><title>The sun</title></head>
  <body>
    <p>The sun shines
      <morph:morphemes>
        <morph:morpheme>bright</morph:morpheme>
        <morph:morpheme>er</morph:morpheme>
      </morph:morphemes>
    </p></body></html>
```

XML::Loy

http://search.cpan.org/~akron/XML-Loy/



Nils Diewald > XML-Loy-0.19 permalink

XML-Loy-0.19

This Release	XML-Loy-0.19 [Download] [Browse] 01 Aug 2013
Other Releases	XML-Loy-0.18 -- 20 May 2013 Goto
Links	[Discussion Forum] [View/Report Bugs] [Dependencies] [Other Tools]
Repository	https://github.com/Akron/XML-Loy
Rating	☆☆☆☆☆ (0 Reviews) [Rate this distribution]
License	The Perl 5 License (Artistic 1 & GPL 1)
Special Files	Changes Makefile.PL META.json LICENSE MANIFEST

Modules

XML::Loy	Extensible XML Reader and Writer	0.19
XML::Loy::ActivityStreams	ActivityStreams Extension for Atom	
XML::Loy::Atom	Atom Syndication Format Extension	
XML::Loy::Atom::Threading	Threading Extension for Atom	
XML::Loy::Date::RFC3339	Date strings according to RFC3339	0.02
XML::Loy::Date::RFC822	Date strings according to RFC822	
XML::Loy::HostMeta	HostMeta Extension for XRD	
XML::Loy::XRD	Extensible Resource Descriptor Extension	

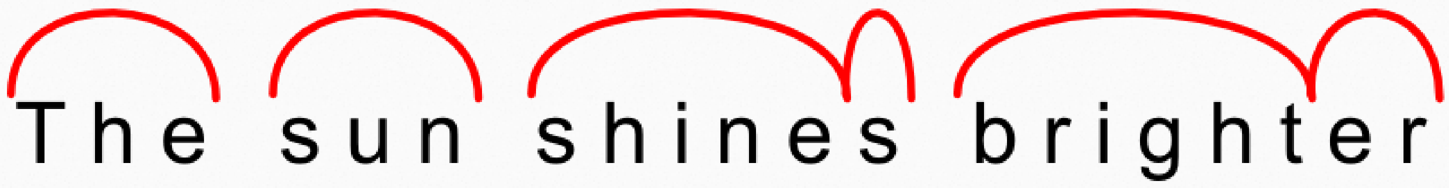
XStandoff

XML::Loy::XStandoff » Challenge

The sun shines brighter

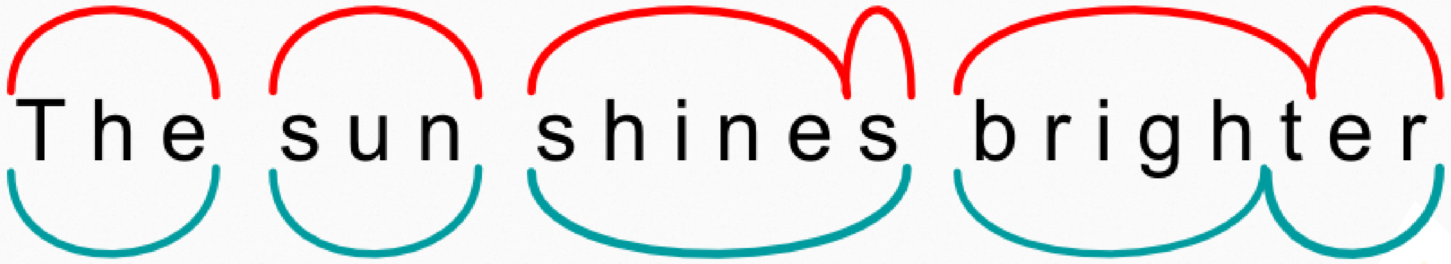
XStandoff

XML::Loy::XStandoff » Challenge



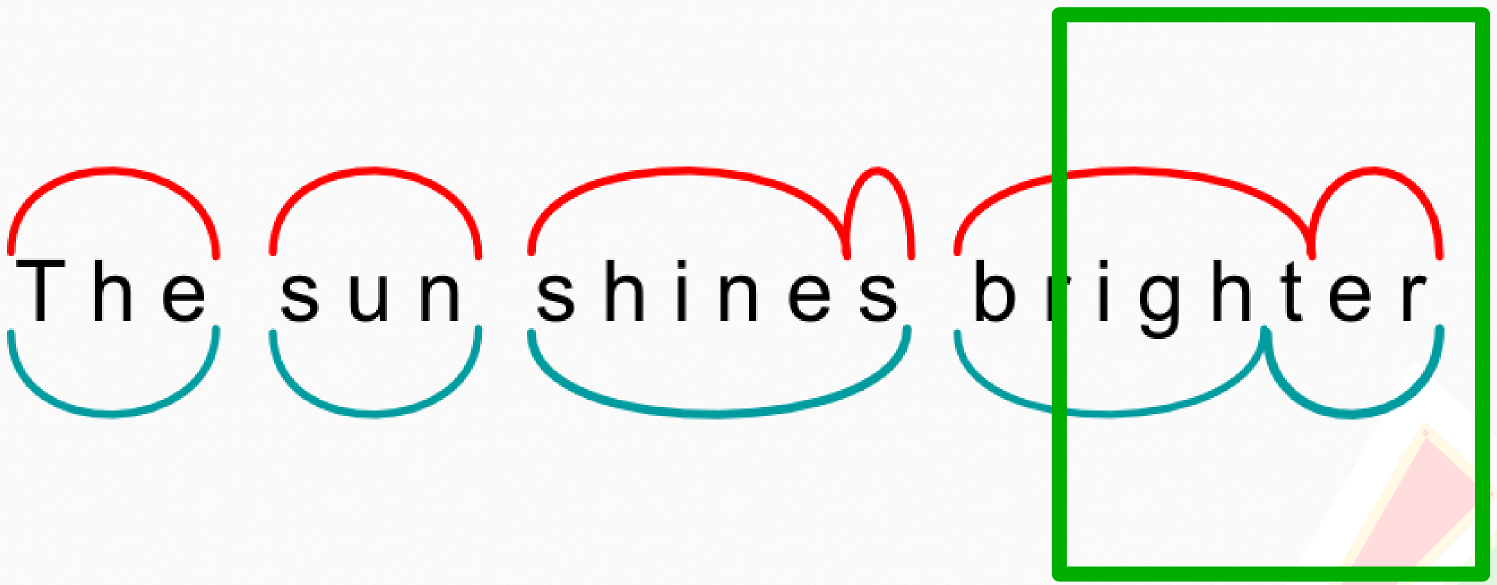
XStandoff

XML::Loy::XStandoff » Challenge



XStandoff

XML::Loy::XStandoff » Challenge



XStandoff

XML::Loy::XStandoff » Challenge

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsf:corpusData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.xstandoff.net/2009/xstandoff/1.1 xsf2_1.1.xsd"
   xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
   xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" xml:id="c1" xsfVersion="2.0">
   <xsf:primaryData start="0" end="24" xml:lang="en" xml:space="preserve" unit="chars">
      <textualContent>The sun shines brighter.</textualContent>
   </xsf:primaryData>
   <xsf:segmentation>
      <xsf:segment xml:id="seg1" type="char" start="0" end="24"/>
      <xsf:segment xml:id="seg2" type="char" start="0" end="3"/>
      <xsf:segment xml:id="seg3" type="char" start="4" end="7"/>
      <xsf:segment xml:id="seg4" type="char" start="8" end="14"/>
      <xsf:segment xml:id="seg5" type="char" start="8" end="13"/>
      <xsf:segment xml:id="seg6" type="char" start="13" end="14"/>
      <xsf:segment xml:id="seg7" type="char" start="15" end="21"/>
      <xsf:segment xml:id="seg8" type="char" start="15" end="20"/>
      <xsf:segment xml:id="seg9" type="char" start="20" end="23"/>
      <xsf:segment xml:id="seg10" type="char" start="21" end="23"/>
   </xsf:segmentation>
   <xsf:annotation>
      <xsf:level xml:id="l_morph">
         <xsf:layer xmlns:morph="http://www.xstandoff.net/morphemes"
            xsi:schemaLocation="http://www.xstandoff.net/morphemes morphemes.xsd">
            <morph:morphemes xsf:segment="seg1">
               <morph:morpheme xsf:segment="seg2"/>
               <morph:morpheme xsf:segment="seg3"/>
               <morph:morpheme xsf:segment="seg5"/>
               <morph:morpheme xsf:segment="seg6"/>
               <morph:morpheme xsf:segment="seg7"/>
               <morph:morpheme xsf:segment="seg10"/>
            </morph:morphemes>
         </xsf:layer>
      </xsf:level>
      <xsf:level xml:id="l_syll">
         <xsf:layer xmlns:syll="http://www.xstandoff.net/syllables"
            xsi:schemaLocation="http://www.xstandoff.net/syllables syllables.xsd">
            <syll:syllables xsf:segment="seg1">
               <syll:syllable xsf:segment="seg2"/>
               <syll:syllable xsf:segment="seg3"/>
               <syll:syllable xsf:segment="seg4"/>
               <syll:syllable xsf:segment="seg8"/>
               <syll:syllable xsf:segment="seg9"/>
            </syll:syllables>
         </xsf:layer>
      </xsf:level>
   </xsf:annotation>
</xsf:corpusData>
```

XStandoff

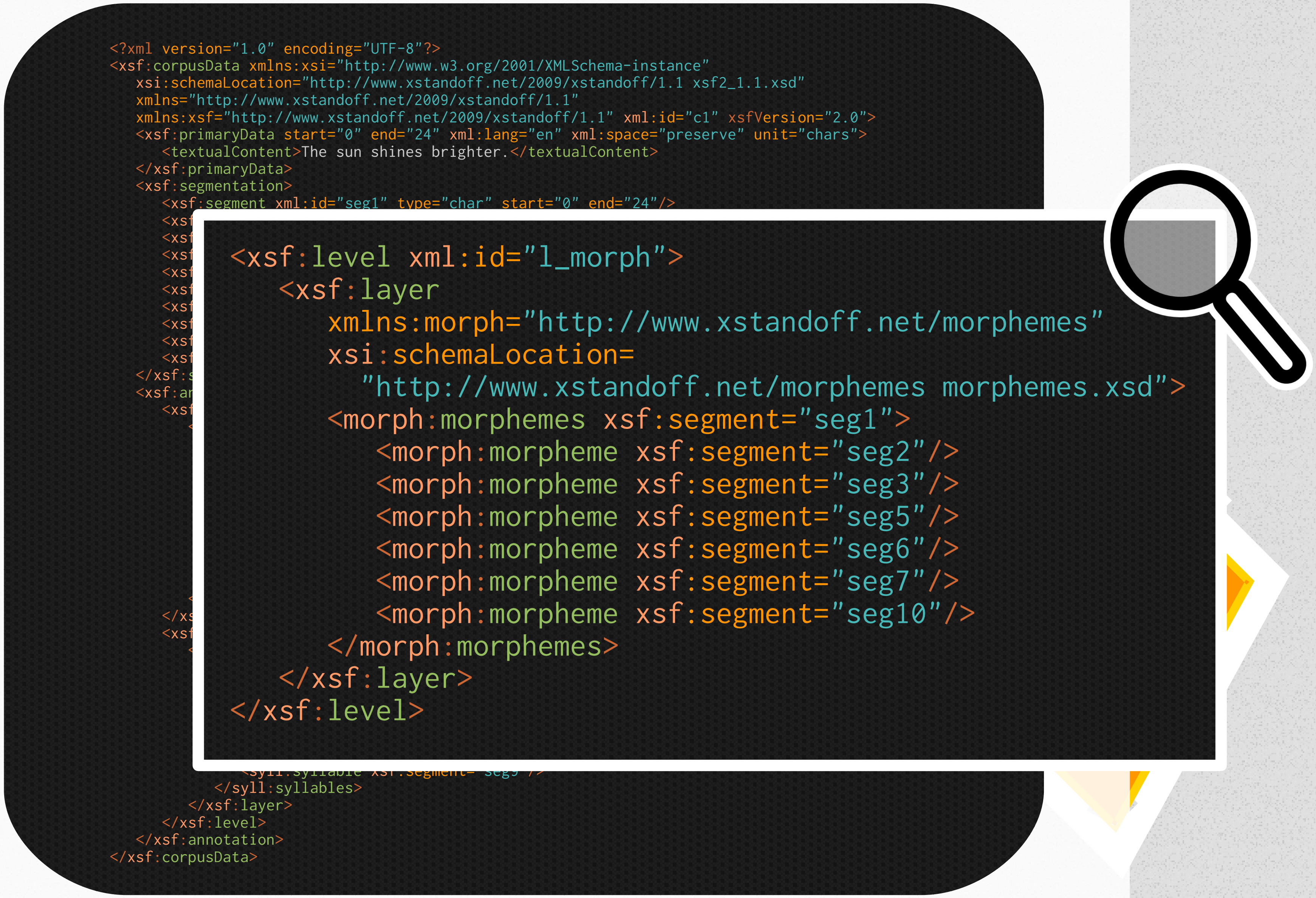
XML::Loy::XStandoff » Challenge

```
<?xml version="1.0" encoding="UTF-8"?>
<xsf:corpusData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.xstandoff.net/2009/xstandoff/1.1 xsf2_1.1.xsd"
   xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
   xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" xml:id="c1" xsfVersion="2.0">
   <xsf:primaryData start="0" end="24" xml:lang="en" xml:space="preserve" unit="chars">
      <textualContent>The sun shines brighter.</textualContent>
   </xsf:primaryData>
   <xsf:segmentation>
      <xsf:segment xml:id="seg1" type="char" start="0" end="24"/>
      <xsf:segment xm
      <xsf:segment xm
      <xsf:segment xm
      <xsf:segment xm
      <xsf:segment xm
      <xsf:segment xm
      <xsf:segment xm
      <xsf:segment xm
      <xsf:segment xm
   </xsf:segmentation
   <xsf:annotation>
      <xsf:level xml:
         <xsf:layer x
            xsi:schem
            <morph:mo
               <morph
               <morph
               <morph
               <morph
               <morph:morpheme xsf:segment="seg7"/>
               <morph:morpheme xsf:segment="seg10"/>
            </morph:morphemes>
         </xsf:layer>
      </xsf:level>
      <xsf:level xml:id="l_syll">
         <xsf:layer xmlns:syll="http://www.xstandoff.net/syllables"
            xsi:schemaLocation="http://www.xstandoff.net/syllables syllables.xsd">
            <syll:syllables xsf:segment="seg1">
               <syll:syllable xsf:segment="seg2"/>
               <syll:syllable xsf:segment="seg3"/>
               <syll:syllable xsf:segment="seg4"/>
               <syll:syllable xsf:segment="seg8"/>
               <syll:syllable xsf:segment="seg9"/>
            </syll:syllables>
         </xsf:layer>
      </xsf:level>
   </xsf:annotation>
</xsf:corpusData>
```

```
<xsf:primaryData
     start="0" end="24" xml:lang="en"
     xml:space="preserve" unit="chars">
   <textualContent>
     The sun shines brighter.
   </textualContent>
</xsf:primaryData>
```

XStandoff

XML::Loy::XStandoff » Challenge



```
<?xml version="1.0" encoding="UTF-8"?>
<xsf:corpusData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.xstandoff.net/2009/xstandoff/1.1 xsf2_1.1.xsd"
   xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
   xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" xml:id="c1" xsfVersion="2.0">
   <xsf:primaryData start="0" end="24" xml:lang="en" xml:space="preserve" unit="chars">
      <textualContent>The sun shines brighter.</textualContent>
   </xsf:primaryData>
   <xsf:segmentation>
      <xsf:segment xml:id="seg1" type="char" start="0" end="24"/>
```

```
<xsf:level xml:id="l_morph">
   <xsf:layer
      xmlns:morph="http://www.xstandoff.net/morphemes"
      xsi:schemaLocation=
        "http://www.xstandoff.net/morphemes morphemes.xsd">
      <morph:morphemes xsf:segment="seg1">
         <morph:morpheme xsf:segment="seg2"/>
         <morph:morpheme xsf:segment="seg3"/>
         <morph:morpheme xsf:segment="seg5"/>
         <morph:morpheme xsf:segment="seg6"/>
         <morph:morpheme xsf:segment="seg7"/>
         <morph:morpheme xsf:segment="seg10"/>
      </morph:morphemes>
   </xsf:layer>
</xsf:level>
```

```
         </syll:syllables>
      </xsf:layer>
   </xsf:level>
  </xsf:annotation>
</xsf:corpusData>
```

XStandoff

XML::Loy::XStandoff » Challenge

```
<?xml version="1.0" encoding="UTF-8"?>
<xsf:corpusData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.xstandoff.net/2009/xstandoff/1.1 xsf2_1.1.xsd"
   xmlns="http://www.xstandoff.net/2009/xstandoff/1.1"
   xmlns:xsf="http://www.xstandoff.net/2009/xstandoff/1.1" xml:id="c1" xsfVersion="2.0">
   <xsf:primaryData start="0" end="24" xml:lang="en" xml:space="preserve" unit="chars">
      <textualContent>The sun shines brighter.</textualContent>
   </xsf:primaryData>
   <xsf:segmentation>
      <xsf:segment xml:id="seg1" type="char" start="0" end="24"/>
```

```
<xsf:segmentation>
   <xsf:segment xml:id="seg1" type="char" start="0" end="24"/>
   <xsf:segment xml:id="seg2" type="char" start="0" end="3"/>
   <xsf:segment xml:id="seg3" type="char" start="4" end="7"/>
   <xsf:segment xml:id="seg4" type="char" start="8" end="14"/>
   <xsf:segment xml:id="seg5" type="char" start="8" end="13"/>
   <xsf:segment xml:id="seg6" type="char" start="13" end="14"/>
   <xsf:segment xml:id="seg7" type="char" start="15" end="21"/>
   <xsf:segment xml:id="seg8" type="char" start="15" end="20"/>
   <xsf:segment xml:id="seg9" type="char" start="20" end="23"/>
   <xsf:segment xml:id="seg10" type="char" start="21" end="23"/>
</xsf:segmentation>
```

```
               <syll:syllables xsf:segment="seg1">
                  <syll:syllable xsf:segment="seg2"/>
                  <syll:syllable xsf:segment="seg3"/>
                  <syll:syllable xsf:segment="seg4"/>
                  <syll:syllable xsf:segment="seg8"/>
                  <syll:syllable xsf:segment="seg9"/>
               </syll:syllables>
            </xsf:layer>
         </xsf:level>
      </xsf:annotation>
</xsf:corpusData>
```

XML::Loy::XStandoff

XML::Loy::XStandoff » Code Example

```
use XML::Loy::XStandoff;

# Create new corpusData
my $cd = XML::Loy::XStandoff->new('corpusData');
```

XML::Loy::XStandoff

XML::Loy::XStandoff » Code Example

```perl
use XML::Loy::XStandoff;

# Create new corpusData
my $cd = XML::Loy::XStandoff->new('corpusData');

# Set textual content embedded
$cd->textual_content('The sun shines brighter');
```

XML::Loy::XStandoff

XML::Loy::XStandoff » Code Example

```perl
use XML::Loy::XStandoff;

# Create new corpusData
my $cd = XML::Loy::XStandoff->new('corpusData');

# Set textual content embedded
$cd->textual_content('The sun shines brighter');

# Create segmentation
my $seg = $cd->segmentation;

# Create segments manually
my $seg1 = $seg->segment(0,24);
my $seg2 = $seg->segment(0, 3);
my $seg3 = $seg->segment(4, 7);
my $seg4 = $seg->segment(8, 13);
my $seg5 = $seg->segment(13, 14);
my $seg6 = $seg->segment(15, 21);
my $seg7 = $seg->segment(21, 23);
```

XML::Loy::XStandoff

XML::Loy::XStandoff » Code Result

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<corpusData xmlns="http://.../xstandoff/1.1"
            xmlns:xsf="http://.../xstandoff/1.1">
  <primaryData start="0" end="23" xml:id="pd-2531FE9A-...">
    <textualContent>The sun shines brighter</textualContent>
  </primaryData>
  <segmentation>
    <segment start="0" end="24"
             type="char" xml:id="seg-2532C88E-..." />
    <segment start="0" end="3"
             type="char" xml:id="seg-25330ACE-..." />
    <segment start="4" end="7"
             type="char" xml:id="seg-25334E9E-..." />
    <segment start="8" end="13"
             type="char" xml:id="seg-2533949E-..." />
    <segment start="13" end="14"
             type="char" xml:id="seg-2533DFE4-..." />
    <segment start="15" end="21"
             type="char" xml:id="seg-25343052-..." />
    <segment start="21" end="23"
             type="char" xml:id="seg-25348368-..." />
  </segmentation></corpusData>
```

XML::Loy::XStandoff

XML::Loy::XStandoff » Primary Data Manipulation

```
# Get segment content
say $seg->segment($seg3)
        ->segment_content;
# 'sun'
```

XML::Loy::XStandoff

XML::Loy::XStandoff » Primary Data Manipulation

```
# Get segment content
say $seg->segment($seg3)
        ->segment_content;
# 'sun'

# Replace segment content
$seg->segment($seg3)
    ->segment_content('moon');
```

XML::Loy::XStandoff

XML::Loy::XStandoff » Primary Data Manipulation

```
# Get segment content
say $seg->segment($seg3)
        ->segment_content;
# 'sun'

# Replace segment content
$seg->segment($seg3)
    ->segment_content('moon');

# Interactively replace segment content
$seg->segment($seg7)->segment_content(
  sub {
    my $t = shift;
    $t =~ s/er//;
    return $t;
  }
);
```

XML::Loy::XStandoff

XML::Loy::XStandoff » Primary Data Manipulation

```
# Show updated textual content
say $cd->textual_content;
# ‚The moon shines bright‘
```

XML::Loy::XStandoff

XML::Loy::XStandoff » Primary Data Manipulation

```
# Show updated textual content
say $cd->textual_content;
# ‚The moon shines bright‘


# Segment positions are updated
# automatically
for ($seg->segment($seg6)) {
  say $_->attr('start'); # 16
  say $_->attr('end');   # 22
};
```

XML::Loy::XStandoff

XML::Loy::XStandoff » New Document

```
use XML::Loy::XStandoff;
```

XML::Loy::XStandoff

XML::Loy::XStandoff » New Document

```perl
use XML::Loy::XStandoff;

my $cd = XML::Loy::XStandoff->new('corpusData');
$cd->extension(-Example::Morphemes,
               -Example::Syllables);
```

XML::Loy::XStandoff

XML::Loy::XStandoff » New Document

```
use XML::Loy::XStandoff;

my $cd = XML::Loy::XStandoff->new('corpusData');
$cd->extension(-Example::Morphemes,
               -Example::Syllables);
$cd->textual_content('The sun shines brighter.');
```

XML::Loy::XStandoff

XML::Loy::XStandoff » New Document

```
use XML::Loy::XStandoff;

my $cd = XML::Loy::XStandoff->new('corpusData');
$cd->extension(-Example::Morphemes,
               -Example::Syllables);
$cd->textual_content('The sun shines brighter.');

my $seg = $cd->segmentation;
my $all = $seg->segment(0, 24);
```

XML::Loy::XStandoff

XML::Loy::XStandoff » New Document

```
use XML::Loy::XStandoff;

my $cd = XML::Loy::XStandoff->new('corpusData');
$cd->extension(-Example::Morphemes,
               -Example::Syllables);
$cd->textual_content('The sun shines brighter.');

my $seg = $cd->segmentation;
my $all = $seg->segment(0, 24);

my $m = $cd->layer->morphemes;
$m->seg($all);
foreach ([0,3], [4,7], [8,13],
         [13,14], [15,21], [21,23]) {
  $m->morpheme->seg(
    $seg->segment($_->[0], $_->[1])
  );
};
```

XML::Loy::XStandoff

XML::Loy::XStandoff » New Document

```
use XML::Loy::XStandoff;

my $cd = XML::Loy::XStandoff->new('corpusData');
$cd->extension(-Example::Morphemes,
               -Example::Syllables);
$cd->textual_content('The sun shines brighter.');

my $seg = $cd->segmentation;
my $all = $seg->segment(0, 24);

# ... morphemes ...
my $s = $cd->layer->syllables;
$s->seg($all);
foreach ([0,3],[4,7],[8,14],[15,20],[20,23]) {
  $s->syllable->seg(
    $seg->segment($_->[0], $_->[1])
  );
};
```

Ideas of Enhancement

- Improve Constraints
- Namespace Islands
- More extensions in a repository
- Generate templates based on Document Grammars
- Speed improvements

Free to ...

- ... use!
  `http://search.cpan.org/~akron/XML-Loy/`
  `https://github.com/Akron/XML-Loy-XStandoff`
- ... investigate!
- ... modify!

Conclusion

Need for simple, extensible and reusable APIs

Conclusion

Need for simple, extensible and reusable APIs

XML::Loy
Foundation for APIs

Conclusion

Need for simple, extensible and reusable APIs

XML::Loy
Foundation for APIs

XML::Loy::XStandoff
Example API, dealing with standoff annotation

More ....

- **XML::Loy**
  http://search.cpan.org/~akron/XML-Loy/
- **XML::Loy::XStandoff**
  https://github.com/Akron/XML-Loy-XStandoff
- **XStandoff**
  http://xstandoff.net
- **Mojo::DOM**
  http://search.cpan.org/~sri/Mojolicious/
- **Sojolicious**
  http://sojolicio.us

Thank you …

…

… Questions?

External Files

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="...-syntax-ns#"
         xmlns:dc="http://purl.org/dc/
                   elements/1.1/">
  <rdf:Description>
    <dc:creator>Nils Diewald</dc:creator>
    <dc:creator>Maik Stührenberg</dc:creator>
    <dc:title>
      An extensible API for documents
      with multiple annotation layers
    </dc:title>
    <dc:language>EN</dc:language>
  </rdf:Description>
</rdf:RDF>
```

External Files

```
# Define the metadata as an external file
$cd->meta(uri => 'files/meta.xml');

# Retrieve the metadata,
# resulting in a new XML::Loy object
my $meta = $cd->meta(
  as => [-Loy, -DublinCore]
);

# The extension is available in
# the newly defined object
print $meta->at('Description')
           ->dc('title');
```

XML::Loy::XStandoff::Tokenizer

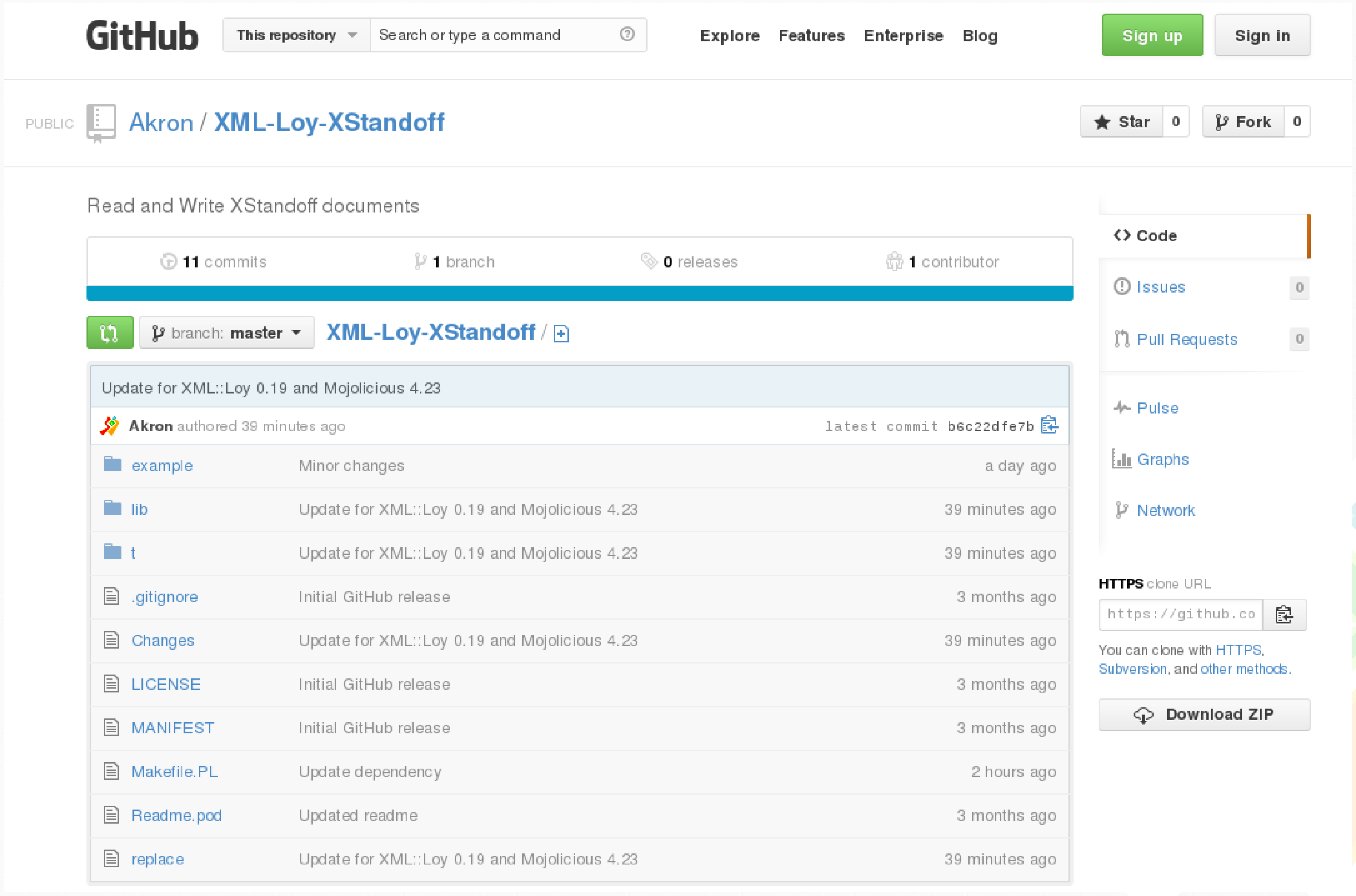
```perl
package XML::Loy::XStandoff::Tokenizer;
use XML::Loy -base;
use utf8;

sub tokenize {
  my $self = shift;

  while ($self->type !~ /^(?:xsf:)?corpusData$/) {
    $self = $self->parent or return;
  };

  my $seg = $self->segmentation;
  my $tc = $self->textual_content;

  my @segments;

  my ($start, $end) = 0;
  foreach my $t (split(/([^-a-zA-ZäüöÖÄÜß]|\s+)/, $tc)) {
    $end = $start + length $t;
    if ($t =~ /\w/) {
      push(@segments, [$t, $seg->segment($start, $end)]);
    };
    $start = $end;
  };

  return @segments;
};
```

XML::Loy::Schema::Validator

```perl
package XML::Loy::Schema::Validator;
use XML::LibXML;
use XML::Loy with => (
  on_init => sub {
    shift->namespace(
      xsi => 'http://www.w3.org/2001/XMLSchema-instance'
    )});

# Validate the document
sub validate {
  my $self = shift;

  my $root = $self->at(':root');
  my ($schema_loc, $ns) = pop;

  unless ($schema_loc) {
    ($ns, $schema_loc) = split /\s/, $root->attr('xsi:schemaLocation');
  };

  $ns = shift || $ns || $root->namespace;

  my $schema = XML::LibXML::Schema->new( location => $schema_loc );

  my $doc = XML::LibXML->load_xml(string => $self->to_pretty_xml );

  eval { $schema->validate($doc) };

  warn $@ and return if $@;

  $root->attr('xsi:schemaLocation' => "$ns $schema_loc");
  return $self;
};
```

XML::Loy::XStandoff

https://github.com/Akron/XML-Loy-XStandoff



Read and Write XStandoff documents

11 commits | 1 branch | 0 releases | 1 contributor

branch: master XML-Loy-XStandoff / +

Update for XML::Loy 0.19 and Mojolicious 4.23

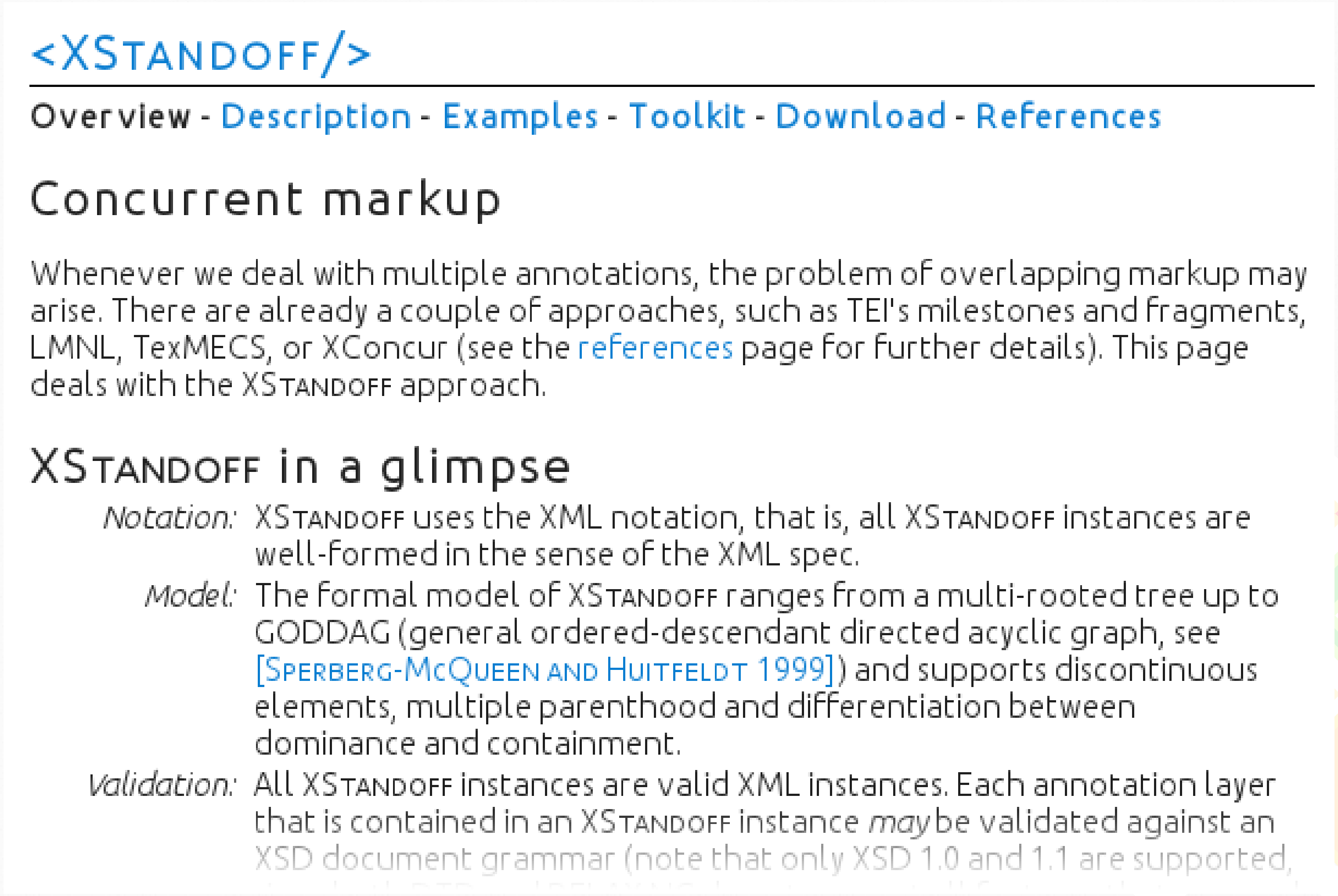
Akron authored 39 minutes ago — latest commit b6c22dfe7b

example	Minor changes	a day ago
lib	Update for XML::Loy 0.19 and Mojolicious 4.23	39 minutes ago
t	Update for XML::Loy 0.19 and Mojolicious 4.23	39 minutes ago
.gitignore	Initial GitHub release	3 months ago
Changes	Update for XML::Loy 0.19 and Mojolicious 4.23	39 minutes ago
LICENSE	Initial GitHub release	3 months ago
MANIFEST	Initial GitHub release	3 months ago
Makefile.PL	Update dependency	2 hours ago
Readme.pod	Updated readme	3 months ago
replace	Update for XML::Loy 0.19 and Mojolicious 4.23	39 minutes ago

You can clone with HTTPS, Subversion, and other methods.

XStandoff

http://xstandoff.net/



<XSTANDOFF/>

Overview - Description - Examples - Toolkit - Download - References

Concurrent markup

Whenever we deal with multiple annotations, the problem of overlapping markup may arise. There are already a couple of approaches, such as TEI's milestones and fragments, LMNL, TexMECS, or XConcur (see the references page for further details). This page deals with the XStandoff approach.

XStandoff in a glimpse

Notation: XStandoff uses the XML notation, that is, all XStandoff instances are well-formed in the sense of the XML spec.

Model: The formal model of XStandoff ranges from a multi-rooted tree up to GODDAG (general ordered-descendant directed acyclic graph, see [Sperberg-McQueen and Huitfeldt 1999]) and supports discontinuous elements, multiple parenthood and differentiation between dominance and containment.

Validation: All XStandoff instances are valid XML instances. Each annotation layer that is contained in an XStandoff instance *may* be validated against an XSD document grammar (note that only XSD 1.0 and 1.1 are supported,