Using Perl to Gather Information from the Web

Tom Hukins

What's the Point?

Web Services Where None Exist

WWW::Mechanize

- LWP
- HTML::Parser

LWP: Retrieve a Page

```
use LWP::Simple;
$doc = get 'http://annoyme.scrubhole.org/bad.html';
```

LWP: Submit a Form

```
use HTTP::Request::Common qw(POST);
use LWP::UserAgent;
$ua = LWP::UserAgent->new;

my $req = POST 'http://www.example.com/login',
    [
        username    => 'japh',
        password    => 'braga',
    ];
```

LWP: Submit a Form

```
use HTTP::Request::Common qw(POST);
use LWP::UserAgent;
$ua = LWP::UserAgent->new;

my $req = POST 'http://www.example.com/login',
    [
        username    => 'japh',
        password    => 'braga',
    ];
```

LWP: Submit a Form

```
use HTTP::Request::Common qw(POST);
use LWP::UserAgent;
$ua = LWP::UserAgent->new;

my $req = POST 'http://www.example.com/login',
    [
        username    => 'japh',
        password    => 'braga',
    ];
```

Parsing Bad HTML

```
<html>
  <title>A title belongs in the <head></title>
  <BODY>
    Here is <a href="some" title="example">some</a>
    <a href="other" title="example2">text</a>.
    <p>
    The end.
  </body>
<html>
```

Parsing Bad HTML

```
<html>
  <title>A title belongs in the <head></title>
  <BODY>
    Here is <a href="some" title="example">some</a>
    <a href="other" title="example2">text</a>.
    <p>
    The end.
  </body>
<html>
```

Parsing Bad HTML

```
<html>
  <title>A title belongs in the <head></title>
  <BODY>
    Here is <a href="some" title="example">some</a>
    <a href="other" title="example2">text</a>.
    <p>
    The end.
  </body>
<html>
```

Parsing Bad HTML

```
<html>
  <title>A title belongs in the <head></title>
  <BODY>
    Here is <a href="some" title="example">some</a>
    <a href="other" title="example2">text</a>.
    <p>
    The end.
  </body>
<html>
```

Parsing Bad HTML

```
<html>
  <title>A title belongs in the <head></title>
  <BODY>
    Here is <a href="some" title="example">some</a>
    <a href="other" title="example2">text</a>.
    <p>
    The end.
  </body>
<html>
```

HTML::Parser

```
use HTML::TokeParser;
my $p = HTML::TokeParser->new('bad.html');

while (my $token = $p->get_tag("a")) {
    my $url = $token->[1]{href};
    print "$url\n";
}
```

HTML::Parser

```
use HTML::TokeParser;
my $p = HTML::TokeParser->new('bad.html');

while (my $token = $p->get_tag("a")) {
    my $url = $token->[1]{href};
    print "$url\n";
}
```

HTML::Parser

```
use HTML::TokeParser;
my $p = HTML::TokeParser->new('bad.html');

while (my $token = $p->get_tag("a")) {
    my $url = $token->[1]{href};
    print "$url\n";
}
```

HTML::Parser

```
use HTML::TokeParser;
my $p = HTML::TokeParser->new('bad.html');

while (my $token = $p->get_tag("a")) {
    my $url = $token->[1]{href};
    print "$url\n";
}
```

WWW::Mechanize

```
use WWW::Mechanize ();
my $mech = WWW::Mechanize->new();

$mech->get('http://annoyme.scrubhole.org/bad.html');
my $link = $mech->find_link( text => 'some' );
print $link->attrs->{title}, "\n";
```

WWW::Mechanize

```
use WWW::Mechanize ();
my $mech = WWW::Mechanize->new();

$mech->get('http://annoyme.scrubhole.org/bad.html');
my $link = $mech->find_link( text => 'some' );
print $link->attrs->{title}, "\n";
```

WWW::Mechanize

```
use WWW::Mechanize ();
my $mech = WWW::Mechanize->new();

$mech->get('http://annoyme.scrubhole.org/bad.html');
my $link = $mech->find_link( text => 'some' );
print $link->attrs->{title}, "\n";
```

Filling in Forms

```
use WWW::Mechanize ();
my $mech = WWW::Mechanize->new();

$mech->get('http://www.example.com/');
$mech->submit_form(
    form_name   => 'login',
    fields      => {
        username    => 'japh',
        password    => 'braga',
    }
);
```

Filling in Forms

```
use WWW::Mechanize ();
my $mech = WWW::Mechanize->new();

$mech->get('http://www.example.com/');
$mech->submit_form(
    form_name   => 'login',
    fields      => {
        username    => 'japh',
        password    => 'braga',
    }
);
```

Filling in Forms

```
use WWW::Mechanize ();
my $mech = WWW::Mechanize->new();

$mech->get('http://www.example.com/');
$mech->submit_form(
    form_name   => 'login',
    fields      => {
        username    => 'japh',
        password    => 'braga',
    }
);
```

Filling in Forms

```
use WWW::Mechanize ();
my $mech = WWW::Mechanize->new();

$mech->get('http://www.example.com/');
$mech->submit_form(
    form_name   => 'login',
    fields      => {
        username    => 'japh',
        password    => 'braga',
    }
);
```

XPath

- XML Parsing Language
- W3C Specification
- Support in many languages (including Perl)

XPath Example

```
<?xml version="1.0"?>
<html>
  <head>
    <title>A title belongs in the</title>
  </head>
  <body>
    <p>
      Here is <a href="some" title="example">some</a>
      <a href="other" title="example2">text</a>.
    </p>
  </body>
</html>
```

XPath Example

/html

```
<?xml version="1.0"?>
<html>
  <head>
    <title>A title belongs in the</title>
  </head>
  <body>
    <p>
      Here is <a href="some" title="example">some</a>
      <a href="other" title="example2">text</a>.
    </p>
  </body>
</html>
```

XPath Example

/html/body

```
<?xml version="1.0"?>
<html>
  <head>
    <title>A title belongs in the</title>
  </head>
  <body>
    <p>
      Here is <a href="some" title="example">some</a>
      <a href="other" title="example2">text</a>.
    </p>
  </body>
</html>
```

XPath Example

/html/body//a

```
<?xml version="1.0"?>
<html>
  <head>
    <title>A title belongs in the</title>
  </head>
  <body>
    <p>
      Here is <a href="some" title="example">some</a>
      <a href="other" title="example2">text</a>.
    </p>
  </body>
</html>
```

XPath Example

/html/body//a/@href

```
<?xml version="1.0"?>
<html>
  <head>
    <title>A title belongs in the</title>
  </head>
  <body>
    <p>
      Here is <a href="some" title="example">some</a>
      <a href="other" title="example2">text</a>.
    </p>
  </body>
</html>
```

XPath Example

```
/html/body//a[@title="example2"]/@href
```

```
<?xml version="1.0"?>
<html>
  <head>
    <title>A title belongs in the</title>
  </head>
  <body>
    <p>
      Here is <a href="some" title="example">some</a>
      <a href="other" title="example2">text</a>.
    </p>
  </body>
</html>
```

Mechanize and XPath

```
use WWW::Mechanize ();
use XML::LibXML ();

my $mech = WWW::Mechanize->new();
$mech->get('http://annoyme.scrubhole.org/average.html');

my $parser = XML::LibXML->new();
$parser->recover(1);

my $doc = $parser->parse_html_string( $mech->content );
print $doc->findvalue(
    '/html/body//a[@title="example2"]/@href'
), "\n";
```

Mechanize and XPath

```
use WWW::Mechanize ();
use XML::LibXML ();

my $mech = WWW::Mechanize->new();
$mech->get('http://annoyme.scrubhole.org/average.html');

my $parser = XML::LibXML->new();
$parser->recover(1);

my $doc = $parser->parse_html_string( $mech->content );
print $doc->findvalue(
    '/html/body//a[@title="example2"]/@href'
), "\n";
```

Mechanize and XPath

```
use WWW::Mechanize ();
use XML::LibXML ();

my $mech = WWW::Mechanize->new();
$mech->get('http://annoyme.scrubhole.org/average.html');

my $parser = XML::LibXML->new();
$parser->recover(1);

my $doc = $parser->parse_html_string( $mech->content );
print $doc->findvalue(
    '/html/body//a[@title="example2"]/@href'
), "\n";
```

Mechanize and XPath

```
use WWW::Mechanize ();
use XML::LibXML ();

my $mech = WWW::Mechanize->new();
$mech->get('http://annoyme.scrubhole.org/average.html');

my $parser = XML::LibXML->new();
$parser->recover(1);

my $doc = $parser->parse_html_string( $mech->content );
print $doc->findvalue(
    '/html/body//a[@title="example2"]/@href'
), "\n";
```

XPath and Bad HTML

```
...

my $args =
  '-ashtml --show-errors 0 -quiet --output-file tidy-temp';

open my $tidy, "| tidy $args" or die $!;
print $tidy $mech->content;
close $tidy;

...
```

XPath and Bad HTML

```
...

my $args =
  '-ashtml --show-errors 0 -quiet --output-file tidy-temp';

open my $tidy, "| tidy $args" or die $!;
print $tidy $mech->content;
close $tidy;

...
```

XPath and Bad HTML

```
...

my $args =
  '-ashtml --show-errors 0 -quiet --output-file tidy-temp';

open my $tidy, "| tidy $args" or die $!;
print $tidy $mech->content;
close $tidy;

...
```

XPath and Bad HTML

```
...

open my $tidy, "| tidy $args" or die $!;
print $tidy $mech->content;
close $tidy;

my $parser = XML::LibXML->new();
my $doc = $parser->parse_html_file('tidy-temp') or die $!;
print $doc->findvalue('/html/body//a[@title="example2"]/
@href'), "\n";

...
```

XPath and Bad HTML

```
...

my $parser = XML::LibXML->new();
my $doc = $parser->parse_html_file('tidy-temp') or die $!;
print $doc->findvalue('/html/body//a[@title="example2"]/
@href'), "\n";

unlink 'tidy-temp';

...
```

xsh: The XML Shell

```
xsh scratch:/> recovering 1
xsh scratch:/> open HTML doc=average.html
parsing average.html
done.
xsh doc:/> cd /html/body
xsh doc:/html/body> ls
<body><p>...</p><p>...</p></body>

Found 1 node(s).
```

xsh: The XML Shell

```
xsh scratch:/> recovering 1
xsh scratch:/> open HTML doc=average.html
parsing average.html
done.
xsh doc:/> cd /html/body
xsh doc:/html/body> ls
<body><p>...</p><p>...</p></body>

Found 1 node(s).
```

xsh: The XML Shell

```
xsh scratch:/> recovering 1
xsh scratch:/> open HTML doc=average.html
parsing average.html
done.
xsh doc:/> cd /html/body
xsh doc:/html/body> ls
<body><p>...</p><p>...</p></body>

Found 1 node(s).
```

xsh: The XML Shell

```
xsh scratch:/> recovering 1
xsh scratch:/> open HTML doc=average.html
parsing average.html
done.
xsh doc:/> cd /html/body
xsh doc:/html/body> ls
<body><p>...</p><p>...</p></body>

Found 1 node(s).
```

xsh: The XML Shell

```
xsh scratch:/> recovering 1
xsh scratch:/> open HTML doc=average.html
parsing average.html
done.
xsh doc:/> cd /html/body
xsh doc:/html/body> ls
<body><p>...</p><p>...</p></body>

Found 1 node(s).
```

xsh: The XML Shell

```
xsh doc:/html/body> ls *
<p>
    Here is <a href="some" title="example">some</a>
    <a href="other" title="example2">text</a>.
    </p>
<p>
    The end.
  </p>

Found 2 node(s).
```

The End!

References

- LWP (libwww-perl on CPAN)
- HTML::Parser (on CPAN)
- XPath (http://www.w3c.org/)
- tidy (http://tidy.sourceforge.net/)
- xsh (XML::XSH on CPAN)

tom@eborcom.com