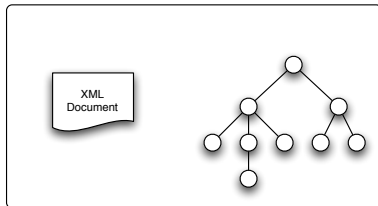


Writing XML files from Perl

Alberto Simões
ambs@cpan.org

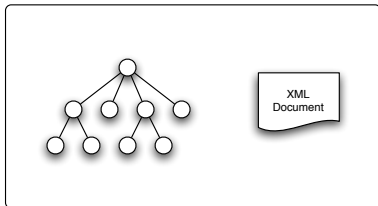
YAPC::EU::2006

XML Parsing



Transformation

XML Generation



- Parsing:
 - **DOM:** XML::Parser, XML::LibXML, XML::DT, XML::Simple, XML::Twig
 - **SAX:** XML::Parser, XML::LibXML
- Generation:
 - **DOM:** XML::Twig, XML::Simple
 - **SAX:** XML::Writer, XML::Writer::Simple

- Why not XML::Writer?
 - error prone: start_tag/end_tag oriented;
 - slow (at least the last time I've tried it);
- So, what is XML::Writer::Simple?
 - is CGI generalized for any tag;
 - huh?
- you just need to declare the **valid** tags:
 - use a list of valid tags;
 - use a sample XML document;
 - use a DTD document;
 - use a schema (oops, not yet!).

- Why not XML::Writer?
 - error prone: start_tag/end_tag oriented;
 - slow (at least the last time I've tried it);
- So, what is XML::Writer::Simple?
 - is CGI generalized for any tag;
 - huh?
- you just need to declare the **valid** tags:
 - use a list of valid tags;
 - use a sample XML document;
 - use a DTD document;
 - use a schema (oops, not yet!).

- Why not XML::Writer?
 - error prone: start_tag/end_tag oriented;
 - slow (at least the last time I've tried it);
- So, what is XML::Writer::Simple?
 - is CGI generalized for any tag;
 - huh?
- you just need to declare the **valid** tags:
 - use a list of valid tags;
 - use a sample XML document;
 - use a DTD document;
 - use a schema (oops, not yet!).

- Why not XML::Writer?
 - error prone: start_tag/end_tag oriented;
 - slow (at least the last time I've tried it);
- So, what is XML::Writer::Simple?
 - is CGI generalized for any tag;
 - huh?
 - want a p tag? use p("contents")
 - want a foo tag? use foo("contents")
- you just need to declare the valid tags:
 - use a list of valid tags;
 - use a sample XML document;
 - use a DTD document;
 - use a schema (oops, not yet!).



- Why not XML::Writer?
 - error prone: start_tag/end_tag oriented;
 - slow (at least the last time I've tried it);
- So, what is XML::Writer::Simple?
 - is CGI generalized for any tag;
 - huh?
 - want a p tag? use p("contents")
 - want a foo tag? use foo("contents")
- you just need to declare the **valid** tags:
 - use a list of valid tags;
 - use a sample XML document;
 - use a DTD document;
 - use a schema (oops, not yet!).

- Why not XML::Writer?
 - error prone: start_tag/end_tag oriented;
 - slow (at least the last time I've tried it);
- So, what is XML::Writer::Simple?
 - is CGI generalized for any tag;
 - huh?
 - want a `p` tag? use `p("contents")`
 - want a `foo` tag? use `foo("contents")`
- you just need to declare the `valid` tags:
 - use a list of valid tags;
 - use a sample XML document;
 - use a DTD document;
 - use a schema (oops, not yet!).

- Why not XML::Writer?
 - error prone: start_tag/end_tag oriented;
 - slow (at least the last time I've tried it);
- So, what is XML::Writer::Simple?
 - is CGI generalized for any tag;
 - huh?
 - want a `p` tag? use `p("contents")`
 - want a `foo` tag? use `foo("contents")`
- you just need to declare the `valid` tags:
 - use a list of valid tags;
 - use a sample XML document;
 - use a DTD document;
 - use a schema (oops, not yet!).

- Why not XML::Writer?
 - error prone: start_tag/end_tag oriented;
 - slow (at least the last time I've tried it);
- So, what is XML::Writer::Simple?
 - is CGI generalized for any tag;
 - huh?
 - want a `p` tag? use `p("contents")`
 - want a `foo` tag? use `foo("contents")`
- you just need to declare the **valid** tags:
 - use a list of valid tags;
 - use a sample XML document;
 - use a DTD document;
 - use a schema (oops, not yet!).

- Why not XML::Writer?
 - error prone: start_tag/end_tag oriented;
 - slow (at least the last time I've tried it);
- So, what is XML::Writer::Simple?
 - is CGI generalized for any tag;
 - huh?
 - want a `p` tag? use `p("contents")`
 - want a `foo` tag? use `foo("contents")`
- you just need to declare the **valid** tags:
 - use a list of valid tags;
 - use a sample XML document;
 - use a DTD document;
 - use a schema (oops, not yet!).

- Why not XML::Writer?
 - error prone: start_tag/end_tag oriented;
 - slow (at least the last time I've tried it);
- So, what is XML::Writer::Simple?
 - is CGI generalized for any tag;
 - huh?
 - want a `p` tag? use `p("contents")`
 - want a `foo` tag? use `foo("contents")`
- you just need to declare the **valid** tags:
 - use a list of valid tags;
 - use a sample XML document;
 - use a DTD document;
 - use a schema (oops, not yet!).

- Why not XML::Writer?
 - error prone: start_tag/end_tag oriented;
 - slow (at least the last time I've tried it);
- So, what is XML::Writer::Simple?
 - is CGI generalized for any tag;
 - huh?
 - want a `p` tag? use `p("contents")`
 - want a `foo` tag? use `foo("contents")`
- you just need to declare the **valid** tags:
 - use a list of valid tags;
 - use a sample XML document;
 - use a DTD document;
 - use a schema (oops, not yet!).

- Why not XML::Writer?
 - error prone: start_tag/end_tag oriented;
 - slow (at least the last time I've tried it);
- So, what is XML::Writer::Simple?
 - is CGI generalized for any tag;
 - huh?
 - want a `p` tag? use `p("contents")`
 - want a `foo` tag? use `foo("contents")`
- you just need to declare the **valid** tags:
 - use a list of valid tags;
 - use a sample XML document;
 - use a DTD document;
 - use a schema (oops, not yet!).

```
use XML::Writer::Simple
    tags => qw/contacts contact name email phone/;

print contacts(
    map { contact(
        name($_->{name}),
        email($_->{mail}),
        phone($_->{phone})) } @contacts )
```

```
use XML::Writer::Simple
    xml => "contacts.xml";
    # xml => [qw/mycontacts.xml morecontacts.xml/];

print contacts(
    map { contact(
        name($_->{name}),
        email($_->{mail}),
        phone($_->{phone})) } @contacts )
```

```
use XML::Writer::Simple dtd => "contacts.dtd";

print contacts(
  map { contact(
    name($_->{name}),
    email($_->{mail}),
    phone($_->{phone})) } @contacts )
```



- use attributes:
 - as with CGI: `a({href=>"foo"}, "bar")`
 - you get: `bar`
- apply tags recursively:
 - what about: `li([qw/a b c/])`
 - the expected: `abc`
- but we can do better:
 - try: `ul_li("a","b","c")`
 - to get: `abc`
- or not...
 - the ugly: `table_tr_td(["a","b","c"],["d","e","f"])`
 - generates what you expect...
- unfortunately ATM you need to declare them:
 - use `XML::Writer::Simple`

```
powertags=>[qw/ul_li ol_li/];
```
 - `powertag("table","tr","td");`



- use attributes:
 - as with CGI: `a({href=>"foo"}, "bar")`
 - you get: `bar`
- apply tags recursively:
 - what about: `li([qw/a b c/])`
 - the expected: `abc`
- but we can do better:
 - try: `ul_li("a","b","c")`
 - to get: `abc`
- or not...
 - the ugly: `table_tr_td(["a","b","c"],["d","e","f"])`
 - generates what you expect...
- unfortunately ATM you need to declare them:
 - use `XML::Writer::Simple`

```
powertags=>[qw/ul_li ol_li/];
```
 - `powertag("table","tr","td");`



- use attributes:
 - as with CGI: `a({href=>"foo"}, "bar")`
 - you get: `bar`
- apply tags recursively:
 - what about: `li([qw/a b c/])`
 - the expected: `abc`
- but we can do better:
 - try: `ul_li("a","b","c")`
 - to get: `abc`
- or not...
 - the ugly: `table_tr_td(["a","b","c"],["d","e","f"])`
 - generates what you expect...
- unfortunately ATM you need to declare them:
 - use `XML::Writer::Simple`

```
powertags=>[qw/ul_li ol_li/];
```
 - `powertag("table","tr","td");`



- use attributes:
 - as with CGI: `a({href=>"foo"}, "bar")`
 - you get: `bar`
- apply tags recursively:
 - what about: `li([qw/a b c/])`
 - the expected: `abc`
- but we can do better:
 - try: `ul_li("a","b","c")`
 - to get: `abc`
- or not...
 - the ugly: `table_tr_td(["a","b","c"],["d","e","f"])`
 - generates what you expect...
- unfortunately ATM you need to declare them:
 - use `XML::Writer::Simple`

```
powertags=>[qw/ul_li ol_li/];
```
 - `powertag("table","tr","td");`



- use attributes:
 - as with CGI: `a({href=>"foo"}, "bar")`
 - you get: `bar`
- apply tags recursively:
 - what about: `li([qw/a b c/])`
 - the expected: `abc`
- but we can do better:
 - try: `ul_li("a","b","c")`
 - to get: `abc`
- or not...
 - the ugly: `table_tr_td(["a","b","c"],["d","e","f"])`
 - generates what you expect...
- unfortunately ATM you need to declare them:
 - use `XML::Writer::Simple`

```
powertags=>[qw/ul_li ol_li/];
```
 - `powertag("table","tr","td");`



- use attributes:
 - as with CGI: `a({href=>"foo"}, "bar")`
 - you get: `bar`
- apply tags recursively:
 - what about: `li([qw/a b c/])`
 - the expected: `abc`
- but we can do better:
 - try: `ul_li("a","b","c")`
 - to get: `abc`
- or not...
 - the ugly: `table_tr_td(["a","b","c"],["d","e","f"])`
 - generates what you expect...
- unfortunately ATM you need to declare them:
 - use `XML::Writer::Simple`

```
powertags=>[qw/ul_li ol_li/];
```
 - `powertag("table","tr","td");`



- use attributes:
 - as with CGI: `a({href=>"foo"}, "bar")`
 - you get: `bar`
- apply tags recursively:
 - what about: `li([qw/a b c/])`
 - the expected: `abc`
- but we can do better:
 - try: `ul_li("a","b","c")`
 - to get: `abc`
- or not...
 - the ugly: `table_tr_td(["a","b","c"],["d","e","f"])`
 - generates what you expect...
- unfortunately ATM you need to declare them:
 - use `XML::Writer::Simple`

```
powertags=>[qw/ul_li ol_li/];
```
 - `powertag("table","tr","td");`



- use attributes:
 - as with CGI: `a({href=>"foo"}, "bar")`
 - you get: `bar`
- apply tags recursively:
 - what about: `li([qw/a b c/])`
 - the expected: `abc`
- but we can do better:
 - try: `ul_li("a","b","c")`
 - to get: `abc`
- or not...
 - the ugly: `table_tr_td(["a","b","c"],["d","e","f"])`
 - generates what you expect...
- unfortunately ATM you need to declare them:
 - use `XML::Writer::Simple`

```
powertags=>[qw/ul_li ol_li/];
```
 - `powertag("table","tr","td");`



- use attributes:
 - as with CGI: `a({href=>"foo"}, "bar")`
 - you get: `bar`
- apply tags recursively:
 - what about: `li([qw/a b c/])`
 - the expected: `abc`
- but we can do better:
 - try: `ul_li("a","b","c")`
 - to get: `abc`
- or not...
 - the ugly: `table_tr_td(["a","b","c"],["d","e","f"])`
 - generates what you expect...
- unfortunately ATM you need to declare them:
 - use `XML::Writer::Simple`

```
powertags=>[qw/ul_li ol_li/];
```
 - `powertag("table","tr","td");`

- use attributes:
 - as with CGI: `a({href=>"foo"}, "bar")`
 - you get: `bar`
- apply tags recursively:
 - what about: `li([qw/a b c/])`
 - the expected: `abc`
- but we can do better:
 - try: `ul_li("a","b","c")`
 - to get: `abc`
- or not...
 - the ugly: `table_tr_td(["a","b","c"],["d","e","f"])`
 - generates what you expect...
- unfortunately ATM you need to declare them:
 - use `XML::Writer::Simple`
 - `powertags=>[qw/ul_li ol_li/];`
 - `powertag("table","tr","td");`



- use attributes:
 - as with CGI: `a({href=>"foo"}, "bar")`
 - you get: `bar`
- apply tags recursively:
 - what about: `li([qw/a b c/])`
 - the expected: `abc`
- but we can do better:
 - try: `ul_li("a","b","c")`
 - to get: `abc`
- or not...
 - the ugly: `table_tr_td(["a","b","c"],["d","e","f"])`
 - generates what you expect...
- unfortunately ATM you need to declare them:
 - use `XML::Writer::Simple`

```
powertags=>[qw/ul_li ol_li/];
```
 - `powertag("table","tr","td");`



- use attributes:
 - as with CGI: `a({href=>"foo"}, "bar")`
 - you get: `bar`
- apply tags recursively:
 - what about: `li([qw/a b c/])`
 - the expected: `abc`
- but we can do better:
 - try: `ul_li("a","b","c")`
 - to get: `abc`
- or not...
 - the ugly: `table_tr_td(["a","b","c"], ["d","e","f"])`
 - generates what you expect...
- unfortunately ATM you need to declare them:
 - use `XML::Writer::Simple`

```
powertags=>[qw/ul_li ol_li/];
```
 - `powertag("table","tr","td");`



- use attributes:
 - as with CGI: `a({href=>"foo"}, "bar")`
 - you get: `bar`
- apply tags recursively:
 - what about: `li([qw/a b c/])`
 - the expected: `abc`
- but we can do better:
 - try: `ul_li("a","b","c")`
 - to get: `abc`
- or not...
 - the ugly: `table_tr_td(["a","b","c"],["d","e","f"])`
 - generates what you expect...
- unfortunately ATM you need to declare them:
 - use `XML::Writer::Simple`

```
powertags=>[qw/ul_li ol_li/];
```
 - `powertag("table","tr","td");`



- use attributes:
 - as with CGI: `a({href=>"foo"}, "bar")`
 - you get: `bar`
- apply tags recursively:
 - what about: `li([qw/a b c/])`
 - the expected: `abc`
- but we can do better:
 - try: `ul_li("a","b","c")`
 - to get: `abc`
- or not...
 - the ugly: `table_tr_td(["a","b","c"], ["d","e","f"])`
 - generates what you expect...
- unfortunately ATM you need to declare them:
 - use `XML::Writer::Simple`

```
powertags=>[qw/ul_li ol_li/];
```
 - `powertag("table","tr","td");`



- use attributes:
 - as with CGI: `a({href=>"foo"}, "bar")`
 - you get: `bar`
- apply tags recursively:
 - what about: `li([qw/a b c/])`
 - the expected: `abc`
- but we can do better:
 - try: `ul_li("a","b","c")`
 - to get: `abc`
- or not...
 - the ugly: `table_tr_td(["a","b","c"],["d","e","f"])`
 - generates what you expect...
- unfortunately ATM you need to declare them:
 - use `XML::Writer::Simple`
`powertags=>[qw/ul_li ol_li/];`
 - `powertag("table","tr","td");`