XML Processing

lab #lab08

James L. Parry
B.C. Institute of Technology

Lab Goals

The purpose of this lab is to help you practice working with SimpleXML, from lesson xml03.

This lab is to be completed with the same team and repository that you setup for Lab 7.

The SimpleXML tutorial is worth a read! It is a walkthrough of a webapp that does similar things to what you need to do in this lab, and you may want to have it open to refer to while you work through the lab.

Suggestion: you may want to skim the slideshow first, before working your way through it.

#123 I will use a box like this to draw your attention to a relevant slide (eg #123) in the SimpleXML tutorial.

Lab Submission

Submit a readme to your Lab 8 dropbox, with a link to the github repository you used for this lab.

Please make sure that your work is in the "master" branch or your repository, and that any other branches have been merged into it.

Due: five days after your lab, at 17:30.

Lab Marking Guideline

- 4 marks for the prescribed `Timetable` model
- 1 marks for the prescribed `Booking` entity model
- 2 marks for `Welcome::index` as prescribed
- 2 marks for `Welcome::search` as prescribed
- 1 mark for reasonable workflow

Lab Preparation

Make sure that your XML documents and DTD are in a `/data` folder inside your project.

Incorporate your CodeIgniter starter into your project, if you have not already done so.

#4 It would be a good idea to define a DATAPATH constant.

#5 You can probably think of your timetable data as something similar to the menu.xml in the example, but you would have your three facets as child elements of the root element, instead of patties, cheeses and toppings.

Basic Model

Create a basic `Timetable` model for your XML data.

Choose appropriate properties to hold each of your timetable facets.

These will get populated in the next step.

#7 Your model will look like this, so far, with a $xml property, and probably a property for each of your facets, each being an array intended to hold bookings.

Entity Model

Make a `Booking` entity model class, which will go inside your `Timetable` model's source code file, at the end.

It should contain O-O properties to match all of the attributes and nested elements in your XML booking data. These properties can be public, for convenience. This is intended to be just a data structure, with no methods.

It should have properties for each of the attributes of child elements in your booking elements. They will get "flattened" into object properties, to hide their origin.

Building a Booking

You could plan to populate these individually, but it might make more sense to have a SimpleXMLElement passed to the `Booking` constructor, from which as much data as possible can be abstracted.

#8 This is closest to the stdClass object, $record.

Instead of something like `$record = new stdClass(); $record->code = (String) $patty['code']; ...` you would have something like `$record = new Booking($patty);` and you would have a Booking constructor like `function __construct($patty) { $this->code = (String) $patty['code']; ...`

However you choose to populate a `Booking`'s properties, the values should be simple PHP data types, and *not* SimpleXMLElements.

Back to Your Timetable Model

You can now populate your Timetable model properties, for each facet.

For instance, `foreach ($this->xml->whatevercorrespondstoacolumn as $time) { $this->timeslots[] = new Booking($time);}`

#9 Your code will be simpler than the example!

Facet Accessors

Build an accessor method for each facet, which returns an array of `Booking` objects appropriate to that facet.

#10 Like the `patties` method. We might need individual ones in a different webapp, but not for this lab.

Basic Homepage

You can now start on your basic homepage, which should report each of the three sets of bookings, one for each facet.

This will be your `Welcome` controller.

#15 Load your Timetable model in the Welcome controller, or else configure it to be autoloaded.

Show Your Data

Your controller's `index()` can build view parameters for your revised homepage view, passing the sets of bookings for each facet.

Alternately, you could use view fragments, like in previous work.

#16 It'll look like this but simpler (you don't need to filter files).

Present search form

Add two methods to your `Timetable` to return the data needed to populate a dropdown for days of the week and another for timeslots. These would each be an associative array, with the keys being the codes you chose inside your XML data and the values being suitable display text.

Suggestion: `Timetable::getTheDayCodesIWantToUseInADropdown()` or perhaps something easier?

Add a "search" HTML form to your homepage, with two dropdowns built from these.

Make the Timetable Searchable

Add search methods to your `Timetable` model, one for each facet. These should return an array of `Booking` objects that match the selected day and timeslot according to the facet they pertain to.

An example: `public function findAnyBookingsInATimetableCellUsingFacet1($day,$slot) {...}`, or perhaps something simpler?

Bingo

Here's the fun part: bind your search form (view) to a `search` method in your `Welcome` controller.

It should search the timetable using each facet, and ... if each of the facet searches returns a single `Booking`, and the three `Bookings` are the same, then you can report "Bingo" and display one of them.

If the searches do not return the same `Booking`, or if they return more than, then communicate that in your output, for instance "by timeslot gives ..." and "by course gives ..." etc.

And then...

If you have complete data, and it is consistent, then you should always get either a "bingo" or else nothing for a given search. If your data is incomplete (for instance, you only provided one or two instances of bookings for a facet), you won't get a "bingo".

The intent is to correctly identify inconsistent data, and not to always get a "bingo"!

The search results can be added to the homepage. In other words, the homepage will show the three facets, the search form, and the result of the immediately prior search (if appropriate).

Tidy Up

Tidy up the presentation for your homepage, and you are done!

Congratulations!

You have completed lab #lab08: XML Processing

If you would take a minute to provide some feedback, we would appreciate it!

The next activity in sequence is: xml-xml02 XML Processing (optional)

You can use your browser's back button to return to the page you were on before starting this activity, or you can jump directly to the course homepage, organizer, or reference page.