

Sarah Kahn
11.30.07
Advanced Internet Applications
RSS Feed Writeup

Dynamically Generated RSS Feed

Part 1 of my project consisted of designing a dynamically generated rss feed for ibiblio.org. The information that I want to display is the collections index entries for newly created collections. This way, users can easily see whats new at [ibiblio](http://ibiblio.org) without having to physically visit our site. My initial plan was to use php to execute a mysql query, retrieve information from the collections index database, and echo that out into rss format.

My initial design was clunky and in the end didn't work. I had blocks of php and blocks of xml in one document, where I would write a php function, then escape out of php, and print some xml, and then go back into php. I tried to call php functions inside of xml tags, something like this: `<rss tag> <?php echo $x, $y?></rss>`

This did not work very well, for reasons which I understand but can't explain very well. I think basically it was just too complicated and it confused the server, also the construction of the php had errors which it was trying to pass into the xml, and it was just wrong. So, I changed the design to use the structure of `echo <<<exit; <xml in here> exit;` where the blocks of php were escaped out of without having to switch between languages.

The next problem that I had to solve was accessing the data that I wanted to format. I sniffed out the login information for what I thought was the collections index database, initially trying to access it from the login server, tribal.ibiblio.org. Come to find out, while this is the server that ibiblio.org/collections pulls its information from, that database is mirrored from somewhere else, and I was getting a feed of older data. The collections index entries don't record a date anywhere, so I used the automatically generated collection id number as the way to determine which are the most recent. I was getting results in the 700s from tribal, when I knew that there were new collections in the 1000s. So after some more looking around, I found that I could access the database server directly, but not from the SILS server Ruby, where the php file was living at the time. So I moved it onto my web directory on [ibiblio](http://ibiblio.org), and then the feed was able to access the more recent mirror.

I also had some trouble with the mysql query. I wanted to pull in information from 3 different tables- the contributor's name, descriptive information about the collection, and information about the collection classification. My original query brought back a lot of repeat information, so I had to play with the joins until I was getting back just one of each item.

Here is the code:

The first thing that I did was to set the output content to the utf-8 character set.

```
<?php
// set the file's content type and character set to utf-8
header("Content-Type: text/xml; charset=utf-8");
```

// connect to the database db2.ibiblio.org is where the updated mirror lives. This is how I originally had it connecting, since then I have modified it to require an .htlogin file for security //purposes. Here is the text of that file:

```
<?php
$db = mysql_connect('db2.ibiblio.org', 'collections', '1b1bl10');
?>
```

//continuing with the script:

```
$db = mysql_connect('db2.ibiblio.org', 'collections', '1b1bl10');
```

// if the connection can't be made, die and display an error message

```
if (!$db){
die
("Error: Could not connect to the database in rss.php");
}
```

//select the database the feed will be pulled from

```
mysql_select_db('collections', $db);
```

// the mysql query- select id, title, description, and other descriptive fields from the "collection" table and first and last name of the contributor from the "person" table.

```
$query = mysql_query('SELECT c.collection_id, c.title, c.description, c.identifier, c.source,
c.meta, c.ready, p.lname, p.fname FROM collection c, person p, collection_person x where
x.collection_id=c.collection_id AND p.username=x.username AND x.role_id=1 ORDER BY
c.collection_id desc LIMIT 0, 10');
```

// if the query fails, die and print an error message, rather than taking down the database server

```
if (!$query){
die
("Error");
}
```

//echo <<<end escapes the php to print out the rss feed header

```
echo <<<END
```

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<rss version="2.0">
```

```
  <channel>
```

```
    <title>ibiblio.org- latest collections</title>
```

```
    <link>http://ibiblio.org</link>
```

```
    <description>ibiblio's latest collections</description>
```

```
    <language>en-us</language>
```

```
END;
```

//loop through the database fields pulling information for each item of the array. My first design was to use the list() function to pull items into the array, but that didn't work, so I backed //down to a less elegant and streamlined design, using mysql_fetch_array(). I decided not to name the individual items in the array, because there are so many, and there could be more //potentially. I thought it would be easier to just call them by position in the array. That way, if they change, all I will have to change is the numbers.

```
while($x = mysql_fetch_array($query)) {  
  
    echo <<<END  
  
    <item>  
        <title>$x[0]: $x[1]</title>  
        <link>$x[3]</link>  
        <description>$x[2] <br />$x[4]</description>  
        <author>$x[8] $x[7]</author>  
    </item>  
    END;  
}  
  
//display end of rss file  
  
echo '  
</channel>  
</rss>';  
  
>
```

This feed is live at ibiblio.org/snkahn. I've titled it as "index.php" because a colleague is attempting to make it talk to a facebook application, and that made his code easier. One thing which I have since changed, is to move the database access information into a .htlogin file and then require that within the php script. This will prevent the database from being hacked, hopefully.

The last problem which I ran into, and which ultimately I couldn't find a way around, was a disconnect between encoding sets on the database server and what is commonly used in rss. The database server uses iso-8859-1, and I was trying to use utf-8. I tried a variety of work-arounds, including php functions `html_entity_decode()` and `utf8_encode()`. These functions both take a string as a parameter, so I had to implode the array `$x` into a string and pass it through them. Neither of these functions worked, so I went on to try `htmlspecialchars()`, `htmlentities()` and finally `iconv()` with no success. After consulting with various ibiblians and some friends in Library Systems, we came up with a theory. The data that is inside of the collections index database is submitted by users via a webform, which is done in perl. This data is in utf-8. The mysql server where the database lives is fairly elderly, and is running an early version of mysql, which defaults to iso-8859-1. The script was causing the utf-8 to be squished into iso-8859-1 without being converted, so some data was being lost.

The fix involves dumping the database, scrubbing the data with `iconv`, and recompiling mysql/apache from source, with utf-8 set as the default. This is slated to happen sometime in the next few months, so hopefully my feed will work properly then.

This feed will be displayed on the front page of ibiblio.org, as well as provide the content for

an ibiblio facebook application that is currently being worked on.