Creating an On-line Recipe Collection in XML

Mark Rich

May 14, 2002

1 Goals

For my project, I wished to create a website to facilitate the electronic exchange of cooking recipes. This project had been on a back burner in my mind for a while, and learning about the possibilities for XML query from class inspired me to finally start working. Previously, I formatted my recipes using HTML. However, when a friend gave me the recipe for their latest culinary creation, I had to manually add the HTML tags. I found this to be rather tedious. With this in mind, I set out four major goals for my project:

- *Use XML* - For years, friends of mine have been singing the praises of XML; I wanted to know what all the hype was about.
- *Cheap and Fast* - My recipe database must be complete within this semester. Also, any tools and technologies used in its creation must be free or inexpensive. I wish for this database to be used by anyone without the annoyance of advertisements, as I saw in other widely available on-line recipe sites.
- *Minimal Supervision* - I also wished for this database to eventually run without much daily care and tweaking. This is for two reasons: first, I don't want to spend all my time fixing broken code and mis-formatted recipes, and second, I want to eventually distribute my code and would like to flatten the mandatory learning curve for other administrators.
- *User Friendliness* - Finally, I wanted my family to be able to enter and exchange their recipes. Therefore, if a decision came down to supporting more features of XML versus confusing the user, I erred on the side of the user.

The On-line Recipe Collection (affectionately known as ORC) is the result of my efforts. The ORC meets these four goals; just last week, my cousin cooked and enjoyed "Soy Garlic Chicken," a recipe contributed by my sister-in-law. I look forward to using and modifying this database project for many years to come. The ORC can be found on-line at `http://www.cs.wisc.edu/~richm/recipe-book/`.

2 Tools Used

2.1 XML and RecipeML

XML, the eXtensible Markup Language, has become ubiquitous on the web. XML allows you to use semantic tags to mark the content of your data, as opposed to HTML which is focused more toward format. This way, the internal data of a recipe could be projected to the user individually or as part of a large list of recipes.

As for a recipe DTD, I at first considered creating my own; many XML introductory tutorials start with a simple example using cooking and recipes. A simple web-search turned up RecipeML, a thorough DTD for recipes in XML. While the current version is only 0.5, RecipeML proved comprehensive enough to store a large variety of recipe formats. More information on RecipeML can be found at `http://www.formatdata.com/recipeml/`.

2.2 Xalan and Xerces

With the decision made to store the data in XML, I now needed a way to translate this data into HTML on demand. I chose to learn XSLT, a style-sheet language for XML, written in XML. XSLT performs template replacement, matching the XML tags in a document with the specified output for that tag. XSLT can be used to project your XML data into any format you would like, including HTML. It includes constructs for both imperative and recursive processing, making it a very flexible language.

Currently, only Internet Explorer version 6 natively includes a way to display XML with a given style-sheet, and requiring this browser did not keep with my goals of user-friendliness. Therefore I needed to find an XSLT processor. I first experimented with SAXON, a Java-based implementation, but found this to be too slow due to the startup costs of the Java Virtual Machine for each processing request. I subsequently found the Xalan processor, with versions in both Java and C++. Xalan relies on the Xerces XML parser and this combination proved to be much faster. Both Xalan and Xerces can be found at `http://xml.apache.org/`.

2.3 Perl

Perl has long been the scripting language of choice for Web Developers. With XML and XSLT in hand, I needed an intermediary to run the parser and generate the HTML web-pages. Perl is excellent for handling text and files, with many resources for web developers on-line and free. This project has given me an opportunity to learn Perl with a purpose. Most of the functionality of this project has been implemented in Perl.

2.4 Niagara

My tools were now in place for the creation and manipulation of recipes in XML. To complete the project, I wanted to provide an interface for searching the recipes. This was first accomplished by creating a master list of recipes, such that all new recipes will be added to this list. However, while the master list does give the user access to all the recipes, this list must be manually searched.

Another approach would be to dynamically generate a new XSLT stylesheet for each query on the data and return only those documents that match the search template, but this was rejected in favor of using Niagara, a search engine for XML data written by the University of Wisconsin - Madison Database Group. Niagara allows users to perform "in context" searches, such that queries can look for tags containing query strings or nested tags. Code for Niagara can be found at `http://www.cs.wisc.edu/nigara/`. For this project, only the Search Engine portion of Niagara was used in combination with an interface in Perl.

2.5 WebGFX

Finally, I needed to make the website pleasing to the eye. Usually, I open up Photoshop and draw some graphics of my own. But for this project, since I am going for the cheapest and fastest options, and Photoshop comes with an expensive price tag. I tested out a recent discovery, WebGFX. WebGFX is a free web-based service for creating template-based web-pages. While I designed the layout myself, I used WebGFX to create the sidebar buttons and the top banner. If you need a prototype website quickly, I encourage you to check out WebGFX at `http://www.webgfx.ch/`.

3 Interface

The backbone of the ORC is organized into a core collection of Perl subroutines contained in `orcsubs.pl`. This is responsible for the processing of XML with XSLT to create HTML for both recipes and full lists, communication with Niagara, and adding the HTML template to all generated web-pages. Most of the other scripts make heavy use of these subroutines, with the only other substantial piece of code in `newrecipe.cgi` for processing recipe additions.

3.1 Home Page

Each of the following sections can be reached through the homepage for the ORC. Currently the homepage keeps a list of all my updates and a brief welcome message. In the future, I hope to make this page generated by Perl with each incoming recipe to display a randomly chosen "featured dish" and some advice on how to best use the site.

3.2 Full List

The first place to visit on the ORC is the Full List of recipes. Here, the user can find a short summary of each recipe, sorted by Title. Each summary row lists the title, any selected categories, the contributor and the addition date. Besides title, the recipes can also be viewed sorted by date and contributor. These lists are static HTML files that are updated when a new recipe is added to or deleted from the database. A temporary list of all files is written in XML, and then is processed by XSLT style-sheets using the `document()` element. These list are static so that the time-delay from XSLT processing occurs infrequently.

Each title is linked to provide access to the full details of the recipe. Recipes are displayed using the `showme.cgi` script by passing in the recipe's unique id. The recipe is processed with a XSLT style-sheet and returned to the user for viewing. This individual recipe processing is dynamic, but since it is only one file, the overhead is minimal.

3.3 Search

Eventually, the list of recipes will become too long and cumbersome, and users will move on to the Search page, powered by Niagara. The ORC provides two interfaces to the Niagara Search Engine. The basic search is aimed toward the casual user familiar with normal Internet search engines. Users can select a field, such as "Category," "Author" or "Ingredient," and search for text within these fields. Also, by selecting the option "Anywhere" users can search the whole document.

In addition, the ORC provides the opportunity to construct advanced searches for those adventurous souls. Using SEQL, a Search Engine Query Language written exclusively for Niagara, a user can make as complex of a query as is necessary. An explanation of SEQL is provided, as well as some example queries using RecipeML tags. I am currently uncertain of how Niagara handles attributes within a tag. The query "cat CONTAINS 'veggie' " will return all documents that have a "cat" element with type "veggie", regardless if the element contains the word "veggie". Surprisingly, the query "cat IS 'veggie' " returns no matches.

Immediately, I began to see the benefit of Niagara's "in context" searching. In a ravenous fit, I searched one night for "Cookie" anywhere in the ORC, hoping someone had added some cookie recipes recently. Initially, the returned searches seemed mighty irrelevant, finding recipes like "Zucchini Pie" and "Chicken Supreme." But these inaccuracies were found to be due to their author, my mother, "Cookie Rich." Upon searching for "Cookie" only in the Title, I immediately found some Oatmeal-Cookie Chews, and my hunger was soon satisfied.

The back-end for both query interfaces is identical. An XML query string is constructed from the users input and passed off to the waiting Niagara server. The resulting list of web-site matches is then processed with an XSLT style-sheet and returned to the user with links to each matching recipe, along with the

original query and the number of matches. Upon the server being disconnected or no matches found, an appropriate error message is returned.

3.4 Add Recipe and Tutorial

The lifeblood of the ORC is the addition of new recipes. Without recipe contributions, the Search, List, and Admin pages are useless. Therefore, this portion of ORC must be the most user-friendly. Many of my friends and relations have tested adding a recipe and provided valuable feedback through both direct and indirect means. To demonstrate how to add a recipe, I will explain the sample recipe in the tutorial and fill in the background scripting details.

Adding a recipe to ORC is easy as pi. To begin the process, you are presented with the recipe header form. Required fields are the Title, Source, and Yields, where Yields must begin with a number. You are asked to add a helpful description of your recipe, such as where the recipe originated, preparation tips, and suggested complementary dishes.

Preparation time should be the total time needed to prepare all the ingredients and follow steps before they are placed in the oven or on the stove. Cooking time will be the time spent in the oven or sauteing on the stove and such.

The ORC is able to accept various category descriptions of your recipe: cuisine, meal, and vegetarian. Currently, cuisine and meal are limited to one selection each.

Vegetarian diet options are defined as follows:

- None: Your Standard American Diet (SAD) focused on meat, eggs, dairy, etc.
- Veggie : Shortened nick-name for an Ovo-Lacto Vegetarian. Same as a Vegan, but will also eat milk products and eggs.
- No Cheese : aka: Ovo Vegetarian. Same as a Vegan, but will also eat eggs.
- No Eggs : aka: Lacto Vegetarian. Same as a Vegan, but will also eat milk products.
- Vegan : Does not eat any animal flesh (meat, poultry, fish and seafood) or animal products (eggs and dairy).
- Strict Vegan : Same as Vegan, but also does not eat Honey or any products derived in any way from an animal (including mono/di-glycerides, casein, gelatin, etc.)

Next, you should input the necessary quantities of equipment, ingredients, and steps. If you're not sure, and you're writing the recipe on the fly, don't worry, just estimate. There will be opportunity to add and delete fields as needed later. The maximum number of any quantity at this stage is 99. You are required to input only ingredients and directions, as not all recipes require equipment.

The recipe form is generated by the `newrecipe.cgi` script. It keeps track of the state of the recipe, and updates the page appropriately when the user clicks on "Update or Submit your Recipe." Error checking is performed on all required fields, and the user is not allowed to proceed to entering the actual equipment, ingredients and directions until the header is properly completed.

Upon selection to continue, The recipe form now expanded to include three more sections, the first of which is for Equipment All fields for tools must be filled for a recipe to be complete. Tools may be declared "Optional", causing them to display in a different format. If you find you have too many tools, check the "Delete" option next to unwanted fields. When you need more tools, they can be added one at a time by selecting the "Add another tool" option. Changes will be reflected next time you click on "Update or Submit Recipe".

The next section involves the ingredients of your recipe. As with equipment, each displayed ingredient must be completed, although the only required field is "Item". "Quantity" is the numerical part, where "Unit" is the unit of measurement, i.e. cup, liter, dash, heap. "Preparation", as in minced, cubed and cooked, is also optional, but heartily recommended. Whole ingredients may be declared "Optional", causing them to display in a different format. If you find you have too many items, check the "Delete" option next to unwanted fields. When you need more ingredients, they can be added one at a time by selecting the "Add another ingredient" option. Changes will be reflected next time you click on "Update or Submit Your Recipe".

Finally, the last stage of recipe entry involves the directions. Again, each displayed step must be filled in to properly process the recipe. Steps may be declared "Optional", causing them to display in a different format. If you find you have too many steps, check the "Delete" option next to unwanted fields. When you need more steps, they can be added one at a time by selecting the "Add another step" option. Changes will be reflected next time you click on "Update or Submit your Recipe".

Behind the scenes, the script again checks to make sure all the required fields are present. When the recipe is ready for submission, the script translates the form input into an XML file. The filename is the time of input into the database. This will produce unique identifiers for each recipe as long as no two recipe transactions have begun at the exact same second. The recipe is placed in the data directory, and the three sorted listing files are generated. Finally, the completed recipe is displayed to the user. Ideally, there should be a preview state between when the user successfully completes the form and the recipe is added; this has yet to be implemented.

3.5 Admin

To facilitate easy administration of the ORC, I created a separate page dealing with deletion and editing. One click of a button will delete a specific recipe and regenerate all the recipe listings. This greatly increased my productivity. Recipe editing is a future feature that will also simplify management of the ORC.

To ensure that only authorized users can delete and edit recipes, the `admin.cgi` and `edit.cgi` scripts are placed in a secure directory protected with `htpasswd` protection. The manager is presented with each recipe in the database sorted from most recent to least recent, and the option to delete each recipe.

4 Future Work

4.1 Recipe Editing

All recipes in the database must be edited by hand. This is very inconvenient for the database administrator. I am currently working on a Edit page under the Admin page, to bring an XML document back into an HTML form page for editing and resubmission. This will be accomplished using an XSLT style-sheet and then posting the output to `newrecipe.cgi`.

4.2 Changes to Niagara

Whenever a recipe is added to the database, it is indexed by Niagara. However, should a recipe be altered or deleted, these changes will not be refreshed in Niagara. This could cause erroneous searches and phantom recipes. To fix this, changes must be made to the internal Niagara code. Only a delete portion is necessary, since an edit can be accomplished by deleting and then re-indexing the modified recipe.

4.3 Importing XML

Other recipe sites, such as `http://www.recipezaar.com/`, use the RecipeML data format to store their recipes, and another future project would be to import recipes found on that website into mine. Also, other sites such as `http://www.recipesource.com/` have their recipes stored in flat text files. It would be nice to automatically convert these recipes into XML so they can be indexed and searched via Niagara.

4.4 My ORC

The current interface only allows users to enter their recipes into the general database. It would be more convenient to allow a user to edit and delete their own recipes, similar to a my.yahoo.com or my.wisc.edu interface. This could be facilitated with granting each new user a password. This could involve some use of cookies, and I hope to find time to learn these details of web-programming soon.

4.5 Distributed Orcster and GPL

Eventually, I would like to place the code freely available on the web. Most people I talk to about this project have small modifications they would like

to see in the project, and an open source Gnu Public License (GPL) would facilitate those changes being made without me having to make them. All these individual clients could connect to the same Niagara server; changes would have to introduce flexibility in the location of the XML data.