EsMS: A Database-Driven E-service Management System

Shiyong Lu
Department of Computer Science
University of New York at Stony Brook
Stony Brook, NY 11794-4400, USA
shiyong@cs.sunysb.edu

Ziyang Duan
Department of Computer Science
University of New York at Stony Brook
Stony Brook, NY 11794-4400 USA
dziyang@cs.sunysb.edu

Abstract

We have built a database-driven e-service management system prototype called EsMS. The key idea is separating the management of e-service data from their visual representation. EsMS provides a template language EsHTML for specifying a web page's representation. This language also serves as the language to compose and incorporate e-services from existing e-services, which might be provided by other companies. E-service data are stored in the database, and the schema management provides the flexibility of delivering different e-services within the same framework. Roles are defined to facilitate the close cooperation of template designers, content editors, and e-service programmers. In this paper, we describe the architecture of the system and how our declarative template language EsHTML can be used to achieve these ideas.

1. Introduction

Today, more and more companies are delivering e-services to business and individual customers through the World Wide Web. These e-services include bill payment, delivery of customized news and e-magazines, stock real-time quotes, today's recommendations, etc.. Some e-services like news delivery require quick change, others require that the change ripples throughout a site rapidly. Furthermore, the growing size of today's dynamic business sites make it impossible for all consistent revisions to be managed by a dozen of people. The complexity and speed of change demand automated ways to manage web content effectively and thus deliver up-to-date, consistent e-services efficiently.

On the other hand, new e-services are emerging each day and need to be incorporated into the existing e-service delivery framework seamlessly. Value-added e-services can be composed from existing e-services, possibly offered by different companies. Apparently, in most companies, these incorporation and composition of e-services are conducted by reengineering the existing e-services delivery framework manually.

To meet these requirements, numerous web tools have been built both in the commercial and research fields, notably data-intensive web site management systems. The key idea of these systems is separating the management of a web site's data and structure from their visual representation. The ARANEUS system [2, 3, 11] developed the view definition language ULIXES to build database views of the web, and another language PENELOPE to define derived hypertextual views from relational views. The STRUDEL system [7, 9, 8, 6] used a declarative query language called STRUQL to specify a site's content and structure, and a template language is used to specify a site's representation. Based on the experience of STRUDEL, the TIRAMISU system [1] was built with an open architecture so that it can support other implementation tools. The Re-Web system [5, 12, 4] used ODMG as the object model, OQL as the transformation language, SERF as the OODB evolution facility, and XML as a middle layer presentation between the object model and the final web constructs in HTML. The system focused on the issue of reusable view generation templates at the content level. Other web tools can be found in this good survey [10].

However, most of these tools assume that the structure of a web site can be predefined and will change less frequently while the content or its presentation will change frequently. This assumption is true for a lot of web sites, but might not be appropriate for web sites which deliver e-services since new e-services might be composed or incorporated into the system each day which ensues structural changes. The complexity and dynamic change of a web site's structure make some web tools infeasible, especially those tools which rely on the explicit specification of a web site's structure.

In this paper, we continue this trend by developing the EsMS system with the new requirements above in mind. EsMS uses a declarative template language called EsHTML for specifying a web page's representation; the high level

tags defined in the system allow us to compose and incorporate new e-services from existing e-services without much programming skill. The rest of the paper describes the architecture of EsMS system and how our declarative template language EsHTML can be used to achieve these ideas.

2. System architecture

In this section, firstly we present the data model of e-service items, then we describe the cooperative model in terms of role definitions and their cooperation, and finally we give a description of the functionalities of the major components of the system.

2.1 The data model

An *e-service item* is a well defined object which has a number of fields of different types. We say an e-service item is *composite* if at least one of its fields is another e-service item or a set of other e-service items, or *basic* otherwise. For example, a piece of e-news is a composite e-service item since it might contain pictures, each of which is a basic e-service item. Other examples of basic items include plain texts, HTML texts, images, videos, audios and other media resources.

The presentation of the e-services are defined by *templates*. Each template is defined using HTML, any applicable scripts and our EsHTML tags. Using templates, the management of e-service data is separated from their visual representation. This mechanism has been used in numerous web tools to separate the management of a web site's data and structure from their visual representation [11, 6, 10]. In summary, the separation of e-service data from their visual representation provides the following benefits:

- This facilitates the consistent and quick change of the feel-and-look of existing e-services. One only needs to republish the specified items or all existing items of that type using a new template.
- Different views of the same e-service item can be delivered to different sites and audiences. For example, the same news item can be delivered to news web pages of different languages.
- Content editors are not required to have any knowledge of HTML and they can focus on the management of materials and e-services items; while template designers can spend time on the designing and authoring of the feel-and-look of e-service items.

Templates are stored in the database like e-services items. They not only provide the proper formatting and corporate branding elements, but also serve as an effective way to incorporate different e-services, including those provided by other companies.

A web site is organized in a hierarchical structure and thus naturally, we organize e-service items in a hierarchical way (Figure 1). E-service items are classified into different categories. The composite items are built from other e-service items, each of which might come from different categories. Finally, at the page level, items are published using predefined template. We consider three kinds of pages under each category, each of which is rendered by the corresponding template:

1. **Index**: the default page (e.g., index.html) under a category. It is rendered by an *index template*, in which any tag except *field tag* can be used. Field tag will be discussed in section 3.
2. **Listing**: a collection of pages which serve as lists of items under a category. They are generated by a *listing template*. Conceptually, they are the same as the index page, the only difference is that they do not generate the default page of a category.
3. **Item**: a collection of pages which correspond to all the items at the page level under a category. They are rendered by *item templates*. We require that all the items at the page level under the same category must be of the same type, but different categories might have items of different types at the page level.

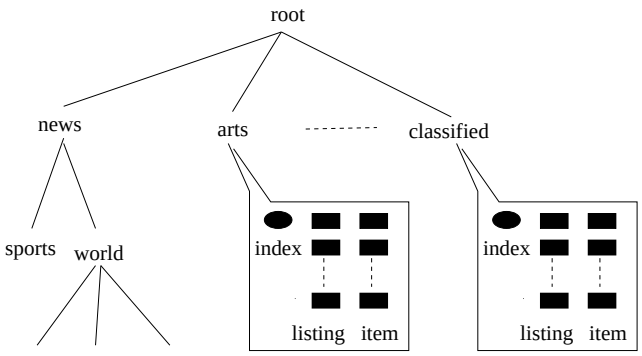

Figure 1. The hierarchical organization of e-service items

2.2 The cooperative model

Roles are defined in terms of the operations that one can perform and the categories that one has privilege to operate on. In EsMS, besides the root who has all the privileges

of all the users in the system, three groups of managers are defined as follows:

1. **Template designers**: They are responsible for designing the presentation of web pages using templates. They generally have artistic talents, and are excellent in the judge of web page formats, layouts, colors, fonts and in using existing web authoring and designing tools to create nice templates. In addition, they are required to know our declarative language EsHTML, which is basically an extension of the standard HTML language with a collection of tag definitions. Each tag corresponds to a presentable e-service item. However, programming skills are not required for them.

2. **Content editors**: These people are responsible for the management of the content of e-service items. They are professionals for the assigned fields. For example, a financial analyst for a financial website and a news editor for a news website. They can use the templates provided by template designers to publish e-service items at the page level. Programming skills as well as HTML knowledge are not required for them.

3. **E-service programmers**: These people define each e-service item in terms of a tag in the EsHTML language and thus provide an interface for the template designers. New tags can be defined and implemented by e-service programmers by complex queries against the database or be composed from existing tags.

EsMS has been built to facilitate the close cooperation of the above three groups of people. Different groups are given different interfaces and thus commit to different responsibilities, so that their work will not interfere with each other. The cooperation interface between e-service programmers and template designers is defined in terms of the set of EsHTML tags, which are to be discussed in section 3; Template designers, on the other hand, design the three kinds of templates for the content editors to render the three kinds of pages under a category, including item pages.

However, in some cases, it is necessary for a particular role to perform some operations that are not usually within his privilege. For example, it might be temporarily necessary for a content editor to add a subcategory although he has no such privilege. This is achieved by the following cooperation: the user submits an authorization request for that operation, and the system will, in term, redirect the request to the owner of that privilege (i.e., whoever has the granting privilege for that operation). The owner of the privilege can choose to grant or reject this request, or perform the requested operation on behalf of the requester. Three kinds of authorizations can be granted: temporary authorization, which is valid only for one operation, short-term authorization, which is valid only for the specified period, and long-term authorization, which is valid until the owner revokes this privilege. To facilitate other general cooperation among the groups and users, a database based email system is incorporated into the system. Each user can either send an email to a particular user, or send to a group of users of the system or of a particular category.

2.3 Major components

Our EsMS system consists of five major components (Figure 2): the category and schema manager, the item manager, the template manager, the page generator, and the EsMS controller. In the following, we describe briefly the functionalities of each component.

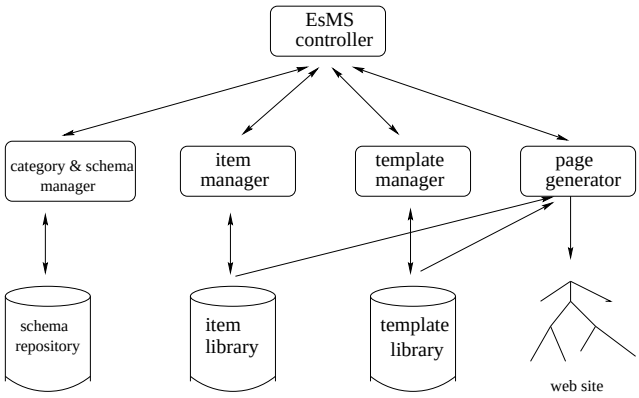

Figure 2. The architecture of EsMS

Category and schema manager

The category and schema manager has two functionalities. One is the management of categories, i.e., the creation, deletion, update and browsing of categories. The other functionality is the management of item schemas. As mentioned above, we assume that the e-service items at the page level must have the same type for each category, but different categories might have items of different types. The schema manager allows one to define the type (i.e., the schema) of items under a category.

Item manager

The item manager allows one to insert, delete or update an item, whose schema is defined by the schema manager. The management of basic items are trivial. For a composite item, the management of component items might be involved. To insert a composite item, based on the schema definition of the item, the system will automatically invoke the insertions of its component items if they do not exist in the system. For example, when a content editor wants to

insert a news item, the system might present an interface automatically to him requiring that related component pictures and audio/video clips to be inserted before the news item is inserted into the system. To delete an existing composite item, its component items are not deleted from the system since they might be shared by other items.

Template manager

The template manager provides the interface of the management of templates. Templates are classified into three types: index template, listing template and item template. They correspond to the three types of pages defined in section 2.1. Several templates of the same type can be defined under a category. This allows easy change of the presentation of a page and multiple presentations of the same service. One can also preview the template using testing items before a template is inserted into the database. For each item type, more than one templates can be defined and one of them is designated as the default template.

Page generator

The page generator interprets the templates to generate the target web pages. For each tag in the template, the page generator will invoke the corresponding program component, which will either retrieve data or perform some tasks, and return the result to the page generator. The result is then formatted according to the predefined format and presented as web pages.

For some e-service items, it might be necessary to republish them after certain period of time. For example, in an index page, there might be an item which corresponds to the 10 most recent news items. As content editors input more news items, this index page needs to be republished periodically. A timer system is incorporated into the page generator. One can subscribe to this system so that a page is republished automatically using a specified template periodically.

EsMS controller

The EsMS controller serves as the coordinator and the controller of other components. In addition, it provides the interface which makes the system web accessible.

3. EsHTML: a template language

Our template language EsHTML is an extension of the standard HTML with a set of well defined tags. The language serves as a bridge between the templates, which define the presentation of e-services at the page level, and the underlying data model. A tag is defined as

```
<# tag_name attribute_list #>
```

where tag_name is the name of a tag, $attribute\_list$ is a list of attribute and value pairs in the form of $attribute = value$. We have defined the following three kinds of tags:

```
<html>
...
<body>
<br>
<# image
  cat ="/images/news"
  name=logo
  format="<img src=\"$(url)\">"
#>
<h1> <# title #> </h1>
<br> time: <# compose_time #>
  source: <# source #>
<p> <# content #>
<br>
<# multimedia
  name=photo
  format="<img src=\"$(url)\">
    <br>$(description)>"
#>
<hr>
<h2> related news: </h2>
<# related #>
...
</html>
```

Figure 3. A template example

1. **Field tag:** Field tags are only used in item templates. They are used to retrieve the fields of the corresponding item that the template is applied, thus, the corresponding item is implicit. For example, tag $title$, $content$ in Figure 3.

2. **Item tag:** Item tags are also used to retrieve the fields of an item, however, this item is explicitly specified in the tag definition. For example, in Figure 3, the $image$ tag is used to retrieve an image named "logo" which belongs to the "/images/news" category. The $\$(url)$ is used to retrieve the url attribute of the image and it is embedded in the *format* attribute such that, when the corresponding item does not exist, none of the string specified in the value of the format attribute will be generated. This is useful for processing pictures in e-news service web site since the number of pictures for a piece of news is not necessarily fixed.

3. **Aggregation tag:** Aggregation tags are used to retrieve the fields of a set of items according to some condition. For example, the following *hot* tag is used to retrieve the top ten hottest sports news.

```
<# hot
   id=x
   category=/news/sports
   number=10
   format="<a href=\"$(url)\">$(title)
      </a><img src=\"$(photo.url)\">"
#>
```

The *id* attribute is optional. If it is present, the rendering result is saved in the x variable, otherwise, it is inserted into the resulting page. It is worthy noting that, the value of format is used repeatedly for each item in the result set and, the fields of other items that is related to the current item can be retrieved. For example, $photo.url$ above is used to retrieve the url of the $photo$ image for each news. This is an example of e-service item composition. When *id* attribute is present, the result set is saved in the variable x temporarily and after which it can be retrieved by the *var* tag to be described next.

4. **Var tag:** As described above, the result item set can be saved in the x variable when the id attribute is used in the aggregation tags. After that, the fields of each item can be retrieved and rendered using the var tag. For example,

```
<# var
   id=x
   index=5
   format="<a href=\"$(url)\">$(title)
      </a><img src=\"$(photo.url)\">"
#>
```

This tag gives the flexibility of rendering each item in the result set.

4. Conclusion and future work

We designed a database driven e-service management system and built a working prototype to demonstrate that our declarative language EsHTML enabled us to separate the content of e-service items from their presentation, thus facilitated the cooperation of different working groups in this framework. To our knowledge, this is one of the first attempts to investigate e-service composition and integration.

The tag system is effective in e-service composition and integration within one EsMS system. We are planning to generalize the tag system to an e-service definition language to support e-service composition and integration within one EsMS system, among several EsMS systems and other third party e-service systems.

Acknowledgements We would like to thank the anonymous referees for their helpful comments.

References

[1] C. Anderson, A. Levy, and D. Weld. Declarative web-site management with TIRAMISU. In *Proceedings of the International Workshop on The Web and Databases (WebDB'99)*, Philadelphia, PA, USA, June 1999.

[2] P. Atzeni, G. Mecca, and P. Merialdo. To weave the web. In *International Conference on Very Large Data Bases*, pages 206–215, 1997.

[3] P. Atzeni, G. Mecca, and P. Merialdo. Design and maintenance of data intensive web sites. In *Proceedings of the Sixth International Conference on Extending Database Technology*, pages 436–450, 1998.

[4] L. Chen, K. T. Claypool, and E. A. Rundensteiner. SERFing the web: The Re-Web approach for web re-structuring. *World Wide Web, Special Issue on "Internet Data Management"*, 3(2), 2000.

[5] K. T. Claypool, J. Jing, and E. A. Rundensteiner. SERF: Schema evolution through an extensible, re-usable and flexible framework. In *International Conference on Information and Knowledge Management (CIKM)*, pages 314–321, Nov 1998.

[6] M. Fernandez, D. Florescu, A. Levy, and D. Suciu. Declarative specification of web sites with STRUDEL. *The VLDB Journal*, 9(1):38–55, 2000.

[7] M. F. Fernandez, D. Florescu, J. Kang, A. Y. Levy, and D. Suciu. STRUDEL: A web-site management system. In J. Peckham, editor, *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 549–552, Tucson, Arizona, USA, May 1997.

[8] M. F. Fernandez, D. Florescu, J. Kang, A. Y. Levy, and D. Suciu. Catching the boat with STRUDEL: Experiences with a web-site management system. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 414–425, Seattle, Washington, USA, June 1998.

[9] M. F. Fernandez, D. Florescu, A. Y. Levy, and D. Suciu. A query language for a web-site management system. *SIGMOD Record*, 26(3):4–11, 1997.

[10] P. Fraternali. Tools and approaches for developing data-intensive web applications: A survey. *Computing Surveys*, 31(3):227–263, 1999.

[11] G. Mecca, P. Atzeni, and P. Merialdo. The ARANEUS web-based management system. *SIGMOD Record*, 27(2):544–546, 1998.

[12] E. A. Rundensteiner, K. T. Claypool, and L. Chen. SERFing the web: A comprehensive approach for web site management. In *Demo Session Proceedings of SIGMOD'2000*, 2000.