Generic Interfaces for Managing Web Data

Oleg Burlaca, *Institute of Mathematics and Computer Science*

Abstract: This paper discusses a generic user interface for managing web data that is incorporated in a content management system. The interface is created at run-time from a set of xml documents stored in database. We accentuate the importance of content analysis phase that leads to a well formed data model. Another important aspect is the use of context in the interface and the hierarchical model to represent multiple relationships between hierarchy items. The proposed event model acts like a glue between data management and application logic.

Keywords: User Interface, Database Access, Conceptual Modeling, Content Management System, XML.

1. Introduction

Significant research have addressed the conceptual modeling [1] and declarative specification [2] of data-intensive websites. Frameworks and paradigms are proposed to built websites from data sources with complex structure, but do not offer means or propose methodologies for managing the underlying data. In other words, there is a lack of tools that offer generic interfaces for managing intertwined(complex) data structures.

The main motivation for this work is NeoSite (http://baraka.neonet.md) a CMS developed by the author for structured web sites. Constantly been improved, it has been in use for the last three years to manage web sites of different complexity.

After the content analysis phase, we try to represent the data of a web site using a lightweight version of the entity relationship model introduced by Chen [3].

This article is structured as follows. The next section describes what incited us to develop a CMS that offers a generic interface. Section 3 explains why the content analysis phase is so important when developing structured web sites. Section 4 highlights some advantages of generic UI, describes some of the problems with web interfaces, reveals our approach to building generic interfaces, and accentuate on relationship and multirelations between entities. A brief outline of the event model that allows to tie the interface to application logic is provided in Section 5. Section 6 gives an overview of related research and tools.

2. Historical Background

Back in 2000, the author entered the web development arena. His duty was to create scripts for web site generation from a set of templates and the data stored in a RDBMS. But someone should somehow, feed the site's data to the database. The most tedious and time consuming task was to develop handy user interfaces (UI) for content authors. Most database systems (we used MySql) contains rudimentary tools for manipulating and querying the data, that are usually insufficient for most users. For this reason, a customized UI to handle interaction between the user and the database was required for each web project.

As the number of web projects grew up, there was an urgent need for a Content Management System (CMS). Using our experience of creating tools for web site management, we started to build our own CMS that will encompass the most needed and reusable components. From our point of view, as developers, at that time, a CMS that offers a tight integration of WWW, DB and FTP services would be sufficient. From a content author and designer perspective, a CMS is a system that enables him to effectively and comfortably publish his content, change the design of the site, notify other CMS users, generate some reports for the administrative staff, etc.

In reality a CMS is a concept rather than a product. It is a concept that embraces a set of processes that will stay at the foundation of the next generation of web sites where content authors will have more privileges, duties and responsibility than designers and developers.

As we mentioned, developing custom UI was (besides information architecture process and conceptual modeling) the most time consuming stage. The two key factors were: 1) complexity and uniqueness of UI being developed 2) the ever changing requirements of web projects (a new field was added to a DB table, the nature (attribute list) of a relation between entities has changed). We desperately needed a high level interface to database; a system that will generate a UI from a set of rules: which fields should be available to the user, how relations are created between records etc. In other words, a generic UI to relational DB systems was needed.

We propose such a generic UI engine, implemented in our NeoSite CMS [4]. NeoSite is a web development environment where the data management facilities plays a central role. The system responsibility is how content are managed (created, updated, related) but not how it is used.

3. Content Analysis

To make a successful CMS for a particular web project, a research phase for content analysis is needed. It helps you reveal patterns and relationships within content and metadata that can be used to better structure, organize, and provide access to that content. The purpose of content analysis is to provide data that's critical to the development of a solid information architecture [5] .

In this work we are mainly concerned with UI to structured web data. Examples of such sites are bookstores, electronic catalogs, e-news. These are sites that include pages composed of data whose overall structure is naturally hierarchical, but exhibits a modest degree of variation. In [6] such pages are called "data rich, ontological narrow, multiple-record web pages". Such sites are composed of index and entity pages. Index pages will be automatically generated from entity(leaf) pages. An entity page consists of 1. internal content: it mainly appears on this page (a long

text, a xml description of a product) 2. external content: attributes that are used to construct index pages, feed search engines. 3. relationships with other entity pages and the specification (attributes) of those relations.

Defining entity types, their relationships and organizing entity collections into a hierarchy that mimics the site structure is a creative process and requires information architects intuition, programmers point of view and customer involvement. After this work is carried out, a XML specification of the UI is elaborated and fed to NeoSite generic UI engine.

Hierarchy is ubiquitous in our lives, because of its pervasiveness, users can easily understand web sites that use hierarchical organization models. They are able to develop a mental model of the site's structure and their location within that structure. This provides context that helps users feel comfortable.

4. Generic User Interfaces

Within the lifecycle of a software project there are often many changes required to the initial design as it progresses, and the developer's software tools are not able to easily cope with these changes. The UI needs to interact with the database and is sensitive to any changes occurred in the database. Such changes will cause the UI to stop working unless the same changes are also applied to the UI. Furthermore, the results of major changes to the underlying database and user interface may require substantial re-testing of the application.

The users desire to view a multitude of data on a single complex screen is usually indulged by the developer even though this may not be desirable. Such complex screens are not scalable and difficult to change.

When developing a database system there are well-established rules of how data should be stored and accessed called Relational Database Theory. "form relieves" said Michelangelo. What if there will be a UI theory with strict rules ? Maybe then, interfaces will be simpler and more consistent than a traditional application, but able to provide much more powerful functionality ? A few obvious advantages of a generic interface are: reduced development time, scalability, no hand coding, less testing, consistent look and feel, standardization, less training.

4.1. Client application vs. Web interface

Almost all commercial CMSs uses a web interface to interact with the user. The central argument is that you don't have to install special software on the client side. You can administer the site from any computer connected to the Internet that have a web browser installed. However, browser interface controls have limited functionality, compared to desktop applications. But the main problem is the stateless editing. HTTP is a stateless protocol and doesn't support stateful interaction with a server. This problem can partially be solved by using a mix of cookies and server sessions imposed over the protocol, but in general, editing content in a web browser is not so pleasant nowadays. The problem becomes even worse when you have to edit a large collection of entities. For example, ERW – a system for handling complex databases through a web browser [7], "solves" this issue by "simulating remote procedure calls". However, their todo list tells that the system will be redesigned from scratch; the first step in this direction is the creation of a working, independent framework for asynchronous remote-script callback invocation.

As you have probably guessed, we are using a client application for managing the site's data. Besides installation issues (each user has to install the software on his machine), there are a lot of "plusses" that are evident when the site is managed by a few number of users.

4.2. The Hierarchy

After the content analysis phase, the identified entity types of the future web site will be hierarchically organized and presented to the user as a treeview. The hierarchy provides user with a decent navigation system, that simplifies the UI by minimizing the number of facilities used to find information. At the same time, the user will always have a clear understanding of his location within that structure due to bread-crumbs. In other words, the *hierarchy provides context*. Another feature is the *implicit relationship* (parent-child) between nodes. Instead of displaying a grid with two columns: 1. product title, 2. category of the product, sometimes a tree with two levels: category as parent with products as children is more appropriate.

However, hierarchies can be limiting from a navigation perspective. You are forced to move up and down and can't jump across branches (lateral navigation) or between multiple levels (vertical navigation). Additionally, some tasks, like editing multiple records with a single operation, will be inconvenient and time consuming when the hierarchy. Shortly, we must have the possibility to view the data from any angle. We tried to address this issue by introducing a special type of entity called "Data shortcut" described in 4.4.

4.3. Entity types

Entity type definitions are a set of xml documents that describes: 1) how an entity will be displayed in the treeview (icon, font); the associated editor plugin 2) which database fields are available, and which editors to use for them 3) which entity types can be related to each other and which attributes describes those relations. 4) triggered events and available actions when working with an entity.

[Fig. 1] illustrates a snippet from the XML definition of an entity type that represents the db fields displayed. [Fig. 2] shows the generated UI.

Database fields that contain large texts and require specialized editors with syntax highlighting are described in the <memos> .. </memos> section.

Besides standard visual editors (ComboBox, CheckBox, DateEdit, TimeEdit…) available in the previous version of NeoSite [4], we have added two useful complex editors: Lookup and ImgEdit. The ImgEdit allows you to browse for an image on your computer for uploading via FTP. The uploading directory from the remote server can be obtained by invoking a server side script with the `id` of the item being edited. The resulting text written to db field is obtained by filling a template like "".

```
…
<memos>
  <f>
     <title> content </title> <lang>en</lang>
     <db_field> content </db_field>
  </f>
</memos>
<fields>
  <f name="thumb" editor="ImgEdit" />
  <category caption="Title" />
  <f name="title_en" caption="En" />
  <f name="title_ro" caption="Ro" />
  <category caption="Brief" />
  <f name="brief_en" editor="Memo" caption="En" height="50"  />
  <f name="brief_ro" editor="Memo" caption="Ro" height="50"  />

  <category caption="Perioada" />
  <multieditor fixed="0" />
     <f name="date1" caption="Start">
        <default>NOW()</default>
     </f>
     <f name="date2" caption="End">
        <default>NOW()</default>
     </f>
  <multieditor />

  <multieditor fixed="0" />
     <f name="ora1" caption="Ora1" editor="Time">
        <options format="HourMin" />
     </f>
     <f name="ora2" caption="Ora2" editor="Time">
        <options format="HourMin" />
     </f>
  <multieditor />
…
```

Figure 1: A snippet from the XML definition of an entity

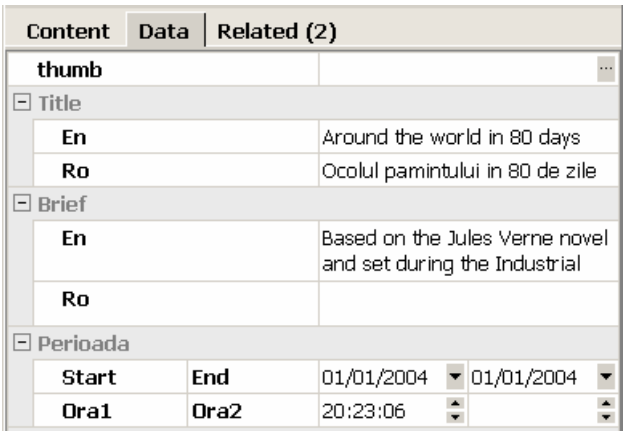

Figure 2: A piece of UI for db field editing

```
<f name="producer_id" editor="Lookup">
   <sql>SELECT id, title FROM site_en
         WHERE parent_id=5  ORDER BY title
   </sql>
   <columns show_header="1">
     <col name="title" caption="" />
     <col name="tel" caption="Phone" width="60" fixed="1" />
   </columns>
</f>
```

Figure 3: Definition of a lookup db field

Such automatizations helps a lot when dealing with lots of images. It also relieves the burden from content authors.
The Lookup editor display dataset records within a dropdown window in a column-based format. These records are fetched based on a SQL Select. [Fig. 3] illustrates the definition of a lookup db field.
NeoSite has a modular architercture based on plugins. A plugin is a DLL library. Each entity type can have it's own plugin for editing.

4.4. Data Shortcuts

It has been said that hierarchies can be limiting from a navigation perspective and have the possibility to view the data from any angle. For this reason, a special editor plugin was developed, called EdSql. It fetches a collection (described by a sql query) of entities from the hierarchy and has two purposes: 1) additional navigation pathway 2) editing multiple records with a single operation. [Fig. 4]

When double-clicking the 'ShortCut' node, a grid is displayed containing live data, the user can use grid inplace editing or press 'Ctrl+Enter' to find&focus the selected item in the hierarchy.

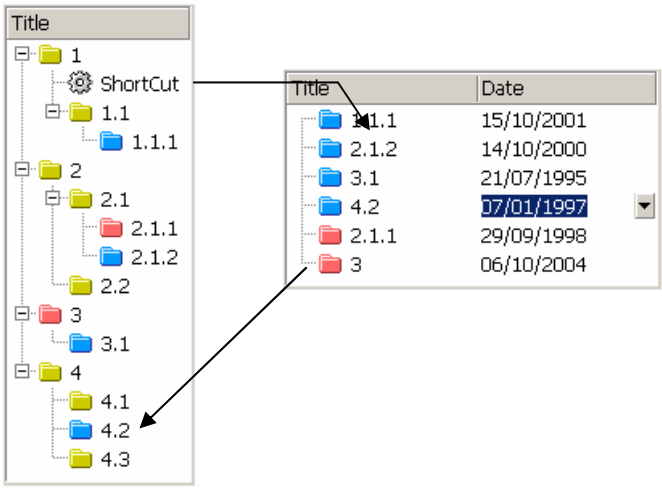

Figure 4: Data Shortcuts

4.5. Relationships

The basic idea of Entity-Relationship modeling, introduced by Chen[3], is that using sets and relations we can model objects of the real world and their inter-relationships.

At first, when designing the database structure for a website, adjusting it to a relational normal form was a priority. In practice the theoretical model is often constrained by the way in which the data will be used and speed of access; compromises are introduced into the design.

Today we try to model site's data using only hierarchies and relations between hierarchy nodes. It greatly reduces and simplifies the user interface. Instead of providing a highly customized interface, we separate complex items into smaller parts, meaningfully organized in the hierarchy. If the context provided by the hierarchy is not sufficient to express the meaning of an item collection, explicit relations and

relationship attributes are additionally used to incorporate more "data semantics".

Relations are defined using drag&drop operations.

4.6. Multiple Relationship

Two entities can be related "more than once". For instance, if a set of entertainment places are given, a place 'x' can be related to place 'y' two times: the first relation means that 'x' is near 'y' and the attribute of this relation is the distance in meters between them. The second one means that 'x' and 'y' are similar (ambiance, age targeting).

To store and display relations, the hierarchy model was used again: in order to relate an item multiple times, you'll have to drag&drop it to different parent nodes [Fig. 5]. (Notice the 'Arcada' place). 'Nearby places' and 'Related places' items are abstract entities, automatically related to entities of 'place' type on their creation. Besides allowing multirelations, these abstract entities provides context, helping the user to better understand what a given relation means.

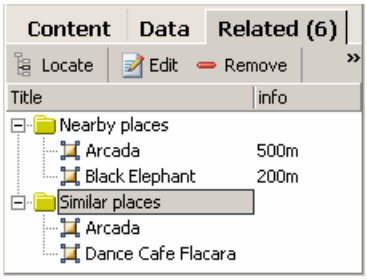

Figure 5: Hierarchical multirelation management

Previous version of NeoSite [4] was able to deal only with single relations, but allowed the user to manage backward references. The practice showed that backward references are seldom used and only bothers the user with additional information.

5. Talking to the Logic

A generic solution will offer only basic functionality. The attempt to endow it with complex features that will encompass a large spectrum of tasks will end up in a flexible (maybe) but hardly manageable software creature. A generic framework should "connect", not "implement" project specific or narrow features. To accomplish this objective, NeoSite offers a high level event and action mechanism. As we have mentioned in Section 4.3, an entity definition contains the description of it's events [Fig. 6] and actions.

Entity action definition are similar to events. Actions are directly activated by the user through a run-time generated toolbar. Each of it's buttons have an associated action. Buttons may have an icon specified in the entity definition.

Events a processed consecutively, the output of an event is one of the input parameters of the next event. This model adds workflow capabilities to our system.

```
<events>
  <OnCreate>
     <e type="sql">INSERT INTO site_en_related (from_id, to_id,
from_tip, to_tip) VALUES ($id, 444, 10, 1)</e>

     <e type="http">
        <params>
           <url>http://neonet.md/cgi/build.cgi</url>
           <prms>id=$id</prms>
           <urgent>0</urgent>
           <method>get</method>
        </params>
     </e>
  </OnCreate>

  <OnCloseAfterCreate>  ...  </OnCloseAfterCreate>
  <OnOpen> ... </OnOpen>
  <OnRelateNode> ...  </OnRelateNode>
  ...
</events>
```

Figure 6: Entity events

6. Related Research and Tools

A generic query tool [8] that dynamically creates its user interface, based on xml configuration files, enables the user to query a metadata store through filters that impose search criteria on attributes. It can be seen as a complementary tool for NeoSite CMS, that shares the same metadata scheme described in a xml file.

Discovering a generic paradigm to manage complex data is a challenging task. The QSByE [6] interface for querying semistructured data allows representing complex objects with arbitrary hierarchy levels, presenting variations in their structure, as nested tables. Such approach can be used to provide an insight of a complex structure, but is not appropriate to edit it.

The Arepo Platform [9] is a set of development tools, used to generate a user interface directly from the database and supplementary metadata. Playing with the online demo, and investigating the docs on their site, we didn't find a way to specify relationships between records.

The closest approach to our tool is ERW [7]: a set of specifications and tools that makes it easy to create, modify and maintain via web a database described by an entity-relationship schema. It has a stable entity-relationship language (ERL) and the associated algorithm. ERW has a well established theoretical foundation, but provides a modest interface to the underlying gears. Because they use a web interface, it makes impracticable to edit large databases. A common "limitation" of both tools: ERW and NeoSite, is that it handles binary relations only. If you really need n-ary relations, you'll have to factor them. The main reason for this limitation is that it is very difficult to device a generic user interface for n-ary relations that will adapt to every situation.

7. Conclusions

This paper has discussed a hierarchical approach to manage entity multirelations in an intuitively clear and convenient way. It also proposed the "Data Shortcut" paradigm as a facility for navigation deep hierarchies and view data from different angles.

Even the grandest project depends on the success of the smallest components. Web projects that were implemented

using the proposed approach and software (NeoSite CMS [4]) demonstrate the effectiveness of the proposed model for managing web data.

As future work, we intend to investigate how to build an interface for managing n-ary relationship.

REFERENCES

[1] Stefano Ceri, Piero Fraternali, Maristella Mareta. "Conceptual Modeling of Data-Intensive Web Applications", *IEEE Internet Computing*, August 2002, pag. 20-30

[2] Mary F. Fernandez, Daniela Florescu, Jaewoo Kang, Alon Y. Levy, Dan Suciu. "Catching the Boat with Strudel: Experiences with a Web-Site Management System", *SIGMOD Conference* 1998: pp. 414-425

[3] P.P. Chen. The entity-relationship model: towards a unified view of data", ACM Trans, on Database Systems, 1(1), 1996, pp. 9-36

[4] Oleg Burlaca, "NeoSite: A Simple Content Management System", *Computer Science Journal of Moldova*, vol.12, no.1(34), 2004, http://baraka.neonet.md

[5] Louis Rosenfeld, Peter Morville, "Information Architecture for the World Wide Web: Designing Large-Scale Web Sites" (Book), *O'Reilly,* 2 edition, August, 2002. Chapter 10.4.2.

[6] Irna M.R. Evangelista Filha, Altigran S. da Silva, Alberto H. F. Laender, David W. Embley "Using Nested Tables for Representing and Querying Semistructured Web Data", *Springer-Verlag,* 2002.

[7] Sebastiano Vigna. "ERW: Entities and relationships on the web", *Poster Proc. of Eleventh International World Wide Web Conference*, Honolulu, USA, 2002. http://vigna.dsi.unimi.it/

[8] B. Verhoeven, E. Duval, H. Olivie, "A Generic Metadata Query Tool", *WebNet*, pp. 1122 - 1127

[9] Arepo Solutions Ltd, http://www.arepo.com