AN ARCHITECTURE FOR THE RAPID DEVELOPMENT OF XML-BASED WEB APPLICATIONS

José Paulo Leal
DCC-FC & CRACS, University of Porto / R. Campo Alegre, 823 – 4150-180 Porto, Portugal
zp@dcc.fc.up.pt

Jorge Braz Gonalves
ESEG, Polytechnic Institute of Guarda / Av. Dr. Francisco S Carneiro, 50 – 6301-559 Guarda, Portugal
jgoncalves@ipg.pt

Keywords: Web application generation, XML transformations, Architectural patterns, Data management applications

Abstract: Our research goal is the generation of working web applications from high level specifications. Based on our experience in using XML transformations for that purpose, we applied this approach to the rapid development of database management applications. The result is an architecture that defines of a web application as a set of XML transformations, and generates these transformations using second order transformations from a database schema. We used the Model-View-Controller architectural pattern to assign different roles to transformations, and defined a pipeline of transformations to process an HTTP request. The definition of these transformations is based on a correspondence between data-oriented XML Schema definitions and the Entity-Relationship model. Using this correspondence we were able produce transformations that implement database operations, forms interfaces generators and application controllers, as well as the second order transformations that produce all of them. This paper includes also a description of a RAD system following this architecture that allowed us to perform a critical evaluation of this proposal.

1 INTRODUCTION

The work described in this paper is part a research effort aimed to the automatic creation of web applications from XML specifications. It builds on previous work (Leal and Domingues, 2007) focused on the development of web interfaces to systems with a loose coupling to the web server. In this previous work we develop an XML domain specific language describing the target system and used XML transformations to process that description. This domain specific language lacks the features to be classified as an Architectural Description Languages (ADL) (Medvidovic and Taylor, 2000) but provides a similar level of abstraction, focusing on system and interconnection of its components rather than on lines of code.

The development of prototypes of applications is the corner stone of the Rapid Application Development (RAD). It was proposed by James Martin as a model of iterative software development (Martin, 1991), aiming to shorten the software development cycle, producing faster results and reducing costs without losing quality. The RAD model became popular and has been used on tools for different database management systems and/or programming languages. Some of these RAD tools are used for development of web applications, as is the case of Omnis, Intraweb, RAD-Studio, Delphi-forPHP, WebSnap, and TurboGears, to name a few.

2 OVERALL ARCHITECTURE

The architecture we propose is strongly based on XML technologies. XML (Bray et al., 2006) was originally created as a formalism for defining web formatting languages, with XHTML – the XML compliant successor of HTML – as the first language to use this paradigm. Nevertheless, XML was quickly adopted to transfer and archive data. Being an auto-descriptive and text-based formalism, XML documents are easily processed by different systems, independently of their platform. Although created also for web formatting, XSLT (Clark, 1999) have outgrown its original role of transforming XML documents to XSL-fo or XHTML, and is routinely used to convert between any two XML languages. In our approach all these XML technologies play a role in the configuration of automatically generated applications, and indeed also in its generation.

Applications software and other programs with graphical interaction are usually structured using the Model-View-Controller (MVC) architectural pattern (Gamma et al., 1994). This pattern was proposed by Trygve Reenskaug in 1978 as a design solution for Smalltalk (Reenskaug, 1979). In this architecture we used the MVC pattern to structure a set of transformations that implement the distinct features of the generated applications.

Model data-to-data transformations implement data management operations;

View data-to-XHTML transformations produce forms based interfaces to interact with portions of the database;

Controller data-to-state transformations manage a set of parameters that controls all transformations.

Having identified a set of transformations that can be used to configure a web application, we need to produce these transformations for a given data management application. Assuming a fix set of data management operations, these transformations can be obtained from the schema of the database.

An XML Schema Definition (XSD) is a standard way to describe the structure and content of XML documents. In a sense, an XSD is similar to the schema of a database, although with some limitation when describing relationships.

Our infrastructure to this transformation-based RAD system works in two stages: first an XSD document is processed and generated a set of transformations for a specific application; then these transformations are used for running the generated applications. In the architecture we propose, each stage corresponds to its own individual component: the application generator and the application engine

The *application generator* is itself a web application for generating web applications from XSD documents and managing applications configurations. The generation of web applications results from applying a set of second order transformations to the uploaded XSD document, to the produce first order transformations that define the applications; then these first order transformations are installed in the application engine.

The *application engine*, with a set of first order transformations, is the actual web application generated by this RAD system. In fact, the application engine is common to a set of web applications generated by the same application generator.

3 Transformations

The distinctive feature of this RAD architecture is the use of XSLT transformations to generate and run web applications. In the applications engine transformations act as participants of the Model-View-Controller pattern and are generated from the XML Schema Document modelling the data. This sections starts with the database operations that are the model of the generated application, proceeds with forms interfaces that provide views to the generated applications an concludes with the control of model and views.

To implement database operations on data-oriented XML documents we need to identify a database schema in its structure, given as an XML Schema Document (XSD). To represent the database schema we use the Entity-Relationship Model (ERM) (Elmasri and Navathe, 2003). It is almost straightforward to map the XSD concepts of element and attribute type definitions, into the ERM concepts of entity, attribute and relationship. Only the last - relationships - pose some difficulties since the XML data model (infoset) is not completely equivalent to the ERM.

In the ERM, entity types define sets of entities and is only natural to relate them with elements types with multiple occurrences. In an XSD, an occurrence indicator specifies the minimum and maximum number of times an element of a given type can be repeated. Thus, we use the indicator `maxOccurs` with value ``` ``unbound'' ``` to identify entities in an XSD.

Given a database schema extracted from a XSD, each operation-entity pair can be implemented by a simple XSLT transformation on the database instance, and a few parameters.

Second order transformations are responsible generating these transformations from the XSD. For each entity detected on the XSD (an element type definitions with unbound repetition) and each supported operation (insertion, query, modification, deletion) an XSLT with the appropriated transformation is generated.

We use a similar approach to create also a forms based graphical users interface from the XSD, using also XSLT transformations. To accomplish this, we started by analysing how RAD systems usually map ERM concepts into forms based interfaces and then used the same correspondence between ERM and XSD.

Entity types are usually mapped to forms in the GUI and their attributes to visual controls (such as text boxes, radio buttons, selectors, etc), depending of their type. There are a few special cases regarding attributes.

XML Schema	Entity-Relationship Model	Users Interface
Repeatable element	Entity	Form
Nested Repeatable element	Entity, Relationship (1:N)	Sub-form
ID attribute	Primary key	Text label
IDREF attribute	Relationship (1:1)	Selector (single selection)
IDREFS attribute	Relationship (1:N)	Selector (multiple selection)
Other Attributes	Attribute	Visual control
Simple element (PCDATA)	Attribute	Visual control
Complex element	Structured attribute	Group of controls

Table 1: Correspondence between XSD, ERM and GUIs

Table 1 presents a summary of the correspondence between XSD, the Entity-Relationship Model (ERM) and Graphical Users Interfaces (GUIs) components. This table is the basis to understand how to create a collection of HTML forms to manage the content of data-oriented XML document. The transformations that generate these forms receive as source the database document and produces an HTML form with the current entity as initial value. The current entity is passed to the transformation using parameters.

As in the transformations that implement database operations, second order transformations process the XSD for generating these HTML forms. For each entity detected on the XSD is generated and HTML page with a form containing the controls associated with its ERM attributes.

The first order transformations discussed previously fulfil the roles of Model and View in the application engine. To extend this approach to the complete MVC pattern we define how the Controller can be generated from an XSD document.

For each user, there is a set of parameters used for controlling each and every transformation, and the transformation pipeline itself, that receives also HTTP parameters. In this architecture, all these parameters are grouped in an XML document that resides solely in memory (i.e. is never serialised). Since this document reflects the users interaction state and bound to each users' state, we will refer to it simply as *State*. Some parameters are required by the infrastructure and for that reason will have reserved names, with a standard prefix.

The State document is used both for as target of Model transformations and for extracting parameters to every transformation. Unlike in the previous participants of the MVC pattern, there is a single Control transformation. This transformation is the first to be processed in the transformation pipeline, changing the current values of parameters and thus controlling the following transformations. The Controller transformation processes both the Data and the State (as transformation parameters) and generated a new instance of the State. As usual, the second order transformations that generate the Controller transformation uses the XSD to extract the entities and attributes required to redefine control parameters.

When an HTTP request is received from a web browser it triggers three transformations in sequence, producing an HTML document that is sent back to the browser in the HTTP response.

Two DOM objects - Data and State – have a central role in this process. The former reflects the application data as persisted in a XML document file; there is a Data object for each application managed by the application engine. The latter is the state of the interaction with each user; there is a State object for each active user session. Both these objects are used in all three transformations: the Data object is the transformation data source and the State object contains the transformation parameters.

4 EXPERIMENTAL EVALUATION

The architecture described in the previous sections was implemented as a Java web application using the Tomcat servlet container. In this section we highlight the main issues encountered in the development.

We managed to implement the infrastructure of our RAD system with a very simple design: its consists of pair of servlets - one for the application generator and other for the application engine - and a collection of objects representing applications.

Both servlets manage a set of applications. Each application instance contains a DOM object and a collection of preloaded transformations. The main method of this class invokes a transformation on the data object that is outputted to different objects according to its type: model transformations are copied to the data object itself and serialized in the file system; controller transformations change the DOM object representing the state; view transformations produce HTML outputted to the HTTP response channel.

When the Application class is instanced the corresponding data file is loaded to its DOM object as well as all it's transformations. As some HTTP requests just change the current view and do not activate model transformations, the data object is serialised to its data file only when it is actually changed. This class is thread safe to ensure data integrity in concurrent operations.

As would be expected, the implementation of second order transformations was more challenging. The first issue we has to solve was the different ways in which an element can be typed in an XSD. Just to give a few examples:

- types in XML Schema can be either anonymous or named, and named types can be reuse in several definitions;
- elements and attributes can be grouped and reused in several definitions;
- type definitions can be created by extending other type definitions.

To simplify the detection of entities, attributes and relationships we start by normalising XSD documents. As would be expected, the normalisation is also an XSLT transformation that unfolds all reusable definitions into an XSD only with anonymous types without any for of reference.

The templates for identifying entities, attributes and relationships are imported by all second order transformations that produce the three classes of XSLT transformations from the normalised XSD. Note that only the Controller transformations is resumed to a single transformation. In this evaluation prototype there is a second order transformation for each database operation The number of view transformations depends on the style of application.

The *style* of application is a particular set o second order transformations that can be selected in the application generator when an XSD is uploaded to the system. We implemented a *simple* application style that produces always a form (depending of entity type) for every operation; this style has a single has a single second order transformation. We implemented also an *extended* application style where search operations generate a listing with all matching entities, with anchors for editing operation on that entity; this style has two second order transformations, one for each kind of visualisation. More complex application styles can also be programed, modifying all the three classes of transformations.

5 CONCLUDING REMARKS

This article presents an architecture for rapid development of web applications based on XML documents and transformations. Our goal was the generation of a functional application from an XML specification, in this case an XML Schema definition.

We used the Model-View-Controller architectural pattern to structure a set of transformations and identified different classes of transformations with each participant in that pattern. We shown that a data-oriented XML Schema definition can be corresponded to the classic Entity-Relationship model used for modelling databases. Using this correspondence we were able produce transformations to implement database operations, forms interfaces and applications control, as well as the meta transformations that produce them. We define also a pipeline to process the first order transformations to process an HTTP request. We implemented a system following this architecture to evaluate experimentally the RAD approach we proposed.

REFERENCES

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., and Yergeau, F. (2006). Extensible markup language (xml). http://www.w3.org/TR/xml/.

Clark, J. (1999). Xsl transformations (xslt) w3c recommendation. http://www.w3.org/TR/xslt.

Elmasri, R. and Navathe, S. (2003). *Fundamentals of Database Systems*. Addison Wesley.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design Patterns - Elements of Reusable Object-Oriented software*. Addison Wesley Professional.

Leal, J. and Domingues, M. (2007). Rapid development of web interfaces to heterogeneous systems. In van Leewen et al., J., editor, *SOFSEM 2007: Current Trends in Theory and Practice of Computer*, number 4262 in Lecture Notes in Computer Science, pages 716–725. Springer-Verlag.

Martin, J. (1991). *Rapid Application Development*. Macmillan Coll Div, New York.

Medvidovic, N. and Taylor, R. N. (2000). A classification and comparison framework for software architecture description languages. *IEEE Transactions on Software Engineering*, 26(1).

Reenskaug, T. (1979). Models - views - controllers. Technical report, Xerox PARC.