RADX - Rapid development of web applications in XML

José Paulo Leal and Jorge Braz Gonçalves

DCC-FC, University of Porto
R. Campo Alegre, 823 – 4150 – 180 Porto, Portugal
zp@dcc.fc.up.pt, jgoncalves@ipg.pt

Abstract. This article presents an on-going project whose goal is the fast development of web applications based on the RAD model. The system resulting from this project - RADX - generates web applications for XML data management. RADX consists of two main components: an application engine to run web applications based on XML documents that it is configured using XSLT transformations; a meta-application for generating and managing applications that run with the application engine. The main feature of the meta-application is the ability to generate XSLT configurations from 2nd order transformations, applied to data document type definitions in XML Schema. These configurations can be changed, allowing customization of the application. RADX intends to be a system rapid development of small web application and prototypes of larger systems.

Keywords: XML, RAD, web applications, framework, prototyping.

1 Motivation

The goal of this project is the implementation of a system for rapid and easy development of web applications based on XML [1] documents. These data management applications, with a simple and intuitive interface, are intended for small data sets and/or applications prototypes. XML is increasingly used to transfer and archive data since it enables the interoperability between different platforms and facilitates its future use.

By using XML for data representation, RADX enables the use of XML technologies in all web application layers, avoiding the need for successive format conversions. Web applications are usually composed of three layers, each with its own data model: the presentation layer based on HTML or XHTML trees, the logical layer based on object graphs, and the data layer based on tables of a relational databases management system. The use of different formats in each layer requires several data conversions that have a significant cost.

As RADX uses XML both in the data layer and the presentation layer, we decided to explore the use of XML also in the logic layer using XSLT [2] transformations on XML documents.

2 An example

This section describes, through an example, the development and use of a web application with RADX. This system has two main components: the *Application Manager* and the *Application Engine*. The former is a meta-application that allows the creation, management, and elimination of applications that run in the latter.

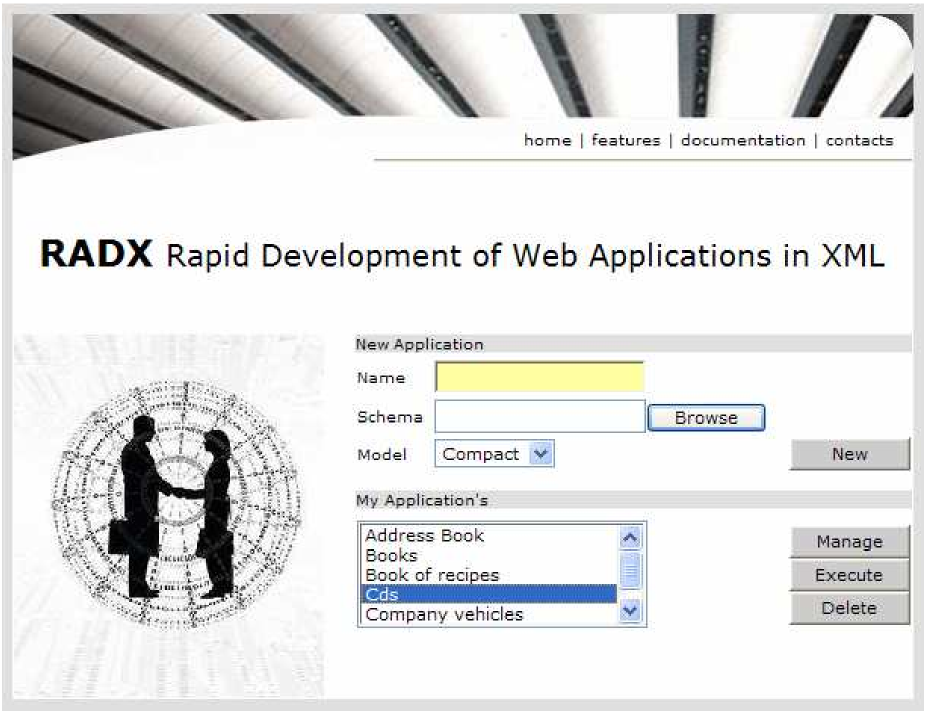


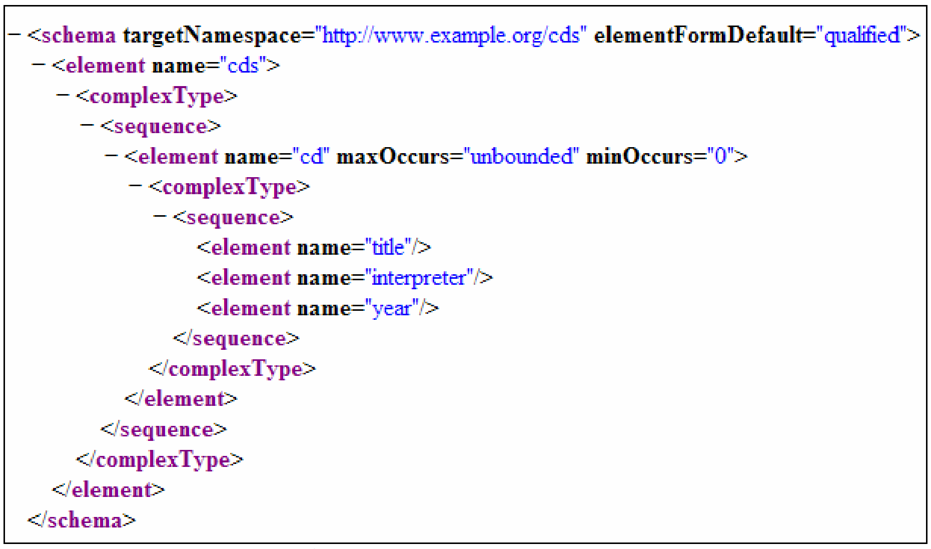
Fig. 1. A screen shot of the Application Manager

The initial screen of the Application Manager is shown on Fig. 1. One of its main features is the ability to generate an application from a XML Schema [3] document. This document describes the structure of the application's data. Usually, this document will be produced in an IDE with specific support for this standard, such as XML Spy, Oxygen or Eclipse.

The upper part of the Application Manager's initial screen is a form for generating a new application. It requires entering the application's name, the location of the XML Schema document and choosing a model for the application's GUI. As part of the creation of a new application, several second order XSLT transformations are executed, producing documents of various types. These documents will be needed by the Application Engine to run this application.

Generated applications are listed on the form on the lower part of the screen (Fig. 1). The selected application can be managed, executed or eliminated. The manage button gives access to a form for editing individual documents generated for an application. The delete button eliminates the application from the system. The execute button launches the selected application on the Application Engine.

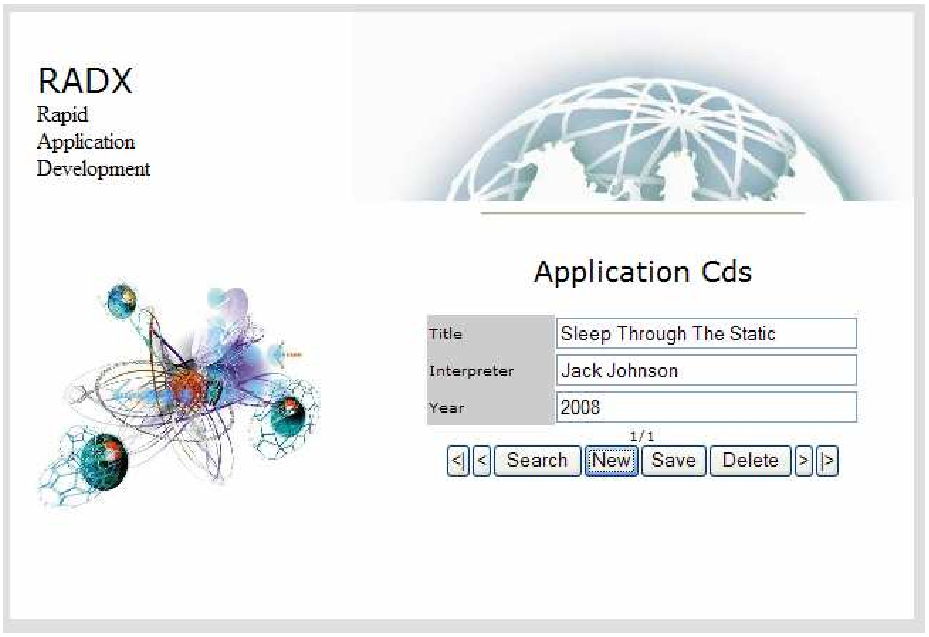
Let us assume we want to generate an application to manage a collection of music CDs. Using a specialized XML Schema editor we have already produced a document type similar to the presented in Fig. 2:



```
- <schema targetNamespace="http://www.example.org/cds" elementFormDefault="qualified">
  - <element name="cds">
    - <complexType>
      - <sequence>
        - <element name="cd" maxOccurs="unbounded" minOccurs="0">
          - <complexType>
            - <sequence>
                <element name="title"/>
                <element name="interpreter"/>
                <element name="year"/>
              </sequence>
            </complexType>
          </element>
        </sequence>
      </complexType>
    </element>
  </schema>
```

Fig. 2. An XML Schema for a CD management application

Using the Application Manager we create a new application named *cds* with the file containing the document on Fig. 2 and the compact GUI model. In the resulting application the user will have access to a range of options: search CDs, create a new CD entry, edit or delete existing CDs and navigate trough them.



Title: Sleep Through The Static
Interpreter: Jack Johnson
Year: 2008

Fig. 3. Application Engine screen shot

The GUI shown on Fig. 3. depends both on the data structure, given by the XML Schema document, and on the selected GUI model. The currently available options are the compact model, used in the example, and the expanded model. The compact model is based on a single form used for all operations: create, modify, search, display, navigate and remove elements. The expanded model has other views for some operations. For instance, the result of a search operation is a list containing the selected elements. Other GUI models are planned for future versions of the Application Manager.

3 Related work

James Martin presented, in 1991, a model of software development known as RAD (Rapid Application Development) with the publication of a book [4] with the same name. This model aims to shorten the software development cycle, producing faster results and reducing costs without losing quality. It also intends to address to the problem of excessive project duration felt in the standard development methodologies commonly used before the 90, when the time to conclude many projects affected their viability.

The RAD model became popular and has been used on tools targeted to 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 [5], Intraweb [6], RAD-Studio [7], Delphi-for–PHP [8], WebSnap [9], TurboGears [10].

However, most of these tools need a significant amount of programming to produce a working application. The goal of RADX is the creation of a working application without any programming. To be sure, in order to customize an application in RADX the programmer may need to edit some configuration files but a working application is created immediately after defining its data schema.

The architecture of RADX is based on the standard MVC (Model-View-Controller) architectural pattern. This pattern is normally used in programs with graphical interaction with the user [11]. This standard was proposed by Trygve Reenskaug in 1978 as a design solution for Smalltalk [12]. Its main purpose is to serve as a mediator between the human mental model and the digital model that actually operates in the computer, facilitating the control of large and complex data structures [13]. In this pattern the model represents the knowledge of the program, the view is a representation of the model in which are highlighted some of their attributes and controller serves as an intermediary between the user and the system, acting on the model and providing views of the model in accordance with the user requests.

The MVC pattern is typically used in the design of graphical applications implemented in object oriented languages. The participants in this pattern are classes of objects. Nevertheless, the MVC concepts of Model, View and Controller can also be used to structure a graphical application without being used in its design. These concepts have been successfully used to separate and structure configuration files in highly configurable web applications [15].

4 Architecture

As mentioned before, RADX integrates two distinct components: an application engine for running web applications operation based on XML, a meta-application for managing applications running on the application engine. This section describes the architecture of each component.

4.1 Application engine

The application engine’s architecture is based on the MVC pattern, a pattern often used in programs with interaction with the user. As outlined in the section 1, we pretend to use XML as the data format, and XSLT transformations as the corner stones of RADX development. Therefore, we try to base each of the participants in the MVC standard - Model, View and Controller in the processing of XML documents.

The model of the RADX applications – the set of its features - is the management of data persisted in XML documents. Therefore, the model can be encoded as a transformation on the data, generating an updated XML document as a result.

The graphical interface of RADX's applications consists of HTML pages that allow the user to view or interact with the data. These HTML pages are the views and are obtained by processing the XML data.

It should be noted that the changes implemented in either the model or view need a set of variables associated with the interaction state as parameters of the transformation. For example, the navigation in a set implies that the present view only shows the element (the equivalent of a record) currently selected. These XSLT transformation parameters are specific to each user and change during his interaction with the application.

The controller, as the name suggests, is responsible for controlling the other constituents of the application, namely the model and view. As explained earlier, these two participants in the MVC pattern are XSLT transformations controlled by a set of parameters that constitute the state. Thus, in order to encode all logic as XSLT files, the model is implemented as a transformation that produces a XML document representing the state.

The application engine is a framework for running web applications. It has three extension points (usually called "hot spots") for each web applications it runs. Those extension points are XSLT transformations and each corresponds to one of the participants of the MVC model.

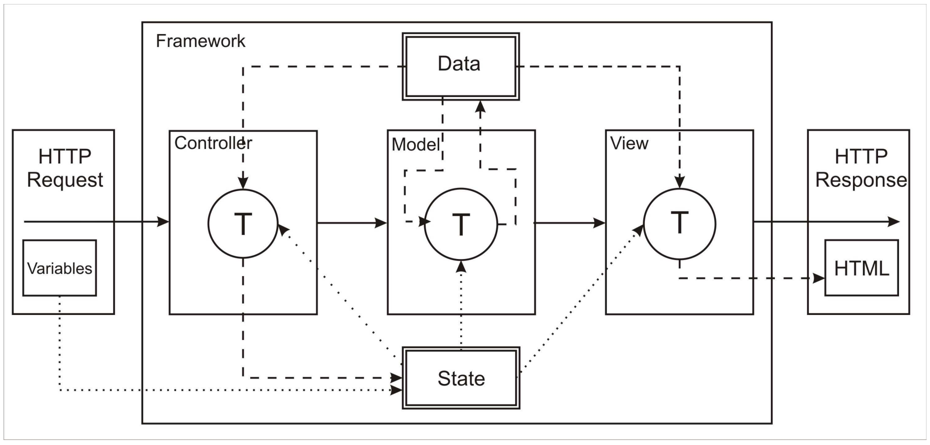


Fig. 4. Processing an HTTP request in the RADX application engine

To better explain this architecture we present in Fig. 1. the processing of an HTTP request received by the RADX application engine. 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. In the diagram we denote execution flow by solid arrows. Transformations are represented by circles with a “T” label, connected by dashed arrows to their input and output documents and by dotted arrows to their parameters.

Two DOM objects – Data and State - have a central role in this process and are both represented by double-line rectangles in the diagram. 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 of the application engine. 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.

In the beginning of this process the variables of the HTTP request are copied to the State object for the current user. The first transformation, representing the controller, uses Data as the original document and changes the State. The second transformation, representing the Model, transforms the Data into itself. The third transformation, representing the View, transforms Data into HTML and sends it in the HTTP response.

4.2 Application Manager

The implementation of the application manager was based on “Model 2” that is the MVC applied to Java web applications. Therefore, the controller is composed by a servlet that receives HTTP requests and according to these requests acts on the model and the view. The Model is a set of Java beans that processes instructions from the servlet. The View is composed of JSPs that uses the model data to produce an HTML output.

The application manager’s function is to manage the applications supported by the application engine. To create the new applications it generates XSLT configuration for the application engine using 2nd order transformations applied to XML Schema type definitions.

5 Implementation

The RADX system was implemented as two independent Java web applications: the application engine and the application manager. They were both implemented using the Tomcat servlet container. In this section we highlight the main issues encountered in the development of these two components.

5.1 Application engine

The application engine has a rather simple design: it has a single servlet that acts as front controller to all applications requests; this servlet instances the Application class for each web application it manages.

The main function of the servlet is to apply the three XSLT transformations associated with the model, view and controller, as explained in the previous section. It also manages the users’ state using the standard session mechanism provided by the servlet container.

Each Application instance contains a DOM object and a collection of 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 produces HTML that is 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 its transformations. It should be noted that some HTTP requests just change the current view and do not activate model transformations. The data object is serialized to its data file only when it actually changes. On the other hand the Application class is thread-safe to ensure data integrity in concurrent operations.

5.2 Application manager

The application manager is a standard model 2 Java web [15] application that allows the creation, customization and deletion of web applications supported by the application engine.

The main feature of the application manager is the creation of a new application when a XML Schema is uploaded. Several second order XSLT transformations are applied to this document in order to create the XSLT files that configure the hot spots of the application engine, as well as the initial XML data file.

The main issue in these transformations is the identification of definitions of elements containing sets of elements of the same type, which would correspond to entities in a relational model. For this purpose we use the XML schema sequence indicator.

Element types in XML Schema can be either named or anonymous. To simplify the detection of type definition with sequence indicators we start by normalizing XML Schema documents. As would be expected, this normalization is also an XSLT transformation that unfolds all named type reference into a XML Schema with only anonymous type definitions.

The same procedure for identifying repeated elements is used for generating each of the three XSLT transformations from the normalized schema. For each set of repeated elements a corresponding set of templates is produced in the target transformation.

Using the application manager the programmer can edit the individual transformations to customize its application. In most cases the XSLT encoding the view will be the first candidate for customization in order to change the application graphical appearance. The controller transformation will need to be edited only to add or remove access to model operations. We expect the model transformation to be the one to require less customization.

6 Conclusion

This article presents RADX, a system for development of web applications prototypes based on XML documents in a rapid and easy way.

The fact that in RADX data persistence and graphical interfaces are both based on XML, led us to use XML transformations in the implementation of the logic layer. The architecture of the Application Engine of RADX is based on applying three successive transformations corresponding to the constituents of the MVC architectural pattern. We also highlight the main issues encountered during the development of the RADX system.

As it is an ongoing project is not yet possible to make an assessment of the RADX system efficiency. In any case, since it is a system designed for prototyping, its main advantage is the ease of applications creation and adequacy of the user's needs.

References

1. Extensible Markup Language (XML), http://www.w3.org/XML/
2. XSL Transformations (XSLT), http://www.w3.org/TR/xslt/
3. XML Schema Definitions (XSD), http://www.w3.org/XML/Schema
4. Martin, J: Rapid Application Development. Macmillan Coll Div, New York, (1991)
5. Omnis Studio - http://www.omnis.net;
6. Intraweb - http://www.atozedsoftware.com/intraWeb;
7. RAD-Studio - http://www.codegear.com/products/radstudio;
8. Delphi-for - PHP - http://www.codegear.com/products/delphi/php;
9. WebSnap - http://dn.codegear.com/article/27404;
10. TurboGears - http://www.turbogears.org.nyud.net/;
11. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns - Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, (1994)
12. Reenskaug, T.: Models-Views-Controllers, Xerox PARC technical note, http://heim.ifi.uio.no/trygver/themes/mvc/mvc-index.html, (1979)
13. Reenskaug, T.: The Model-View-Controller (MVC) - Its Past and Present, http://heim.ifi.uio.no/trygver/themes/mvc/mvc-index.html, (2003)
14. Leal, J., Domingues, M., Rapid development of web interfaces to heterogeneous systems, SOFSEM 2007: Current Trends in Theory and Practice of Computer, pp 716-725, Lecture Notes in Computer Science, Springer-Verlag.
15. Singh, T., Sterns, B., Johnson, M., et al.: Designing Enterprise Applications with the J2EE Platform, Addison-Wesley, (2002)