Theme Park Soil: A Case Study for Using Template Technology in Web-based Environmental Information Systems

Clemens Düpmeier, Werner Geiger
Forschungszentrum Karlsruhe
Institut für Angewandte Informatik

The acceptance of environmental protection measures and an active participation of the persons concerned in their implementation often require a fundamental understanding of the processes and interdependencies in the respective environmental field by the public. In order to support this, web-based multimedia presentations of environmental topics are developed for the interested public. These systems must be attractive for the users, their preparation must be efficient for the developers, and the contents of the systems should be maintainable by the environmental experts themselves. To build such systems, a clean separation between the assignments of marketing and design, preparation and maintenance of (environmental) contents, and technological implementation is necessary. In this paper we will show how template technology can help in achieving this goal and how it is used and extended for the prototype of a concrete system, called Theme Park Soil (in German "Themenpark Boden"), a web-based information system about soil.

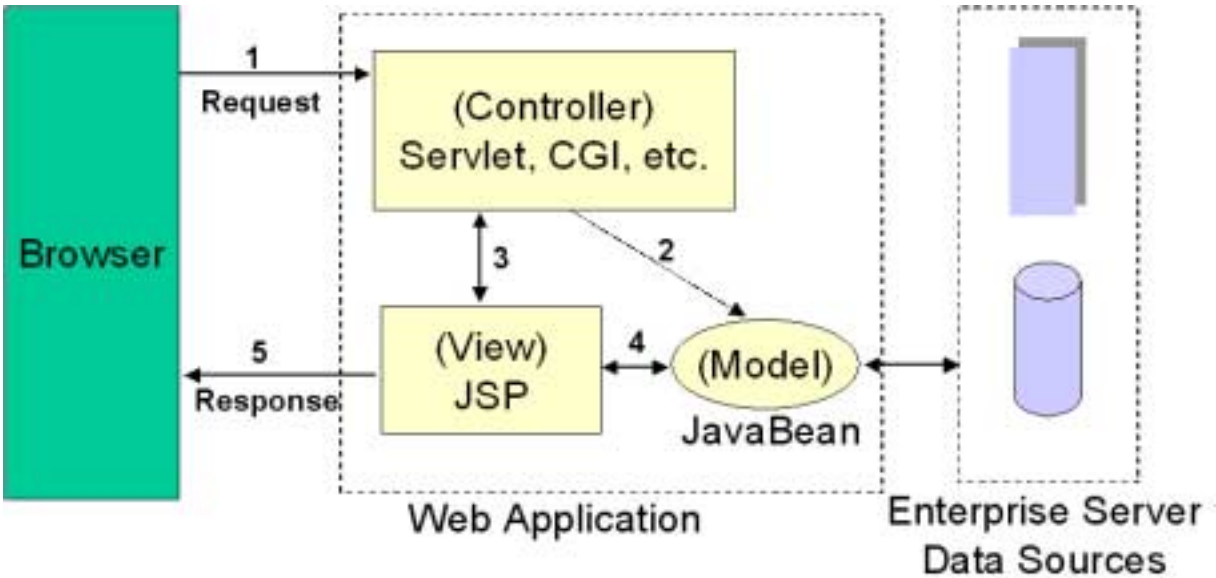

Figure 1: "Model 2 Architecture" for java-based web applications

In modern web-based applications, the presentation of information is often separated from the data model and control flow of the application using a model-view-controller architecture, which is a well-known user interface paradigm in object-oriented programming [1]. In the context of the Java language, the latest versions of the Servlet and JSP (Java Server Pages) frameworks for server-side programming of web applications recommend the use of an ar-

chitecture for web application design, which is known as “Model 2 Architecture” [2], as shown in Figure 1.

In this architecture, the controller of the model-view-controller pattern is implemented as a servlet (in other languages as CGI program), the model as a set of Java Beans (Java Enterprise Beans), and the view as a set of JSP pages (HTML pages containing embedded Java code). When a request arrives, the controller servlet (1) analyses the request and generates some data objects (2) which handle the data requirements related to the request. It then calls an appropriate JSP page (3) which should return the response (5) to the user. The JSP page uses the instantiated data objects (4) to fill in parts of the data into the response pages.

While this design clearly separates the functionality of the application into different parts, the JSP technology (or equivalent technologies like ASP, PHP) allows to mix complex Java code into the HTML pages of the web application far too easily, resulting in pages which are extremely difficult to read both by the programmer and the web designer [3]. Because of this disadvantage of embedding general programming language code into web pages, several template frameworks (like the Velocity framework [4]) have been developed. They define special template languages, which can be used easily by pure web designers to enrich their web pages with dynamic data functionality.

```
<h3>Ozone values in Baden-Württemberg</h3>
<p>
<table>
#foreach ($location in $ozoneStations)
    <tr>
       <td>$location.Name</td>
       <td>$location.ozoneValue</td>
    </tr>
#end
<tr><td colspan=”2”>
      Average: $ozoneStations.averageValue()</td></tr>
</table>
</p>
<p>
Best regards<br>
State Agency of Environmental Affairs
</p>
```

Figure 2: Typical usage of a template language (using the Velocity Markup Language)

As template languages are designed to meet the needs of web designers, their functionality is focused on the job of allowing access to dynamic data objects within the presentation pages and the control of the presentation. Their syntax is easy to understand by programmers as well as by designers. Figure 2 shows the typical usage of a template language based on the Velocity template framework. In this example, the resulting page will print out a table with current ozone values at different places in Baden-Württemberg, a federal state of Germany. In a `#foreach` loop, values are printed out for all locations, each as one row of an HTML table. Every row contains the name of the location and the ozone value at this location. In the last row of the table, averages of the different values are printed.

The programmer’s part of a template framework is used by the controller programmer to select which templates and which data model objects should be used to return the presentation layer of the web application to the user. Therefore, the application programmer places several data model objects into the context (in case of Velocity, a container object associated with the template) of an appropriate template and then calls the template framework to render the template. All data objects within the context of the template are available to the page designer through the template language associated with the chosen template framework.

This usage scenario of template frameworks clearly separates the presentation language of the web designer from the programming environment of the application programmer.

Figure 3 shows how the “Model 2 Architecture” of web applications applies to the usage of template frameworks. In this figure, the JSP view component is substituted by the template framework used.

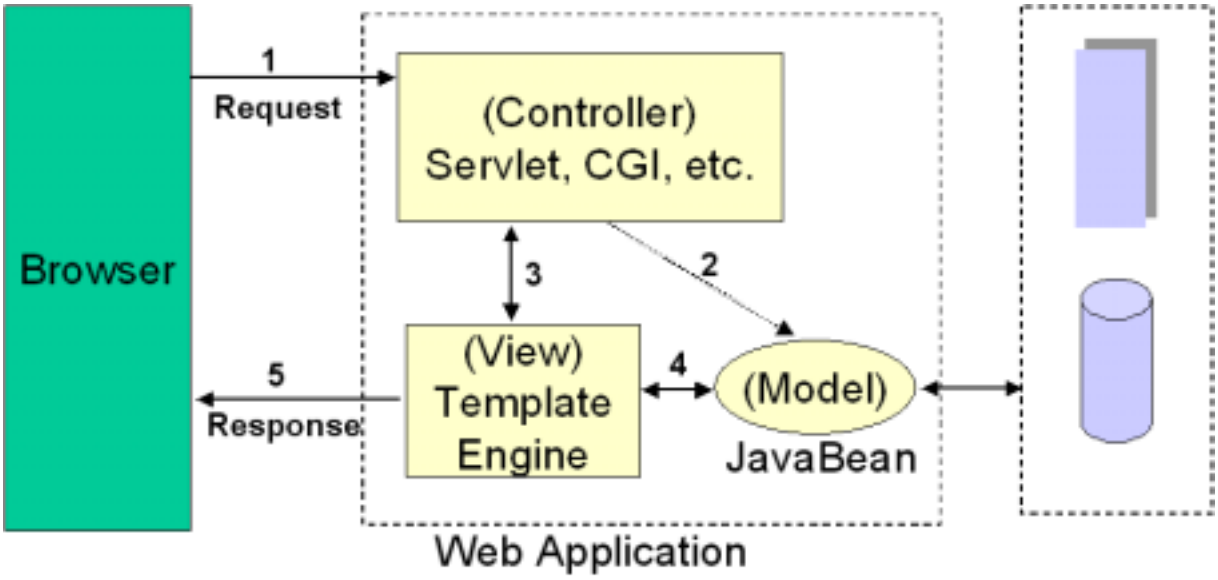

Figure 3: „Model 2 Architecture“ using template framework

The Velocity template framework is used in a web-based environmental information system, called Theme Park Soil [5]. The Theme Park Soil, which is currently implemented as a prototype for the Ministry of Environment and Transport and the State Agency of Environmental Protection of Baden-Württemberg, is a web-based environmental information system describing specific objects in the landscape of Baden-Württemberg, which are related to soil (like moors, soil measurement stations, geological formations associated with specific soil aspects). It is intended to serve as an information system for the general public, which illustrates the significance of different objects and different aspects of soil itself to interested parties. An additional purpose of the system is to make citizens aware of the relationship between these objects and their personal lives, the cultural history, and their environment. In the future, it is planned to extend the Theme Park Soil to a more comprehensive Theme Park Environment, a theme park providing information on landscape objects in Baden-Württemberg, which are related to environmental protection in general.

The data which describe the landscape objects include different multimedia types, like images, 360° panorama images, small videos, database data (like data of measurements) or GIS data. There are some web-based information systems in Germany, which describe soil or other environmental information in an educational manner suitable for non-experts and therefore use multimedia data (see for example [6],[7]). These systems are usually handmade, which means that the web pages and multimedia data of these systems are created manually, often by multimedia companies editing and enhancing the contents of the domain experts in application areas like soil. On the other hand, there are many developments in the area of information science for environmental protection, which are aimed at automatically generating easy-to-understand multimedia presentations of structured database data for the web. Such systems often use GIS or 2D and 3D model technology for their visualizations (see [8], [9]). Web-based environmental information systems, which combine both aspects are the exception (like the “Nature Detectives on the Internet” [10] project, which addressed students and teachers).

The main goal of the Theme Park Soil architecture is the seamless integration of both worlds, mixing different data sources (structured and unstructured data, text and multimedia data, GIS data or automatically generated animations) into consistent views (HTML pages describing the objects) of the presented environmental objects. Because of the complex requirements, we need an architecture of the underlying web application, which allows to combine single information parts from different data sources and maybe other web systems in flexible presentations of the larger environmental objects. Template technology is well suited to fulfil this requirement on the presentation side of our application.

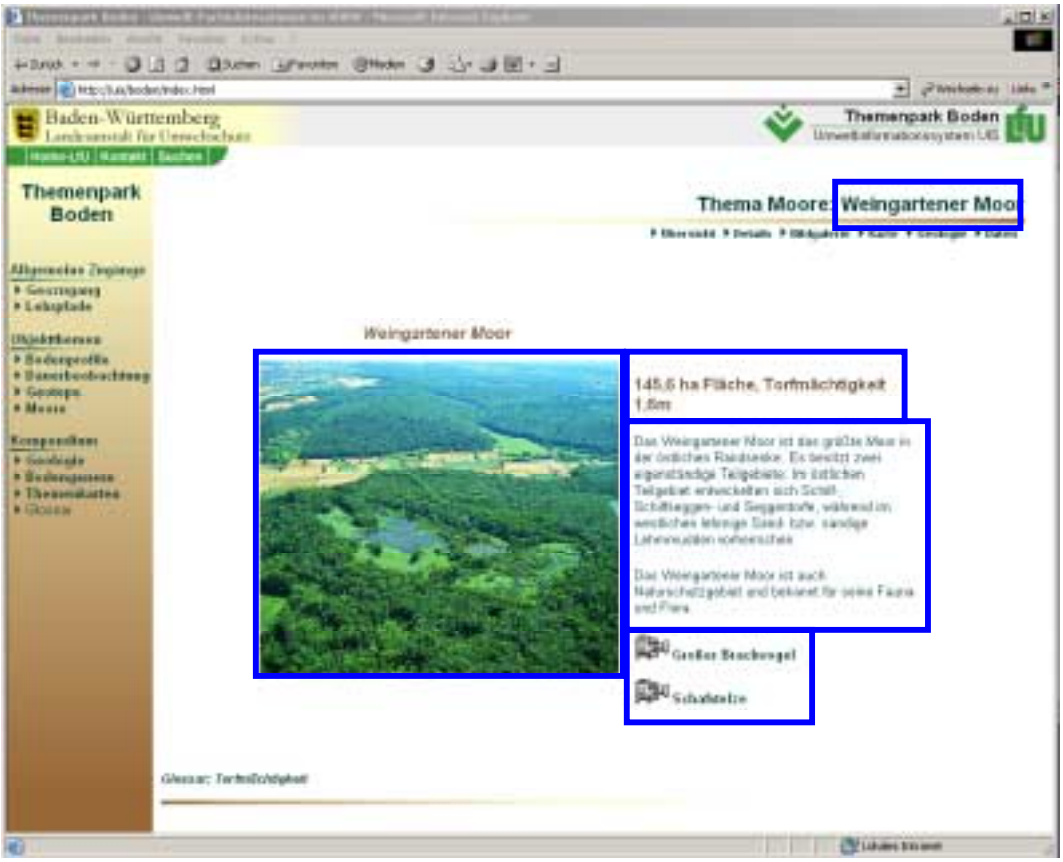

Figure 4: Introduction page of the “Weingartener Moor” object

As examples, Figures 4 and 5 show, how two views of a moor landscape object in the Theme Park Soil prototype are disassembled into the underlying information pieces, which are dynamically inserted into the templates corresponding to the views. On the java side of the information system, these information pieces are parts of data objects (the model) assembled from the different data sources. For example, the URL of the embedded moor image in Figure 4 is part of a `ThemeImage` object, which, apart from the URL, contains further meta information about the image, like a short description of the image, width, and height. These information elements are used by the web designer to integrate the image into this special view of the landscape object. Other information, like the description text or the name of the object, are parts of a data object of the type `ThemeObject`, which incorporates basic information about the shown landscape object. Multimedia data associated with the landscape object (in this example, two videos) are internally managed as a Java Collection object (a list of objects of the type `ThemeMultiMediaObject`). They are inserted into the view as a table of hypertext references to web pages with an embedded quicktime player.

Figure 5 shows a different view of the same environmental object, in which a larger part of the page is inserted into the template through the use of a very special data object (`MapObject`), which is represented in the page by the client side of a map server application [11].

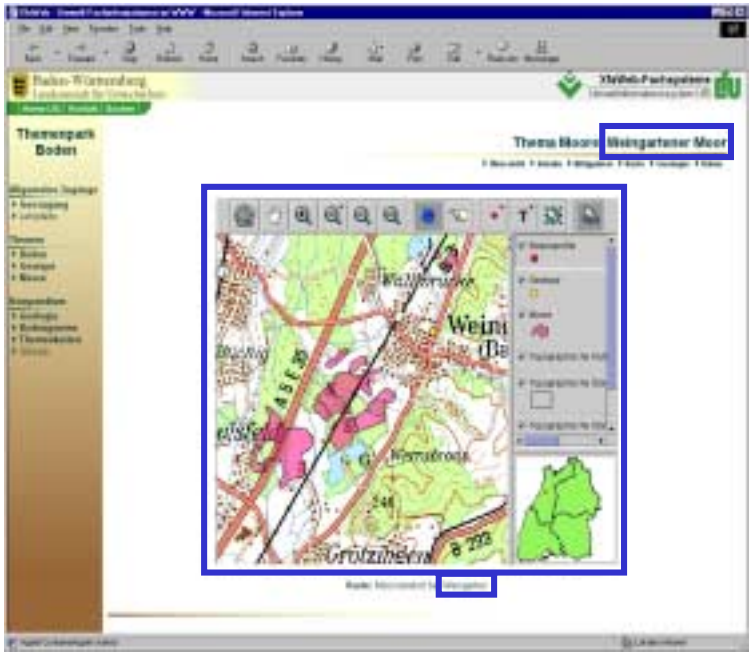

Figure 5: Use of a map server map within the Theme Park Soil

While the application of template technology in the Theme Park Soil seems to be straightforward, the really interesting and conceptually new aspects are hidden in the controlling servlet(s). In simple applications of template frameworks, the association of requests, model data objects, and templates are often hard coded into the controlling servlet(s) of the web application. This makes the implementation straightforward, but the resulting web application lacks flexibility with regard to extensions of the contents, new associations between different parts of the content, etc. There are several web application frameworks, like Turbine ([12]) and Cocoon ([13]), which help to overcome this problem. Cocoon, for example, has an XML-based configuration language allowing to associate different URL patterns with special Cocoon execution pipelines. Each execution pipeline is associated with some data sources supplying data in the XML format and transformer components, which transform the given data into some format (for example, HTML) which can be returned to the requesting user. A Cocoon application can be extended by editing the XML-based configuration file and programming new data source and transformer classes.

For the Theme Park Soil, this idea shall be further extended and specialized by creating an XML-based configuration language and different helper classes that allow to generate complex web-based (environmental) information systems by specifying data sources, models for the data sources, the relationship between different model objects in the form of an application-specific meta data model, and the views of the information system through templates in a set of XML configuration files of the information system.

The contents of such information systems will be stored by expert authors using appropriate back-end data storage systems, like content management systems for textual and multimedia data or database systems for structured data. The implemented helper classes will provide the information system with an easy access to these data by querying them, supplying meta data as specified in the configuration files, and transforming the queried data into runtime objects of the information system, which can then be placed into the context of the corresponding view templates for presentation.

References

[1] G. E. Krasner and S. T. Pope: A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. Journal of Object-Oriented Programming, 1(3):26-49, August/September 1988

[2] G. Seshadri: Understanding JavaServer Pages Model 2 Architecture. JavaWorld Online Article, http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html, December 1999

[3] J Hunter: The Problems with JSP. Servlets.com Online Article, http://www.serlvets.com/soapbox/problems-jsp.html, January 2000

[4] Velocity Template Framework Home Page, http://jakarta.apache.org/velocity

[5] R. Weidemann, C. Düpmeier, W. Geiger, C. Greceanu, Ch. Grieß; Th. Schlachter, G. Zilly (2001): XfaWeb 2001 – Realisierung neuer Funktionalitäten und Inhalte bei den XfaWeb-Systemen aufgrund der Erfahrungen in Baden-Württemberg und Sachsen. in Mayer-Föll, R.; Keitel, A.; Geiger, W. (Eds): Projekt AJA, Anwendung JAVA-basierter Lösungen in den Bereichen Umwelt, Verkehr und Verwaltung, Phase II 2001, Forschungszentrum Karlsruhe, wissenschaftliche Berichte, FZKA 6700, pp. 33 – 56

[6] Bodenwelten, http://www.bodenwelten.de

[7] Grünes Oval, http://www.gruenesOval.de

[8] K. M. Bauer, C. Freytag, L. Neumann: Online-Visualisierung aktueller Umweltdaten innerhalb des WWW-basierten Informationssystems MARS. In H.-D. Haasis, K.C. Ranze (Eds): Umweltinformatik `98, Vernetzte Strukturen in Informatik, Umwelt und Wirtschaft, 12. Internationales Symposium „Informatik für den Umweltschutz“ der Gesellschaft für Informatik (GI), Bremen 1998, pp. 743 – 756

[9] J. Pecar-Ilic, I. Ruzic: Dynamic Web Applications for Temporal and Spatial Presentation of Environmental Data. In L. M. Hilty, P. W. Gilgen (Eds): Sustainability in the Information Society, 15. International Symposium Informatics for Environmental Protection, Part 1: Impacts and Applications, Zurich 2001, pp. 431 – 436

[10] H. Freiberg, H. Voss, M. Drabe: „Naturdetektive im Internet“ – Erlebnisorientierte Umweltbildung über das Internet. In A. B. Cremers, K. Greve (Eds): Umweltinformatik 00, Umweltinformation für Planung, Politik und Öffentlichkeit, 14. Internationales Symposium „Informatik im Umweltschutz“ der Gesellschaft für Informatik (GI), Bonn 2000, pp. 114 - 130

[11] C. Hofmann, A. Otterstätter, R. Caserta, M. Briesen, N. Howind; (2001): disy Cadenza – Die technologische Plattform für UIS-Berichtssysteme in Baden-Württemberg, Bund, Ländern und Kommunen. in Mayer-Föll, R.; Keitel, A.; Geiger, W. (Hrsg): Projekt AJA, Anwendung JAVA-basierter Lösungen in den Bereichen Umwelt, Verkehr und Verwaltung, Phase II 2001, Forschungszentrum Karlsruhe, wissenschaftliche Berichte, FZKA 6700, S. 73 – 98

[12] Turbine Web Application Framework Home Page, http://jakarta.apache.org/turbine

[13] Cocoon XML Framework Home Page, http://xml.apache.org/cocoon