NOVI SAD J. MATH.
VOL. 39, NO. 1, 2009, 111-121

A LIGHTWEIGHT CONFERENCE MANAGEMENT SYSTEM ON ASP.NET

Đorđe Herceg[1], **Željko Marčićević**[2]

Abstract. In this paper, a lightweight conference management system is presented, which was developed on ASP.NET and .NET Framework 3. The system was used to support two conferences, held in 2008 and 2009. Features of the system include participant registration, abstract and article submission, download of conference materials, multilingual user interface, visit logging, role-based security, user management and export of data in XML format. Complex software systems can be difficult to maintain and, in an event of software or hardware failure, they can cause unacceptable delays. Our aim was to create a system that is easy to deploy, maintain, backup and restore.

AMS Mathematics Subject Classification (2000): 68U35

Key words and phrases: Conference Management Systems, ASP.NET

1. Introduction

A conference management system (CMS) is web-based software application that supports the organization of scientific conferences. It helps conference organizers, authors and reviewers perform their conference related duties.

There exist a variety of conference management systems. EasyChair [11] is a well known system that is free, flexible, and easy to use, and has many features. Its primary use is to help program chairs manage the refereeing process. However, it is not possible to install EasyChair on one's own server. All EasyChair installations are hosted at the server located at the University of Manchester Computer Science Department, and access to EasyChair is available on request. ConfTool [12] is another system that provides similar services. Two versions exist: a free, open/shared source version, which is limited to a maximum of 150 participants, and a commercial version which is more flexible and has support for advanced features such as multiple payment options, user role based access, bulk mail and data export functions. ConfTool can be installed on one's own server, and a hosted solution is also offered. OpenConf [13], based on PHP/MySQL, also exists in two versions: the Community Edition is free, and the Professional Edition is offered as a commercial product, with technical support and hosting service. The free version can be installed on one's own server, but it is subject to certain limitations imposed by the OpenConf License, which prohibits hiring a third-party PHP/MySQL professional to set up and maintain the OpenConf installation. COMS

[1]Faculty of Sciences, University of Novi Sad, Trg Dositeja Obradovića 4, Novi Sad, Serbia
e-mail: herceg@dmi.uns.ac.rs

[2]Higher School of Professional and Business Studies, Vladimira Perića - Valtera 4, Novi Sad, Serbia
e-mail: marcic@uns.ac.rs

[14] is a commercial system which is available as a Web application only. It provides functions similar to those of other conference management systems.

The decision to develop our own conference management system, instead of using one of readily available systems, was based on our evaluation of existing conference management systems and several restrictions placed before us, most notably that the system must be free of charge, easy to backup and restore, and that all data must be available even in the event of an unrecoverable system failure. One of the consequences of these restrictions was that the system had to be hosted on our server.

The CMS has to provide the following features:

- Online application of conference participants
- Online submittal of lecture abstracts and conference papers
- Tracking of participants' lectures and their submissions
- Support for multiple languages in the user interface
- Support for a download section with conference materials and access control
- Different access rights for guests, authenticated users and administrators
- Simple user management
- Visit logging
- Easy backup of data
- Easy recovery from system failures
- Access to relevant data even in an event of a system failure

2. System architecture and usage scenarios

Microsoft Windows Server 2003 with Internet Information Services and ASP.NET was chosen as the application server platform. Microsoft SQL Server 2008 was chosen for data storage, except for the download section, which is kept in the file system. Figure 1 shows the overall structure of the CMS. Both the application server and SQL Server are hosted on one computer, running under Microsoft Windows Server 2003. Since the majority of users of the CMS come from the Internet, domain user accounts are not necessary, and the server does not need to be a domain member. ASP.NET Forms Authentication with SQL membership [8] was chosen instead. The user database is kept on the SQL Server.

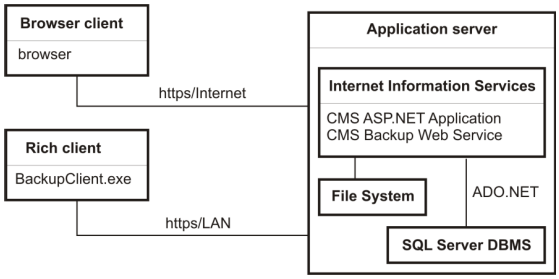

Figure 1: The structure of the CMS

Conference participants access the CMS from the Internet, using their web browsers and a secure (https) channel. Conference organizers can access the CMS either with the web browser, or using a rich client application, which performs a full database backup to an XML file. The rich client application connects to a web service, which provides read-only access to all tables within the CMS database. A simplified form of backup in XML format can also be obtained from the web application. In this case, the data from multiple tables is aggregated in one denormalized table, which is then streamed to the client. This table can be imported into Microsoft Office applications (Excel and Access) for further processing.

The users are presented a web UI, consisting of a menu on the left, and the main content on the right (Figure 2). The menu items are generated dynamically, and their content depends on the user's access level.

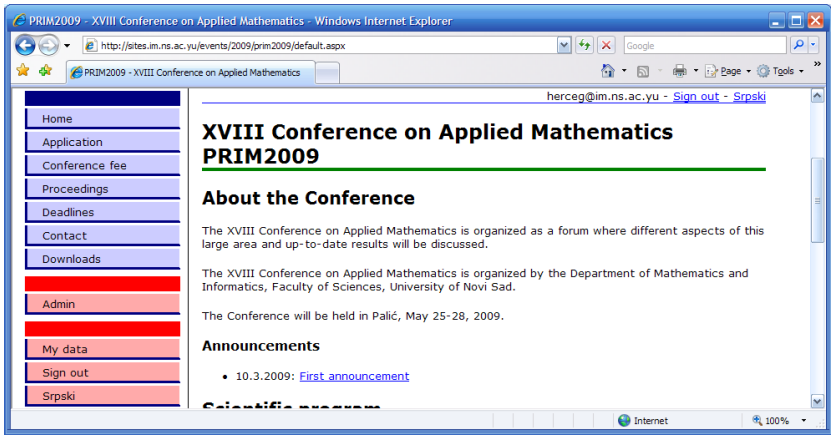

Figure 2: CMS main page, an administrator logged in

A typical usage scenario consists of the following steps:

- A guest user accesses the public part of the CMS and chooses to register.
- The user is presented the choice of entering existing user credentials or creating a new username and password. An option to email a forgotten password to the user is also provided.
- If the user chooses to create a new username and password, he/she does so. For simplicity, user's email address is his/her username.
- The user logs in to the CMS. New hyperlinks appear, enabling access to the data entry forms.
- The user fills out the data entry forms and uploads his/her files (if any), or
- The user downloads files assigned to him/her by the administrators, such as reviewer's comments, or preprints.
- The user then logs off the system, or chooses to browse, correct or cancel previously entered data.

The organizers need to log in to the CMS in the same manner as the ordinary users, and then can perform the following tasks:

- View and download all submissions, or submissions grouped by section,
- Upload new files to the download section,
- Export all submissions into an XML file,
- Activate and deactivate user accounts,
- Reset users' passwords.

Guests can perform the following tasks:

- Access all the public information in the CMS
- Download public files from the download section, such as conference announcements, instructions to authors, submission templates, etc.

3. Security in the CMS

Security is an important aspect of every web application [10]. Security in the CMS is implemented using Forms authentication with URL authorization, which controls access to specific web application directories by the role a user belongs to (Figure 3). The root of a restricted directory subtree contains a web.config file, which specifies access control entries (Listing 1). When an unauthenticated user attempts to access a restricted page, he/she is redirected to the login page, where user credentials can be entered.

An additional level of security is implemented in the download handler, which itself resides in the publicly accessible part of the web site. Since the files from the download section are not affected by the built-in security infrastructure, any user can request any file from the download section. As the download section can contain both the publicly accessible files, and the protected ones, we implemented custom security model, which relies on custom XML files to specify per-folder security. A simple XML file, named folderinfo.xml, containing only the <authorization> tag from web.config and its contents, can be placed in any folder of the download section. Upon receiving client's request for a file, the download handler first searches for and reads the folderinfo.xml file. Download is then either granted or denied, based on the access control entries in the file. On the other hand, if the file is not present, access is granted by default.

```
<configuration>
   ...
   <system.web>
       ...
       <authorization>
           <allow roles="Administrators" />
           <deny users="*" />
       </authorization>
   </system.web>
</configuration>
```

Listing 1: Authorization configuration section of the web.config file

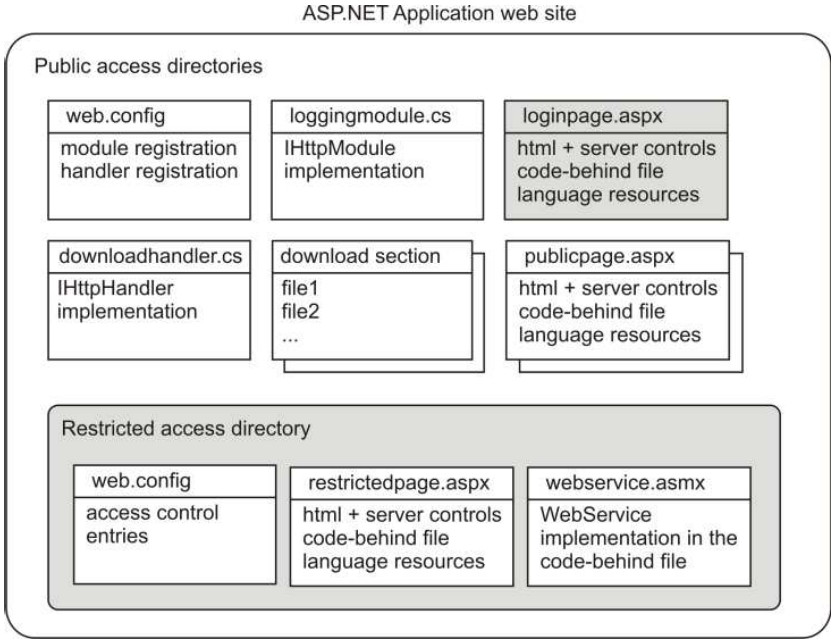

Figure 3: URL authorization in the CMS application web site

Three levels of access are defined for the web application: guest, user and administrator. The first level contains publicly available pages and data, which is accessible to anyone. The second level, containing data entry forms, is available only to authenticated users (i.e. to the users which have a user account and a password). The data displayed on web forms is filtered by the current user's name, effectively restricting the data that a particular user can see and modify. Users at the second level can only view and modify their own data. The third level is for conference organizers (i.e. administrators) and it provides access to all the data, user management and database backup.

The LoginView ASP.NET control was used to restrict the HTML menu items displayed to the user, based on role membership. Listing 2 shows two blocks of server-side code, one of which is processed and presented to the user at a time. The <AnonymousTemplate> contains hyperlinks which are accessible to all users, and the <LoggedInTemplate> contains hyperlinks which are visible to the authenticated users only. In the cases where more subtle control of the user interface was required, a combination of server-side code and named HTML blocks was used for precise control of visibility of specific parts of the user interface.

```
<asp:LoginView ID="LoginView1" runat="server">
    <AnonymousTemplate>
        <p class="HeadingAdmin"> </p>
        <p class="Admin">
            <asp:HyperLink ID="hlLogin" runat="server"
             NavigateUrl="~/Apps/Login/PLogin.aspx"
             Text="Log in" meta:resourcekey="hlLogin" /></p>
```

```
        <p class="Admin">
            <asp:HyperLink ID="hlCreateAccount" runat="server"
             NavigateUrl="~/Apps/Login/CreateUser.aspx"
             text="Create account"
                  meta:resourcekey="hlCreateAccount"/></p>
    </AnonymousTemplate>
    <LoggedInTemplate>
        <p class="HeadingAdmin"> </p>
        <p class="Admin">
            <asp:HyperLink ID="hlMembers"
                 runat="server" NavigateUrl="~/Apps/Members"
             Text="My data" meta:resourcekey="hlMembers" /></p>
        <p class="Admin">
            <asp:HyperLink ID="hlSignOut" runat="server"
             NavigateUrl="~/Apps/Login/Logout.aspx" Text="Sign out"
            meta:resourcekey="hlSignOut" /></p>
    </LoggedInTemplate>
</asp:LoginView>
```

Listing 2: Selective display of menu items based on whether the user has logged in

4. Extensions to the ASP.NET HTTP pipeline

The ASP.NET core infrastructure is based on the HTTP pipeline [6]. It is an extendable, general-purpose framework for server-side programming that serves as the foundation for ASP.NET pages, as well as web services. In order to implement visit logging, we developed a HTTP module (marked with italics in Figure 4) which examines and logs all relevant HTTP requests. Visit logging is triggered by the Application.EndRequest event (Listing 3). As the HTTP module is processed before control is passed to the particular ASP.NET web page, all HTTP requests can be logged, even the invalid ones. We chose to log only requests for certain file types, such as .aspx, .html and .htm.

The download handler, which streams the files from the download section to the client, was implemented as a HTTP handler (marked with italics in Figure 4).

```
public class LoggingModule: IhttpModule {
    ...
    private void Application_EndRequest(Object source, EventArgs e)
    {
        HttpApplication application = (HttpApplication)source;
        HttpContext context = application.Context;
        if (context.AllErrors != null)
            return;
        string path0 = context.Request.Url.LocalPath.ToLower();
        if (((path0.EndsWith(".aspx")) || (path0.EndsWith(".htm")) ||
            path0.EndsWith(".html"))) ||
            (path0.LastIndexOf('.') < path0.LastIndexOf('/')))
        {
            DbDMIWeb.Instance.LogVisit(context);
        }
    }
}
```

Listing 3: The Visit logging HttpModule

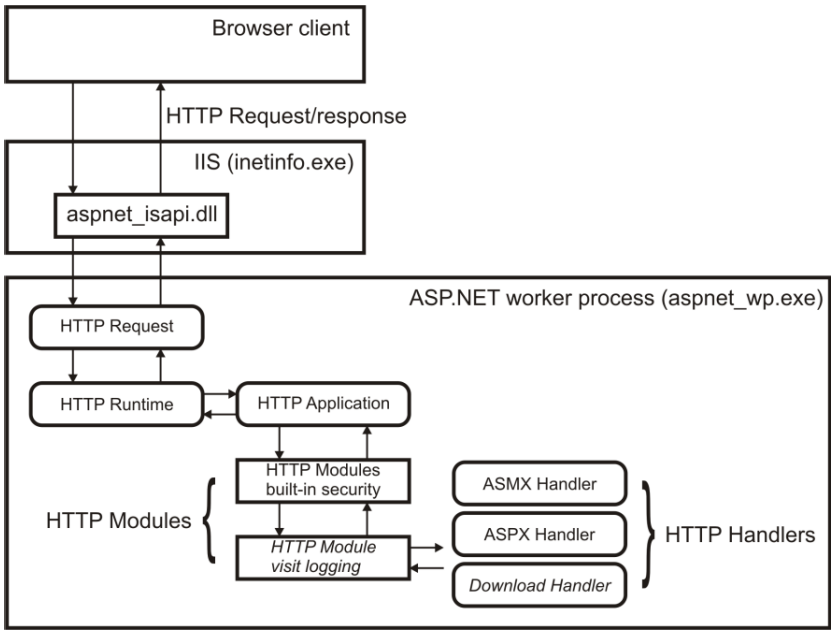

Figure 4: Overview of the extensions to the ASP.NET HTTP pipeline

The visit logging HTTP module and the download HTTP handler are registered in the ASP.NET HTTP pipeline in the system.web section of the web.config file (Listing 4).

```
<httpModules>
   <add name="LoggingModule" type="LoggingModule"/>
</httpModules>
<httpHandlers>
    <add verb="*" path="/Apps/Repository/Download.aspx"
         type="Download"/>
</httpHandlers>
```

Listing 4: Registration of HTTP modules and handlers in web.config

5. Multilanguage support

ASP.NET provides support for localization, which is implemented in several classes in the System.Resources and System.Globalization namespaces. Basically, the .NET Framework has a mechanism for packaging sets of localized resources with ASP.NET applications. Every web page in the application is supposed to have as many resource sets as there are supported languages, for example, a web page main.aspx should have the files **main.aspx.resx** and **main.aspx.sr.resx**, which contain the strings in English and Serbian language respectively. The localized resources are bound to ASP.NET controls declaratively, using the meta:resourcekey attribute, as shown in Listing 3.

```
file MasterPage.master:
    ...
    <asp:HyperLink ID="hlDeadlines" runat="server"
                   NavigateUrl="~/Datumi.aspx"
                   Text="Važni datumi"
                   meta:resourcekey="hlDeadlines">
                   </asp:HyperLink>

file MasterPage.master.resx:
    <root>
      ...
      <data name="hlDeadlines.Text" xml:space="preserve">
        <value>Deadlines</value>
      </data>
    </root>

file MasterPage.master.sr.resx:
    <root>
      ...
      <data name="hlDeadlines.Text" xml:space="preserve">
        <value>Važni datumi</value>
      </data>
    </root>
```

Listing 5: Declarative binding of local resources to an ASP.NET control

One of the problems with the built-in multilanguage support is that each web page is processed separately on the web server, and the choice of language is not persisted between different web pages. Therefore, each time a web page is processed, the culture of the processing thread has to be set to the desired language.

This problem was addressed by implementing the BasePage class, which inherits from System.Web.UI.Page. The BasePage class provides a special implementation of the InitializeCulture method (Listing 6), enabling the web page to keep information about the chosen language as a session variable. This way, each time a BasePage is instantiated, its thread culture is set automatically to the chosen language. All web pages in the CMS inherit from BasePage.

The user switches languages by clicking a dedicated hyperlink, which executes code on the server, that sets the value of the session variable "lang" accordingly. In addition, current user's name is extracted from the HttpContext and put into the appropriate session variable. The UserName session variable is used throughout the CMS for user-specific data filtering in data access forms.

```
public class BasePage {
    ...
    protected override void InitializeCulture()
    {
        Session["Today"] = DateTime.Today;
        Session["UserName"] = HttpContext.Current.User.Identity.Name;

        string lan = Session["lang"] as string;

        if ((lan == null) || (lan == string.Empty))
        {
```

```
            lan = "en";
            Session["lang"] = lan;
        }
        else
     lan = lan.ToLower();

        CultureInfo ci = CultureInfo.CreateSpecificCulture(lan);
        Thread.CurrentThread.CurrentUICulture = ci;
        Thread.CurrentThread.CurrentCulture = ci;
        this.lang = lan;
        base.InitializeCulture();
    }
}
```

Listing 6: Setting the current thread's culture in the BasePage class

6. Backup and export to XML

Backing up the CMS is simple, as only the SQL Server database and the web site files need to be backed up. This task is easily automated using SQL Server maintenance plans and a simple shell script. In the event of a critical failure, the CMS can be restored in only two steps: by restoring the SQL Server database, and the CMS website files.

The application data are represented by a strongly typed dataset and are easily exported to the XML format for backup or further processing. Using the built-in XML support and a web service, which was developed specifically to provide backup from a remote location, we provided export of all the tables from the database in XML format. A specialized backup client was developed for that purpose. A simplified form of backup in XML format can also be obtained from the web application (Figure 5). In this case, the data from several tables is aggregated in one denormalized table, which is then streamed to the client. This table can be imported into Microsoft Excel or Microsoft Access for further processing.

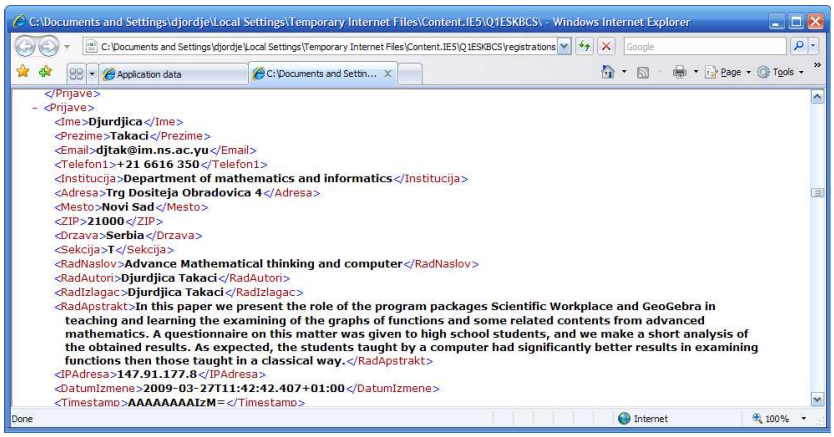

Figure 5: A simplified backup of the conference application data in XML format

7. Conclusion

A Conference Management System was developed in ASP.NET on .NET Framework 3, IIS and SQL Server. The system was designed with ease of use and easy recovery in mind. Some special design requests, such as different levels of authorization for different parts of the web site and multilanguage support were addressed using a combination of the built-in features of ASP.NET and our extensions. Visit logging was implemented in such a way that it enables visitor tracking and behavior anaylsis. In order to satisfy design requirements, features of ASP.NET HTTP pipeline were utilized, extended and customized as necessary.

References

[1] MacDonald, M., Microsoft .NET Distributed Applications: Integrating XML Web Services and .NET Remoting. Microsoft Press, 2003.

[2] Spackman, D., Speaker, M., Enterprise Integration Solutions. Microsoft Press, 2005.

[3] Nilsson, J., .NET Enterprise Design with Visual Basic .NET and SQL Server. Sams Publishing, 2002.

[4] Exstensible Markup Language 1.0 (Second Edition). W3C Recommendation, October 6, 2000.

[5] Ewald, T., Brown, K., Securely Implement Request Processing, Filtering, and Content Redirection with HTTP Pipelines in ASP.NET. MSDN Magazine Online (2002), http://msdn.microsoft.com/en-us/magazine/cc188942.aspx (visited May 2009)

[6] HTTP Handlers and HTTP Modules Overview, MSDN Library, http://msdn.microsoft.com/en-us/library/bb398986.aspx (visited May 2009)

[7] The New Dynamic Language Extensibility Model for ASP.NET, Microsoft ASP.NET, http://www.asp.net/DynamicLanguages/whitepaper/(visited May 2009)

[8] Managing Users by Using Membership, MSDN Library, http://msdn.microsoft.com/en-us/library/tw292whz.aspx (visited May 2009)

[9] Microsoft ASP.NET 2.0 Membership API Extented. CoDe Magazine (2007), http://www.code-magazine.com/Article.aspx?quickid=0703071, (visited May 2009).

[10] Securing ASP.NET Applications, Novologies (2009), http://www.novologies.com/post/2009/04/08/Securing-ASPNET-Applications.aspx (visited May 2009)

[11] EasyChair conference management system, http://www.easychair.org/ (visited May 2009)

[12] ConfTool conference management system, http://www.conftool.net/en/features.html (visited May 2009)

[13] OpenConf conference management system, http://www.openconf.com/ (visited May 2009)

[14] Conference Online-Management System (COMS), `http://www.conference-service.com/conference-support/conference-management-system.html` (visited May 2009)

Received by the editors May 5, 2009