**Beata Panczyk**[1]

## EFFECTIVE WEB APPLICATIONS DEVELOPMENT

***The fundamentals of web applications development for business are some simple approaches: ability to focus on user tasks, avoiding trying to solve all problems and asking relevant question at the early stages of a project. Developer's job is not to solve any business issues but simply to create the requested application in reasonable time and the ability to use the best programming patterns to achieve good performance. The main goal of this paper is to show the possibilities of two modern and competitive technologies (Java Server Faces and ASP.NET MVC) that are very helpful for web applications developers. To demonstrate and compare these technologies, the simple CRUD application will be developed, both in NetBeans IDE (JSF) and Visual Studio Express for Web (ASP.NET MVC).***

***Keywords:*** *web frameworks comparison, JSF, ASP.NET MVC, NetBeans, Visual Studio.*

**Беата Паньчик**

## РОЗРОБКА ЕФЕКТИВНИХ ВЕБ-ДОДАТКІВ

***У статті показано, що в основі розробки веб-додатків для бізнесу лежать кілька простих підходів: здатність зосередитися на користувацьких завданнях, неможливість вирішити всі завдання одночасно і постановка релевантних питань на ранніх стадіях проекту. Завдання розробника – не вирішення всіх ділових питань, а створення програми в розумні терміни з використанням сучасних досягнень програмування. Показано можливості двох сучасних конкурентоспроможних технологій (Java Server Faces і ASP.NET MVC), які корисні для веб-розробників додатків. Щоб продемонструвати і порівняти ці технології, розроблено простий додаток CRUD в NetBeans IDE (JSF) і Visual Studio Express для Інтернету (ASP.NET MVC).***

***Ключові слова:*** *порівняння веб-платформ, JSF, ASP.NET MVC, NetBeans, Visual Studio.*
***Табл. 2. Рис. 4. Літ .13.***

**Беата Паньчик**

## РАЗРАБОТКА ЭФФЕКТИВНЫХ ВЕБ-ПРИЛОЖЕНИЙ

***В статье показано, что в основе разработки веб-приложений для бизнеса лежат несколько простых подходов: способность сосредоточиться на пользовательских задачах, невозможность решить все задачи одновременно и постановка релевантных вопросов на ранних стадиях проекта. Задача разработчика – не решение всех деловых вопросов, а создание приложения в разумные сроки с использованием современных достижений программирования. Показаны возможности двух современных конкурентоспособных технологий (Java Server Faces и ASP.NET MVC), которые полезны для веб-разработчиков приложений. Чтобы продемонстрировать и сравнить эти технологии, разработано простое приложение CRUD в NetBeans IDE (JSF) и Visual Studio Express для Интернета (ASP.NET MVC).***

***Ключевые слова:*** *сравнение веб-платформ, JSF, ASP.NET MVC, NetBeans, Visual Studio.*

Usually web applications are structured around 3 physical tiers: client, application, and database. The application contains the business logic, running on a server and communicates with the client using HTTP. The client, on web applications is a web browser that runs HTML generated by the application layer. Figure 1 presents the structure of the 3 layers of a web application.

[1] PhD, Senior Lecturer, Lublin University of Technology, Poland.

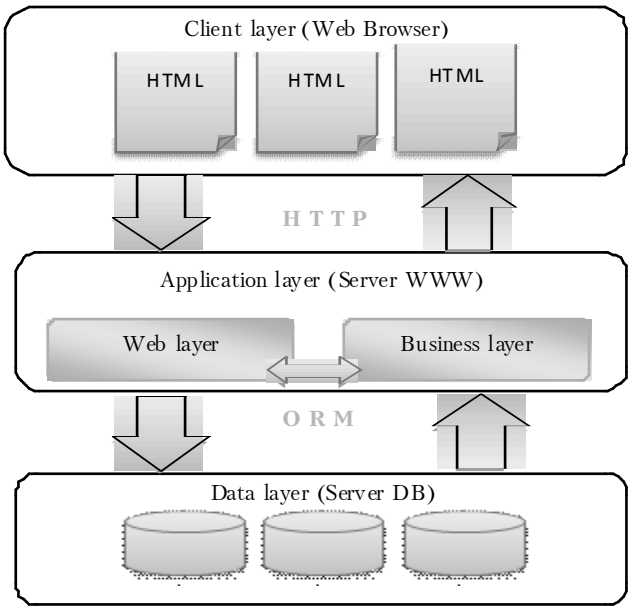


*Figure 1.* **The three-layer web application architecture**

Web applications are built using standard web technologies. They work in any modern Web browser, and can be developed using different tools. A web application framework is designed to support the development of web applications. Frameworks provide libraries for database access, the templates for view and session management.

Most web application frameworks are based on the model-view-controller (MVC) pattern, which separate the data model with business rules from user interface. This is generally considered a good practice as it modularizes code, promotes code reuse, and allows multiple interfaces to be applied. A model represents the state of a particular aspect of the application. A controller handles interactions and updates the model to reflect a change in state of the application, and then passes information to the view. A view accepts necessary information from the controller and renders a user interface to display that information.

Effective web application development will be briefly described on the example of two modern and competitive frameworks: JavaServer Faces (Java Enterprise Edition Platform from Oracle) and ASP.NET MVC (by Microsoft). Figure 2 presents their market position among the most popular web frameworks.

**CRUD application**

The acronym CRUD (Create Read Update Delete) refers to all major functions implemented in relational database applications. Each letter in the acronym can map to a standard SQL statement and HTTP method (Table 1).

*Table 1.* **SQL and HTTP operations (WikiDB, 2013)**

| Operation | SQL | HTTP |
|---|---|---|
| Create | INSERT | POST |
| Read (Retrieve) | SELECT | GET |
| Update (Modify) | UPDATE | PUT / PATCH |
| Delete (Destroy) | DELETE | DELETE |

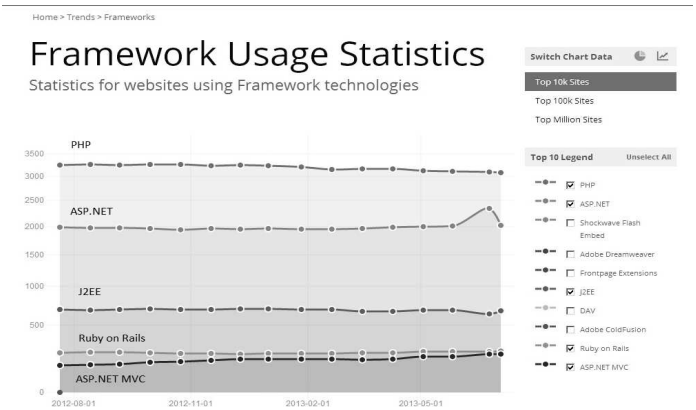


*Figure 2.* **Statistic for websites using Framework technologies (Statistics, 2013)**

Although a relational database provides a common persistence layer in software applications, numerous other persistence layers exist. CRUD functionality can be implemented with an object database, an XML database, text files, custom file formats etc. (WikiDB, 2013).

As a minimum, software must allow a user to (WikiDB, 2013):

– create or add new entries,
– read, retrieve, search, or view existing entries,
– update or edit existing entries,
– delete/deactivate existing entries.

Without at least these 4 operations, a software cannot be considered complete. Because these operations are fundamental, the simple CRUD application will be implemented both in JSF and ASP.NET MVC frameworks. Appropriate programming environments (NetBeans and VisualStudio) enable effective development of such applications.

**JSF and NetBeans**

JavaServer Faces (JSF) (Geary et al., 2008) is a Java specification for building component-based user interfaces for web applications (JSF, 2013; Hall et al., 2009). It is part of the Java Platform, Enterprise Edition (JEE, 2013). JSF 2 uses Facelets as its default templating system. Other view technologies can also be employed (WikiJSF, 2013).

NetBeans is an integrated development environment (IDE) for Java and provides numerous features that enable built-in support for JSF. The IDE's JSF 2.x support builds upon its previous support for JavaServer Faces, and includes convenient editor enhancements for Facelets pages, various facilities for working with entity classes, and a suite of JSF wizards for common development tasks, such as creating JSF managed beans, Facelets templates and composite components (NetBeans, 2013).

If a developer uses Java persistence in his application and has entity classes based on database schema, then the IDE provides functionality that lets him work efficiently with entity class data. To create entity classes from a database table, IDE's Entity Classes from Database wizard may be used. For example, after generating the

entity class from data base test for one table Students (with columns: PESEL, NAME, EMAIL, AVERAGE), the Students Entity class, similar to that shown on the Listing 1, will be automatically generated.

Next, developer can use the IDE's JSF Pages from Entity Classes wizard to create a web interface for displaying and modifying entity class data. The code generated by the wizard is based on persistence annotations contained in the entity classes.

*Listing 1a.* **Students.java (data model) generated from Data Base**

```
@Entity
@Table(name = "STUDENTS")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Students.findAll", query = "SELECT s FROM Students s"),
    @NamedQuery(name = "Students.findByPesel", query = "SELECT s FROM Students s
WHERE s.pesel = :pesel"),
    ...
    })
public class Students implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 11)
    @Column(name = "PESEL")
    private String pesel;
    private String name;
    private String email;
    private double average;
    public Students() {    }
    public Students(String pesel) {        this.pesel = pesel;    }
    public Students(String pesel, double average) {        this.pesel = pesel;        this.average
= average;    }
    public String getPesel() {        return pesel;    }
    public void setPesel(String pesel) {        this.pesel = pesel;    }
...
}
```

For each entity class, the wizard generates the following (NetBeans, 2013):

– a stateless session bean for creation, retrieval, modification and removal of entity instances,

– a JSF session-scoped, managed bean,

– a directory containing four Facelets files for CRUD capabilities (Create.xhtml, Edit.xhtml, List.xhtml, and View.xhtml),

– utility classes used by the JSF managed beans (JsfUtil, PaginationHelper),

– a properties bundle for localized messages, and a corresponding entry in the project's Faces,

– configuration file (a faces-config.xml file is created if one does not already exist),

– auxiliary web files, including a default style sheet for rendered components (jsfcrud.css), and a Facelets template file (template.xhtml).

For example, if we apply the wizard JSF Pages from Entity Classes to the Students entity class, the settings shown in Figure 3 will be generated. Of course it is

possible specify the locations of the generating files. Among others the JSF session-scoped, managed bean named StudentController file (Listing 2) will be created.

The IDE allows also automatically generate code for Facelets page. It is possible to use the data table from Entity dialog to generate a JSF data table that contains columns for all properties contained in an entity class. In order to make use of this facility, we must already have a JSF managed bean created to handle any back-end data associated with the entity class (e.g. Student class). For example, a list of students is displayed using the view page List.xhtml (source code is presented on Listing 3a) and final results in the browser are shown in Figure 4.

**ASP.NET MVC and Visual Studio**

Basing on ASP.NET, ASP.NET MVC (ASPMVC, 2013) allows software developers to build a web application using MVC model. ASP.NET MVC framework is a lightweight, highly testable presentation framework that is integrated with existing ASP.NET features. Some of these integrated features are master pages and membership-based authentication. The ASP.NET MVC Framework couples the models, views, and controllers using interface-based contracts, thereby allowing each component to be easily tested independently. In March 2012 Microsoft had released a part of their web stack (including ASP.NET MVC, Razor and Web API) under an open source license (Apache License 2.0) (WikiASP, 2013). ASP.NET MVC 4 is a framework for building scalable, standards-based web applications using well-established design patterns and the power of ASP.NET and the .NET Framework (Chadwick et al., 2013).

Microsoft Visual Studio is an IDE from Microsoft used to develop web applications for all platforms supported .NET Framework.

To create Web application, Visual Studio uses a default template for the ASP.NET MVC project. To create a view template file, the new Razor view engine introduced with ASP.NET MVC 3 may be used. Razor-based view templates have a .cshtml file extension, and provide an elegant way to create HTML output using C#. Razor minimizes the number of characters and keystrokes required when writing a view template, and enables a fast, fluid coding workflow.

The .NET Framework data-access technology known as the Entity Framework to define and work with these model classes can be used. The Entity Framework (often referred to as EF) supports a development paradigm called Code First. Code First allows creating model objects by writing simple classes. Then the database may be created on the fly from the classes, which enables a very clean and rapid development workflow (ASPMVC, 2013). For example, each instance of the Student object will correspond to a row within a database table, and each property of the Student class will map to a column in the table. An important role is also the StudentDBContext class representing the Entity Framework Student database context, which handles fetching, storing, and updating Student class instances in a database. That class handles the task of connecting to the database and mapping Student objects to database records. A connection information is written in the Web.config file of the application.

ASP.NET MVC automatically creates the CRUD action methods and views (known as scaffolding). So a fully functional web application that lets creating, listing, editing, and deleting Student entries may be quickly generated.

*Listing 1b.* **Student.cs (data model) with the Student and StudentDBContext classes definitions**

```
namespace ASPMVCTest.Models
{
    public class Student
    {
        public int ID { get; set; }
        public string Name { get; set; }
        public string Email { get; set; }
        public string Pesel { get; set; }
        public double Average { get; set; }
    }
    public class StudentDBContext : DbContext
    {
        public DbSet<Student> Student { get; set; }
    }
}
```

**CRUD application development example**

The simple CRUD application (working with the students data) was created using both JSF and ASP.NET MVC frameworks. Figure 3 presents the structure of applications. Gray background indicates the most important and helper files used by MVC model. The comparison of the fragment of the StudentController file is presented in Listing 2a (JSF) and 2b (ASP.NET MVC). Listing 3a presents the fragment of the students list view page generated by JSF and Listing 3b the same for ASP.NET MVC. One of the automatically generated view pages for JSF is shown in Figure 4. Almost the same view is generated by ASP.NET MVC.

Listing 2a. StudentsController.java

```
public class StudentsController implements
Serializable {    private Students current;
    private DataModel items = null;
    private int selectedItemIndex;
    public StudentsController() {    }
    public String prepareList() {
        recreateModel();   return "List";
    }
...
}
```

Listing 2b. StudentController.cs

```
public class StudentController : Controller
{  private StudentDBContext db = new

StudentDBContext();
        // GET: /Student/
        public ActionResult Index()
        { return View(db.Student.ToList());
        }
...
}
```

*Listing 3a.* **The fragment of the JSF View page (List.xhtml)**

```
    <h:dataTable value="#{studentsController.items}" var="item" border="0" cellpadding="2"
cellspacing="0"
                         rowClasses="jsfcrud_odd_row,jsfcrud_even_row" rules="all" style="bor-
der:solid 1px">
        <h:column>
            <f:facet name="header"> <h:outputText
value="#{bundle.ListStudentsTitle_pesel}"/></f:facet>
            <h:outputText value="#{item.pesel}"/>
         </h:column>
    ...
        <h:column>
              <h:commandLink action="#{studentsController.prepareView}"
value="#{bundle.ListStudentsViewLink}"/>
            ...
            </h:column>
```

```
    </h:dataTable>
```

Listing 3b. The fragment of theASP.NET MVC View page (inde x.cshtml)

```
<table>
    <tr>    <th>  @Html.DisplayNameFor(model => model.Name)        </th>
     ...
    </tr>
@foreach (var item in Model) {
    <tr>
        <td>            @Html.DisplayFor(modelItem => item.Name)      </td>
        ...
        <td>            @Html.ActionLink("Edit", "Edit", new { id=item.ID }) |
                          ...
        </td>
    </tr>
}
</table>
```

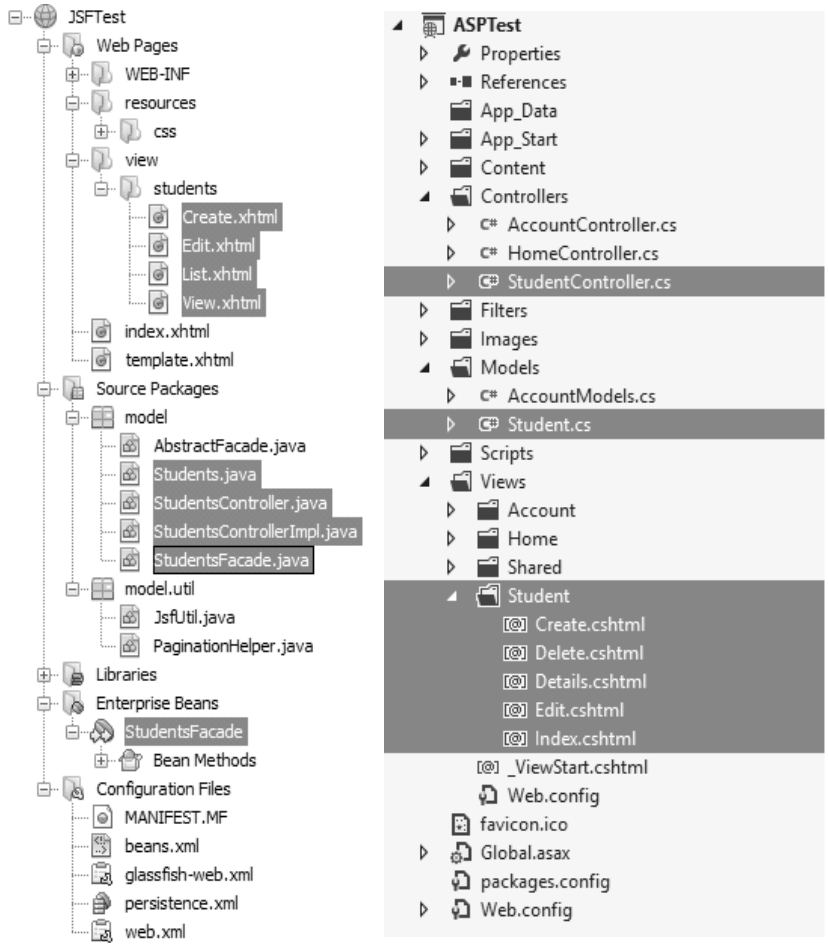


*Figure 3.*

**a) Structure of JSF web application** **b) Structure of ASP.NET MVC web application**

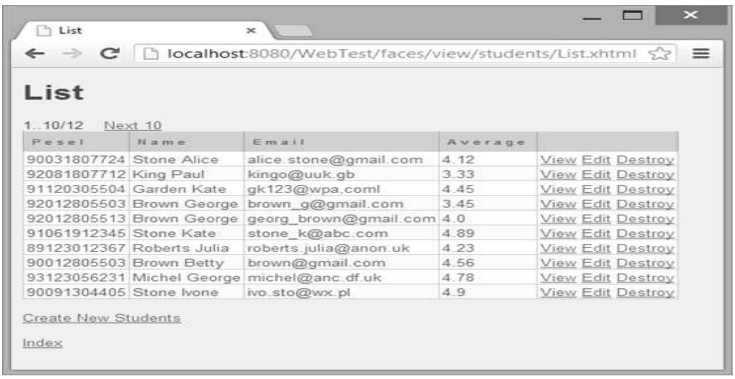


*Figure 4.* **View page with the list of students (JSF)**

Table 2 presents the comparison of the other features of both frameworks.

*Table 2.* **The main features of frameworks**

| | JSF | ASP.NET MVC |
|---|---|---|
| Company | Oracle (earlier Sun Microsystem) | Microsoft |
| Initial specification | March 2004 (JSF 1.0) | May 2000 (ASP)<br>December (2007 ASP.NET MVC) |
| Final release | May 2013 (JSF 2.2) | August 2012 (ASP.NET MVC 4) |
| Programming language | Java | C# |
| Actual Developer Kit | Java EE 7 SDK and GlassFish Server Open Source Edition 4.0 | Microsoft .NET Framework 4.5 (free) and Internet Information Services (IIS 8.0 –Windows 8, Windows Server 2012) |
| IDE | NetBeans (free- Sun), Eclipse(free - IBM) | Visual Studio 2012 (Microsoft) or Visual Studio Express 2012 for Web (free) |
| The recommended view engine | Facelet (introduced in JSF 2.0) | Razor (introduced in ASP.NET MVC 3) |
| Others possible view engines | JSP | Web Forms, NDjango, SharpTiles, String Templates etc. |
| Runtime and application servers | Server and IDE-providers to choose from – better flexibility but some extra overhead for the developer to get the different pieces to work together | All from Microsoft as a standard package, required Windows servers |
| Open source servers | GlashFish, Apache Tomcat | No (only IIS) |
| Support | Well used and supported in this market | Well used and supported in this market |
| License agreements | An open-source license | An open source license (Apache License 2.0). |
| Standards based | Yes | Yes |
| View page standard | XHTML1.1 (HTML5 possible in JSF 2.2) | HTML5 (ASP.NET MVC 3 and 4) |
| Platform independence | Runs on several operating systems including Windows, Mac and Linux | Primarily for Windows. Although the open source project Mono is developing a multi-platform runtime for .Net |
| Weight of the simple CRUD application | 300kB (53 files, 28 folders) | 68,3MB (532 files, 218 folders) |
| Functionality | There is not a big difference between the two | |

**Conclusion.**

In a brief article it is not possible to present all the possibilities of both technologies. Nevertheless the ASP.NET MVC 4 and JSF 2.2 are good choice for web application developers. Generally both technologies offer modern integrated development environment and allow programming web applications according to the actual networks standards. Because efficient web application developing is today one of the most wanted ability in IT companies, knowledge of various modern frameworks is indicated for all developers. It is also very important in education of students at computer science departments.

**References:**

ASPMVC (2013). http://www.asp.net/mvc/tutorials, ASP.NET MVC tutorial, accessed Jun 2013.

*Chadwick, J., Snyder, T., Panda, H.* (2013). ASP.NET MVC 4, O'REILLY, Helion.

*Geary, D., Horstmann, C. S.* (2008). Core JavaServer Faces. Helion, Gliwice.

*Hall, M., Brown, L.* (2009). Java Servlet i JavaServer Pages. Tom I i II, Helion, Gliwice.

JEE (2013). http://docs.oracle.com/javaee/6/tutorial/doc/bnaax.html, JEE model, accessed Jun 2013.

JSF (2013). http://docs.oracle.com/javaee/6/tutorial/doc/bnaph.html, JSF tutorial, accessed Jun 2013.

NetBeans (2013). https://netbeans.org/kb/docs/web/jsf20-support.html, JSF 2.x Support in NetBeans IDE, accessed Jun 2013.

*Rychlicki-Kicior, K.* (2010). Java EE6 Programowanie aplikacji WWW, Helion, Gliwice.

*Sanderson, S., Freeman, A.* (2012). ASP.NET MVC 3 Framework. Zaawansowane programowanie, APRESS, Helion.

Statistics (2013). http://trends.builtwith.com/framework, Framework Usage Statistic, accessed Jun 2013.

WikiASP (2013). http://en.wikipedia.org/wiki/ASP.NET_MVC_Framework, accessed Jun 2013.

WikiDB (2013) http://en.wikipedia.org/wiki/Create,_read,_update_and_delete#Database_applications, accessed Jun 2013.

WikiJSF (2013). http://en.wikipedia.org/wiki/JavaServer_Faces, accessed Jun 2013

Стаття надійшла до редакції 14.07.2013.