CZASOPISMO TECHNICZNE M
WYDAWNICTWO POLITECHNIKI KRAKOWSKIEJ

z. 3-M/2008
ISSN 0011-4561
ISSN 1897-6328

KAMIL NOWAK[*]

CREATING FLEXIBLE WEB APPLICATIONS ON EXAMPLE OF CONTENT MANAGEMENT SYSTEM

PROJEKTOWANIE ELASTYCZNYCH APLIKACJI INTERNETOWYCH NA PRZYKŁADZIE SYSTEMÓW ZARZĄDZANIA TREŚCIĄ

A b s t r a c t

In the paper a way of creating flexible web applications has been shown. It describes division of application into layers to ensure control separation of the user (controller), generated content (view) and internal decision process (model). It also presents a technique which allows independence of data source. It shows how to design an application that can be easily extended in a safe way.

Keywords: *flexibility*, *MVC pattern*, *Ajax*, *Database Abstraction Layer*, *unit tests*

S t r e s z c z e n i e

W niniejszym artykule zaprezentowano jedną z metod do budowania elastycznych aplikacji internetowych. Omówiono sposób podziału aplikacji na warstwy w celu jak najlepszego odseparowania sterowania użytkownika (kontroler), generowanej treści (widok) oraz wewnętrznego procesu podejmowania decyzji (model). Zaprezentowano sposób uniezależnienia się od źródła danych. Pokazano, jak zaprojektować aplikację, aby była łatwo rozszerzalna oraz aby proces ten przebiegał bezpiecznie dla systemu.

Słowa kluczowe: *elastyczność*, *wzorzec MVC*, *Ajax*, *warstwa abstrakcji bazy danych* (*DAL*), *testy jednostkowe*

[*]Kamil Nowak, V year student, Institute of Applied Informatics, Cracow University of Technology.

1. Introduction

Nowadays, creating web applications is not limited only to implementation of a system that meets functional requirements. Continuous modifications take place on initiated programme. Therefore, adding a new feature should be fast and without interference in other components. Each new feature should be carefully tested, but classical, functional testing can not be used by virtue of risk of system's damage or loss of data.

Due to web systems flexibility it is possible to modify application in an easy way, with no need to change the existing code. It also allows using once written components in other applications. It entails the need of data source independence and ability to easily change the view layer.

The present paper describes an approach to creating flexible applications on an example of Content Management System. This program is designed for updating and extending web applications, and it is addressed to people without much experience in creating WWW pages.

2. Architecture of application

To ensure the flexibility of the created system, a few guidelines have to be followed. The most important features are extension and modification with ease. These features were granted by dividing the application into modules and differentiating it in layers. To make the program more functional it is sufficient to add a new module without interference in the already existing one. Frequent changes (for example changing the subject) or considerable interference in the code (for example changing source of data) can be made easier by dividing the application into characteristic layers. One of them is responsible for data source independence. Because potential application users do not know HTML syntax, the user interface should be similar to desktop application interface.

2.1. Layer structure of application

The created system was divided, according to MVC pattern, into three loosely connected layers: Model, View and Controller (Fig. 1). In this approach one layer can be modified with only minor influence on the others. The main advantage of applying MVC pattern is separation the presentation (view) from the data (model) and the way of interconnecting them. Individual layers can be described as:

1. Model – a layer of business objects and access methods. These objects are related to discipline for which the application is created.
2. View – this layer defines how to present the objects from the model layer and user interface. It is responsible for correct data formation depending on the way of presentation (WWW browser, RSS, PDF).
3. Controller – is a driver. In reply to user's actions it takes the objects from the Model and passes them on to an appropriate View.

Most of WWW applications use databases – transactions on these databases make up models. XHTML templates, responsible for data display are part of Views. The Code responsible for execution of specific operations is created by controllers.

This layer model assists the process of expanding the functionality. The application code is legible to persons who are familiar with MVC pattern. Creating new modules does not require full knowledge about the structure of application.

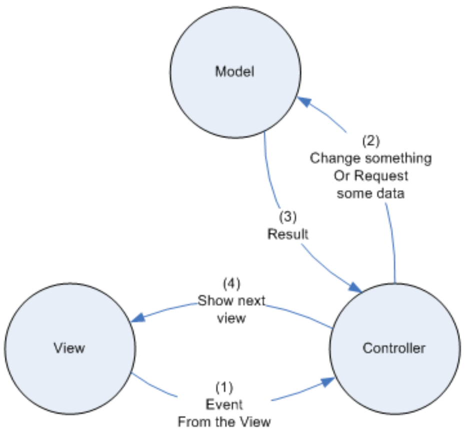

Fig. 1. MVC scheme
Rys. 1. Schemat wzorca MVC

To avoid implementation of MVC pattern another time, Zend Framework was used. The important advantage of this framework is the conventional, modular structure of directories. It allows splitting different MVC applications into separate units and to use them with different front controllers. It fits well the modular structure of the created CMS. Thanks to this feature the system can be easily extended even by users who know the architecture of the system only superficially.

The structure of directories is shown in Fig. 2.

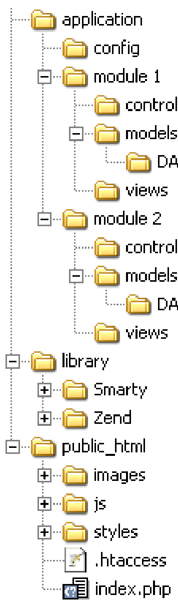
```
application
  config
  module 1
    controllers
    models
      DAO
    views
  module 2
    controllers
    models
      DAO
    views
library
  Smarty
  Zend
public_html
  images
  js
  styles
  .htaccess
  index.php
```

Fig. 2. Application layout
Rys. 2. Struktura katalogów aplikacji

Thanks to the structure planned in this way, adding a new module is reduced to placing another directory in the application structure. Moreover, each layer has a different directory in the module. Apart from directory named application, which includes logic characteristic of the application, another directory – library – is located. This directory includes scripts not related to the application domain. All scripts, except bootstrap and JavaScript, were placed outside DocumentRoot. It increases the safety of application.

2.2. Database abstraction layer

The created CMS requires database connection, such as MySQL, PostgreSQL or Oracle. To avoid replacement of the code responsible for communication with the database (for example, changing function mysql_query into pg_query) it is necessary to place, between the application and database, a database abstraction layer (DAL). There are many such solutions (for example ADOdb, PEAR::DB or Creole) but PDO (PHP Data Object) is the most efficient interface. A comparison of these drivers is shown in Fig. 3.

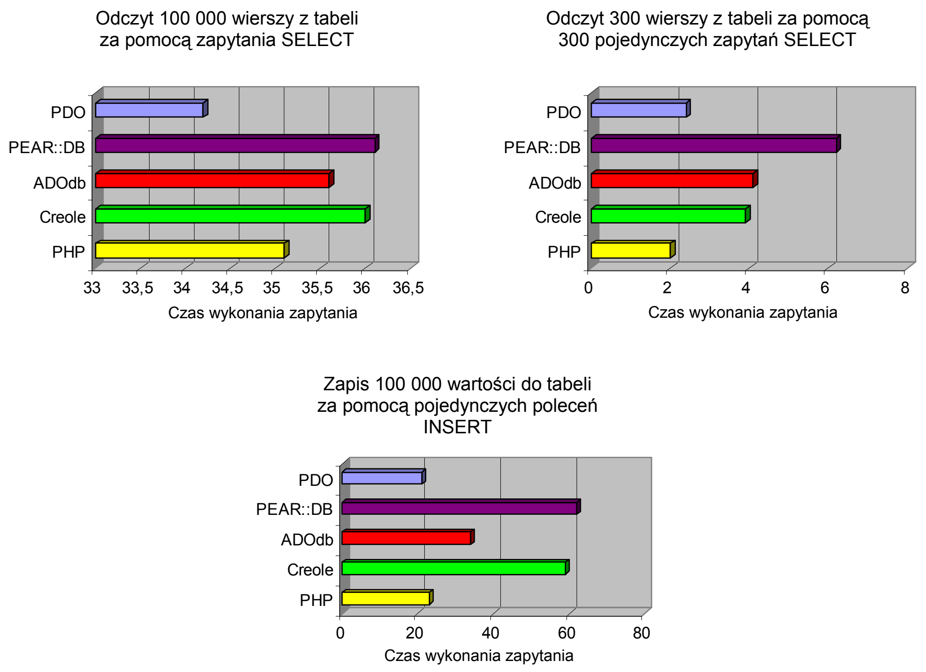

Fig. 3. Comparison of different DAL
Rys. 3. Porównanie wydajności różnych rozwiązań abstrakcji baz danych

Thanks to PDO, the created system is independent of RDBMS and it works quicker.

Apart from the DAL layer, another layer – Data Access Object – was created. It is responsible for getting business objects form data sources. Such solution allows grouping SQL queries in one place. In case when the table structure, its name or column change, thanks to DAO, there is no need to search for links to this table in the whole application.

2.3. Authorisation

The improved flexibility of the application is also related to users management and granting them rights. This aim was achieved by creating access control lists (ACL). To this end a division into the role and reserve has to be done. The reserve is an object to which access is controlled and the role demands access to the reserve. Users groups have the role character, the reserve corresponds to controller's actions. Thanks to such approach we can freely define functions which the given user can execute in the system. The privileges were verified by creating a plug-in for the front controller. Before every loop of dispatching process, this plug-in checks if the given user has a right to execute the given function.

2.4. Ajax

The user interface Ajax (Asynchronous JavaScript and XML) was created. It allows a better interaction with the user. It does not depend on the solution request-answer (as done so far), but on sending requests asynchronously (Fig. 3). The client and server communication is unnoticeable for the user. Ajax ensures the balance between functionality on the client and server side. It spreads loading evenly on both sides.

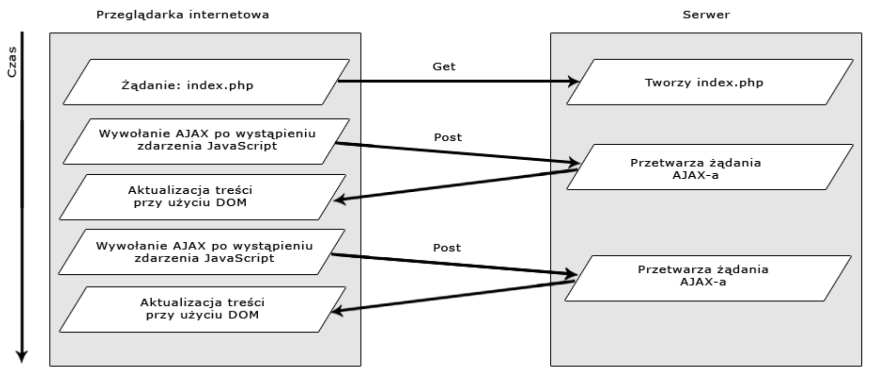
Przeglądarka internetowa
Serwer
Czas
Żądanie: index.php
Get
Tworzy index.php
Wywołanie AJAX po wystąpieniu zdarzenia JavaScript
Post
Przetwarza żądania AJAX-a
Aktualizacja treści przy użyciu DOM
Wywołanie AJAX po wystąpieniu zdarzenia JavaScript
Post
Przetwarza żądania AJAX-a
Aktualizacja treści przy użyciu DOM

Fig. 4. Typical Ajax request
Rys. 4. Typowe wywołanie Ajax

The interaction with the system is similar to the interaction with desktop applications. There is a possibility to apply drag-and-drop technique or double click. It will make the work with the system easier, especially for a person not familiar with computer science.

2.5. Tests

In the paper, creating CMS system is based on the use of interfaces (not implementation). Thanks to it each class can be easily modified, replaced, or extended. This method also allows on simple code refactoring and testing application classes at the early stage of formation (unit tests). These tests permit using Test Driver Development (TDD) technique. It is necessary to prepare tests for interfaces. It is done on the basis of requirements described for classes (using SimpleTest framework or phpUnit). Then a body of class will be created and tested. This technique allows elimination of most errors before the component is placed in application.

3. Summary

Thanks to the method described in the paper we gain the possibility of creating more flexible applications. Such project allows code refactoring and extending the application with minimal costs. Subsequent work with such application is faster and easier. Adding new functionality, changing data source or subject do not cause any trouble. Tests module preparation ensures safety of the application. Use of Ajax increases application speed and makes work with the program more intuitive and ergonomic.

References

[1] Lecky-Thompson E., Eide-Goldman H., Nowicki S.D., Cove A., *Professional PHP*, Wydawnictwo Helion, Gliwice 2005.
[2] Zakas N.C., *Professional JavaScript for Web Developers*, Wydawnictwo Helion, Gliwice 2006.
[3] Zakas N.C., McPeack J., Fawcet J., *Professional Ajax*, Wydawnictwo Helion, Gliwice 2007.
[4] http://framework.zend.com.