MVC Framework Utilization in Development of Web Application

Vojtěch ONDRYHAL

Dept. of Communication and Information Systems, University of Defence, Kounicova 65, 662 10 Brno, Czech Republic

vojtech.ondryhal@unob.cz

Abstract. *The System for Scientific Information Division is a Web Based Client-Server application build mostly on top of non-commercial components with students' participation. The development is based on iterative approach with Unified Modeling Language (UML) support.*

The paper provides a case study implementation of MVC (Model-View-Controller) framework in real project. The paper discusses in detail MVC framework implementation, the approach utilized for controller, model and view implementation on the webserver.

At the end of the paper the advantage of MVC framework utilization is described together with Use Case realization.

Keywords

MVC, Software architecture, Web application, AJAX.

1.Introduction

MVC (Model-View-Controller) framework is quite an old framework. The MVC pattern was originally formulated in the late 1970s by Trygve Reenskaug at Xerox PARC, as part of the Smalltalk system.

In the paper the framework is used for both server and client development.

1.1 About the system under development

The information system for Scientific Information Division is being developed since 2009 at the Communication and Information Department of University of Defence; the first version was publicly available in April 2010.

The goal of the project is to support of the day-to-day activities connected to administration of research. The following processes are (or will be) implemented in the system:

- The building of the register of the scientific results at the university. The register is used for the annual summary creation, for the export into national register of scientific results, for the project administration, and also for the personal purposes of researchers.
- News for researchers. Management provides news and requests for members of a university at the research field.
- The management of projects at the university, including economic aspects likes funding and budget preparation.
- The support for material and services (including business trips) purchasing using requests based system is provided, including configurable workflow process of request review.
- Other registers like activities, citations, person workload, etc. are also available. Many reports in pdf, doc or xls formats are provided.

The main processes are supported by several core and technology processes like user management, security or logging system.

1.2 Architecture goals

Development of any system requires definition of the main architecture goals at the start of the development. Such goals form constraints for the consequential design and implementation. The architecture is usually a compromise between user expectations and developer possibilities. The following are the principles defined for the project:

- MVC (Model-View-Controller) design pattern is strictly applied for all the components. The pattern splits the data processing, program control and user interface development into separate streams.
- "Make it simple" principle. Avoid creating complex user screens and difficult processing tasks. It is more feasible to create several simple screens comparing to complex, tricky and confusing one.

- To enhance user experience, the AJAX (Asynchronous Java and XML) technology is utilized in edit forms; for example, automatic saving during editing, constant update of user interface after the change of data – e.g. for bibliographic record or computations, autocompleting etc.

To support previously defined principles the sufficient technology and tools must be selected. It is vise to build the system on top of advanced framework which provides basic system functionality and supportability as a part of such framework. For the project the Codeigniter PHP framework [2] (based on MVC) and jQuery Javascript [8] library have been selected.

2. MVC on a Server

The architecture of the server is based on MVC paradigm. The user interaction (controllers), data manipulation (models) and presentation (views) are separated into independent classes (Figure 1).

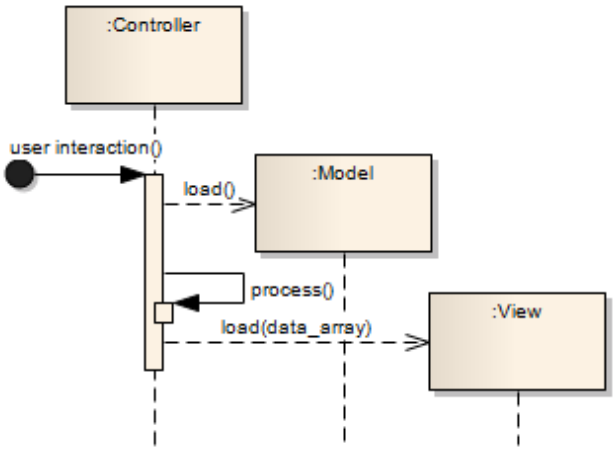

Fig. 1. Basic MVC communication

2.1 Controllers

Controllers are the heart of the system, as they determine how the HTTP requests should be handled. In the system, there are two kinds of controllers. The basic controllers handle single HTTP requests and the AJAX controllers that handle HTTP AJAX requests. All the basic controllers inherit from *MY_Controller* class. The class provides generic methods and properties for security, logging and error handling. All the AJAX controllers inherit from *AJAX_Controller* class. The class provides similar functionality for AJAX controllers as the *MY_Controller* class.

From the application point of view, there are usually three different controllers in each module.

- The first controller handles HTTP requests for a collection of objects, like collection of projects, activities, etc. Controller handles functionality consisting of complex calculation on objects in the collection, providing manipulation of lists of single objects in different situations.
- The second controller handles single instance of class, like a single person, project or request. The first two controllers are descendants of the *MY_Controller* class.
- The third controller handles AJAX requests for the module. On the Figure 2 is an example of controllers in the Projects module.

In case any AJAX controller is involved, only the page fragments are refreshed, otherwise the whole page is refreshed.

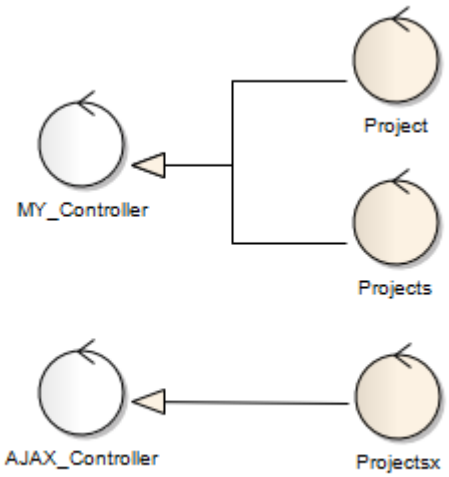

Fig. 2. Controllers in module *Projects*

2.2 Models

Models are classes that are designed to work with information in the database or any other form of persistent data storages (text files, XML data, JSON data etc.). Model class usually contains functions to insert, update, and retrieve data from data storages.

All model classes communicating with the database inherit from *MY_Model* class. Each table in database has associated matching model class (Figure 3). Simple model classes contain only properties with name of the matching table, name of the primary key in the table and implicit order of records. All other functionality is handled by parent class. It is obvious that adding a new table manipulation (model) class is quite simple.

More complex model classes provide calculations, collection to multidimensional arrays transformations,

associated database views manipulations and other functionality depending on user requirements.

The number of model classes differs in each module. The total depends on module complexity, responsibility and detail design.

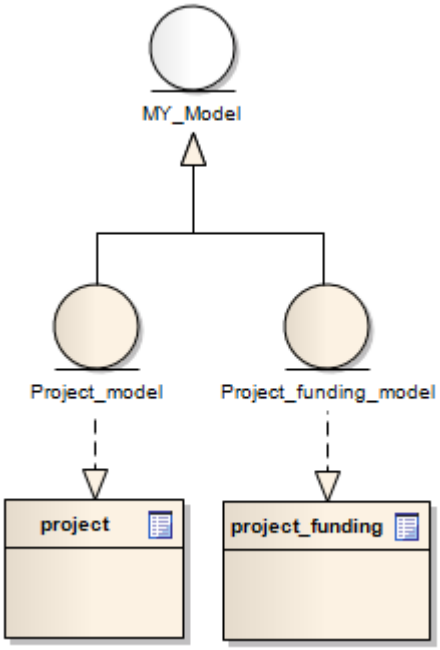

Fig. 3. Simplified diagram with Models in module *Projects*

2.3 Views

The last component of MVC framework is a *view*. A view in our system is a web page, or a page fragment, like a header, footer, sidebar, etc. Views can flexibly be embedded within other views. Views are never called directly; they must be loaded by a controller.

The number of views in large-scale system can run up to hundreds. In the system under discussion each method in controller leads into a single view. Views are structured into subfolders according to controller name.

Every page is constructed from a *Layout* view. This view is always composed of *Menu* (navigation), *Main* (content) and *Info* (help) views. In addition, based on current case, the set of dialog views is associated with the *Main* view (Figure 4).

Requests originated in Menu view are handled by Menu controller. Menu controller redirects handling to appropriate basic module controller, usually collection controller. Requests originated in Main view are handled directly by module controllers, both basic and AJAX and requests originated in dialogs are always handled by AJAX controllers.

2.4 Other server components

In addition to mentioned components, other components are loaded during the application processing. E.g. export libraries into pdf, doc, xls, and standalone toolbar classes and config files for the modules.

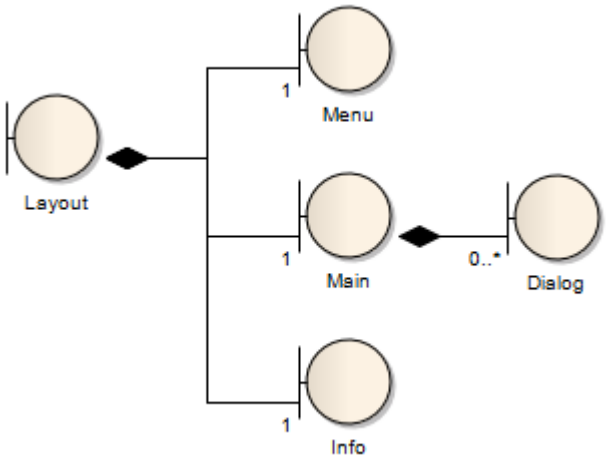

Fig. 4. Organization of views

3. MVC on a Client

The client, in this case, is a web browser and web page inside a browser. Web page itself consists of html code, CSS styles and JavaScript segments.

At the high level view, separation of look, processing and data is a modern approach for web development on the client side. Alongside with separation of look from HTML with Cascade Stylesheets, the separation of processing is a next step; especially by applying unobtrusive approach for JavaScript development [7]. jQuery is the example of the mentioned approach [8].

In connection with server part, in the system the JavaScript files are created in the same way the views are created. For each method in controller an independent JavaScript file is created if required.

4. Use Case Realization and MVC

According to [1] the use case is a collection of related success and failure scenarios that describe an actor using a system to support a goal. The scenario is a specific sequence of actions and interactions between actors and systems.

In our approach realization of a Use Case scenario is a collection of the following parts:

- A method in non-AJAX controller. The call of the method is a start point into scenario.

- The view associated with a controller method.
- The JavaScript file associated with controller method that enables dynamic user interaction basics.
- The required shared models, libraries and JavaScript files which are loaded.
- Set of AJAX base controllers, usually one associated with module for AJAX based user interaction.

The names of view and JavaScript file correspondent with name of method in the controller.

The described final components of Use Case scenario realization suit to generic principles of software development – Low Coupling and High Cohesion [1]. Developed components are highly focused on single user need, the components can be easily added and removed from system. The components can be easily tested.

Conclusion

By selecting proper software technologies the development can be changed dramatically. MVC framework is such a technology which helps creating more modular, easily maintainable code. Developers can be more specialized on development of specific tasks as of user interfaces, application logic or database communication development. Code is more easily reusable, e.g. utilization of a single model in different controllers.

Large system is usually described with the set of use cases and its scenarios. The MVC is a technology background on which the realization of scenario can be developed as independent and easily recognized component.

The described approach also easily fulfills the Iterative Development. Single user scenarios are included into iteration and simple output components are produced (a method in controller, a view and a JavaScript file), usually independent on other existing components.

The process of the development is utilized also in lectures in two subjects at the university; Information system development and Web application development. The development of the system, used tools and methodology are discussed in detail during lectures.

Acknowledgements

The article is prepared as a component of the research project for the Development of the CIS Department, Faculty of Military Technologies, and University of Defence (PRO K209, FVT). Name of the project: *Prospective Technologies in Communication and Information Systems.* It introduces some outcomes of the solutions in Command and Control Information Systems (C2IS) and Knowledge Management Systems (KMS). Our results are part of the education process within the University of Defence, Brno.

References

[1] Larman, C. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development.* Prentice Hall, 2004. ISBN 978-0131489066.

[2] CodeIgniter – Open source PHP web application framework [Software]. Available from http://codeigniter.com.

[3] Enterprise Architect - Model Driven UML Tools [Software]. Available from http://www.sparxsystems.com.au/.

[4] Zervaas, Q. *Practical Web 2.0 Applications with PHP*. Apress 2008. ISBN 978-1-59059-906-8.

[5] McArthur, K. Pro PHP – Patterns, Frameworks, Testing and More. Apress 2008. ISBN 978-1-59059-819-1.

[6] Harmes, R., Diaz, D. Pro JavaScript Design Patterns. Apress 2008. ISBN 978-1-59059-908-2.

[7] Resig, J. *Pro JavaScript techniques*. Apress 2006. ISBN 978-1-59059-727-9.

[8] Bibeault, B., Katz, Y. *jQuery in Action*. Manning Publications 2010. ISBN 978-1-935182-32-0.

About Authors ...

Vojtech ONDRYHAL, Ph.D (1970) is a lecturer at the University of Defence, Military Technology Faculty, Communication and Information Systems Department, Brno, Czech Republic. His skills include database systems, data mining, software engineering (Java, PHP, Python and JavaScript languages) ontology engineering and information systems modeling.

Author is also a software architect and lead developer of the System for Scientific Information Division described in the paper.