Nick Yaitsky
Russell Myers
Matthew McElhaney
10/02/07

Project Proposal

Motivation and Objectives

Currently Multicast Media Networks (“Multicast”) employs both Nick Yaitsky and Russell Myers as developers on a variety of web-related products that they deliver. These products require an extensive amount of interaction with databases to provide their content. The majority of these products serve as client management systems in which one or many users may log into the application, change various aspects, and have them presented to consumers as well.

The client management systems generally supported by Multicast Media Networks are codenamed “DMMS” (standing for Digital Media Management System) and “Vidego”, both supporting their own subset of clients. Both of these products essentially do as the first product acronym suggests, they provide a way to manage video content and present it in both stream and on demand forms. This media may be branded, advertised, and developed upon at any point by any end user.

The Vidego Pro web application is built using the Data Access Object / Value Object (DAO/VO) System, originally developed by Nick Yaitsky. Please note that this system is not Vidego’s framework, but it is the framework that Vidego is built on. This project is aimed at augmenting the DAO/VO System to add features, functionality, and heavy documentation. Documentation for the framework will help future developers of Multicast Media Networks who may need to work with the existing system, and who have never seen this framework before. All documentation will have no direct linkage to Vidego as the documentation is for the framework and not for Vidego.

The DAO/VO System is an interface between a MySQL database backend and a PHP functionality wrapper. By the end of this project, the system will be able to map arbitrary PHP variable names within entities to tables in the database layer. The programmer will then not need to have any knowledge of the field names or table names within the database, and only focus on coding the PHP functionality. The system will then also be able to translate data into XML strings for Webservice data transfers. With only a few configuration options, this framework will be able to parse an existing database, and create the PHP frame around it to abstract the database. This follows the common practices in web application development of tier structure in which various layers of abstraction are built into an application for a variety of reasons. In this case, the project is a modification of the typical 3-tier design: Database Layer, Business Logic, and Front-End Web Pages.

This project is endorsed by Multicast Media Networks under the premise that the company retains rights to the product and should be able to claim any enhancements, source code, documentation, and presentation materials presented during this project's lifespan. In *no way* may the Georgia Institute of Technology nor the persons involved with this project be able to claim ownership. This is the agreement that the board of Multicast Media Networks signed off on.

Related work

The proposed work is not entirely novel, but it is an efficient method of creating a lightweight framework in the PHP language. There are many frameworks out there that do similar things, but none are as lightweight and easy to use. There are systems in other languages that create similar results in relation to a database interface. One of those is the Microsoft .NET framework, in which MS SQL databases can be generated into fully fledged API's.

Here is a list of the most popular PHP Frameworks in use today:

The Zend Framework (http://framework.zend.com) is considered to be the number one framework on the market. However, it is overpowering, and cannot be applied to an already existing project. It relies heavily on naming conventions to auto-magically instantiate classes, and it uses up most of the available memory.

CakePHP (http://www.cakephp.org) is the next most common framework. It completely splits off the model view and controller to create a very robust web application. The drawbacks to this framework are similar to Zend: You cannot apply it to an already existing project, and are forced to rewrite the entire project from scratch.

The Symphony Project (http://www.symfony-project.com) combined the business logic with a PHP abstracted HTML generator. The developer does not use HTML and JavaScript in most cases because much of the functionality is abstracted out to the PHP side. The drawback is that you can't do any complex tasks that aren't natively supported by the framework. When developing a small web application with little pizzazz, this is an excellent way to go, but for larger projects, it does not suffice.

Seagull PHP Framework (http://seagull.phpkitchen.com) is very similar to the Symphony Project in many ways with only modularization differences. Seagull uses modules which make the system very robust, and provides the developer the ability to remove unwanted elements very easily. However, the configuration settings for it are very complex, and require much experience to use effectively.

Proposed work

The proposed work is to be able to easily generate and regenerate various elements in the framework, such as Entities, Logic files, and Page Builders. One of the key new features is a

method to take an existing database and web application, and convert it to the DAO/VO standard. Some of the few requirements are the following. First, the database must be a MySQL system. The server must support PHP 5.1+. All database tables must have singular integer based primary keys (naming is irrelevant). If a field named “id” exists, it must be the primary key. Beyond this, the idea is to present any other novel features that may be of interest to the team as well as fully document the design schema and the generated file structure.

By the end of the project, the framework should be able to satisfy these needs as well as have documentation that would allow a new developer to easily use the system’s key features, as well as at least a few of the more complex / hidden elements.
Note: Diagrams, file structure schemas, and algorithm descriptions will be provided as part of the proposed work, and are therefore not included here, as that is what we are developing.

Plan of action (what resources, schedule, plan for evaluation)

Scheduling has not yet been cemented as there are three developers working on it, and all three have very busy schedules themselves. Since this is a company product, there is a great deal of desire to do the majority of the work during allotted work time. The preliminary schedule states that 60-70% of the work will be done on an individual basis among the team members, and the rest of the work will be done in a group over weekends at the College of Computing.

Evaluation will consist of the comprehensiveness and usability of new features, as well as effectual modifications to the existing infrastructure. It will also consist of the readability and extensiveness of the provided documentation.

All testing and development will occur on an Ubuntu system using PHP 5.1.6 and MySQL 5.0. A public (password protected) web application will be created on a sub-domain of http://nickyaitsky.com (name of the sub-domain is still to be determined). The test web application will demonstrate all of the new features as well as showcase most of the existing features in a reasonably secured environment.

Evaluation and Testing Method

The proposed testing scenario is a black box case in which we create several database schemas and run the framework generation against those schemas. The generated code must then be able to perform a series of CRUD commands, those being insert, update, delete and select. The generated web services must be able to perform similar functionality, but also translate to and from XML. Included in this will be the ability to recognize table relations so that the entities may have simple foreign key associations and be able to retrieve data based on these relations.