CERN Summer Student Program 2013 Report

Stanislav Pelák

E-mail: stanislav.pelak@cern.ch / pelaksta@gmail.com

Abstract. This report describes the work and achievements of Stanislav Pelák, during his stay at CERN as a Summer Student and as an assistant to technical manager for CERN School of Computing.

Introduction

My three-month internship could be divided into three phases: maintaining CSC's (CERN School of Computing) web applications, database and preparation for the School, CERN School of Computing - technical support and finally, after-school development focused on Drupal content management framework.

I will describe each of this phases in following sections.

Internship details

Time period:	01.07. – 27.09.2013
Department:	IT
Project:	CERN School of Computing 2013
Supervisor:	Giuseppe Lo Presti
CSC 2013 director:	François Flückiger

1. Pre-School phase

For the first six weeks of my internship at CERN, my work was focused on:

- Designing the structure of the CSC's[1] **Oracle DB** and performing the changes (together with another student).
- Editing the **Portal** application (new functionality, reflecting the changes in database, refactoring).
- Editing the **AdminApp** application (reflecting the changes in database, minor refactoring).

1.1. ***Oracle database***

The main reason to start redesigning the structure of a CSC Oracle database was, that because of having a year as a primary key for school entity, the structure didn't support several schools in one year. This started to be a problem with the Thematic School, which took place for the first time in 2013.

As a result, the scheme was changed to be able to support this use case. In addition, several other changes were made:

- Better distinction between *applicants* and *former students*.
- Automatization of school closing process by implementing a **procedure**:
  1. move all students from *Applicants* to *FormerStudents* table,
  2. clear tables *Selections, Selected, Login, UserDevice* and *Applicants*,
  3. change school status to "Closed".
- Prevention of integrity constraints violation by **triggers** (only one opened school at a time).

1.2. ***Portal***

Portal application is used by students to submit their applications and application-related documents for CSC. There were two major changes implemented in this application:

- refactoring,
- submission of a reference letter by student's supervisor.

1.2.1. ***Refactoring*** The goal of refactoring of the application was to improve application's security, maintainability and make its components loosely coupled.

Even though in the **initial solution** a "central point" for database access existed, PHP source files opened and closed a database connection, sent a pre-made SQL queries to it and processed results on their own.

Implemented solution separates the application into three layers according to the **MVC**[2] design pattern.

- Source files exposed to the user contain mostly an HTML code to be displayed. Logic behind the *View* was moved to its *Controller* layer.
- *Controllers* access the database via the **Database Adapter** object. *Database adapter* implements a general *IDBAdapter* interface, so it can be easily replaced.
- *Controller* retrieves the right singleton instance of a *IDBAdapter* interface from *Model*.
- **Model** only reads from the configuration file, which *DBAdapter* it should load and provides its instance to *Controllers*.
- Configuration of the application was separated from its source files.

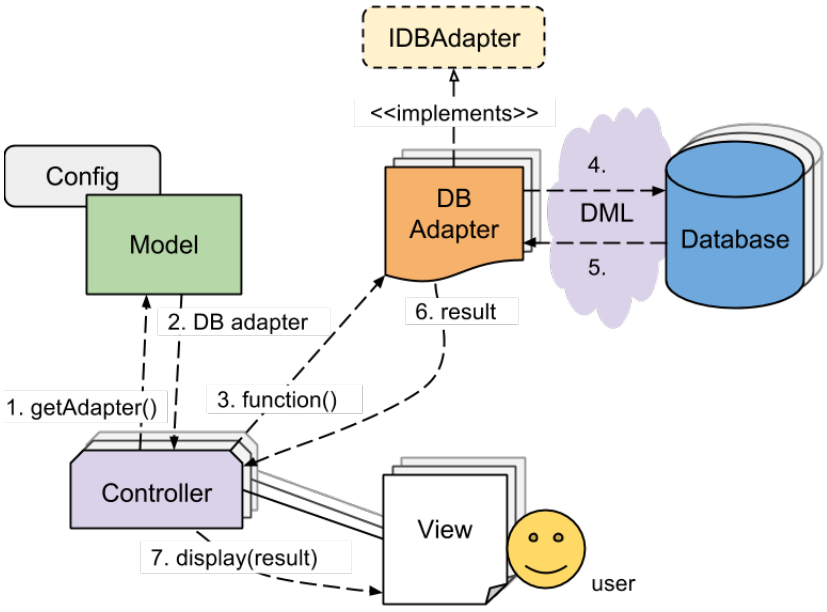

Figure 1. Structure of a Portal application and mechanism of accessing the database.
1. and **2.** – Controller retrieves an instance of *IDBAdapter* interface.
3. – Controller calls a function on retrieved instance of an IDBAdapter interface.
4. and **5.** – DB adapter alters its database using a particular data manipulation language (DML).
6. – DB adapter returns a result of a function.
7. – Controller updates the View.

The whole mechanism is displayed on figure 1.

After the refactoring, the application is **more secure** and **easier to configure** (before, even files that were not supposed to be visible were in the same folder with the other files and exposed to the user), **better maintainable** (layers – separation of concerns) and **loosely coupled** (i.e. in case of changing the database, only new *adapter* is needed).

1.2.2. ***Submission of a reference letter by student's supervisor*** Originally, the reference letter from a supervisor was submitted by the student, or sent to CSC administrator via email. The new requirement was to allow the supervisor submit the reference letter on his/her own.

Current solution generates a token, which is a part of an URL link sent to the supervisor at the moment of student's registration. Since the token is created by hashing the student's credentials (ID) and a timestamp (also a part of the link), it's not possible to guess it (student doesn't know his/her ID) and it can be used to authorize the supervisor and to retrieve the right student.

Using this link, the supervisor is able to submit a reference letter for the student via the dedicated form.

1.3. ***AdminApp***

AdminApp web application offers an administrative interface for selecting and managing the applicants for CSC in general.

[1] CSC – CERN School of Computing
[2] Model–View–Controller

Several changes were made also to this application:

- Reflection of changes in the structure of the Oracle database.
- Process of creating and closing school updated.
- Several user interface changes (view information from past schools, optimization of available menu links etc.).

These changes, as well as changes of the database structure, has their origin in updated use case, which requires to have several schools in one year.

2. CSC 2013

During the CERN School of Computing 2013 in Nicosia, Cyprus, I was responsible for administering the School's web applications and MySQL database, which were used by students.

I was maintaining already existing PHP code, in which I implemented several new functionalities and also some bugs were found and fixed.

- **Functionalities**
  - Support for complex events consisting of several atomic activities.
  - Modification of the participants overviews – view by evnt or atomic activity.
- **Bug fixes**
  - *Security by obscurity* – exam questions, list of students, their answers and scores back to 2009 were basically exposed and could have been found by guessing the URL.
  - Recovery of a session cookie in case of its expiration during the exam.

Apart from my duties, I also managed to follow the series of CSC 2013 lectures.

3. Post-School phase

After the CSC 2013, I was working with Drupal – open-source content management PHP framework, which should replace current CSC site in the future.

My task was to design the most suitable architecture of a Drupal site, investigate whether Drupal is able to satisfy our requirements for the CSC website, find techniques (i.e. modules) and propose recommendations to achieve desired functionalities and finally, demonstrate the solutions.

Three main questions I focused on were:

- How to manage embedding of a content into another content?
- How to clone the site for a new school?
- How to effectively find a desired content?

As a result of my work, I managed to propose a solution for all of these questions. In addition, I designed a new **core structure** of the application in order to maximize its usability.

With the new approach, application is divided into several "logical" sections, that appear to the user as independent websites. One of these sections contains general information about the whole CSC (i.e. what, who, how to apply etc.). Each of the other sections will contain only the content of a single (i/t)CSC (mostly for students of a certain School).

By separating the content logically, the users see only what is important or interesting for them and nothing more (thanks to separation). On the other hand, all the content of all sections is managed "in one place", which wouldn't be possible if the sections would be divided physically (as independent Drupal sites). This approach therefore combines good attributes of both having a single and several websites.

For **cloning the school**, I have implemented a module, which clones all desired content including the structure of the menu, creates and assigns a new taxonomy term to the new content to simplify the filtering and sets the URL alias, so the base of the site is ready "on-a-click".

To improve **filtering** effectiveness, I proposed using a combination of custom Content Types with taxonomy. By using these techniques, it is possible to set some categories, and even hierarchy to otherwise flat-structured general nodes. In addition, I also recommend to use third-party module (*Enterprise Base*), which provides the content filtering with several very useful filters.

Detailed description about this part of my work can be found in a separate document *Design principles of the Drupal CSC website* available in CERN's CDS under the reference:
CERN-STUDENTS-Note-2013-219.

4. Conclusion and further work

During my internship and work with the applications mentioned above, I found several parts that I would strongly recommend to improve.

The first of them is the examination form, which is rather fragile and is either unable to recover from a mistake (no partial saves), or it relies on an user not to perform a forbidden action (i.e. refresh the page).

I would also recommend to redesign the structure of a main CSC site in order to have less and strictly specific menu items, simplify the navigation on a website and to display only information that is relevant for a user browsing the page. I tried to make a first step in this direction by proposing separation of a "general" site and "school-specific" sites and I think, that migration to Drupal, which is currently being developed, is an excellent opportunity to start this discussion.

Since the Drupal seems to be the future for CSC website, it would be practical to merge the functionality of as many currently used web applications as possible to it.