Timetabling RIA in action

Florent Devin and **Yannick Le Nir**
L@RIS - EISTI
26 avenue des lilas
64062 Pau Cedex 9
FRANCE

Abstract

We present an Rich Internet Application to create timetabling for real engineering school. In this application, we have resources and constraints. Resources describe the timetable's context. Constraints are restrictions of resources. All lectures are placed on a timetable composed of time-slots. For timetabling, we use a computational web service written in Prolog. Our application presents two different views, one for the user, and one for the *admin*.

Users can view all generated timetables and put in their own constraints. The input of constraints can be done by two ways, directly from our application, or via Google calendar. To use Google calendar, users have to update the link of their Google calendar.

Admin has many tasks to do. He has to plan all different lectures that occur in a year. He also has to specify which contributors teach which lecture, and to whom the lecture is being given. Moreover, he has to input constraints for classes, and possibly rooms' unavailability. Then he has to validate or invalidate users constraints. Finally he can generate one or more timetables. He can also export all generated timetables to Google calendar.

Introduction

In this paper we present an original approach to timetabling. This approach is based on the concept of web services. To be able to create timetables, there are two different parts in our application. One is the Rich Internet Application (RIA), and the other is the computational part. Before viewing how our application runs, we have to define several terms. Then we explain why we have chosen to create an RIA. Finally, we describe the demonstration itself.

Definitions

First we have to define *time-slot* and *timetable*. For us a time-slot is a period with a start time and a fixed duration. So a time slot is the minimal time interval we can find on a timetable. A timetable is a consecutive list of time-slots on which resources are planned.

Then we have to define *resources* describing the context of our application. In fact, we consider three resources:

- main resources: elements to plan (lectures, meetings, ...);
- static resources: elements which are linked to main resources (contributors, classes, ...);
- dynamic resources: elements to include during the computation (rooms, materials, ...).

Finally we have to define *availability* of resources. This is all the possible associations between resources and the amount of consecutive time-slots. *Unavailability* is the complementary of *availability* in the timetable. Unavailability can also called a *constraint*.

Rich Internet Application

Rich Internet Application provides a very viable technology (Duhl 2003), offering most desktop features. This technology can address many users, without any requirement, because they are run on a web browser. By using a web browser, users do not have to learned either a particular operating system, or a particular software to use our application[1] (Rogowski 2007), (Driver and Rogowski 2007).

Our application is written with the ZK framework. ZK is an open source Ajax web application framework (Seiler 2009). Yeh(2006) shows numerous advantages of using this framework. Also ZK uses a centric server approach, which simplifies the security of the application, and saves time during the coding phase. Thus our application is written with Java, and we can use Hibernates' tools.

CSP and Prolog

Constraint Satisfaction Problem can efficiently model the timetabling problem (Wallace 1996), (Frühwirth and Abdennadher 2003), (Jaffar and Maher 1994), (Carter and Laporte 1998). We can translate the timetabling problem with a CSP on finite domain. The implementation of the computation is written with Prolog. It communicates with the RIA using web services, and also directly with our database using a classic ODBC.

Application's architecture

Before describing our demonstration, we now introduce the general architecture of our application. The central point, for users, is the RIA. This RIA communicates with other

[1] Except our application

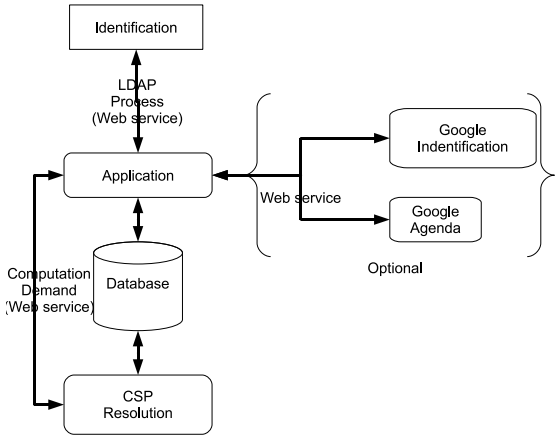

Figure 1: General architecture system

applications using web services. Currently, we have three web services:

- Identification process: as our application is used by our contributors, we have to include it in the existing IT system. When someone tries to log into our application, we ask the LDAP[2] system if the user is authorizes to access our IT system, via a web service.
- Computation process: the computation process is written in Prolog. The RIA is written in Java. In order to communicate with each other, RIA and Prolog have to interact. For the communication from the RIA to the Prolog part, we use a web service.
- Constraints process: we can use Google calendar to input constraints as the demonstration will show. Using Google services is like using web services.

To store data, we use a database. This database is a MySQL database. The RIA create/retrieve/update or delete data with the help of Hibernate. The Prolog part, as mentioned above, uses a classical ODBC.

Timetabling RIA in action

Our demonstration will show the two different views of the application, the one for users, and the one for the admin. In this demonstration, you will see a lot of timetables. You will note that the timetable has many colored time-slots. In fact, there is one color for one feature's classroom.

User demonstration

User can view all previously generated timetables, as timetables are public[3] information. There are three different views for timetables, one for contributor, one for class, and one for classroom. A user can also modify his personal data, like a link to his own Google calendar. He can also put unavailability. A screen is dedicated to this function. In this screen, he can see all previously input constraints. He can also view the state of the constraint, that is to say if the constraint has been validated, or invalidated by the admin. User can do nothing more.

Admin demonstration

The most important part of our demonstration is for the admin's functions. The admin have many tasks to do:

- generation of timetables;
- constraint keyboarding/checking;
- lectures planning function;
- lectures adjustment;
- contributors lectures association.

At the very beginning we have to create the classes, and moreover the subgroups in each class. For example, as you can see in the video, the class named *"ING 1"* is divided into two subgroups, named *A* and *B*. Doing this, we also specify the size of each group, this will be useful later when we want to create the timetable to consider a room's size and group's size.

We may, also at the very beginning, define all existing classroom. A classroom is defined by its location, features and size. This step is essential as the computation considers room's size and feature to plan a course.

Then we have to specify which lectures can occur for which class. Which means that we have to name the lecture, and associate the class with it. Over the years this step does not have to be repeated, as lectures for a year do not often change.

Then we have to plan in advance the lectures that occur during a year, a term, or a period. This phase is shown on the third admin's video. By planning the lecture in advance, we also specify the number of theoretical courses per week, and also the number of practical courses per week. At the same time, we also indicate the duration of courses (per week). The difference between theoretical courses and practical courses is that theoretical courses are done for the entire class once, and practical courses are done once for each subclass.

Once this is done, we can associate the contributors, as shown also on the third video. We do not create contributors, as we can synchronize the contributors via our LDAP services. The only thing to do is to synchronize our database. Once our database is updated, we can indicate which contributor teach which courses, and also what is the feature they need.

With this data, we can generate a valid timetable, if contributor does not have any constraints. To generate a timetable, we just have to choose the week, or a starting week and an ending week, and ask for computation. This can take time, about 20 seconds for a week. And you will note that using a web service is totally appropriate. During this step we can also ask for the export to Google calendar. If we ask for it, all the generated timetables will be exported. The export is done for all contributors, classes, classrooms calendar. If contributors have their own calendar, we have control the other calendars. These can be shared to offer

[2]LDAP stands for Lightweight Directory Access Protocol

[3]By public, we mean that all contributors can be informed of other timetables.

a complete view of the timetable for students, contributors, and administration.

There is also a screen for changing a lecture in duration for a particular week. We can also change the number of courses for a week. For instance, if we have planned a lecture occurring once a week both for theoretical and practice, we can assume that for a particular week, there is no practical course, but the theoretical course is longer. If we are in this case, or whatever similar case, we will use this screen.

We can also deal with constraints. To do this, there are two screens, one more textual (as is shown on the user's video), and one more graphical (as is shown on the second admin's video).

As shown on the first admin's video, we can put in an exceptional event. This event will be considered by the computational part as a hard constraint.

All this functionality, user and admin, will be shown during the demonstration, except the configuration phase. The configuration phase is the keyboarding of contributors, the naming of lectures, and the specification of rooms and classes. If the audience wants, we can show this phase too.

Conclusion

We present an RIA for timetabling, which is fully functional. This RIA uses web services to timetabling. It is important to note that the workload of the admin is considerably reduced by using our application rather than handwriting timetables. We have paid a particular attention to simplifying the admin's works. Also this application can evolve by the use of other web services. We also consider users' habits, by providing a way to use their own calendar.

References

Abbas, A., and Tsang, E. 2001. Constraint-based timetabling-a case study. *Computer systems and applications, ACS/IEEE international conference on* 0:0067.

Abdennadher, S.; Aly, M.; and Edward, M. 2007. Constraint-based timetabling system for the german university in cairo. In *INAP/WLP*, 69–81.

Carter, M. W., and Laporte, G. 1998. Recent developments in practical course timetabling. In *PATAT '97: Selected papers from the second international conference on practice and theory of automated timetabling II*, 3–19. London, UK: Springer-Verlag.

Driver, E., and Rogowski, R. 2007. Rias bring people-centered design to information workplaces. Forester Research.

Duhl, J. 2003. White paper : rich internet application. IDC.

Frühwirth, T., and Abdennadher, S. 2003. *Essentials of constraint programming*. Springer Verlag.

Jaffar, J., and Maher, M. J. 1994. Constraint logic programming: a survey. *J. Log. Program.* 19/20:503–581.

Qu, R.; Burke, E. K.; Mccollum, B.; Merlot, L.; and Lee, S. Y. 2009. A survey of search methodologies and automated system development for examination timetabling. *J. of scheduling* 12:55–89.

Rogowski, R. 2007. The business case for rich internet application. Forrester Research.

Seiler, D. 2009. Ria with zk. In *JAZOON09*.

Wallace, M. 1996. Practical applications of constraint programming. *Constraints* 1:139–168.

Yeh, T. M. 2006. Zk ajax but non javascript.