

Insert here your thesis' task.

CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF SOFTWARE ENGINEERING



Bachelor's thesis

Specialized Information System Maintaining Patients Participating in Epileptosurgical Programme – Reporting Module

Martin Dvořáček

Supervisor: Ing. Petr Ježdík Ph.D.

1st March 2014

Acknowledgements

THANKS

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60(1) of the Act.

In Prague on 1st March 2014

.....

Czech Technical University in Prague

Faculty of Information Technology

© 2014 Martin Dvořáček. All rights reserved.

This thesis is a school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).

Citation of this thesis

Dvořáček, Martin. *Specialized Information System Maintaining Patients Participating in Epileptosurgical Programme – Reporting Module*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2014.

Abstract

Summarize the contents and contribution of your work in a few sentences in English language.

Keywords Replace with comma-separated list of keywords in English.

Abstrakt

V několika větách shrňte obsah a přínos této práce v českém jazyce.

Klíčová slova Replace with comma-separated list of keywords in Czech.

Contents

List of Figures

Introduction

The reporting module, part of the GENEPI - the information system, adds an important function to this software. Thanks to this module the user will be able to export data saved in the system to sundry formats. This is useful for the doctors that take care of patients with epilepsy, as well as for the researchers that make analysis above the data from this system. In this thesis I'll describe the design and implementation of the reporting module, as well the process of final testing of this product.

Analysis and design

The reporting module, as well as the whole information system, was designed given the requested robustness, accessibility, reliability and the cleanness of the code. GENEPI has a three-tier architecture, uses access according to the roles of the users via Spring Security and thanks to the optimized data layer it saves the computing resources.

Due to the fact, that users, who should work with the exported data, have different access levels and different requirements on the format of the exported data, there is a need to make the modul to be able to anonymize sensitive data as well as to export data to different formats. The contracting authority also requested that the user could choose which data does he want to export. For research purposes is also important to determine, if data that should be exported, contain only patients, whose data were verified by the povereny user.

2.1 GENEPI - the information system

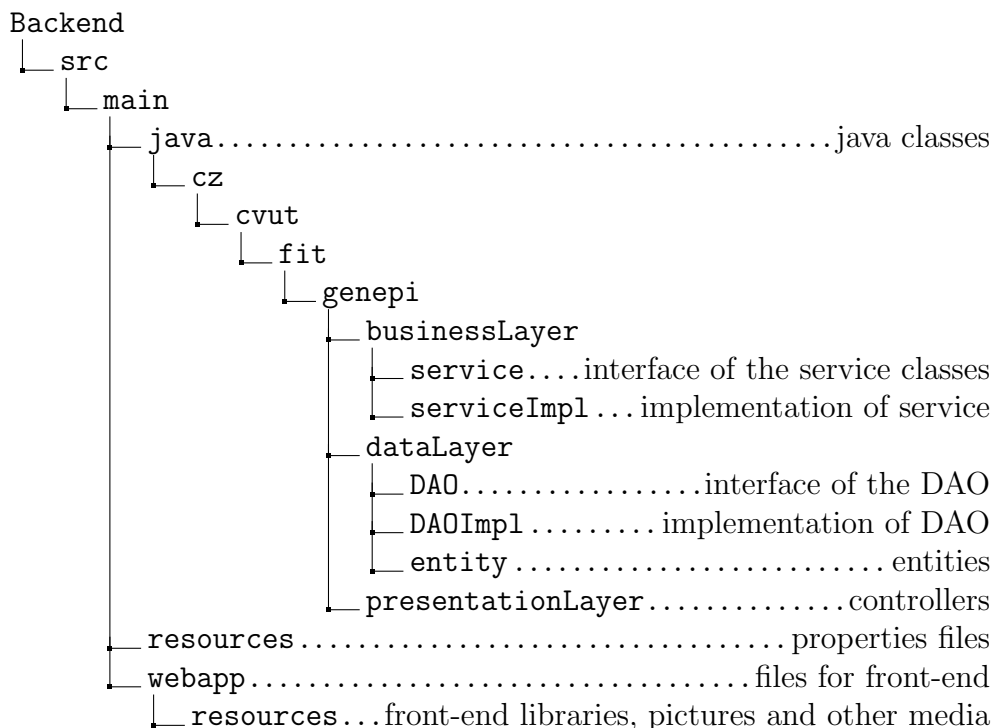
That the reader could fully understand the functioning of the reporting module, first of all it is needed to introduce the information system that it extends. GENEPI - the information system was being created within the subjects BI-SP1 and BI-SP2 on the Czech Technical University, Faculty of Information Technologies in the school year 2012/2013 as a student's project. It should replace the original information system that was used in the Faculty Hospital Motol in Prague for maintaining patients in epileptosurgical programme. The main reasons for replacing the original system was a fact, that it didn't comply with the current requirements of the medics, contained bugs and its design wasn't optimal. GENEPI IS is being developed in the java programming language, using frameworks Spring 3 and

Hibernate 4.2. Front-end part is realized via JSP, using HTML5, Javascript and CSS. Libraries that were used to ease the programming of the front-end were Twitter Bootstrap and jQuery. As a database has been chosen MySQL 5.5 and as an application server Apache Tomcat 7. All of the used libraries are distributed under some kind of free license. The GENEPI is a bilingual software due to the expected deployment in the Miami Children's hospital in Florida, the United States of America. Languages that are currently supported, are czech and english, nevertheless thanks to the sui * TODO tably implemented localization it is easy to extend the application and add any other language. During the whole period of the programming of the system, the team consulted the approach often and regularly with contracting authority, to fully meet the needs of the medics.

2.2 Design of the reporting module

Architecture of the reporting module doesn't differ from the architecture of the rest of the information system. Thanks to this, I could guarantee the robustness and the security of this module. It has also provided me an easy and elegant way to access the other components in the system.

In this tree diagram you can see the parts of the IS, ktere se tykaji of reporting module:





2.2.1 Design of the back-end part

Back-end part of the reporting module was designed to follow the three-tier architecture of the rest of the system. Thus the classes that this modul uses, are devided into three different packages. Every package contains classes that belong to the same tier. These packages are called Presentation layer, Data layer and Business Layer.

2.2.1.1 Presentation layer

In the presentation layer there is a controller - a Spring MVC component that calls function according to the HTML request and its URL that is mapped to a particular function. This function executes simple operations such as verification of the passed parameters and calls to the other layers. The results of the functions from other layers may be saved to the `org.springframework.ui.Model` object and passed to the front-end. This layer never executes any more complicated actions, as those should be done within the business layer. Functions of the controller usually return the name of the view that should be displayed to the user.

2.2.1.2 Data layer

This layer contains three packages - DAO, DAOImpl and entity. In DAO package, there are interfaces of the data access objects stored, in DAOImpl are the classes that implement the interfaces from DAO package and finally in entity package, we can find the entity classes. In DAOImpl classes you can find the functions, that are used to access the database and to get some data from it. There is direct use of hibernate functions or queries in HQL. In entity classes is described, how the structures of the tables in the database look like. Every entity contains a property for every single column of the table. The type of this variable must match the type of the column of the table. Every property has to have a getter and a setter as well. Thanks to anotations from `javax.*` and `org.hibernate.validator.constraints.*` it is possible to check the requirements for perks of the input that should be saved to the database, ie. minimal or maximal size, regexp pattern or if the input may be blank. Very important is hibernate anotation `@Column`, as it describes the mapping of the property to the certain table in the database.

I would like to point out the generic GenericDAO and GenericDAOImpl classes that have been implemented. These are abstract classes that were implemented to ease the development team the programming of other DAO and DAOImpl classes. In the GenericDAOImpl class there are implemented the most used methods, that may be used by any entity such as save, delete, getCount, getCountOfUnhidden, findById and others methods. The other DAO and DAOImpl methods then extend these generic classes so then it's not needed to implement these basic functions to every single class and the programmer had to programme only those functions, that were unique for that certain class.

2.2.1.3 Business layer

2.2.2 Design of the frontend part

Realisation

-Vsechno, co se zde popisuje se nachazi v business layer, v par slovech popsati obecny postupy pri realizaci a na co se zvlast kladl duraz

3.1 Customization of the export

There was a strong demand from the contracting authority to create module that would let the user to customize the report. This is because every patient can have filled in fifteen cards. This means more than 340 properties that can be stored for every single patient. Nevertheless not all of them are needed to be exported in certain situations. Therefore was needed to implement some solution, that would allow user to select only those cards or properties of those cards, that should be exported. Vzhledem k high amount of those properties, it was quite challenging to create solution, that would meet the requirements and will be user-friendly at the same time. Treeview on the front-end and special entity on the back-end.

On the front-end there was implemented a customized treeview, where every item has a checkbox. Patient's cards are represented by the nodes, whereas properties of these cards are represented by the leaves of this tree. By checking and unchecking these items, the user can choose, what all should be involved in the exported data. Here I attach a simple example of the treeview I was writing about:

```

├─ Anamnesis.....anamnesis card
│   ├── Neco
│   └── Neco jineho
└─ Seizures.....seizure card
    ├── Seizure property
    └── Another seizure property
  
```

3. REALISATION

After confirming the export form, the state of the treeview is saved to an entity called `ExportParams`. This entity has a boolean property for every single value from the treeview in export form. Thanks to this, I am able to determine, which properties the user wanted to export and which didn't. It gives me another advantage - I can save the current setting of the treeview for later usage, but I will speak about this feature later.

OBRÁZEK TREE VIEW

3.2 Export to the particular formats

During the programming of the classes that procure the logic of the reporting, I was trying to use the fact, that there already exist java libraries that can export data to docx, pdf, xls and csv formats. I also avoided the duplication of the code by transforming data from one format to another. Thanks to these measures, it is much easier to do changes in the code of the classes that handle export itself and it also eases the understanding of the code to anybody, who would read it.

3.2.1 txt

Export to the text format is realized by components from `java.io.*` package. Specifically `java.io.FileOutputStream`, `java.io.OutputStreamWriter` and `java.io.BufferedWriter`. Output is encoded to UTF-8 format. Every property is printed out to a new line, sections are delimited by dash lines, star lines or empty lines.

3.2.2 docx

Export to the docx format was not as easy as to the txt, so I decided to look up suitable libraries, compare them and use that one, which would best suit my needs. After researching the possible solutions I ended up with two libraries - apache POI and docx4j. o POI o docx4j Proc jsem se rozhodl o docx4j

3.2.3 xls

As well as for the docx format, I used that there were already programmed libraries for export to xls. POI Neco dalsiho Proc jsem se rozhodl pro POI

3.2.4 pdf

There are of course java libraries for export data to pdf as well, nevertheless I've chosen different approach. While I already had implemented the export to docx, I've chosen not to implement export to pdf, but to create file with data exported to docx and using the classes from `apache.poi.xwpf.converter.*` package convert this file to pdf. K cemu je to dobry

3.2.5 csv

Before I started to implement the export to csv format, I was also thinking, if I should implement whole logic of export to csv, but then I decided to proceed obdobre as in the case of implementation of the export to pdf. Nevertheless now I don't use file with data exported to docx, but file with export to xls, as this format can be easily transformed to csv and vice versa. During the export to csv I create xls file with exported data at first and then I walk through this file and print out the values to the file via the classes from `java.io.*` package. These are the same classes, that have been used in the export to txt. Based on the requirements of the contracting authority, the comma, as a standard delimiter in csv format, has been replaced with semicolon. The main reason of this change was a fact, that every card, that patient may have, contains comment, which is a text, which may also contain commas. As the semicolon is a much less used character in the sentences, it was decided to use it as a delimiter. Otherwise the report could look confusingly for the user

3.3 Security and anonymization

In GENEPI - the information system there are 3 main levels of an access, according to the visibility of the patient's data.

1. Administrators

Administrators don't have any access to the patient's data. They are not even able to access the URL to view or export these data.

2. Researchers

Researchers have limited access to patient's data - they are not allowed to see sensitive data.

3. Doctors

Doctors have full access to patient's data

Because of this fact, it was needed to implement some way, how to anonymize the exported data. I have implemented two types of anonymization - volitelnou a povinnou.

Volitelna anonymization is accessible only to doctors. Before they commit the export, they can choose if they want to anonymize the data. This could be useful for example in the situations, when the doctor needs to provide the data to somebody, who normally doesn't have an access to the system and therefore can't see the sensitive informations. On the other hand, the doctor still has a possibility to export unanonymized data, which might be useful for the medical purposes. Povinná anonymization is done automatically, when somebody, who doesn't have dostatecne pristupove upraveni wants to export data. This feature will prevent anybody unauthorized from seeing the sensitive data.

3.4 Export of the verified patients

3.5 Custom sets

CHAPTER 4

Testing

CHAPTER 5

Conclusion

Acronyms

GUI Graphical user interface

XML Extensible markup language

HQL Hibernate query language

Contents of enclosed CD

```

|  readme.txt ..... the file with CD contents description
|  └─ exe ..... the directory with executables
|  └─ src ..... the directory of source codes
|      └─ wbdcm ..... implementation sources
|      └─ thesis ..... the directory of LATEX source codes of the thesis
|  └─ text ..... the thesis text directory
|      └─ thesis.pdf ..... the thesis text in PDF format
|      └─ thesis.ps ..... the thesis text in PS format

```