# Maintenance Report

*SPICSwound Web Application*

183.634 Software Maintenance and Evolution 2015S

| ***Group 10*** | | |
|---|---|---|
| Peter Eder | 0827388 | e0926758@student.tuwien.ac.at |
| Martin Dietl | 0927710 | e0430004@student.tuwien.ac.at |
| Christian Detamble | 0827283 | christian.detamble@hotmail.com |

Christian Detamble - 0827283
Martin Dietl - 0430004
Peter Eder - 0926758

# Corrective Maintenance

## Bug Report 1

*“Last development steps caused a severe bug which prevents the application from starting. (See the JBoss logs for further details.)”*

The scope of an annotated collection with @DataModel is restricted to ScopeType.UNSPECIFIED or ScopeType.PAGE[1]. Previously it was set to ScopeType.SESSION.

- /spics/src/action/bean/ParticipationManagerBean.java:52 → scope changed from ScopeType.SESSION to ScopeType.PAGE

## Bug Report 2

*“The customer complained about following error, when creating new documents and there are currently no patients created and a new patient should be recorded a wrong hint-message is shown "Es konnten keine Patienten mit der gegebenen ID gefunden werden!", as there is no search run for any patients the message should be: "Es wurde noch kein Patient für diese Dokumentation angelegt”.”*

The message was simply referenced to the wrong message resource property.

- /spics/resources/messages_de.properties, /spics/resources/messages_en.properties → added new property for message: nopatients6.info=Es wurde noch kein Patient für diese Dokumentation angelegt!
- /spics/view/viewPatients.xhtml:40 → changed message key to #{messages['nopatients6.info']}

## Bug Report 3

*“The search for users (menu item "Administration") is defective. With the provided test data a search for "mus" leads to no results. Instead the user "Dr. Muster" should be found. Correct this defect and make sure that the fields "Name" and "Anzeigename" are searched case-insensitive.”*

The search query in /spics/src/action/db/impl/UserDAO.java:69 did not meet the specified requirements. The search criterias were not case-insensitive and only matched users by given username or firstname or lastname.

- /spics/src/action/db/impl/UserDAO.java:79 → Changed query to meet requirements.

[1] http://docs.jboss.org/seam/1.1GA/api/org/jboss/seam/annotations/datamodel/DataModel.html

### Bug Report 4

*“While running some tests a strange error occurred: when documenting wound healing ("Wundheilung") with a single documentation sheet (menu item "Blätter definieren") the upwards arrow is visible the arrow is not visible in the "WHAT Studie". Find the cause of this error and correct it. (Hint: it does not have to be a programming error.)”*

Collections.sort(trialForms) in /spics/src/action/bean/TrialFormsViewerBean.java:109 caused problems if the sort attribute of a single TrialForm in a Trial is set to 1 instead of 0. The compareTo function utilizes the sort attribute in a TrialForm entity and is used by Collection.sort(...).

- /spics/src/action/bean/TestDataGeneratorBean.java:163 → Changed sort attribute to 0 since it is the first generated entity.

## Adaptive Maintenance

### Database Replacement

*“Replace the currently used database with a MySQL or Postgres database. Document all necessary steps.”*

We replaced the used database with a PostgreSQL database using the following steps.

1. Download PostgreSQL JDBC Driver from https://jdbc.postgresql.org/download.html (postgresql-9.4-1201.jdbc4.jar[2])
2. Copy driver to [JBOSS_HOME]/server/default/lib/
3. Install PostgreSQL Core Distribution to local machine
4. /spics/resources/SPICSwound-dev-ds.xml:17 → Changed local datasource specification to PostgreSQL with required credentials
5. /spics/resources/META-INF/persistence-dev.xml:13 → Changed Hibernate Dialect to PostgreSqlDialect. This ensures that Hibernate is able to communicate correctly with the database.
6. Startup PostgreSQL server and create a new table with the name “spics”

[2] https://www.google.com/url?q=https%3A%2F%2Fjdbc.postgresql.org%2Fdownload%2Fpostgresql-9.4-1201.jdbc4.jar&sa=D&sntz=1&usg=AFQjCNGbQ2UDjB39nJSR8TU4alTt-6Fkxg

## Application Server Porting

*"At the moment SPICS is currently running only on a JBoss server. In meetings with the customer the customer complaint about that fact and stated that he wanted us to support multiple application servers. Concretely they want us to support Glassfish. Evaluate the porting to Glassfish and make a cost estimation. (Bonus points will be awarded for providing an implementation.)*
*Document all necessary steps and changes you have made."*

According to the official Seam documentation[3] it is possible to use the Seam Framework on a GlassFish Server by applying the following adaptations.

| **Task** | **Description** | **Estimated Time** |
|---|---|---|
| Dependency Management | We have tried to deploy the project to GlassFish 3.1.2 but the deployment failed due to missing dependencies. The task here would be to identify the correct libraries and to remove/change unnecessary ones. | 2.5 h |
| Change of build configuration | Since libraries have to be changed we have to adapt the build file. | 2 h |
| Removal of unrelated JBoss files | The following files are JBoss related and can therefore be deleted.<br>● resources/META-INF/jboss-app.xml<br>● resources/*-ds.xml | 0.5 h |
| Configuration for PostgreSQL | In order to be able to use a PostgreSQL database in combination with the GlassFish Server several configurations need to be done using the GlassFish Administration Console[4]. | 1 h |
| Several Code changes | The alternative data export requires a directory to write the exported file to and this directory must also be accessable through a HTTP GET. It is also requried to add EJB references to the web.xml and to evaluate if the authentifaction is still working on GlassFish. | 4 h |
| **Total** | | **10 h** |

We assume a salary of 70 Euro per hour for the executing developer and thus get an estimated total cost of 10 x 70 Euro = **700 Euro**

[3] https://docs.jboss.org/seam/2.1.1.GA/reference/en-US/html/glassfish.html#d0e32275
[4] http://www.liferay.com/de/community/wiki/-/wiki/Main/PostgreSQL+with+Glassfish

## Preventive Maintenance

### Dependency Management

*“With the current build system all dependencies have to be managed manually. This could lead to problems when upgrading new versions. The standard for dependency management in java is the freely available Apache Maven project.*
*Use maven to manage the dependencies and change the ant-build-script so that it uses the libs from the maven repository. The build system should not be changed. (Hint: you could use the ant-maven-plugin. As spics.jsf.jar is available in the official maven repositories you have to add it to your local repository.)”*

Our approach is to use maven for managing all dependencies where copy the downloaded dependencies to the lib folder of the project just before compilation. Therefore we use the Maven Ant Task[5] plugin that allows to reference maven dependencies inside the ant build.xml file. To get rid of the version numbers in the jar filenames a versionmapper is used. The rest of the build script stays the same since the libraries are copied before compilation and are then used in the same way as before.

In order to prepare the pom.xml file for Maven, we inspected the existing libs and tried to identify the correct versions of each library. Typically, the version number of a specific library can be found inside the META-INF/MANIFEST.MF file. If we couldn’t find a version number we tried the newest one.

[5] http://maven.apache.org/ant-tasks/index.html

## Preventive Maintenance

### Data Export

*“At the moment the data export is implemented in a very inefficient way which leads to the problem that the JVM is running out of memory when exporting large amounts of data. As a result of that the system is getting unstable and will eventually crash. Design a solution to prevent this problem and implement your solution.*
*Document all necessary steps and changes you have made. Don't forget to measure your improvements and compare them to the measurement results before the improvements.”*

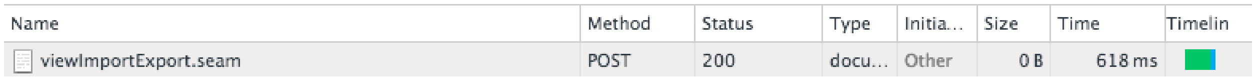
The main performance problem is the way how data is loaded for the desired export. Each TrialData entity has a list of generic values which are loaded lazy. For each CSV line the method “/spics/src/action/util/excel/FullTrialDataExport.java:buildTrialData” is called, which also calls getValues() for a TrialData entity. Now Hibernate/JPA is forced to execute an additional database request to retrieve all values for this entry, since the list of values are lazy loaded. Our solution simply pre-loads all values for the requested TrialData entities to prevent executing additional database queries for each entity.
Another improvement is an iterative approach to fetch TrialData entities. We extended /spics/src/action/db/impl/TrialDataDAO.java by to new methods which return a limited result set instead of all.

- /spics/src/action/db/impl/ValueDAO.java:69 → Add DAO method to retrieve all values for a given list of TrialData ID’s.
- /spics/src/action/db/interfaces/IValueDAO.java:23 → Add implemented method to the interface specification.
- /spics/src/action/util/excel/FullTrialDataExport.java:86 → Override export method, pass value list to method buildTrialData, set old export method to deprecated.
- /spics/src/action/util/excel/FullTrialDataExport.java:143 → Override method buildTrialData, use value list to fill-up line with remaining fields.
- /spics/src/action/db/impl/TrialDataDAO.java:125/147 → Add two new methods for retrieving a limited set of TrialData.
- /spics/src/action/bean/ViewImportExportBean.java:231 → Implemented an iterative approach to prevent a stack overflow.
- /spics/src/action/bean/ViewImportExportBean.java:249 → Call new DAO method and pass the values list to FullTrialDataExport by calling export.

**Improved Performance:**

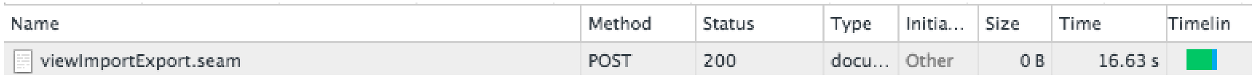
Measured by comparing the HTTP response time between the old version and the new version of the export mechanism. For this purpose we generated 1000 TrialData entities with related values in /spics/src/action/bean/TestDataGeneratorBean.java:556

Before improvements:



| Name | Method | Status | Type | Initia... | Size | Time | Timelin |
|---|---|---|---|---|---|---|---|
| viewImportExport.seam | POST | 200 | docu... | Other | 0 B | 618 ms | 🟩 |

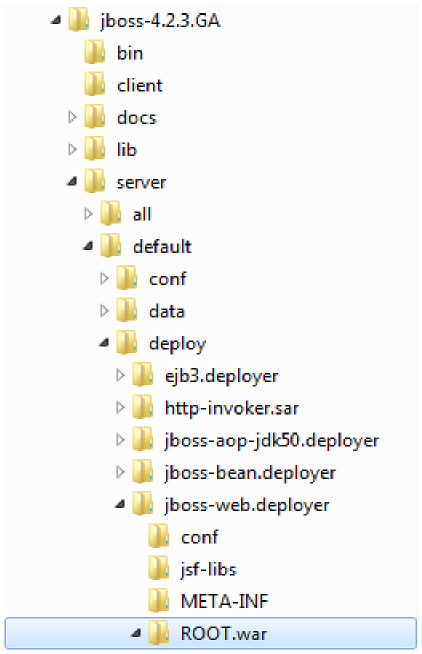
After improvements:



| Name | Method | Status | Type | Initia... | Size | Time | Timelin |
|---|---|---|---|---|---|---|---|
| viewImportExport.seam | POST | 200 | docu... | Other | 0 B | 16.63 s | 🟩 |

**Alternative: Direct File Export to the Filesystem**

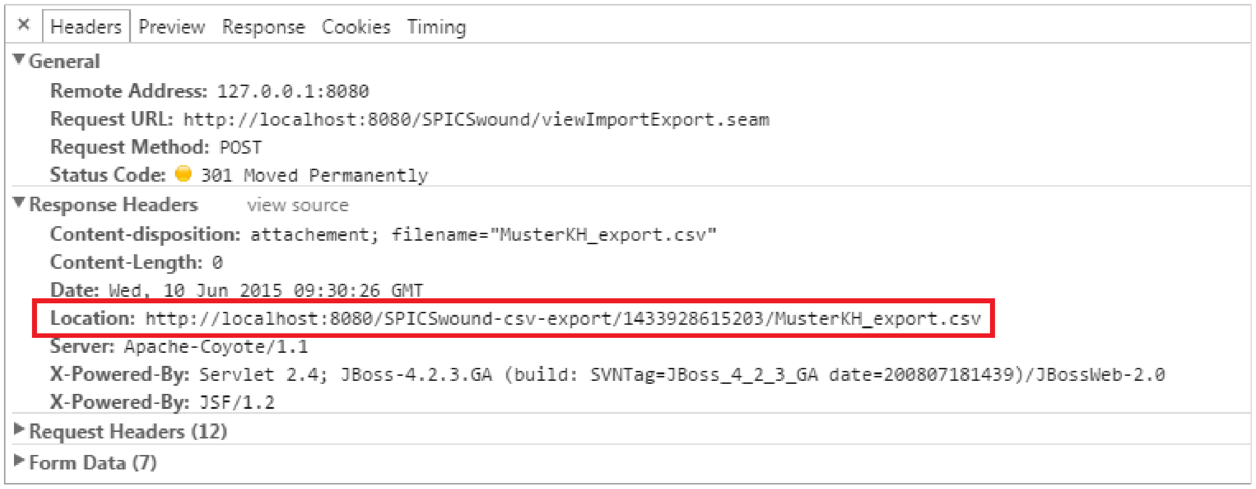
Another approach to significantly improve the stability of the data export against exceeding memory usage is to directly write the read data from the database line-wise to a temporary file on the file system and to redirect the browser to that file. With that approach it is not longer needed to serialize the complete data into memory and to write it to the OutputStream of the HTTP Respose. Thus memory errors are prevented. As can be seen in Figure 1, the [JBOSS_HOME]/server/default/deploy/jboss-web.developer/ROOT.war directory can be used to store the exported files. Inside this directory, we create the “SPICSwound-csv-export” folder and for each export we create another folder named after the current timestamp we get from System.currentTimeMillis() in order to avoid write errors.



```
jboss-4.2.3.GA
  bin
  client
  docs
  lib
  server
    all
    default
      conf
      data
      deploy
        ejb3.deployer
        http-invoker.sar
        jboss-aop-jdk50.deployer
        jboss-bean.deployer
        jboss-web.deployer
          conf
          jsf-libs
          META-INF
          ROOT.war
```

Figure 1: Public JBoss Directory

After the export, the browser receives a HTTP Response that redirects the user to the exported file, as can be seen in Figure 2. Exported files from the past are being deleted whenever a new data export is requested in order to save disk space.



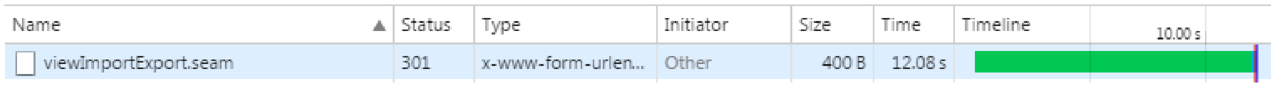
```
▼General
    Remote Address: 127.0.0.1:8080
    Request URL: http://localhost:8080/SPICSwound/viewImportExport.seam
    Request Method: POST
    Status Code: ● 301 Moved Permanently
▼Response Headers    view source
    Content-disposition: attachement; filename="MusterKH_export.csv"
    Content-Length: 0
    Date: Wed, 10 Jun 2015 09:30:26 GMT
    Location: http://localhost:8080/SPICSwound-csv-export/1433928615203/MusterKH_export.csv
    Server: Apache-Coyote/1.1
    X-Powered-By: Servlet 2.4; JBoss-4.2.3.GA (build: SVNTag=JBoss_4_2_3_GA date=200807181439)/JBossWeb-2.0
    X-Powered-By: JSF/1.2
▶Request Headers (12)
▶Form Data (7)
```

Figure 2: HTTP response of the data export

The following code changes have to be done in order to implement this improvement.

- /spics/src/action/bean/ViewImportExportBean.java:212 → A second method for directly exporting the data to the file system and to return a HTTP response that redirects the user to the created file
- /spics/src/action/util/excel/FullTrialDataExport.java:122 → Iterate over all read TrialData objects from the database, build the strings to export and directly write them to the filesystem using the given FileWriter

As we have not incorporated the same speed improvements as in the maintenance measurement from above, just gain a little performance (~4 sec) improvement besides the significant stability improvement.



| Name | Status | Type | Initiator | Size | Time | Timeline |
|---|---|---|---|---|---|---|
| viewImportExport.seam | 301 | x-www-form-urlen... | Other | 400 B | 12.08 s | 🟩 |