Sophie Export to iPad and Android Project

Code Audit Results

October 29, 2011

A report of results in fulfillment of a grant to the University of Southern California
from The Andrew W. Mellon Foundation

Table of Contents

1. Introduction

The Andrew W. Mellon Foundation funded a project to develop new software for the Sophie system to allow it to export e-books that can be viewed on both iPad and Android devices.

As part of the development process, an independent code review was performed on all software developed during the grant term. An independent software analyst was retained for the review. The effort was done incrementally over the last three months of development and following release, comprising three interim reviews and a final review. Interim reports were shared with development staff, providing information useful in improving the code during the process, rather than having such information available only after development had been completed.

The code was developed by Astea Solutions AD, Sofia Bulgaria. The code auditor was Thavatchai Makphaibulchoke, PhD, Fort Collins, Colorado. This document includes incremental code audit reports and information supplied to the code auditor by the development group during the process.

The final code was released as Sophie 2.1 on October 21, 2011. The code review in this document applies to that release.

2. Code Review Results

In the final audit report, Dr Makphaibulchoke adjudged the reviewed computer code and documentation developed during this project to be of professional quality, following accepted practices for agile development environments. See section 8 below.

3. Initial Review Report

T. Makphaibulchoke, PhD
Code Reviewer
July 17, 2011

Sophie Export to iPad and Android Project Code Audit Report

Coding practice

- Good practice shown with the use of the override directives to ensure methods are correctly overwritten.
- Extensive uses of parameterized class help make the code highly reusable.
- Extensive uses of inheritance, making the code highly extensible and reusable. Easy to maintain
- Moderate uses of various types of supersswarnings directives, such as superwarnings("rawtypes"). Presumably this is done only in the early phase of the project. The final product should be suppresswarnings clean.
- Extensive use of final attribute to help compiler generate highly optimized object.
- Most classes have no constructor dependence upon the assumption that the base class has a default constructor.
- Most of the functions do not do sanity check to see if any of the object reference parameters are not null and proceed to use it in a way that could cause errors. At the least there should be an assert test to make sure that these arguments are not null.
- Java guidelines are not followed for most class order of declarations, variables, method, etc.

Naming

- Generally good variable naming.
- In a few cases, variable names are too terse and do not convey meaning.

Comments

- Most of the packages contain very minimal comments
- Many of the comment do not follow java guidelines

Testability

- No support for testing functionality.
- Insufficient error condition checking.

Comments for each individual package

Jsonexport Package:

JsonExportModule

- Order of method declaration does not follow java guideline – instance variable should precede inner class declaration, which should precede static method, which should come before instance method

- Method DefineExtensionPoitn(List<T<?>> res) – should at least have an assert on res not null, as res.add methods are invoked on res in a number of areas

  No constructor – SophieModule based class must have a default constructor

abstract class JSONPersister<T> extends Persister<T, JSONObject>

- Static method should come before instance method

- No constructor – Persister<T, JSONobject> must have a default constructor

Convert Package:

- Not sure what the swp file is for.

IpadVideoConverter extends FfmpegMediaConverter

- Extension, FormatDescription, MaxHeight and MaxWidth are class's attribute. It would be better if they were declared as a class's final variables and have all the corresponding get attribute function return the proper value.

- To be safe, checkAdditionalForFormat should check the integrity of the ffmpegOutput both to validate whether the reference to the string array itself is null and also if any of its strings are null. An assert would be sufficient.

- checkAdditionalForFormat – it would be more modular if the based class were split into two modules, one for video and the other for audio, instead of using an input flag.

- checkAdditionalForFormat – string video – line wrap.

- To be safe, getExecutableCommand – may want to check if both target and dest are not null, with an assert.

- AudioCodecs, VideoCodecs and Containers List are classs's attributes. It would be better if they were declared as class's static varaiables and returned through the appropriate get method.

XoomVideoConverter extends FfmpegMediaConverter

- Same here - Format, Extension, Description, Containers, MaxHeight, MaxWidth, videoCodecs and AudioCodecs would be better as class variables.

- Same - getExecutableCommand should check if target and dest are not null.

model package:
ExportInfoH extends ResourceH

- To be safe, methods that take object references, eg: ExportInfoH, getExportId, etc, should check to make sure that the object is not null, at least by an assert.

Net package:
`IpadExportVersion extends BaseExportVersion`

- Should at least have an assert on object reference arguments not null, eg: getDownloadInfo() uses string bookLocation extensively.

- No constructor – Persister<T, JSONobject> must have a default constructor

- Declaration of members and methods does not follow java guidline.

- uploadSpecific - recommend checking if uploaded is true before creating string dir and then not calling kind.upload on dir later.

- ConvertSpecific – why just retry only twice?

JsonBookInfo implements Comparable<JsonBookInfo>

- Does not check to make sure object reference parameters are not null, especially in the constructor for parameters that contain vital infomation. These parameters are assigned to member variables and could propagate null values somewhere else.

- Use of the this prefix is debatable as it could be too cluttering. Should be consistent across the project.

- All modifyXXX functions' names are misleading. It's actually cloning with new XXX. Also, each function could just call new with the member variables without having to call the appropriate get fucntion to get its own member variables.

- Equals – could compare other == null first.

abstract class BaseExportVersion

- JSON_TO_HTML_SCHEMA and BOOK_TO_JSON_SCHEMA should have the final attribute.

- String getCoverLocation should check to make sure String is not null

- PersistenceOptions should check to see whether View or file is not null.

FtpServerKind extends ServerKind

- Connect should check to see if both connection and loginInfo are not null
- Wouldn't userName.equlas("") be more readable than "".equals(userName).
- All method that has object reference as parameter, such as upload and download, should check if all object parameters are not null.
- The message "Could not close disconnect" should be "Could not disconnect"
- InformationDescription should be implemented as class's variable.
- status.equals(ConnectionStatus.DISCONNECTED) would be more readable than ConnectionStatus.DISCONNECTED.equals(status)

LibraryXml
- Equals – could compare other == null first.
- addBook is not really adding a book to an LibrayXml object. It creates a new LibraryXml object whose books are the original book + new book.

XoomExportVersion extends BaseExportVersion
- No constructor
- saveDownloadInfo – first arg bookInfo is never used.

Logic Package:

BrowserFrameViewJSONPersister extends FrameJSONPersister<NativeBrowserFrameView>
- Export – line wrap

abstract class ChannelJSONPersister<T, C extends Channel<T, C>> extends JSONPersister<C>
- Export- why JSONObject currentJsonCycle = new JSONObject()?; it is first created before the loop. First thing in the loop, it re-creates a new currentJsonCyle.
- isPrev - returns hasTime && hasProperty && sameMillis && samePhase && sameProp; doesn't really need to check hasTime && hasProperty again as we are in the then part.
- hasTarget – always returns false, assume it is to be implemented.
- persistedTarget – always returns null; assume it is to be implemented

CommentFrameJSONPersister extends FrameJSONPersister<CommentFrameView>

- getSchema – a good candidate for class method; always returns the same value for all objects.

CommentFrameJSONPersister extends FrameJSONPersister<CommentFrameView>

- getSchema – a good candidate for class method; it always returns the same value for all objects.

EmbeddedBookFrameJSONPersister extends FrameJSONPersister<EmbeddedBookFrameView>

- Line after JSONManifestUtil.addingInfoForManifest is in an unusual format

- getSchema could be a class method.

ExportVersionDialogLogic implements OperationDef

- versions.remove(version) – could attempt to remove a non-existing version

HotStyleDefJSONPersister extends JSONPersister<HotStyleDef>

- export does not check for null StyeDef

- FtpService contains un-encrypted user/password. This should be reviewed for possible security issues

model package:

ExportInfoH extends ResourceH

- To be safe, methods that take object references, eg: ExportInfoH, getExportId, etc, should check to make sure that the object is not null, at least by an assert.

view package:

- No comments.

4. Second Review Report

T. Makphaibulchoke, PhD
Code Reviewer
August 21, 2011

Code Review – Sophie iPad / Android Export Summary

Comments

- The comments have resolved a number of concerns.

Documentation

- The documentation provided is insufficient. Do you have design and data flow documents to review?

5. Developer Response to Initial Review Report

Sophie Development Team
Astea Solutions AD
September 16, 2011

The review was quite extensive and brought some good points to our attention. We took all the notes under consideration and modified the code in accordance with some of them. For some of the issues that were listed, however, our approach is somewhat different. Attached is a document with our comments on the review notes as well as some general information about our coding conventions and guidelines.

Review comments

This document contains our comments to the review notes. For some of the notes we have already implemented the suggested fixes or have planned to do so in the near future.

On some of the other notes, however, we would prefer to stick with our existing guidelines as these have been established for the whole project. An insight into our coding conventions and general guidelines can be provided by the document here: http://sophie2.org/trac/wiki/GoodCodeExamples

I. Below are listed our comments on the general notes in the review:

* About the suppress warnings for raw-types there are some cases, for example when dealing with collections, in which there is no good way to remove them. One way would be turning off the warnings for raw-types/unchecked, which in other cases is bad. However, for all other types of suppressing directives, we are trying to stay clean of them.

* About the constructors, our policy is against adding empty constructors that do nothing, it is overhead for us.

* About the sanity checks, we think your review brings a good point. We need to add assert tests, and we will try to add them where necessary.

* Java guidelines indeed are not followed for most classes as far as the order of declarations, variables, method, etc is concerned. We will try to follow these guidelines from now on.

* About the comments we do not follow the java guidelines, we have our own Java-Doc practices that are defined in http://sophie2.org/trac/wiki/GoodCodeExamples

* About package comments - we need to add package comments. We will try to do it from now on.

* About testing - We have unit tests, the problem is that many of them are broken or are not usable. We are currently trying to add new unit tests and fix older ones. However the tests in the 'export to json' module are good ones.

* The review found some things like forgotten final modifiers or bad messages, which we will fix; it gave us some ideas for beter practices, which we will employ.

II. Here are our comments on the individual classes review:

1. Converters - the SWP file is probably a temporary file for some text editors. It's probably generated for your text editor, as we haven't been able to find it on our end.
   - IpadVideoConverter, XoomVideoConverter :
     * we turned all the mentioned data into final class variables
     * we added the sanity checks.
     * checkAdditionalForFormat - it is used only for video for now, we should divide it into two methods indeed.

2. ExportInfoH - we added the sanity checks.

3. IpadExportVersion - we added the sanity checks and modified uploadSpecific. About the retries, in the other note, the last argument is different, because for some videos if the first try doesn't pass, this difference fixes the issue.

4. JsonBookInfo - we have our own naming convention for some operations, modifyXXX for example is for creating a new immutable object from an old immutable one. Fixed other items.

5. BaseExportVersion - fixed the issues.

5. FtpServerKind - we added the sanity checks. Also, when calling the 'equals' method we have a politic of using the CONSTANT.equals(objectInstance) format

6. LibraryXml - about the addBook - for immutable collections we use the add method for generating a new immutable collection with the objects from the old collection plus a new object. LibraryXml is kind of immutable collection.

7. ChannelJSONPersister - fixed the issues. Yes your assumptions are right.

8. About all the persisters - getSchema is a method that must be implemented. It is one and the same for the objects, but only one instance for each of the persisters is registered anyway. The issues for persisters in general are fixed.

9. FtpService - the FTP servers used are just book repositories. We don't think there can be security issues when using them from the desktop application.

6. Developer Response to Second Review Report

Sophie Development Team
Astea Solutions AD
October 14, 2011

Regarding design documentation and data flow, the data is spread over the different tickets, as is the case with most Agile-developed projects. We include references below.

Sophie HTML5 Export Documentation

The goal of this document is to provide a summary of all the documentation that is available about the process of exporting a Sophie book and packaging it into a HTML format.

I. JSON based data model

At the base of the export functionality lies a JSON based data model that describes the content of every page of the book as well as the book in terms of the pages that make it up. The format of the data model can be seen in detail here:
http://sophiehtml5.asteasolutions.net/trac/wiki/JSON%20Data%20Model

II. Process overview

An overview of the process that the data goes through from Sophie 2 book to an HTML5 package is provided here:
http://sophiehtml5.asteasolutions.net/trac/wiki/Process%20Overview

III. Export to JSON

The JSON export logic uses the Sophie 2 Persistence API, which converts the Sophie model to other data models with the help of persisters. In Sophie 2.1 the Persistence API has been expanded to convert Sophie 2 resources to JSON based.
For the majority of the items the persistence process takes a Sophie component and persists it into its respective JSON formatted version. There are some specific aspects to the export though, which can be seen in detail in the tracking system:

1. Some of the media resource formats that are supported by Sophie 2 are not supported by iPad or Motorola Xoom (for example the h264 codec based file formats) and hence cannot be played on the respective devices. Therefore the video resources are preprocessed and converted before they are exported. A new media conversion module for Sophie was implemented for Windows, MacOS, and LINUX. The work on the conversion module is documented in the following tickets:

- http://sophiehtml5.asteasolutions.net/trac/ticket/66: first prototype of the video conversion logic, supports conversion only for iPad target system, bundled in with the JSON export Sophie 2 module.
- http://sophiehtml5.asteasolutions.net/trac/ticket/71: added parameters such as key-frames and target size when converting video files
- http://sophiehtml5.asteasolutions.net/trac/ticket/129: a new module for converting logic is added and conversion for Android based devices (in this instance Motorola Xoom) is implemented
- http://sophiehtml5.asteasolutions.net/trac/ticket/162: media export optimizations

2. Commenting books in Sophie works only with the Sophie server. Therefore we needed to implement a special kind of server that can be used by the comment frames in the exported books. A URL is specified for every comment frame in Sophie and the server at that location is then used by the comment frame in the exported version. The comment export and the comment server are documented in the following tickets:

- http://sophiehtml5.asteasolutions.net/trac/ticket/84: Design of the JSON format for the new comment frames.
- http://sophiehtml5.asteasolutions.net/trac/ticket/85: Design for the HTML 5 comment frames.
- http://sophiehtml5.asteasolutions.net/trac/ticket/99: Comment frames export process.
- http://sophiehtml5.asteasolutions.net/trac/ticket/89: Starting development on the comment server.
- http://sophiehtml5.asteasolutions.net/trac/ticket/90: Picking up and using a database to store the comment's content on the server.
- http://sophiehtml5.asteasolutions.net/trac/ticket/91: The server declares a few services to be used by the comment frame client.
- http://sophiehtml5.asteasolutions.net/trac/ticket/87: Comment frames-server communication.
- http://sophiehtml5.asteasolutions.net/trac/ticket/88: Comment content fetched from the server is displayed in the comment frames.
- http://sophiehtml5.asteasolutions.net/trac/ticket/230: UI design for comment frames implemented.

3. The images in Sophie can be quite large, but when exported they should be resized to the size of the frame that contains them to reduce the exported book's size. This process is documented in the following tickets:

- http://sophiehtml5.asteasolutions.net/trac/ticket/65

4. The exported HTML 5 books can be uploaded to FTP servers, but the directory in which they are uploaded through Sophie 2 becomes a special library directory. There is logic for choosing an FTP server location to upload a book and for converting the location into a Sophie library for HTML exported books if not already converted to one. The following tickets document this logic:

- http://sophiehtml5.asteasolutions.net/trac/ticket/100: UI design how the user should add and choose to upload to FTP server.
- http://sophiehtml5.asteasolutions.net/trac/ticket/103: In the library all the iPad compatible books have manifest file, which contains the relative paths to all the resources in the books. This can be used by off-line application to download these resources.
- http://sophiehtml5.asteasolutions.net/trac/ticket/107: thumbnails for every page are exported for the library
- http://sophiehtml5.asteasolutions.net/trac/ticket/127: a library XML file that defines a remote directory as a Sophie HTML 5 library is generated and updated when uploading a book
- http://sophiehtml5.asteasolutions.net/trac/ticket/140: t he user can chose between export versions (iPad, Android XOOM, etc.) when uploading to the server.
- http://sophiehtml5.asteasolutions.net/trac/ticket/151: in the library the books have unique ID’s
- http://sophiehtml5.asteasolutions.net/trac/ticket/199: an FTP server can contain multiple libraries.
- http://sophiehtml5.asteasolutions.net/trac/ticket/181: books can be recognized by their ID and when uploaded can be updated.
- http://sophiehtml5.asteasolutions.net/trac/ticket/269 : books exported for Motorola Xoom have different locations in the library.

5. The export process takes a while (for example converting of video files is a slow process) so a progress bar was integrated for it in Sophie 2. The following tickets document the implementation of this functionality:

- http://sophiehtml5.asteasolutions.net/trac/ticket/247: the progress bar implementation for exporting a book and saving it on server
- http://sophiehtml5.asteasolutions.net/trac/ticket/289: the export process can be canceled from the progress dialog.

6. Fonts should be embedded in the exported books, because they do not look the same as the original in the iPad (font size, font weight, etc are different). The embed functionality is implemented in the following tickets:

- http://sophiehtml5.asteasolutions.net/trac/ticket/266: design of how fonts should be exported the fonts from Sophie
- http://sophiehtml5.asteasolutions.net/trac/ticket/273: research and prototype

implementation.

- http://sophiehtml5.asteasolutions.net/trac/ticket/277: embed fonts implementation

7. Video poster should be exported, because in the exported book before playing a video the user should see a preview frame for it:
http://sophiehtml5.asteasolutions.net/trac/ticket/267: implementation of the functionality

7. Third Review Report

T. Makphaibulchoke, PhD
Code Reviewer
October 18, 2011

Sophie Export to iPad and Android Project Code Audit Report

The documentation package that you have supplied has been reviewed, and meets or exceeds good practice in Agile development. After reviewing the areas pointed out as earlier statements of concern, your responses and changes have resolved these issues.

There are no additional coding concerns at this review.

8. Final Review Report

T. Makphaibulchoke, PhD
Code Reviewer
October 25, 2011

Sophie Export to iPad and Android Project Code Audit Report

Based on a final review of the released computer code developed for this project, reviewer has found it to be of professional quality and properly documented.

9. Intellectual Property

Sophie 2.1 is free and open source; the code and documentation can be found within the “Developers” section on the Sophie 2.1 website: sophiecommons.org. The code is licensed under the ECL 2.1, and all other information is licensed under the Creative Commons By Attribution 3.0 United States License. Licensing information is available from the Sophie Website.

10. iPad / Android Export Project Staff and Contributors

Elizabeth Daley and Holly Willis, Co-Principal Investigators

Nick Matelan, Project Manager

Ludmil Pandeff and the Sophie development group at Astea Solutions

Thavatchai Makphaibulchoke, Code Auditor

11. References

1. Reading Sophie Books on iPad and Android, A Proposal to the Andrew W. Mellon Foundation.