

Castle (PDF Filler)Design Document

By Cameron Lilly, Alex Mitchell, and Daniel Miller

11/3/2014

An overview of the problem

Project Castle is a PDF report generating application. Many businesses require filling out reports where duplicate information is required to be added multiple times. This application solves that problem and allows users to input answers to a question wizard to automatically reflow and generate a finalized PDF report from a template.

Users will import a PDF document that contains various fields that are specific to a particular instance of the document. These fields will be detected and have a particular prompt assigned for each unique field. The user then responds to the prompts. When all responses have been answered, they are inserted into the document and presented to the user in a PDF. The final exported PDF will not reflect that responses were inserted, but will appear that the document was uniquely created.

The target users are small companies with PDF generation needs. i.e. Home Inspectors who have one PDF that will need to be filled out upon inspection. The application will be used multiple times to answer similar questions for the different homeowners. The use of the application will allow for easy accessing and completion of the PDF document in the field.

This project will be able to be deconstructed and analysed in order for the sponsor to form a fully marketable product. The project may also be sent to specific users for testing and actual use.

Required Features:

- Will allow users to provide responses to prompts in an easy to use and understandable interface.
- Will support the following response types:
 - Text Box
 - Selection Box
 - Radio Buttons
 - Check Boxes
- The final exported PDF will not show that the responses were added to the template. (Document will *reflow* to fit responses seamlessly)

Stretch Goals (in order of importance to sponsor):

- Allow the user to import their own PDF templates and parse through document to prompt for and fill with responses (as opposed to a preset prompts)
- Add additional response types:
 - Signatures
 - Photos taken with Android device
 - Pre-defined drawings and images
- Make application compatible for Android devices

An overview of the design / approach

Our application will accept a PDF Template that contains keywords that are meant to be parsed by the main Java application. The main application will have a question builder that will allow the user to associate a question to each of the different key words from the template. Once a template has key words and questions, the main application will allow the user to save the new PDF Template/question combo so that the user can go through a question wizard to input answers to all of the prompted questions. The main application will then interface with a PDF API in order to take the answers, insert them into the template where the key words were specified, and then export the result as a new edited PDF. The process will be repeatable and the user can use the same template with new answers to the questions to continue to generate new formatted PDFs.

A system level description of how the major components will interact

The user will open the app which will create a CastleApp class that will then have a QuestionBuilder, QuestionPrompter, and PDF (Which will inturn have an XMLBuilder). The CastleApp will have a view that will allow allow a person to Load and Quit.

The most common user story will be that a user will see a menu bar with Load and Quit. They will then click the Load button. This will allow the user to choose a template that they have made. This PDF will then find the XML that is in it and call the XMLBuilder that will help parse the hashes out of the XML and save them in a list. The hashes that came out of the XMLBuilder will then become the keys for the Questions and Answers maps in the PDF. The PDF class, now finished with loading, will give control back to the CastleApp class.

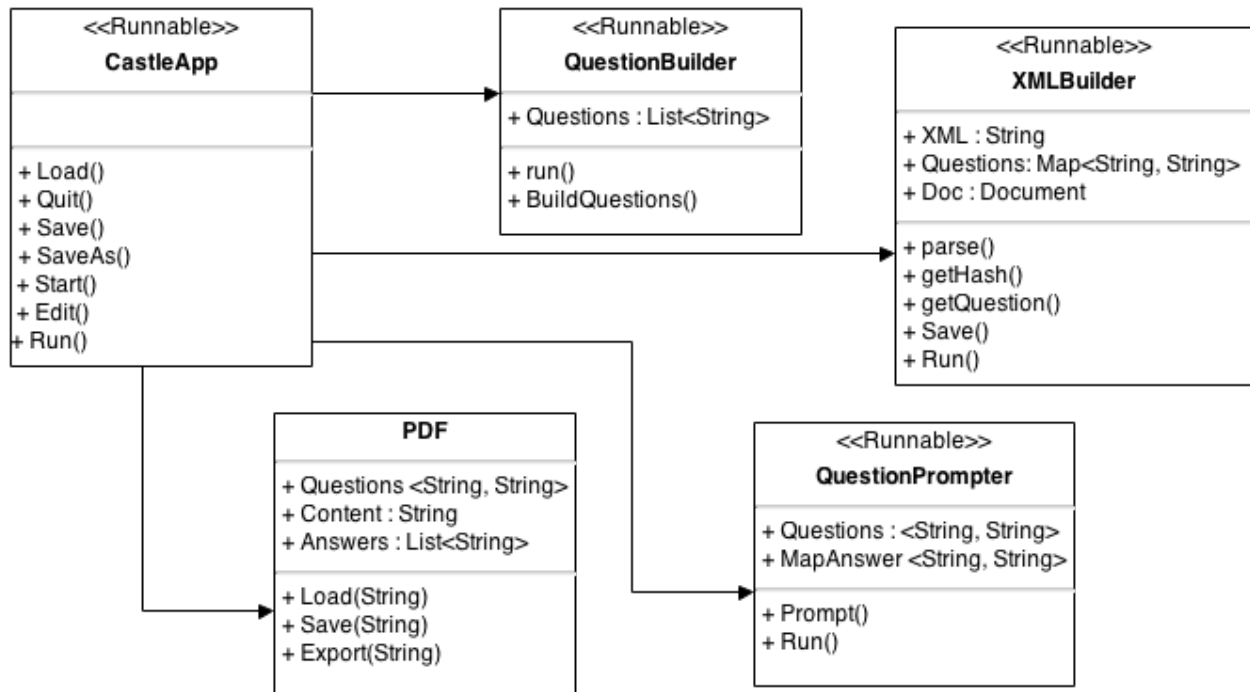
Assuming that the PDF does not yet have questions for each hash the user will be asked if they would like to start building questions. The user will then say yes and the QuestionBuilder will be ran. This will build a question for each unique hash. The CastleApp will set PDF's questions to the updated questions that came back from QuestionBuilder.

The user will then see the menu with Load, Save, Save As, Edit, Start, and Quit. The user will then save the template and this will call the CastleApp save method which will then call the PDF save method and sending in the file that they loaded from. The PDF will then add the content and will then call the XMLBuilder's save sending that same file. The XMLBuilder will then add in the necessary XML. When all is said and done the user should not even see a difference in the ui.

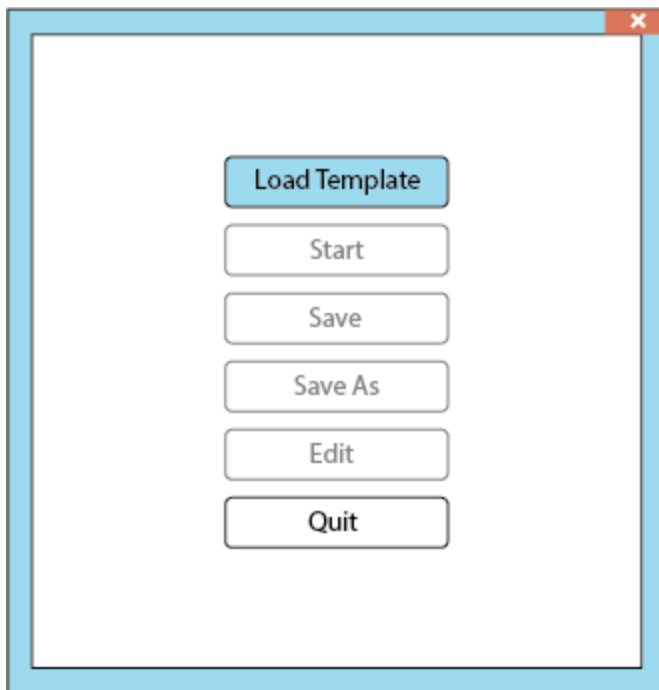
Now the user will start the questionare. The QuestionPrompter's questions will be set from getting the PDF's questions and will begin prompting for responses for each question. After all questions have been answered the PDF's answers will be set from getting the QuestionPrompter's answers. The user will then be prompted on where to save the completed document. The user will select a location and CastleApp will then call PDF's export sending in that location. The PDF will then be generated, reflowing as necessary and then will call the XMLBuilder and send it the file and have XMLBuilder add the backend XML to the document.

Now the user will be done using the program and will click Quit. CastleApp will then call it's closing method and quits the program.

UML showing the planned classes and relations among them



User interface plans



A window with a light blue border and a red close button in the top right corner. The window contains a vertical stack of seven buttons: 'Load Template' (highlighted in light blue), 'Start', 'Save', 'Save As', 'Edit', and 'Quit'.

Load Template

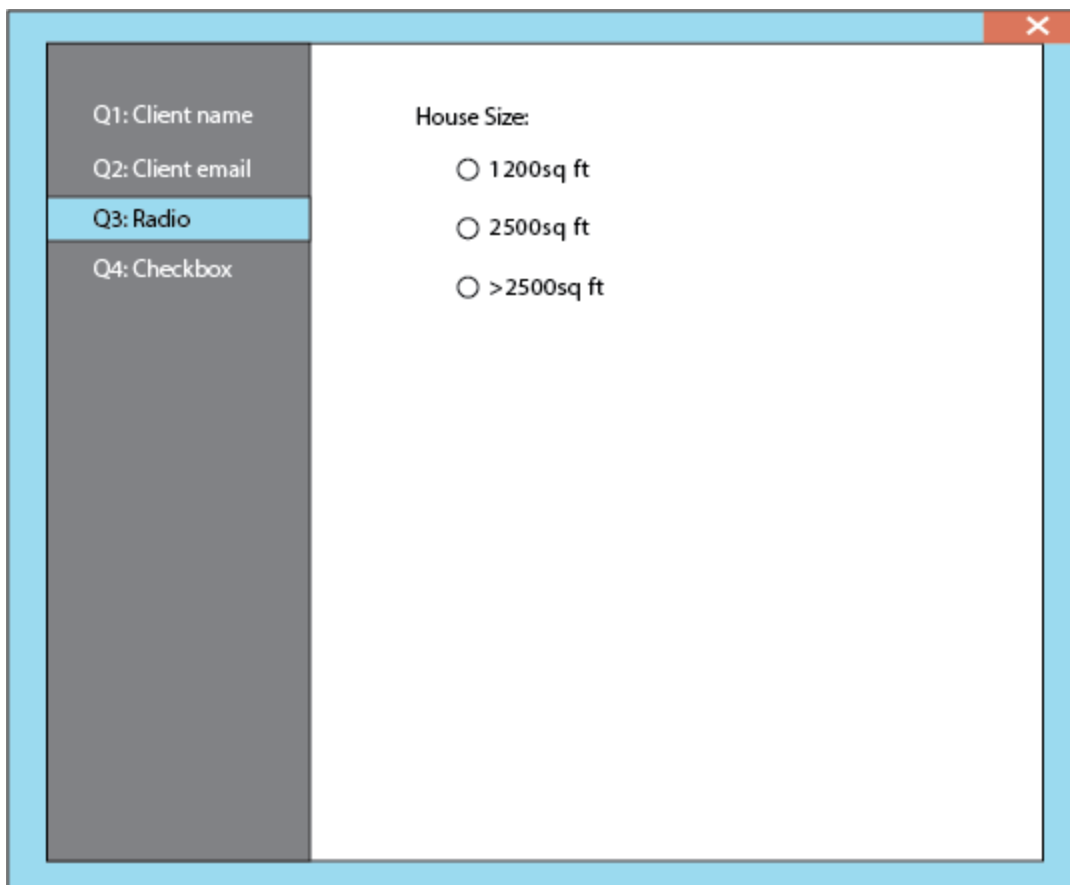
Start

Save

Save As

Edit

Quit



A window with a light blue border and a red close button in the top right corner. The window is divided into two main sections. On the left is a dark gray sidebar with four labels: 'Q1: Client name', 'Q2: Client email', 'Q3: Radio' (highlighted in light blue), and 'Q4: Checkbox'. On the right is a white area with the label 'House Size:' followed by three radio button options: '1200sq ft', '2500sq ft', and '>2500sq ft'.

Q1: Client name

Q2: Client email

Q3: Radio

Q4: Checkbox

House Size:

☐ 1200sq ft

☐ 2500sq ft

☐ >2500sq ft