Expertiza Final Document: Replacing Goldberg

Chad Brady, Hemalatha Elangovan, and Mark Fashing

Table of Contents

1 - Introduction

Purpose

The current version of Expertiza works well with Rails 2.2.2 but it is not compatible with newer versions of Rails. This prevents developers from taking advantage of new features available in more recent versions of Rails. This project focuses on porting the current implementation of Expertiza to Rails 2.3.8. As a secondary objective, it may be desirable to replace certain components of Goldberg with more modern libraries.

Problem Definition

The main source of incompatibility between Expertiza and Rails 2.3.8 is the Goldberg CMS framework. Goldberg is not compatible with Rails versions higher than 2.2.2. Areas where Goldberg has its roots are :

- authentication
- rights and roles
- navigation structure
- content pages

It is unlikely that it will be necessary to replace Goldberg entirely. Instead, there are probably only a handful of version-related incompatibilities that prevent Goldberg from operating properly in Rails 2.3.8. Identifying and resolving these incompatibilities is key to completing this project successfully.

Since this project focuses on identifying issues with Goldberg and the primary developers would like to see all of Goldberg replaced, it also presents an opportunity to replace some components of Goldberg with better maintained and better documented software libraries. In particular, the authentication library Authlogic and the authorization library Acl9 are good candidates for replacing the authentication and rights and roles components of Goldberg.

2 - Design Plan

Upgrade to Rails 2.3.8

The primary goal of the project is to upgrade Expertiza to Rails 2.3.8 and enable it to move forward as Rails itself is upgraded.

A known incompatibility between Goldberg and Rails 2.3.8 is that Goldberg stores a substantial amount of information about the current session in a session variable named :p_store. Since Rails 2.3, :p_store is no longer supported. Instead, client-side session data must be kept in a CookieStore, which is limited to 4kb in size. This encourages developers to store as much session data as possible in the database rather than with the client.

Since Goldberg relies on the existence of :p_store for functionality like content management and navigation which are necessary for using Expertiza, it is likely that this issue will have to be resolved before it will be possible to identify and resolve other incompatabilities between Expertiza and Rails 2.3.8. Resolving this issue will probably require identifying a minimal subset of the session information in :p_store and moving it into a CookieStore. The rest of the session information will need to be moved into the database.

Aside from this issue, there may be others which are unanticipated. In order to ensure that Expertiza functions properly in Rails 2.3.8, it will be necessary to test Expertiza for incompatibilities. This will necessitate developing a test plan which covers common use cases and as much of the functionality of Expertiza as possible. Since this test plan should be developed under Rails 2.2.2, development of the test plan may be done concurrently with addressing the :p_store issue.

Once the :p_store issue has been addressed, the use cases in the test plan will be exercised to identify version incompatibilities with Rails 2.3.8. Just as the :p_store issue may hamper testing, issues that arise while exercising the use cases may prevent some functionality from being exercises. Therefore, it is possible that testing will require an iterative test and debug approach with several cycles before all use cases have been fully tested.

The results of the test plan will determine how feasible it will be to meet the secondary objectives. For example, if there are only a few simple incompatibilities, these may be assigned to a single team member, while other team members work on replacing Goldberg authentication with Authlogic.

Replace Goldberg authentication with Authlogic

A secondary goal of this project is to replace Goldberg authentication with Authlogic. The Authlogic library provides user authentication and has broad support in the Rails community, so it makes a good replacement for the Goldberg authentication. Authlogic supports a broad range of authentication and encryption options and is designed by the authors to be extensible, which will enable easier upgrades to the authentication in Expertiza in the future.

To install Authlogic into Expertiza config.gem “authlogic” will be added into config/environment.rb. After the upgrade to Authlogic is complete, old passwords must continue to work with the new system.

An Authlogic::CryptoProviders:SHA1 provider from Authlogic will be used to match the SHA1 password format currently used by Goldberg. This will ensure that all current passwords stored in the Expertiza database will remain compatible. There are three functions within app/models/user.rb that will require modification: before_save, check_password, and send_password.

There is a pg_user.rb that contains comments that indicate it is there to prevent the main Goldberg user.rb from being modified, but that obviously happened anyway. The functionality from pg_user.rb will be integrated into user.rb to consolidate all of the User model into one file. To enable Authlogic within the User model, acts_as_authentic will be added to app/models/user.rb.

Authlogic will expect the user table in the database to have a minimum list of fields. A db migration will be configured to add the appropriate fields and indexes to the user table. The exact migration hasn't been worked out yet, but the final form for the table is expected to be similar to the column descriptions in the table below. The final database table fields may differ from the excerpt below once the Authlogic requirements are analyzed further.

Column	**Type**
login	string
crypted_password	string
password_salt	string
persistence_token	string
login_count	integer
failed_login_count	integer
last_request_at	datetime
current_login_at	datetime
last_login_at	datetime
current_login_ip	string
last_login_ip	string

user table description from http://www.releasereality.com/blogs/2009/6/21-getting-started-with-user-authentication-in-rails-authlogic

Authlogic also supports a UserSession model which tracks the current session for the user. The UserSession will be added to Expertiza and will support the login and logout operations.

Replace Goldberg authorization with acl9

Another secondary goal of this project is to replace Goldberg permissions with Acl9. Authorization is the task of allowing the authenticated user to perform specific actions. It is different from authentication. In the Expertiza authorization is performed by Goldberg plugin where the current user has been checked and then allowed to access the page or object.The session stores the entire information needed for the

currently logged in user to access different objects.

Most of the permission checking is done in the vendor/plugins/goldberg_filters/lib/goldberg_filters.rb.
It has to be replaced by a method which doesnot use Goldberg plugin.acI9 is a role-based authorization in Rails. acl9 can be installed as a gem or installed as a plugin. Here all the role checking process has been boiled down to calling a single method subject.has_role?(role, object).
It consists of two subsystems which can be used separately.

- Role control subsystem allows to set and query user roles for various objects.
- Access control subsystem allows to specify different role-based access rules inside controllers.

3 - Actual Implementation - Files Modified

environment.rb

- Added AuthLogic gem to the project in environment.rb
- Updated environment.rb to set the session store to ActiveRecordStore.

login.rb

- Removed login.rb from the project because the Login model wasn't being used.

user.rb

- Removed validates_presence_of :name and validates_uniqueness_of :name because this is guaranteed by Authlogic
- Removed before_save and after_save from user.rb as the encryption work is handled by Authlogic
- Moved initialize function from pg_user.rb to user.rb
- Deleted pg_user.rb from the project because user.rb had already been customized
- Updated check_password to use Authlogic for password comparison
- Updated send_password to use Authlogic for password update
- Set Authlogic encryption stretches property to 1 in initialize function to ensure that the password encryption matches the old password scheme.
- Updated check_password to match the password.

user_controller.rb

- Removed password confirmation check from create function in because it is handled in Authlogic
- Removed password confirmation check from update function in because it is handled in Authlogic
- DRY principle: Updated the edit function to use the getRole function instead of repeating the same functionality
- Updated index function to use redirect_to for list action.

_login.html.erb

- Fixed an issue with HTML table tags on line 22 of _login.html.erb. This was a bug that existed with the initial branch of Expertiza.

application.rb

- Added current_user and current_user_session methods to application.rb
- application.rb renamed to application_controller.rb

survey_response_controller.rb

- Commented out all AuthController actions in survey_response_controller.rb because the methods should simply redirect the user instead of logging the user out. This eliminates our need to recreate many of the methods that were being used with our new authentication model.

auth_controller.rb

- Commented out session.delete in logout method because the session parameters get set to nil anyway

assignment.rb

- Commented out require 'ftools' because there was a conflict with Rails 2.3.10, and it doesn't appear that ftools functions were used anyway.
- Removed :secret option from protect_from_forgery, since this has been deprecated.

assignment_controller.rb

- Commented out require 'ftools' because there was a conflict with Rails 2.3.10, and it doesn't appear that ftools functions were used anyway.

export_file_controller.rb

- Changed the FasterCSV to CSV because FasterCSV became the mainline CSV release as of Ruby 1.9.

boot.rb

- Several updates to boot.rb were made automatically by the Rails upgrade

Migrations made

- Created new migration #20101128192024 to add persistence_token column to User table

scripts/

- Fixed paths of the require statement in all scripts
- Removed deprecated scripts

tests/

- Fixed paths of the require statement in all test scripts

4 - Actual Implementation - Issues Faced

Rails Upgrade

At the time of this writing, Rails 2.3.10 is the latest version of Rails 2.3, so Expertiza has been upgrade

to Rails 2.3.10 rather than 2.3.8 as was indicated in the design plan. In light of the fact that other teams are working specifically on developing more comprehensive tests for Expertiza, the development of a test plan was abandoned in favor of examining documented deprecations and ad-hoc manual tests.

In the initial testing, Ruby 1.8.7 was installed on the development machine and selected as the Ruby library for the Expertiza project. Ruby was upgraded to 1.9.2 due to some incompatibilities found when using Ruby 1.8.7 and Rails 2.3.10. With Ruby 1.8.7, Rails would incorrectly assume that certain classes that are part of the Rails standard libraries were not initialized. Upgrading to Rails 1.9.2 eliminated the incompatibilities.

The upgrade of Ruby to 1.9.2 in itself introduced further issues that needed to be addressed. The FasterCSV library that was used in the file export controller is no longer supported after Ruby 1.9. Fortunately, the FasterCSV library was rolled into Ruby as the main CSV library so all of the methods that were used were still available.

The assignment.rb and assignment_controller.rb files had ftools as a required library, but ftools is incompatible with Rails 2.3.10. Although the assignment model does perform file and directory operations in several methods, none of the functions require ftools. Therefore, the require statement was simply commented out to remove the incompatibility.

Some Rails features that existed in Rails 2.2 have been deprecated in Rails 2.3. A list of these can be found in the release notes.

- http://guides.rubyonrails.org/2_3_release_notes.html

Expertiza was examined for deprecated code, and it has been replaced where it was found.

The directory script/process and file script/performance/request were deprecated in Rails 2.3. The file script/breakpointer was deprecated in Rails 2.0, but was still in Expertiza. These scripts have all been deleted from Expertiza.

The :secret option of protect_from_forgery has been deprecated in Rails 2.3. It has been removed from app/controllers/application_controller.rb:6.

In addition, ActiveRecord::Errors.default_error_messages was deprecated in Rails 2.2. In test/unit/user_test.rb, it has been replaced with I18n.translate('activerecord.errors.messages'). This deprecation is documented in the Rails 2.2 release notes.

- http://guides.rubyonrails.org/2_2_release_notes.htm

Ruby 1.9.2 does not implictly include "." in path names. In the script and test directories, virtually every script has a require that begins with "/../" which is expected to evaluate to "../../" but in Ruby 1.9.2 evaluates to "../" instead. In the script directory this was corrected by replacing "require File.dirname(__FILE__) + '/../config/boot'" with "require File.expand_path('../../config/boot',

__FILE__)" in all of the scripts. In the test directory this was corrected by replacing "require File.dirname(__FILE__) + '/../test_helper'" with "require './' + File.dirname(__FILE__) + '/../test_helper'".

Replacing authentication with Authlogic

Authlogic uses a similar model for authentication as that used by Goldberg. The User model and controller needed were very similar to what already existed. The user model had to be modified to replace some of the existing password encryption work and let Authlogic handle those features. There were also a few simple updates required with the user controller to allow Authlogic to handle comparing the password and password check when users are created or updated.

The user login functionality proved much more difficult to decouple from the Goldberg implementation and tie into the Authlogic template. Authlogic is very flexible and can be adapted to many different styles, but the ideal method is to create a user_sessions controller and use standard new, create, destroy methods like would be used with many other controllers. The Goldberg auth_controller is setup a bit differently with distinct login and logout functions. The Goldberg modules were heavily tied in with the auth_controller and proved timely to remove. Instead of forcing changes to many of the other Goldberg modules, the auth_controller methods were simply updated so that the interface wasn't changed. This proved the best way to keep the other Goldberg modules working.

By default, Authlogic re-encrypts the password 10 times to produce a SHA1 hash. Previous Expertiza password hashes were produced with a single pass through SHA1, so Authlogic has been configured to do the same so that existing passwords will remain valid.

Replacing Goldberg Authorization

About acl9 :

In Expertiza after an user is authenticated to enter the system the user is allowed access to certain areas of expertiza based on the role assigned. Authorization is the another area where goldberg plays major role. In expertiza an user can have the following roles:

1. Student
2. Instructor
3. Administrator
4. Super-Administrator
5. Teaching Assistant

Based on the roles assigned user can navigate the system. acl9 is a gem plugin which provides authorization related functions. Authorization provided by acl9 through its two subsytems. Users can access an object based on the roles assigned to them. Such functionality is provided by role-based subsystem. Access based subsystem provides access to users for particular actions, controllers based on the roles assigned.

Installation:

Installation done by two ways:

1. Under config/environment.rb add

config.gem "acl9", :source => "http://gemcutter.org", :lib => "acl9". Then do rake gems:install.

2. Install acl9 as a plugin by running the script script/plugin install git://github.com/be9/acl9.git

Goldberg's role in Authorization:

Goldberg maintains some tables in the database to promote authorization. Some of those tables include

roles (id,name,parent_id,description,cache)

users(id,name,role_id)

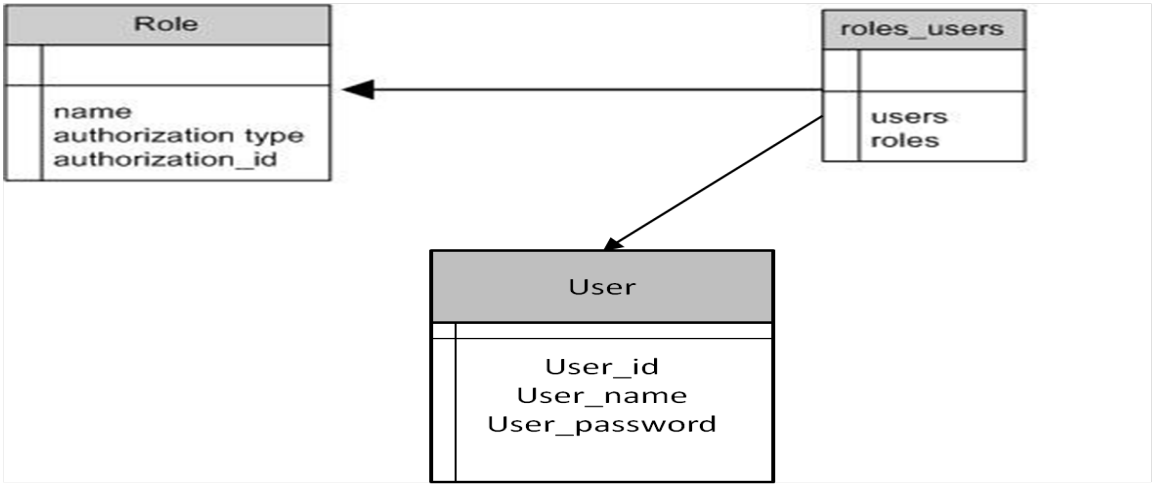
roles_permissions(id,role_id,permission_id)

permissions(id,name)

Cache in roles table maintains all the actions,controllers that are accessed by a role in the system. Cache contains information of menu , actions , controllers , pages.The whole cache for the particular role related with the currently logged in user is stored in session. Upon each request from the client the request is routed to vendor/plugins/goldberg_filters/lib/goldberg_filters.rb where the ActionController::Base.class_eval is invoked which has the method goldberg_security_filter to check if the requested page is present in the cache,is the session timed out ? etc . If present the corresponding action is taken by invoking the action in the controller. Thus Goldberg plays the role of authorizing the user for the requested resources.

Achieving authorization with acl9 :

acl9 mainly works around three dbs Roles table, User table , users_roles table- joining table of user and roles In expertiza we have Roles, User table and we dont need users_roles table as the roleid and userid are already joined in Users table. But we can create users_roles table to avoid further confusion. users_roles tables has columns referring to userid of users table and roleid of roles table. Databased design given below:



To give user access to particular objects there are three rules to be added.
1. In role.rb add acts_as_authorization_role under class Role
2.In user.rb add acts_as_authorization_subject :association_name => :roles under class User.
3.Add acts_as_authorization_object under each object which has to be accessed based on role.

After the initial rules addition, we can assign objects by the function
subject.has_role!(role, object = nil)

Example : If assignment has to be accessed by a user whose role is student , it is created by
user.has_role(:student,assignment).

The controller actions can also be accessed based upon the rules we defined. For example,
In Assignment Controller , to allow a admin to access the assignment object and
create an assigment , the following rule specified in Assignment Controller
allow :admin , :of => assignment , :to => [create]

Thus each user and role and corresponding object accessed can be assigned.

Constraints faced while trying to implement the feature :

1. Requested url is directed by goldberg routes to golberg filter module and it handles the authorization. If we bypass this and make it directly respond to requests this could be done. But the session time out feature is handled by goldberg. Faced issues regarding the menu selected and session.
2. And also the objects , controllers assigned to each role is huge which consumes more time.

5 - Future Goals

The Expertiza project is tightly dependent on Goldberg. Goldberg provides authorization , navigation strucutre and content pages. As we have discussed previously authorization can be replaced by acl9 library. And we did a fair amount of analysis of acl9 and while implementing we faced some constraints and also goldberg authorization is so much bundled with the navigation and content management structures. Hence implementing acl9 might be a good future goal, but replacement of the Goldberg content management is a key requirement before the authorization can be effectively replaced.

Also, the authentication should be updated to removed the authentication controller and adapt to a template more consistent with the Authlogic standard. Making this update will make upgrades of Authlogic easier to roll into Expertiza since the code will be less reliant on the lower level interface of Authlogic as it is with the resultant implementation from this project.

6 - Conclusion

Expertiza has been successfully upgraded from Rails 2.2.2 to Rails 2.3.10. In the process it was also necessary to upgrade from Ruby 1.8.7 to Ruby 1.9.2. As predicted, version incompatibilities introduced a number of bugs, and efforts were taken to track down and resolve as many of these as possible. Launching an Expertiza server and logging in with a web browser indicates that basic functionality like browsing questionnaires and adding users is working.

Responsibility for authentication in Expertiza has been migrated from Goldberg to Authlogic. Authlogic is a popular authentication library for Rails. Since it is more widely adopted than Goldberg, it is also better maintained and the project is more likely to remain active. Care was taken to ensure that existing password hashes will remain valid.

Migrating responsibility for authorization in Expertiza from Golberg to acl9 was investigated. However, it was determined that too much of the other functionality that Goldberg provides, like content management, is closely tied with Goldberg's authorization strategy. So it was deemed impractical to proceed given the short timeline of this project. A lot of useful information on how authorization responsibilities in Expertiza could be migrated away from Goldberg and to a library like acl9 was learned in the process and is presented in this report.

To test the upgraded Expertiza, install Ruby 1.9.2 and make it the default version of Ruby. Install the Rails 2.3.10 gem. Expertiza will require additional gems, like RedCloth and rgl. In theory, "rake gems:install" should automatically identify and install all the necessary gems to run Expertiza, but your mileage may vary. Create a database if necessary with a "rake db:create". Do a "rake db:migrate". Run a server either through Eclipse or from the command line with "script/server". If you created a new database, the login name and password are "admin" and "admin".