AgentTeamwork-Lite Web Application

Dom Roselli

1. OVERVIEW

This AgentTeamwork-Lite Web application allows users to create and run agent teamwork lite jobs, view job results, and provide web service API's for mobile applications to interact with application.

2. GOALS

Like any software project, the goals have changed and evolved over the course of the quarter. At the beginning, the goals were minimal. As the team began to ask questions and give feedback on each of the iterations, we were able to further refine more meaningful goals for a much richer web application.

2.1 Use a web application development framework that affords meeting the project requirements.
2.2 Store user and job information in a database for maintaining user and job state.
2.3 Users can run jobs and view job results.
2.4 Allow external accessibility to website.
2.5 Provide administrative affordance for web application service.
2.6 Provide RESTful API for mobile application.
2.7 Create an aesthetically pleasing user interface.

3. DEFINITIONS

These are terms that are used throughout the document. Also refer to the references section for background information about these terms and concepts.

3.1 The system - The web application to be developed.
3.2 Job – An AgentTeamwork-Lite job.
3.3 Agent Teamwork Lite - dispatches and moves a user job to the most appropriate computing resources.
3.4 Sensor Map
3.5 Connector – Set of tools designed to help programs retrieve remote data.
3.6 MASS Library
3.7 Ruby on Rails – Web application development frame work based on the Ruby language.
3.8 RVM (Ruby Version Manager) – used to manage Ruby versions and Ruby gems.
3.9 Ruby Gem – standard package manager for distributing Ruby code and libraries.
3.10 Git – Code repository application typically used with Rails projects.

4. REQUIREMENTS

The requirements as of the latest iteration of the ATL web application. Much of the underlying framework of the application is complete. The remaining work involves the ability to run the java code that makes up a job and allowing a user to upload a custom java class.

Requirement	Completed

4.1 Users must be able to create and edit accounts on system.	**X**
4.2 Users must be able to securely log in to, and log out of, the system.	**X**
4.3 User passwords must be stored securely.	**X**
4.4 Users must be able to access the web site external to the UW network.	**X**
4.5 Users must be able to create/edit/delete their own jobs.	**X**
4.6 Users must be able to run their own jobs.	
4.7 Users must be able to view their job's results.	
4.8 Users must be able to view their job's history.	
4.9 Users must be able to upload a jar file that contains the program to execute.	
4.10 Users must only interact with their jobs, unless in admin role.	**X**
4.11 Users must be able to view job progress on each machine in cloud.	
4.12 Users must be able to change their password, email address, etc…	**X**
4.13 Admins should be given affordance to stop and start agents on machines in cloud.	
4.14 Admins must be given affordance to view all items in web application.	**X**
4.15 System must securely store lab account information to run jobs.	
4.16 Web service API's must be made available to mobile application developers.	**X**
4.17 System must be able to use existing java classes.	**X**

5. DETAILS

Web Site URL
http://cssvm03.uwb.edu:8080/

Web Application Directory
/opt/rails_apps/atlweb

Code Repository
https://github.com/dr00/atlweb

Starting the Web Application
From the terminal:

```
1. sudo /usr/local/jruby-1.6.3/bin/jruby -S trinidad --config trinidad.yml
2. sudo /etc/init.d/trinidad start
```

Stopping the Web Application
From the terminal:

```
sudo /etc/init.d/trinidad stop
```

Technologies Used

- **5.1** JRuby v.1.63
- **5.2** Ruby on Rails v.3.0
- **5.3** RVM (Ruby Version Manager)
- **5.4** Various ruby gems (see the Gemfile in the application root directory)

Intention

The decision to use JRuby Ruby on Rails was four-fold:

- The ability to generate a quick prototype the web application.
- The necessity to call existing java classes from the web application.
- The integration of data and web application into a single framework.
- RESTful web service API's (for the mobile application) are an inherent part of the framework.
- Desire to challenge the team learning a new, cutting edge technology.

Current State

Currently, the web application performs just more than half of the required work. Much of the framework for a rich web application has been implemented, but the core functionality of running jobs, viewing job status, viewing job results, etc… is not. The remaining work is indicated by the incomplete requirements in section 4. However, the foundation is built and ready.

Lessons Learned

It was an ambitious undertaking to learn the number of new technologies this project presented. I had to learn the basics of the AgentTeamwork-Lite application, general web application development, Ruby on Rails, Ruby and JRuby, setting up the Rails framework, implementing an authentication scheme in a web app, using RVM, installing a Rails application to run as a service on Linux, implementing RESTful web service API's for the mobile phone app, and many more details. Much of this I learned on my own by doing research, reading books, taking web based training, because our team did not have the expertise at the time.

Much of the time initially was spent learning getting up to speed on the technologies, setting up the development and production environment, and building the foundation for the web app (the data object model, the views, authentication, etc…). I thought it would be easier to implement the authentication and roles piece upfront, instead of trying to shim it in later.

In the end, I probably would not have chosen Rails as the development framework given the amount of time we had to complete the project. I still think it is the right choice for the AgentTeamwork-Lite web application, because it is a flexible, robust framework and permits fast iterations of ideas. But, it's probably not the right choice for a novice web developer with only two months to develop.

I am still pleased to have learned Rails and worked on the AgentTeamwork-Lite project. I learned a lot and gained some new skills that are relevant to today's marketplace.

REFERENCES

- http://depts.washington.edu/dslab/SensorCloud/index.html
- http://depts.washington.edu/dslab/SensorCloud/report/connectorManual.pdf
- /net/metis/home3/dslab/doc/nsf2010/sensor_cloud.pdf
- http://ruby.railstutorial.org/
- http://railsforzombies.org/

- http://progit.org/book/