A Case Study in Web 2.0 Application Development

Paul R. Marganian

Software Engineer – National Radio Astronomy Observatory, Green Bank
Authors:

Paul Marganian
Mark Clark
Amy Shelton
Mike McCarty
Eric Sessoms
Introduction

- Discussing our recent experiences creating a Web 2.0 application: the new Dynamic Scheduling System (DSS) for the Green Bank Telescope (GBT).
- Addresses ADASS focus topic: Commonly available development environments & tools
- What is Web 2.0?
  - Wikipedia: “is commonly associated with web development and web design that facilitates interactive information sharing, interoperability, user-centered design and collaboration on the World Wide Web.”
  - Desktop quality applications for the Web Browser.
Overview

Background

Beta Test

Release
Vegetable ONLY
Background: The GBT

- The Robert C. Byrd Green Bank Telescope is the world's largest fully steerable radio telescope.
- The Green Bank Telescope (GBT) is located at the National Radio Astronomy Observatory's site in Green Bank, WV.
- 100 m diameter
- Can observe over a larger range of frequencies than other comparable centimeter/millimeter single-dish telescopes.
- Is located in a continental, mid-latitude region where weather is dominated by water vapor and small scale effects.
Background – The Problem: Dynamic Scheduling

- The primary goal of the DSS is to improve the efficiency of GBT observations by matching the observing schedule to current weather conditions, while allowing each observer to retain interactive control of the telescope.
- Example: use the GBT for high-frequency science for clear, calm weather, and schedule low-frequency science when the weather is otherwise.
- Exceptional Scheduling Paradigm: not scheduling scripts, but scheduling observers.
Background – Problem: Dynamic Scheduling

• How it works:
  – Each day the DSS will examine the weather forecast, equipment availability, observer availability, and other factors, and set an observing schedule for the 24-hour period beginning the next day.
  – Observers will therefore get 24-48 hours notice before their project will observe.
  – Additionally, observers will have the opportunity to pause their observing program, set blackout dates indicating when they are unavailable for observing and back out of current observations if they find the observing conditions are not suitable to their science goals.

• ADASS talk on this last year, by site directory, Dr. Karen O’Neil; along with several posters.
Background – The Problem: Dynamic Scheduling

- Since we’re scheduling observers, the observers have to have a high level of interaction with the system.
- Hundreds of geographically dispersed users.
Background – The Solution

- What is required to dynamically schedule the GBT is not just complex scheduling algorithms (though we certainly need that).
- We also require complex interactions with all the users.
- **Solution:** A Web Application!
- And not just a traditional series of web pages, but a Web 2.0 Application.
Background – The Green Bank Software Development Division

• We have to implement this solution.
• Who’s ‘we’? ~2.5 FTEs over the project life-time.
• Until quite recently, we have focused on three main areas of activity, utilizing three languages:
  – Monitor & Control – C/C++
  – Observation Management – Python
  – Data Analysis – IDL
• What type of skill-set and knowledge-base are required to make a web application?
Background – Web Applications, Grossly Simplified

**Concepts:** Database, Server, HTTP, Authentication, HTML, CSS, JavaScript, DOM, AJAX, XML, JSON
Overview

Background

Beta Test

Release
Frameworks

• As discussed previously, many diverse components involved in a web application
• A framework pulls a lot of these components together for you with a number of benefits:
  – Manages many crucial activities by default
  – Enables a web application to get up off the ground quickly
  – Allows you to work in the language of your choice
• Many frameworks available, several in the language of our choice, Python
• For the Beta Test (2008), the choice came down to two: TurboGears & Django
TurboGears – a Python web framework

• All in python.
• TurboGears follows the MVC design pattern: isolates business logic from input and presentation.
  – Model: the domain-specific representation of the data on which the application operates.
  – View: Renders the model into a form suitable for interaction
  – Controller: Receives input and initiates a response by making calls on model objects.
Model – Object Relational Mapper (ORM)

• Wikipedia: “in computer software is a programming technique for converting data between incompatible type systems in relational databases and object-oriented programming languages”
• Can be a good compromise if data persists in a DB, but you’d like to work with objects.
• We used SqlAlchemy as our ORM
• Worked very well for our purposes:
  – DSS DB is fairly complex, but mapped to objects fine
  – Performance not an issue; DSS DB size is moderate
Controller - CherryPy

- Wraps the HTTP Protocol
- CherryPy maps URLs from the browser to python methods on the server
- Each method can then query the model, then render a template (a page refresh to the user), or redirect to another URL.
- Example: url = /addValues?a=1&b=2
- Calls method addValues, with keyword-value pairs a = 1, b = 2
View – Genshi Templates

- A templating language that allows you to embed python in html.
- Excellent way of separating your display style from your content.
- Example: building a table dynamically.
TurboGears – other features

• With all the previously mentioned tools, you can get a functioning web application up and running quickly.
• More tools for making working with forms (validation, etc.) very easy.
• But apart from a few special widgets (e.g. date picker), web pages using just these components are mostly form based (very Web 1.0).
• TurboGears allows you to enhance your web pages with CSS and JavaScript, and supports AJAX.
• This is the basic approach we took to the Beta Test’s Observers’ Interface
**DSS Beta Test Observers Interface**

Created with:
- TurboGears
- CSS package: Blueprint
- a sprinkle of JavaScript.
There some other technologies that bear mentioning in this interface:

- iCalendar format – wikipedia: “a computer file format which allows internet users to send meeting requests and tasks to other internet users, via email, or sharing files with an .ics extension”
DSS Beta Test Observers Interface

• Other tools used in the observers interface:
  – RSS – wikipedia: “Rich Site Summary” is a family of web feed formats used to publish frequently updated works—such as blog entries, news headlines, audio, and video—in a standardized format.”
  – Chat software – we incorporated third party chat software so that observers can get live help from GBT staff.
• Although a more sophisticated interface for observers was desired, these simpler web pages worked well.
• However, for the scheduler’s interface, some of the requirements for the interface required us to enhance what the browser was doing.
DSS Beta Test Scheduler Interface

• Basically, we had to provide a graphical calendar that supported the graphical manipulation of observing periods on the telescope, with all the feedback necessary so that the scheduler can maximize the observing efficiency of the GBT.
• So, we had to move to JavaScript., and make use of AJAX. We took advantage of a JavaScript library, JQuery.
• JQuery provides:
  – DOM element selections
  – DOM traversal and modification
  – Events
  – CSS manipulation
  – Effects and animation
DSS Beta Test Schedulers Interface

Features:

- Drag and drop calendar
- Feedback
- Color coded efficiencies
- Easy access to project information

Only supported Firefox 2.0
DSS Beta Science Algorithms

• Without these there’d be no need for the Web Application.
• Implemented in Python.
• Interfacing them with the web application was trivial: simply imported and called from the CherryPy controller methods.
• Example:
  – Scheduler hits ‘Schedule’ in the interface,
  – which gets mapped to a ‘schedule’ method on the server
  – Calls ‘schedule’ algorithm in the science python modules
Beta Test: Lessons Learned

- Feedback from users from helpdesk and survey:
  - Observers interface was actually as sophisticated enough for users.
  - Scheduler’s interface was *not* sophisticated enough.
- Experience of developers:
  - TurboGears framework worked well for us.
  - Working with JavaScript was pain-full.
  - Science Algorithm code much too slow, and error prone.
- Used lessons learned to prepare for the production release (Oct. 1, 2009)
Overview

Background

Beta Test

Release
Release: Django Framework

- TurboGears served us well in the Beta Test, but in the mean time, the Django framework really took off.
- Django framework is extremely similar to TurboGears. Switching over was very simple.
- Instead of bringing together different 3rd party libraries like TurboGears, Django uses its own ORM, templates, etc.
- On the whole offers more features than TurboGears.
- Larger user base meant more ‘freebies’. Some of these turned out to save us a great deal of time.
We were able to quickly develop an observer’s interface using mostly features from Django.

Project Calendar using the FullCalendar plugin for jQuery.
Release: integrating with other NRAO systems

- share a single database of user contact info (avoiding the Beta Test nightmare)
- other NRAO systems provide URL query services that serve up XML encoded data.
- Ex: user contact information
Release: Authentication with CAS

- NRAO is using an open-source single-sign-on service: Central Authentication Service (CAS)
- This way, users can log in to one NRAO service (ex: Proposal Submission Tool), and seamlessly move to another serve (ex: DSS) without logging in again.
- Django allows you to specify the authentication backend you want to use.
Release: Django Admin Tool

- Django comes with a built-in Admin Tool, which is basically a sophisticated view into your database (like phpMyAdmin).
- Is extensible to handle custom actions.
- Result: functionality that may not be covered in your own interface is often covered by the Admin Tool.
Release: Schedulers Interface

• Recall that the Beta Test’s Schedulers Interface was not sophisticated enough.
• But creating a more sophisticated interface in the browser would mean more JavaScript, and we’d had enough of that in the Beta Test (even supporting just one browser!).
• Decided to be bold and try out a completely new technology: Google Web Toolkit
Google Web Toolkit

- **Google Web Toolkit** (GWT) is an open source set of tools that allows web developers to create and maintain complex JavaScript front-end applications in Java.
- Using GWT, developers can rapidly develop and debug AJAX applications in the Java language using the Java development tools of their choice.
- When the application is deployed, the GWT cross-compiler translates the Java application to standalone JavaScript files that are optionally obfuscated and deeply optimized.
- Creating an advanced UI for the browser feels like developing a typical Java Desktop interface.
**Google Web Toolkit**

- GWT Java-to-JavaScript Compiler – code in Java!
- GWT Hosted Web Browser – run app as Java in the JVM; great for debugging
- JRE emulation library – common features of java.lang and util for JavaScript
- GWT Web UI class library – a set of custom widgets
- A set of custom interfaces and classes for creating widgets.
- GWT handles all cross-browser issues for the developer.
- JUnit integration
- Lot’s of 3rd party libraries; including one for even fancier widgets: Ext GWT
GWT and Eclipse

We found it easiest to develop GWT with the GWT plugin for Eclipse:
Ext GWT

• Provides a library of advanced widgets for GWT
• See the demo at: http://www.extjs.com/explorer
• We took advantage of the Grid libraries
• Created using GWT and the Ext GWT Grid library.

• Emulates a spreadsheet for project data.
Release: Scheduler’s Tools with GWT

- Created using GWT and the gwt-cal package
- Emulates a google-style calendar
Scheduler’s Tools with GWT: conclusions

- Team was inexperienced using Java & Eclipse. A learning curve when we really didn’t have time for it.
- Using GWT to emulate some of the more basic Desktop widgets (panels, combo-boxes, etc.) was relatively straight-forward.
- Implementing the more advanced widgets, such as the grids (Project Explorer) with Ext GWT were more difficult.
- So, is the problem Ext GWT, GWT, or us?
DSS Release: Science Algorithms

- Recall that in the Beta Test, our Science Algorithms were coded in Python.
- This made it trivial to integrate into the application, but was slow and error prone.
- For the release, we again decided to make the bold move of rewriting the algorithms in a completely different language: Haskell
Science Algorithms with Haskell

- **Haskell** is a standardized, general-purpose purely functional programming language, with non-strict semantics and strong static typing.
- **Purely functional** is a term in computing used to describe algorithms, data structures or programming languages that exclude destructive modifications (updates). According to this restriction, variables are used in a mathematical sense, with identifiers referring to immutable, persistent values.
- A different programming paradigm than Python, C/C++.
- Referential Transparency: put the same thing in, always get the same thing out. It’s State-less.
- No ‘for’ loops: everything is recursion and pattern matching
Science Algorithms in Haskell

• Pros:
  – Reduced code size by a factor of 2
  – Increased performance by a factor of 10 !!!
  – Mathematical nature of Haskell more closely resembles science.
  – Code easier to understand by team scientists

• Cons:
  – Steep learning curve: a completely different programming paradigm for us
  – Not as many practical third party libraries as a language like Python (though enough)
  – Will someone be around to support this in 10 years?
  – Doesn’t interface easily with other languages (just C right now)
Putting it all together:

- When you use what you think is the tool for each part of your problem, how do you get them to all work together?
- Proxy server – let the browser think there’s just one server
Conclusions

- Inexperienced team delivered two web applications (Beta, Release) on time and on budget, using new tools and environments.
- We would strongly recommend using some kind of framework.
- There’s lots of tools and resources to help build a better user experience (iCal, RSS, etc.).
- GWT has a lot of promise. Would strongly recommend it if you are already familiar with Java.
- Haskell is a great programming language, but is it worth the risk?