2. What We Started With

We needed a framework to help us develop our application. Our main constraint was that we wanted one that was Python based. We have been developing in Python for the last few years, mostly for our observation management tools. Python is a dynamic object-oriented programming language that offers an extensive source of libraries, especially for Astronomy. We ended up looking at a number of the more popular Python frameworks, including

- Django, TurboGears (TG), and Zope. In the spring of 2007, we completed a project using TurboGears, and decided to start developing the DSS with this framework as well.

TurboGears uses the Model-View-Controller (MVC) architectural pattern. MVC decouples data access and business logic (Model) from data presentation and user interaction (View) through the controller. TurboGears comes with default packages for implementing each part of the MVC pattern (see Figure 1).

For the Beta Test we used some alternative packages and additional JavaScript to enhance user interactivity (see Figure 2).

3. What We Added

In addition to the above-mentioned packages, we also enhanced our user experience through the use of JavaScript, as well as the application of industry standards, such as RSS and iCalendar.

3.1 JavaScript, RSS, iCalendar

For example, we created a page for the GBT Scheduler in which we were able to drag, drop, slide, and resize telescope scheduling elements around a calendar (see Figure 3). This was accomplished through a collection of JavaScript objects, which communicated to the server via JSON calls to TG’s controller methods. Originally this work was done using the JavaScript libraries Prototype 1.6.0 and script.aculo.us 1.8.0, but we then also included JQuery 1.0.

One of the requirements of the DSS is that observers are informed about their project(s) in a clear and timely fashion, without overwhelming them with information (e.g., swamping their inboxes with emails). We utilized RSS feeds to notify observers of changes in their project data or their project’s observing schedule. RSS is a subscription service, so users are free to subscribe to those feeds that they are interested in. For those users who were more comfortable with traditional email, we recommended the use of third party solutions, such as RSSReaderLive and RSS FuJD. In addition, schedules of the GBT, as well as schedules regarding specific projects and their observer were provided using the iCalendar standard, which is understood by a variety of calendar applications. (see Figure 4) We also embedded a Google Calendar of the GBT schedule. For online help, we took advantage of Instant Messaging technologies (libragnlp), so that users could ‘chat’ with GBT support staff.

We also leveraged the CSS framework Blueprint 0.7.1 for much of our CSS needs, as in this page used by schedulers and observers to manage project information. (see Figure 5)

3.2 Other Tools

- Python 2.5.2
- Authentication – We created our own home-grown authentication system.
- Provides templates for generating web pages using XHTML and embedded python
- MySQL database server
- decrs 2.0.2 – For revision control
- pyfigs 1.3 – For reading GBT FITS files
- hsshell (gch 6.8.3) – For specific server-side algorithms related to the DSS
- sqlite 3.5.9 – For use in conjunction with our unit test framework
- nosetests 0.10 – For unit testing

3.3 Deployment

- Apache Server
- Red Hat Enterprise 5

4. What we learned

4.1 Experience with TurboGears 1.0

The summer 2008 Beta Test was a chance for us to try out the TurboGears framework w/ our chosen components. Here is how we evaluated our experience w/ them:

- Pros
  - TG, as with many frameworks, helps you get a Web Application off the ground fairly quickly
  - SqlAlchemy mapped our classes to the database, making data persistence a simple matter
  - TG handles database connections for you

- Cons
  - TG does not handle multiple databases well
  - SqlAlchemy objects can slow down computation for heavy computations, separate non-SqAlchemy classes needed to be used
  - SqAlchemy does not handle well mapping tables residing in multiple databases
  - TG does not support RESTful URL’s; we had to find a workaround this ourselves

For the Beta Test, we failed to take full advantage of all of TurboGears’ features. For example, TG supports a wealth of widgets (fancy forms, date-time pickers, etc.) that we did not use, and also supports the ability to define alternative authentication backends (we customized our own solutions in both cases).

4.2 Other Lessons Learned

- The RSS forwarding services we recommended proved to be too slow. In the future we will probably need to customize a solution for those users who want to exclusively use email.
- The embedded Google Calendar sometimes took hours to update, which caused confusion and delay. Again, we may have to customize our own solution.