More dBs Software Design

David Cuddeback  dcuddeba@calpoly.edu
Jason Anderson  jander06@calpoly.edu

31 October 2008
1 Basic Architecture

More dBs will be based on Ruby on Rails. One of Ruby on Rails’ key principles is “convention over configuration.” The Rails’ convention imposes a database-driven model-view-controller architecture which follows the REST principles. As such, the basic architecture of More dBs follows the architecture of Ruby on Rails.

1.1 Models

The models closely resemble the database schema, where each model class corresponds to a table in the database. The models are responsible for maintaining the integrity of the data. Before data is committed to the database, the model is responsible for checking the validity of the data and doing whatever is necessary to safeguard the data such as using transactions when they are necessary.

1.1.1 Relationships

Many tables have relationships to other tables through foreign keys. These relationships can be one-to-one, one-to-many, or many-to-many. These relationships are available in the model classes through accessor methods. For example, a session can contain many packets but each packet belongs to one pass. This is a one-to-many relationship. In the SessionModel class, there is a method (packets) to access the packets. This method will return a list of PacketModel objects. Likewise, each PacketModel can return the SessionModel that it belongs to by calling its session method.

1.2 Controllers

Controllers handle user actions, manipulate the models, and select the correct view to display.

1.3 Views

Views are templates whose sole purpose is to display data. They do not contain any application logic, and they do not interact with the models.

1.4 REST

REST (Representational State Transfer) is a way of writing web services that focuses on resources. Following REST principles, all resources are addressable and commands are defined in simple terms. In a web service, resources are addressed by URL and the commands are HTTP commands: GET, POST, PUT, and DELETE. This functionality is built into Rails 2.
2 Public Area

The “public area” of the website encompasses everything that is accessible to the public. Generally, this means non-proprietary information regarding our satellites in orbit. Figure 1 shows the site map for the public area of the website.

![Site map of public area.](image)

2.1 Home Page and Top Level Navigation

With the exception of “satellite data” and “statistics,” every link in the top-level navigation links to a static web page that will be edited manually. The home page will also be a static web page. The content of these pages are not specified by this design document. Table 1 lists the URLs for these pages.
<table>
<thead>
<tr>
<th>Page</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home Page</td>
<td>/</td>
</tr>
<tr>
<td>Downloads</td>
<td>/downloads</td>
</tr>
<tr>
<td>About</td>
<td>/about</td>
</tr>
<tr>
<td>Get Involved</td>
<td>/about/get_involved</td>
</tr>
<tr>
<td>Contact</td>
<td>/contact</td>
</tr>
</tbody>
</table>

Table 1: URLs for the static top-level pages.

<table>
<thead>
<tr>
<th>Page</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satellite Data Entry Page</td>
<td>/satellites</td>
</tr>
<tr>
<td>Individual Satellite Data Pages</td>
<td>/satellites/:name</td>
</tr>
<tr>
<td>Satellite Pass Log</td>
<td>/satellites/:name/passes</td>
</tr>
<tr>
<td>Pass Data</td>
<td>/satellites/:name/passes/:id</td>
</tr>
<tr>
<td>Reports</td>
<td>/satellites/:name/reports</td>
</tr>
</tbody>
</table>
| Report Data                   | /satellites/:name/reports/???
| Packets                       | /satellites/:name/packets  |
| Packet Details                | /satellites/:name/packets/:id |
| Pass Schedule                 | /satellites/:name/pass_schedule |

Table 2: URLs for pages in the Satellite Data section.

## 2.2 Satellite Data

### 2.2.1 Entry Page

### 2.2.2 Passes

**RSS** The satellite passes page will contain an RSS feed, which will give subscribers access to the pass log via an RSS reader.

### 2.2.3 Packets

The design of the Packets section are TBD in a later stage.

### 2.2.4 Reports

The design of the Reports section are TBD in a later stage.

## 2.3 Statistics

The exact statistics displayed on the statistics page are TBD in a later stage.

**URL** The URL for the statistics page is /stats.
2.4 Account Management

<table>
<thead>
<tr>
<th>Page</th>
<th>URL</th>
<th>Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register</td>
<td>/users/register</td>
<td>users</td>
</tr>
<tr>
<td>Sign In</td>
<td>/users/signin</td>
<td>users</td>
</tr>
<tr>
<td>Sign Out</td>
<td>/users/signout</td>
<td>users</td>
</tr>
<tr>
<td>My Account</td>
<td>/users/profile</td>
<td>users</td>
</tr>
<tr>
<td>Users’ Profiles</td>
<td>/users/:id</td>
<td>users</td>
</tr>
</tbody>
</table>

Table 3: URLs for pages in the Account Management section.

3 Administrative Area

4 Database Schema

4.1 Satellite-Related Tables

4.1.1 satellite_models

The satellite_models table defines different satellite models, e.g., CP3, CP4, CP6. For each satellite model, there can be multiple units (see 4.1.3, satellite_units), but they share certain information, such as their link parameters (4.1.2, satellite_modes) and packet formats (4.3.3, packet_types).

4.1.2 satellite_modes

The satellite_modes table contains the information needed to communicate with the satellite through ham radio. Earth station software can use the information in this table to tune radios.

4.1.3 satellite_units

For any particular satellite model, there may be many different units of that particular model. The satellite_units table lists all the units of a particular model. Satellite units contain unit specific data such as current TLE and launch date (although these will not apply for lab units).

4.1.4 public_satellites View

public_satellites is a view which contains the satellite units that are to be included in the public area of the website. The view is a composite of the satellite_models and satellite_units tables, containing only units for which public is TRUE. The columns in the view are the columns in both tables except for name in satellite_units. This means
that when viewed by the public, satellites will have the name of the model, e.g., CP3, CP4, CP6, just as they are referred to publicly. The in-house names which are given to individual units (Flight, BenchSat, TestSat) are not available in this view. The view is based on the following SQL query:

```sql
SELECT t1.*, t2.id, t2.object_number, t2.status_notes, t2.launch_date
FROM satellite_models AS t1
LEFT JOIN satellite_units AS t2 ON (t2.model_id = t1.id);
```

4.2 User-Related Tables

4.2.1 users

The `users` table includes all information relating to a user of the system. This user can be a HAM, PolySat member, or other external entity with a earth station. It is important that the password hash and not the password is stored in this table for security reasons.
4.2.2 earthstations

The earthstations table contains all the data associated with a given earth station. It also contains all of the supported capabilities of the earth station such as which radio features are supported and which antennas are available.

4.3 Session-Related Tables

4.3.1 sessions

A session can be thought of as a single pass of one satellite. Therefore the sessions table includes all pass data such as the operator, the earth station used, and the satellite tracked. Since this data is described in other tables, foreign keys should be used to specify the session data.

4.3.2 packets

The packets table contains all the common data for each packet. This data includes the receive date/time and the Az/El/Range of when the packet as well as the original binary data received from the packet. This table will be a parent table for specific packet tables such as “cp3_sensor_snaps”.

4.3.3 packet_types

The packet_types table contains the information on the different packet types supported by the system. This table serves as a look up between the general packet data in the packet table and the specific table which has all the data parsed into individual fields.

4.3.4 comments
Figure 3: Session related tables.
Figure 4: User related tables.
A Design Tradeoffs

A.1 Downloads Page: Static vs. Dynamic

This tradeoff is about whether the downloads page should be a static HTML page that has to be edited manually or a dynamic page that loads its information from the database. Table 4 lists the benefits of a static downloads page versus one that loads its information from a database.

<table>
<thead>
<tr>
<th>Static</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Much less work upfront.</td>
<td>• Look-and-feel standards are enforced by templates.</td>
</tr>
<tr>
<td>• More flexible for edge cases.</td>
<td>• Scalable to many downloadable files.</td>
</tr>
<tr>
<td></td>
<td>• Easier to maintain links from other pages.</td>
</tr>
</tbody>
</table>

Table 4: Benefits of a static versus a dynamic downloads page.

**Decision**  The decision was made to have the downloads page be a static HTML page due to the amount of work necessary to make it dynamic. To alleviate the risk of a programmer breaking the look-and-feel or implicit standards, procedures will be put in place for making changes to the page. Also, the anticipated number of files and frequency of updates to the downloads page are small, so scalability is not much of a concern. The downloads page will likely consist of CP6 Data Decoder and maybe a few images taken by CP6.

In order to implement a dynamic downloads page, many database tables would be needed and the associations between those tables would have to be carefully considered for issues such as organization of downloads, operating system compatibility (for executable files), and file type.

A.2 Inclusion of Packet Submission Page