Evolution of pgAdmin: pgAdmin4

• Dave Page | 3 December 2016
About me

• PostgreSQL:
  – pgAdmin Project Lead
  – Core Team member
  – Web & Sysadmin Teams
  – Director (secretary) of PostgreSQL Europe
  – Chairman of PostgreSQL Community Association of Canada

• EDB:
  – Vice President & Chief Architect, Tools & Installers
pgAdmin III

- What is it?
- What was wrong with it?
- How do we fix it?
What is it?

- The leading Open Source GUI management tool for PostgreSQL
- Third generation (2002); replacing earlier tools written in VB (1998, 2001)
- Written in C++, using the wxWidgets cross-platform framework
- Ships standalone, and with the EDB ‘one-click’ PostgreSQL installers
- Supports PostgreSQL derivatives; EDB Postgres Advanced Server, and Greenplum Database
What is it?
What was wrong with it?

- C++ code dating back to 2002:
  - Code has grown messy over time
  - Very hard to find C++ developers
- Dated look and feel
- Desktop application in a web based world
- Dependent on troublesome third party libraries:

### Open tickets reported by you

<table>
<thead>
<tr>
<th>Ticket</th>
<th>Summary</th>
<th>Type</th>
<th>Component</th>
<th>Patch</th>
<th>Status</th>
<th>Created</th>
<th>Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>#4408</td>
<td>STC cannot handle long lines</td>
<td>defect</td>
<td>wXStyledText</td>
<td>False</td>
<td>confirmed</td>
<td>9 years</td>
<td>8 years</td>
</tr>
<tr>
<td>#4396</td>
<td>wxSTC AutoComplete menu broken with native wxListCtrl</td>
<td>defect</td>
<td>old wxOSX/Carbon port</td>
<td>False</td>
<td>confirmed</td>
<td>9 years</td>
<td>8 years</td>
</tr>
<tr>
<td>#4313</td>
<td>Numeric keypad doesn’t work with wxSTC</td>
<td>defect</td>
<td>old wxOSX/Carbon port</td>
<td>False</td>
<td>confirmed</td>
<td>9 years</td>
<td>22 months</td>
</tr>
<tr>
<td>#4251</td>
<td>Keyboard navigation impossible</td>
<td>defect</td>
<td>wxAui</td>
<td>False</td>
<td>confirmed</td>
<td>9 years</td>
<td>8 years</td>
</tr>
</tbody>
</table>
How do we fix it?

- Change technology stack:
  - Easier to find developers
  - Could be web based
  - Much easier to change the look and feel
  - No complex and buggy cross-platform libraries

This means a complete rewrite!
pgAdmin 4

- Basic expectations
- Technology choice
- Non-functional requirements
- Functional requirements
Basic expectations

• The application should be able to run in both web and desktop modes
• The technology stack should be based on language(s) popular in the PostgreSQL community
• The core functionality of pgAdmin III should be re-implemented
• Features of pgAdmin III that are known to be used very little, if at all, should be excluded
Technology choice

• Python
  – Mature language
  – Used extensively in the postgresql.org infrastructure
  – Works well with PostgreSQL

• Javascript/jQuery/Bootstrap
  – Tried and tested technologies
  – Lots of developers with experience

• Flask micro-framework
  – A mature, yet lightweight Python web application framework
  – Very similar to the core of Django, which is well known in the community
Non-functional requirements (1)

• Framework:
  – The application should provide a framework for extensibility:
    – TREEVIEW nodes are all plugins
    – Individual tools are plugins
    – Database drivers are plugins, to allow support for PostgreSQL derivatives

• Must be deployable:
  – In a desktop runtime, in single-user mode
  – On a web server using WSGI, in multi-user mode
Non-functional requirements (2)

• Packaging:
  – Windows/Mac packages for desktop deployment
  – Linux RPMs/DEBs, for desktop or server deployment
  – PIP wheel for server deployment

• Python compatibility:
  – All common versions of Python through to the latest
    – 2.7.x
    – 3.0.x – 3.5.x
Non-functional requirements (3)

• **Key needs:**
  – Use of one or more servers simultaneously
  – Support for all PostgreSQL datatypes
  – Support for UTF-8
  – i18n

• **Speed:**
  – Fast response, so it feels like a desktop application
  – No full page reloads; AJAX everywhere
Functional requirements (1)

• Support for all common database object types:
  − Servers, database, tablespaces, roles, extensions, schemas, tables, indexes, constraints, triggers, functions etc.

• Query Tool, with integrated data editing:
  − Merge the existing Query Tool and Edit Grid into a single tool

• Dashboards for simple realtime monitoring

• Procedural Language debugger

• Grant Wizard

• Backup and Restore
Functional requirements (2)

• Maintenance (VACUUM, ANALYZE etc)
• Utilities (Pause/resume WAL replay, add named restore point etc.)
• Online help for use of the application
• Links from object dialogues to relevant PostgreSQL documentation
• User manager with self-service password management for use in multi-user mode
pgAdmin4 – The Project

• The team
• Constraints
• Project management
• QA
• Release plan
• Stats
The team

• Almost all EDB staff, until we got the basics right (because I can boss them around 😊):
  - 3 developers/committers
  - 6 developers
  - 1 designer
  - 1 project manager
  - 4 quality assurance
  - 2 technical writers
  - 3 packagers

• Community
  - 15 contributors
  - ~75(?) bug reporters
Constraints

- Most of the work was being done by the Postgres Enterprise Manager development and QA teams:
  - Needs to start after PEM 6.0 is released
  - Needs to finish in time to start on PEM 7.0
  - Need to allow time for PEM update releases
  - Need to allow time for support escalations

- Code needed to be ready in time for inclusion in the EDB ‘one click’ PostgreSQL 9.6 installers
Project management

• Development schedule managed using traditional methods for ease of scheduling
  – projectmanager.com – multi-project/team aware online equivalent to Microsoft Project or OmniPlan

• Patch management for committers handled through a Kanban chart
  – Kanbanchi – online Kanban charts, integrated with Google Apps

• Post-development QA and bug tracking in the community
  – redmine.postgresql.org
Kanbanchi
Quality assurance

• Manual test suite developed by EDB QA Team:
  – Managed using Testrail QA

• Automated test framework by the QA team:
  – Built on the Python Unittest2 module

• Automated UI testing
  – Not yet underway
  – Will likely utilise Selenium

• Community testing
  – Ad-hoc; no guarantee of participants
  – But… have had mostly great feedback so far 😊
  – As well as some negative feedback 😞
Testrail QA
Regression tests

Run 148 tests in 31.499s
OK

Test Result Summary

Regression - FGAS 9.5:
145 tests passed
3 tests failed
8 tests skipped

Regression - FG 9.5:
133 tests passed
3 tests failed
12 tests skipped:
  PackageAddTestCase
  PackageDeleteTestCase
  PackageGetTestCase
  PackagePutTestCase
  SynonymAddTestCase
  SynonymDeleteTestCase
  SynonymGetTestCase
  SynonymPutTestCase
  ResourceGroupsAddTestCase
  ResourceGroupsDeleteTestCase
  ResourceGroupsPutTestCase
  ResourceGroupsGetTestCase

Regression - FG 9.4:
133 tests passed
3 tests failed
12 tests skipped:
  PackageAddTestCase
  PackageDeleteTestCase
  PackageGetTestCase
  PackagePutTestCase
  SynonymAddTestCase
  SynonymDeleteTestCase
  SynonymGetTestCase
  SynonymPutTestCase
  ResourceGroupsAddTestCase
  ResourceGroupsDeleteTestCase
  ResourceGroupsPutTestCase
  ResourceGroupsGetTestCase

Please check output in file: /Users/cpage/git/pgadmin4/web/regression/regression.log

(cpgadmin4)@piranha:regression cpage$
Release timeline

• Beta 1 released on 7th June
• Beta 2 released on 24th June (PostgreSQL 9.6 Beta 2)
• Beta 3 released on 21st July (PostgreSQL 9.6 Beta 3)
• Beta 4 released on 18th August (PostgreSQL 9.6 Beta 4)
• RC 1 released on 1st September (PostgreSQL 9.6 RC 1)
• v1.0 released on 29th September (PostgreSQL 9.6.0)
• v1.1 released on 27th October (PostgreSQL 9.6.1):
  – 39 bug fixes & 2 new features
• V1.2 released on… TBD!
  – 25 bug fixes & 3 new features, as of 2016-11-22
Stats (as of 22\textsuperscript{nd} November)

- 20 EDB contributors, totalling \(~15,000\) hours of effort
- \(~90\) community contributors
- 1555 commits
- 108 bugs outstanding
- 20 bug fixes in QA
- 69 feature requests
- 380 bugs confirmed resolved by QA
Stats (as of 22nd November)

• 239,008 lines of code:
  – 149,715 – Javascript (inc. libraries)
  – 42,979 – Python
  – 21,174 – CSS (inc. libraries)
  – 16,107 – SQL

• 6,307 lines of documentation source:
  – 87 RST files
  – 304 screen shots
More info

- Website:
  [https://www.pgadmin.org/](https://www.pgadmin.org/)

- Source code:
  [https://git.postgresql.org/gitweb/?p=pgadmin4.git](https://git.postgresql.org/gitweb/?p=pgadmin4.git)

- Mailing lists:
  [pgadmin-support@postgresql.org](mailto:pgadmin-support@postgresql.org)
  [pgadmin-hackers@postgresql.org](mailto:pgadmin-hackers@postgresql.org)
THANK YOU