Improve bug filing interface for bugzilla.mozilla.org

03.04.2017

Proposal for Google Summer of Code 2017

(Mentors - Dylan Hardison, Emma Humphries)

Personal Details

Name: Sebastin Santy

Email: sebastinssanty@gmail.com

IRC Nick: seban

Telephone: +91 777 400 8257

Country of Residence: India

Timezone: Indian Standard Time (UTC +5:30)

Primary Language: English

Profiles: Github, Bugzilla, Mozillians

Project Proposal

bugzilla.mozilla.org (BMO) has been in production since 1998 and is undergoing major developmental changes especially in two parts of the code: User Interface and Performance. After having detailed discussions with Dylan over the past few months, working with the BMO extensively and working through the issues and bugs in BMO, I hereby propose to work on the below mentioned feature which I feel can drastically improve the performance of BMO and its experience to the users at large.

Existing Issues with enter_bug.cgi

The bug filing page (“enter_bug.cgi”) has several problems, and currently shares no code or UI elements with the new bug detail page (show_bug.cgi).

Some of the reported problems that users/developers face are:

I. **Multiple Views** - There are three views (simple, advanced, and the “guided” wizard-like system).
II. **Communication Problem** - Because of the above, new and existing contributors have trouble communicating because what one type sees, the other does not.
III. **Increased Work** - The guided bug entry form increases work for users triaging bugs because it funnels bugs into the wrong components.
IV. **Legacy Code** - Making any substantial changes to enter_bug.cgi is difficult because of legacy concerns (custom forms, complicated spaghetti code).
V. **Increased Bug Filing Time** - Core developers of Firefox routinely leave open tabs because loading bug filing page for the Core product takes upwards of 8 seconds.

Some of the internal problems that make enter_bug.cgi particularly unwieldy are:

I. **Two hop process** - enter_bug.cgi is a stateless CGI that posts form data to post_bug.cgi which subsequently makes a database entry. This is easily confused with process_bug.cgi, which is the backend for show_bug.cgi. Having a unified place in the web ui for bug creation would be a cleaner design.
II. **Backend HTML rendering** - enter_bug.cgi is completely dependent on Template Toolkit rendering of html pages. The parts of the templates that are slow involve loops over huge lists of things.
III. **Critical Code** - Making changes to enter_bug.cgi itself are difficult to impossible because it is actively used, and the risk of breaking it is too high (an inability to file firefox bugs would be critical).

Solution for Improvement

To solve this, a **greenfield approach** to bug filing should be attempted. Thus writing a new bug filing page that doesn't share the legacy constraints of enter_bug.cgi would be considered a better way to handle this problem. Enter **new_bug.cgi**. It aims to solve the above problems:

I. new_bug.cgi would create a streamline process of direct access with the database, instead of two hop access.
II. All the heavy tasks like data fetching would be done by JS using the Bugzilla API. Hence Template Toolkit, would only be used for a basic rendering of the page. This would decrease the already applied speed penalty due to TK.
III. Considering the above process parts of the bug filing like instantaneous product/component selector, pulling of flags etc. can be incorporated using Bugzilla API into new_bug.cgi
IV. As mentioned earlier, huge loops can be avoided by using the very fast JSON serializer perl module which is written in C - JSON::XS

Schedule of Deliverables

Before the coding period begins, that is during the community bonding period, I intend to further work on my Perl skills and familiarise myself with the codebase and other aspects of BMO. Though I have worked on several bugs for BMO since past nine months, majority of them were related to the User Interface. As I am proposing a new bug page to be made, it would definitely be better if it is inspired from the new bug modal design incorporated in show_bug.cgi.

The following is a tentative to-do list that I think will be suitable for the project timeframe. Of course, one must keep in mind that there can be situations which might delay or lead to an early completion.

The whole project is broadly divided into five phases:

I. Development of UI with barebones bug filing functionality
II. Testing (Alpha release) and Security Issues
III. Completion of new_bug.cgi
IV. Provide test coverage to new_bug.cgi
V. Document the implemented functionalities in the project

I. Development of UI with barebones bug filing functionality

- **Bug Modal** - The UI would be borrowed/inspired from the new bug modal skin of show_bug.cgi. The major hurdle with this is that the show_bug.cgi UI isn't modular, and hence can't be directly copied. The work here would first include modularizing the UI.
- **Selectors** - At present, the enter_bug.cgi needs a preselection of product on the go. To streamline the process of filing bugs, new_bug.cgi will be a single page filer. To achieve this, an instant product/component selector needs to be deployed which uses Bugzilla API calls (this can be copied from the existing ProdCompSearch extension that is used elsewhere in BMO).
- **Single Interface** - new_bug.cgi would be a combination of the bug filing interface of enter_bug.cgi and the db interface of post_bug.cgi.
- **Bare number of fields** - At this stage, only bare number of fields necessary for a basic filing of bugs will be included.
- **Estimated Time** - 2 weeks

II. Testing (Alpha release) and Security Issues

- **CSRF Protection** - Use BMO's CSRF token system, before having the new_bug.cgi released into the wild, to prevent CSRF attacks.
- **Alpha Release** - Set up an option in the preferences to prepare for a minor release of new_bug.cgi for testing purposes.
- **Feedback** - Free from the constraints of the existing system, solicit the advice and direction from Emma Humphries who is the point-person at Mozilla for all things related to the bug reporting/QA process at Mozilla (the "Bug Master").
- **Process Iteration** - Iterate on the design weekly, using not only feedback from the Bug Master but actual BMO users as well.

- **Initial Estimated Time** - 2 weeks (for iterated steps, 3 days of feedback would be enough).

III. Completion of new_bug.cgi

- **Continuous Development** - As this particular project heavily depends on Bug Filers and their opinions, hence a weekly 1:1 would be the best way to refine new_bug.cgi.
- **Including all Fields** - As the initial work would exclude some types of fields, this portion is where the most important ones would be added.
- **Profiling** - Will need to ask BMO admins on the performance data for comparison between response times of enter_bug.cgi and new_bug.cgi to validate the work.
- **Estimated Time** - 4 weeks (with changes happening in each week)

IV. Provide test coverage to new_bug.cgi

- **Test-Driven** - Tests will be written every week, to the features implemented in that week.
- **Coverage** - Provide exhaustive test coverage and stitch up all the tests written individually per week.
- **Estimated Time** - 1 week (final coverage)

V. Document the implemented functionalities in the project

- **Documentation** - I intend to reserve last day of every week into documenting all the significant changes I have made in that week, along with completing tests, as earlier mentioned.
- **Blog** - To keep track of my GSoC progress, I plan to start blogging. The blog will contain an overview of my previous week's work, links to the accepted pull requests, and my plan for the next week

Approximate Timeline

My summer vacation starts from 13th May, which is well in advance to the Coding Period and extends upto 1st August, till which I'll plan to complete my majority of work. I have no other commitments during GSoC that I currently know of, and will be able to concentrate fully on the project.

I. Community Bonding Period: May 2 - May 29

- As mentioned, become adroit in Perl and Template Toolkit for seamless integration with the backend. Also get a hang of the codebase and the working of BMO.

II. Work Period 1: May 30 - June 30

- **Week 1** *(May 30 - June 6)*
  Create brand new new_bug.cgi skeleton code. This will be the outline of the script, so that the assumptions can be verified.
- **Week 2** *(June 7 - June 13)*
  Begin work of trying to have new_bug.cgi present a form based on the bug modal view. It won't accept form submissions, but it should display some of the fields.
- **Week 3** *(June 14 - June 20)*
  At this point a decision will be made as to how many fields can be supported for the rough prototype -- a minimum of product, component, summary, user story, and description should be possible, but the previous week's effort will guide to what else can be reasonably represented (considering flags, tracking flags, keywords, whiteboard status, custom fields, attachments, etc) .
  With this decision, work should proceed to making new_bug.cgi accept these form submissions and create bugs.
- **Week 4** *(June 21 - June 27)*
  This week would be for finishing up any remaining parts of the prototype phase.

- **Phase I Evaluations** *(June 26 - June 30)*

  Output: new_bug.cgi should be in for review, in a state where it could be deployed and used.

III. Work Period 2: July 1 - July 28

- **Week 5** *(July 1 - July 4)*

  Based on which fields are currently visible, meet with Emma and Bug Filers and ask which is the most important thing to add next. This is heavily dependent on what is already available and is difficult to predict, but it is likely that Tracking Flags will be both difficult to handle and important, but there will have already been constant validation and feedback. With that said, let us assume that week 5 is all about tracking flags.
- **Week 6** *(July 5 - July 11)*

  This week would be about attachments (subject to re-prioritization).
- **Week 7** *(July 12 - July 18)*

  Regular flags -- most important *needinfo?* As *review?* And *feedback?*
- **Week 8** *(July 19 - July 25)*

  Refinement, possibly security group testing. There are a lot of complicated interactions between filing bugs and security groups to consider and this task would be risky.
- **Phase2 Evaluations** *(July 24 - July 28)*

  Output: At this point new_bug.cgi should be able file bugs for the majority of people. In the preceding week, it would have already been "dog-fooded" for the filing of bugs relating to its own feature development. It is noted that filing security bugs might not work at this point.

IV. Work Period 3: July 29 - August 29

- **Week 9** *(July 29 - August 1)*

  Taking stock of what is the form can do, remaining issues should be resolved and the code put into a state for continued development.

- **Week 10** *(August 2 - August 8)*

  Wrap up the new_bug.cgi with proper documentation. This week would involve proper structuring of the documentation done at the end of every week. As this week would be prior to the major release period, prepare a documentation to make users/developers aware of its use and also the benefits of using this new bug interface
- **Week 11** *(August 9 - August 15)*

  Beta Release new_bug.cgi among bugzilla devs/ admins. Polish the code further with suggestions from the community and Bugzilla Team. This would be a buffer time for any unforeseen contingencies that can occur.
- **Week 12** *(August 16 - August 21)*

  Major release period - Publicize among all users - Get Feedback
- **Final Evaluations** *(August 21 - August 29)*

  Output: A new bug filing interface which would be fast, easy to use, cross-compatible.

Further Development

Buglist.cgi also faces problems similar to the bug filing interface. It is another core part of BMO, which lists the search queries and is frequented a lot by the Mozilla contributors. According to profiling analysis of buglist.cgi, it is found out that almost half of the speed penalty is in rendering the html from backend. A plausible way to overcome this, as discussed regarding enter_bug.cgi, would be to have a html page which will retrieve data using XMLHttpRequest/AJAX and also lazy loads, which can provide a major speed boost.

Open Source Development Experience

I've been contributing to Mozilla for the last nine months and have worked with the Engineering Productivity Team (Bugzilla) (#bteam). I've completed some projects with the Bugzilla Team, which includes covering one of their milestones for 2017: Deliver sample custom form via REST application. Regarding my involvement, please check the following links

My Open Source contributions to Mozilla include :-

- Custom Form Example - Made a custom form example application which uses REST API (also auth delegation callbacks) to file the application as bugs on bugzilla.mozilla.org. It is also designed for handling dynamic class models (with metaclasses) based on the feedback of the author. I maintain this repository along with Dylan Hardison (:dylan) now. Live demo here.
- bugzilla.mozilla.org - I have been contributing to BMO since August last year. Till now my work involves majority of front-end parts. I am also getting accustomed to its Perl based backend, a language that is a tad bit new to me.
- Treeherder - I started working for treeherder from January for a project under the guidance of William Lachance (:wlach). It involves replacing the REST APIs with GraphQL.
- Firefox - Initially I started contributing to Mozilla with working on some Firefox bugs. I was a highly active contributor during the June/July period in which I have contributed to the Security and UI parts of Gecko. My contributions here.

Other Projects -

- Protocol Analyzer - BITS Darshini is a modular, concurrent web application that stores the experimental meta-data and allows users to specify custom parse graphs in P4 language. We are hoping to publish a paper based on the research at ACM Internet Measurement Conference, London.
- AutolabCLI - This CLI helps committing the code to local gitlab server, submit the code from gitlab server to the evaluation nodes and retrieve the results using a socket connection.
- Github-Bot - Automation Bot to respond to issues/PR's on github. Meant to be used on Bugzilla/BMO Organization. It works on receiving calls from a webhook for activation.

Work / Internship Experience

I have worked several projects though not professionally, but more from the aspect of a position of responsibility. These include -

- Developer, Academic Registration and Counselling, BITS Goa
- Building and maintaining websites of festivals held at BITS Goa which include Waves(cultural fest), Quark(technical fest), BITSMUN etc.

- Member of OSDLabs, an initiative to foster the open source culture and innovation of BITS Goa (which is already famous for producing a lot of open source enthusiasts as evident from the no. of selections in GSoC)

Academic Experience

I am studying Electronics and Instrumentation Engineering (E&I) at BITS Pilani - K.K. Birla Goa Campus, India. I started my higher studies with interest in electronics, but realised that I also had a great inclination towards software and other fields of computer science. I try to merge my interests by experimenting and hacking things. It gives me immense pleasure and pride when I contribute to projects which are used by people across the globe and so lies my entry to open source.

Why Me

I started using Bugzilla since my start of contribution to Mozilla. I have worked for a long time and have become very familiar with its codebase especially the frontend. Being attached to Bugzilla for a long time, I really want to improve the existing features and also add essential enhancements. I share a good rapport with the Bugzilla Team—Dylan, Mark, and Emma, so working with them would be very comfortable and also a pleasure.

Why Mozilla

I have always aimed to make a better internet using bleeding-edge technologies and at the same time make it open to everybody. Mozilla, one of the best open source organizations, strives to achieve exactly that. Hence, I volunteered for Mozilla and contributed to it with my fullest potential. Mozilla also has given me an amazing opportunity to learn the latest technologies and work on them.