Firefox Input v4

Prepared by: Aakash Desai

March 4, 2011

Mozilla 650 Castro Street Suite 300 Mountain View, CA 94041 **T** ext 370 adesai@mozilla.com

Summary

Problem

Currently, Firefox Nightly users can only offer feedback via Socorro, our Crash Stats web interface. The only insight we get from users is around one specific use case, “How stable are our nightlies?”. As a very technical demographic, these users are capable of offering a lot more insights if easy-to-access and easy-to-use tools are created to support them. We’d like to incorporate a feedback mechanism that allows Firefox Nightly users to offer feedback to a more generally-focused dashboard of information.

Goals

1. **Support Firefox Nightly Users**
2. **Integrate with other Mozilla web properties and initiatives**

New Features

#	Feature	Priority
4.0.1	Nightly Submissions	P1
4.0.2	API	P1
4.0.3	Stats per Search	P1

4.0.1 - Nightly Submissions

Submission Form (Webpage)

Mock: http://www.flickr.com/photos/aakashhdesai/5477064591/

Notable Features

1. User Agent Sniffing: To get better constructive feedback from the user, we only want Nightly users on the latest Firefox Nightly to submit information to us. So, the following actions should be performed to safeguard against improper submissions.
    a. Latest Firefox Nightlies (on current date): Direct the user to the “report” submission page
    b. All Other Browser User Agents: Direct the user to a page with a download link to get the latest Firefox nightly build
2. Submission: A HTTP POST, with the following fields, should be sent to the production database. The user should then be directed to a “Thank You” page
    a. Sentiment: A new sentiment category, called “report”, should be submitted.
    b. Text Input: The user should be able to give mostly free-form feedback (profanity is blocked). The max limit of the message should be 140 characters in length.
    c. URL Input: An optional text field should allow a user to submit a valid URL as a part of their “report”.
    d. Build Id: The user-agent will be a part of the form header which will be parsed by Input and added as an element to the entire feedback row entry.

3. Thank You Page Follow-Up: On submission of the feedback, the user should be re-directed to the “Thank You” page.

Reports Dashboard (Webpage)

Mock: http://www.flickr.com/photos/aakashhdesai/5497693495/

Notable Remarks

1. Controls
    a. Product A combobox which allows a user to select between various Mozilla browser products (i.e. Firefox and Mobile Firefox).
    b. When: The user should be offered three options: “30D” (i.e. thirty days), “7D” (i.e. seven days), and “1D” (i.e. one day) where each direct the user to a search over the feedback received within a within the specified time period. When checked, the page should update with four updated “Aspect” graphs that relate to the set of feedback for the selected time period.
    c. Platform: A grouping of checkboxes where each is assigned to a specific platform seen on our database of feedback. When checked, the page should update with four updated “Aspect” graphs that relate to the set of feedback for the selected platform.
    d. Locale: A grouping of checkboxes were each is assigned to a specific locale seen on our database of feedback. When checked, the page should update with four update “Aspect” graphs that relate to the set of feedback for the selected locale.
2. Feedback v. Build Number Graph
    a. Graph:
        i. X Axis (Build Number): The build number (notated by changeset available in about:buildconfig) for which feedback was received on.
        ii. Submission Axis (Y): Number of feedback received.
        iii. Mouse Hover Effects:
            1. Cursor inside the Graph: Once a user hovers over a specific build number point on the graph, the messages received on that build should be shown in a box colored the same as its line.
            2. Mouse outside the Graph: The on hover effects should disappear from view on the graph.

4.0.2 - API

Defined Use Cases

1. Product Drivers
    a. Translate spikes (i.e. 1-2 magnitude's higher than the average over a version) in 'sad' or "broken' feedback over specific search terms and URLs into a newly filed bug in Bugzilla. The tool would most likely poll for data over a manually-set list of terms.
2. SUMO
    a. Match clustered themes to support thread topics using an admin-accessible button per thread
3. Metrics
    a. Retrieve feedback per locale and feedback type (i.e. "happy" and "sad") per beta version to create a world map visualization of user sentiment per locale
    b. Retrieve a cluster ID and work with the messages within them to determine other forms of data analysis
    c. Experiment with locale-based feedback to create possible stop-words for multi-language clustering
4. Localization
5. RRRT
    a. Correlate messages per version within "broken", "sad" and "ideas" feedback types with bug summaries in Bugzilla to find matches. Possibly increment "votes"
    b. Correlate messages and URLs per version within "broken" and "sad" feedback types with comments and URLs in crash-stats' reports over the same version.
    c. Translate queries and basic trends/statistics feedback data over the current version of the user's build id into a set of graphs and counters for an extension similar to the blocker-report's extension
6. AMO
    a. Add-ons developers should be able to retrieve feedback for their Add-on's name over the "happy", "sad", "ideas" (release and beta versions) feedback types using a simple "see feedback" button in the Developer Hub.

4.0.3 - Often Mentioned/While Visiting per Search

Notable Features

1. Global Search View: The beta dashboard uses a search view only when a modifier is selected or a search term is set. Unfortunately, this leaves the main and search dashboards with varying feature sets that belong in both. The first part of this solution will be to make the main dashboard a search view as well.
2. Often Mentioned Snippet: The second part of this solution is to incorporate the most often mentioned words in the search result set into a list on the right hand side of the dashboard view. Each word should have a bar listed underneath it which normalizes (to 100%) the number of times that word is seen in messages within the query. The list should be ordered by the number of times each are mentioned in the query.
3. White Visiting Snippet: The second part of this solution is to incorporate the most submitted domains in the search result set into a list on the right hand side of the dashboard view. Each domain should have a bar listed underneath it which normalizes (to 100%) the number of times it is seen in messages within the query. The list should be ordered by the number of messages each are seen in the query.