**DQ2: RFC**

PHP develops as a result of discussions about new features that are the subject of Request for Comments (RFC). Select two or more proposed additions to PHP from the list at <https://wiki.php.net/rfc>. Analyse the proposals and assess their likely impact. Conduct a brief critical review of the RFC system, and compare it with any other procedures for changing open source software that you can identify.

1. Introduction
   1. What is RFCs for PHP?
      1. Jones, 2013
         1. The PHP RFC process has been in place for a while, and users new to core PHP development are starting to use RFCs to propose desirable features.
         2. In summary, the PHP development process is an organic process and subject to flexibility. It is based heavily around users contributing features that they require. This therefore requires high investment from users who want change. Very rarely do PHP core developers have bandwidth to take on external ideas, no matter how good they are. Feature acceptance has to be pragmatic. The core PHP contributors would love to see more people get commit karma and contribute to PHP.
   2. University of Liverpool/Laureate Online Education (2014)
      1. PHP develops as a result of discussions about new features that are the subject of Request for Comments (RFC).
2. Proposed Addition 1
   1. Identify
   2. Impact
3. Proposed Addition 2
   1. Identify
   2. Impact
4. Review of RFC system
   1. Overview/Procedure
      1. PHP.net Wiki, 2013a
         1. Proposal Initiation
            1. Proposal is formally initiated by creating an RFC on PHP wiki and announcing it on the list. If the proposal is a repeated discussion of an existing RFC, with or without modification, it still should be announced on the list for discussion.
            2. The announcement will be done in a way that's easy to flag & follow, e.g. - by [RFC] in the subject line followed by the title of the RFC.
            3. The proposal should be initiated by one of its authors. If the proposal is a repeated one, re-proposed by somebody else, the proposer should discuss it with the original author, if possible, and add himself to the RFC author list before proposing it.
            4. If the proposer is not a member of php.net and thus can not create RFCs on the wiki, they should recruit one of the members for help or request membership.
         2. Discussion period
            1. There'd be a minimum of 2 weeks between when an RFC that touches the language is brought up on this list and when it's voted on is required. Other RFCs might use a smaller timeframe, but it should be at least a week. The effective date will not be when the RFC was published on the PHP wiki - but when it was announced on internals@, by the author, with the intention of voting on it. This period can be extended when circumstances warrant it - such as major conferences, key people being busy, force major events, or when discussion merits it - but should never be less than minimal time.
            2. This does not preclude discussion on the merits on any idea or proposal on the list without formally submitting it as a proposal, but the discussion time is measured only since the formal discussion announcement as described above.
         3. Voting
            1. The author decides when it's time to move ahead and call a vote on the RFC. If the author feels that there's still healthy discussion going on, they can decide not to move ahead to request a vote after the minimal period, but extend it as needed. On the other hand, if they feel that the discussion is being derailed - they can always move ahead to a vote as long as the minimum discussion time elapsed.
            2. The vote is announced on the mailing list in a separate thread by sending an email with the subject [VOTE]. It should reference the RFCs being voted on and if there are different options discussed, explain these options. It should also contain the URL of the page where the vote is taking place.
            3. The voting has minimal period of 1 week, which can be extended when circumstances warrant it.
         4. Required Majority
            1. Given that changes to languages (as opposed to changes to apps or even frameworks) are for the most part irreversible - the purpose of the vote is to ensure that there's strong support for the proposed feature. It needs to be clear that there are a lot more people actively supporting the proposal, vs. people actively opposing it. We also need to ensure, as much as possible, that the decision isn't based on some arbitrary circumstances (such as a temporary marginal majority for a certain school of thought). For these reasons, a feature affecting the language itself (new syntax for example) will be considered as 'accepted' if it wins a 2/3 of the votes. Other RFCs require 50% + 1 votes to get 'accepted'.
         5. Resurrecting Rejected Proposals
            1. In order to save valuable time, it will not be allowed to bring up a rejected proposal up for another vote, unless one of the following happens:

6 months pass from the time of the previous vote, OR

The author(s) make substantial changes to the proposal. While it's impossible to put clear definitions on what constitutes 'substantial' changes, they should be material enough so that they'll significantly effect the outcome of another vote.

* + - 1. Who can vote
         1. There's no way around this 'small' issue. Changes made to the PHP language will affect millions of people, and theoretically, each and every one of them should have a say in what we do. For obvious reasons, though, this isn't a practical approach.
         2. The proposal here is for two audiences to participate in the voting process:
         3. People with php.net SVN accounts that have contributed code to PHP
         4. Representatives from the PHP community, that will be chosen by those with php.net SVN accounts
         5. Lead developers of PHP based projects (frameworks, cms, tools, etc.)
         6. regular participant of internals discussions
      2. RFC Proposer
         1. Proposers vote with +1 on their own RFC per default if they are allowed to vote
  1. How to Create an RFC (PHP.net Wiki, 2014)
     1. Email internals@lists.php.net to measure reaction to your intended proposal. State who would implement the feature, or whether the proposal is only a “concept”. Proceed with an RFC if feedback is not negative or if a detailed RFC will clarify the proposal. Mail list subscription is at http://php.net/mailing-lists.php. (Reminder: always "bottom post" your replies. Never “top post”.)
     2. Get wiki RFC karma (this is only required if you don't have a VCS account for php.net. PHP developers can log on with their credentials and already have the necessary karma to create RFCs):
        1. Register for a wiki account at https://wiki.php.net/start?do=register.
        2. Email php-webmaster@lists.php.net requesting RFC karma for your wiki account. In the email, remind people about the RFC you plan to create. Note that RFC karma does not automatically give you karma to vote. See https://wiki.php.net/rfc/voting#rfc\_proposer
     3. Create the RFC:
        1. Log into the wiki with your wiki account.
        2. Navigate to a URL like https://wiki.php.net/RFC/my\_RFC (use your feature name in the URL).
        3. Push the “Create this page” button, fill out the supplied template and save.
        4. Edit https://wiki.php.net/RFC and link your RFC under the “In Draft” status.
        5. Write the RFC.
     4. When your RFC is ready for discussion:
        1. Change the status of your RFC page to “Under Discussion”
        2. Change its section on https://wiki.php.net/RFC to “Under Discussion”
        3. Send email to internals@lists.php.net introducing your RFC.
     5. Listen to the feedback, and try to answer/resolve all questions. Update your RFC to document all the issues and discussions. Cover both the positive and negative arguments. Put the RFC URL into all your replies.
     6. When discussion ends, and a minimum period of two weeks has passed since you mailed internals@lists.php.net in step 4, then you can move your RFC to “Voting” status. There should be no open questions in the RFC. Review https://wiki.php.net/RFC/voting#voting first. Consider whether the feature concept should be voted on before you invest time in implementation. Or perhaps an implementation is needed to show the feature is feasible; in this case make it clear whether or not the current implementation is being voted on.
        1. Update your RFC page to “Voting” status
        2. Add the doodle voting macro, for example:
        3. Move your RFC on https://wiki.php.net/RFC to “Voting”.
        4. Send email to internals@lists.php.net announcing the start of voting for your RFC. Start a new mail thread and put ”[VOTE]” in the subject. Set a deadline for voting; the minimum period is one week.
     7. Based on the result of the votes and the discussion there are three possible outcomes:
        1. Your RFC is accepted: update the status of your RFC page and its section on https://wiki.php.net/RFC to “Accepted” and close the vote via changing closed=“false” to closed=“true” in the voting macro. When the code is merged, update the RFC and section to state which PHP version the code was merged into.
        2. Your RFC is declined: update the status of your RFC page and its section on https://wiki.php.net/RFC to “Declined” and close the vote via changing closed=“false” to closed=“true” in the voting macro.
        3. A serious issue with your RFC needs to be addressed: update the status of your RFC page and its section on https://wiki.php.net/RFC to “Under Discussion” and continue again from step 5.
     8. When your feature is implemented in PHP, update the RFC with:
        1. the version(s) it was merged to
        2. a link to the git commit(s)
        3. a link to the PHP manual entry for the feature
  2. Critique
  3. Comparison with other procedures for changing open source software.
     1. Putty, 2014a

1. Conclusion

References

Jones, C. (2013) ‘The Mysterious PHP RFC Process and How You Can Change the Web’, *Oracle Blogs* [Online]. Available from: <https://blogs.oracle.com/opal/entry/the_mysterious_php_rfc_process> (Accessed: 20 June 2014)

PHP.net Wiki (2013a) Request for Comments: Voting on PHP features [Online]. Available from: <https://wiki.php.net/rfc/voting> (Accessed: 20 June 2014)

PHP.net Wiki (2014) How to Create an RFC [Online]. Available from: <https://wiki.php.net/rfc/howto> (Accessed: 21 June 2014)

PuTTY (2014a) PuTTY Feedback and Bug Reporting [Online]. Available from: <http://www.chiark.greenend.org.uk/~sgtatham/putty/feedback.html> (Accessed: 21 June 2014)

University of Liverpool/Laureate Online Education (2014) *DQ2 from Programming the Internet Module Seminar 6* [Online]. Available from: University of Liverpool/Laureate Online Education Blackboard (Accessed: 20 June 2014).