7. In an example, don’t try to solve multiple problems at a time since it can be confusing.

8. It’s important to make your code flexible so it will apply to many use cases. However, since it’s not possible to create code for every possible use case, try to cover the most common ones.

9. It’s important to make people feel comfortable. Providing a good documentation is a first step. The second is providing a proof that your code works as expected and will work with further updates. The best way to do it is a set of unit tests.

10. Extension should be maintained, at least until it’s stable and there are no more feature requests and bug reports. So expect questions and reports, and reserve some time to work on the code further. If you can’t devote more time to maintain extensions, but it’s very innovative and no one did it before, it’s still worth sharing. If the community likes it, someone will definitely offer his or her help.

11. Finally, you need to make extensions available. Create the Composer package from your extension, push it on GitHub or other shared repository storage, and publish it on the <https://packagist.org> site.

12. Each extension should have a version number and a change log. It will allow the community to check if they have the latest version and check what is changed before upgrading. We recommend to follow the Semantic Versioning rules from the <http://semver.org> site.

13. Even if your extension is relatively simple and documentation is good, there could be questions, and for the first time, the only person who can answer them is you. Typically, questions are asked at official forums, so it is better to create a topic where people can discuss your code and provide a link at the extension page.

How it works...

If you want to share an extension with the community and be sure it will be useful and popular, you need to do more than just write code. Making extensions distribution-ready is much more work to do. It can be even more than creating an extension itself. So, why is it good to share extensions with the community in the first place?

Making the code you use in your own projects open source has its pros. You are getting people, a lot more people than you can get to test your closed source project. People who are using your extension are testing it, giving valuable feedback, and reporting bugs. If your code is popular, there will be passionate developers who will try to improve your code, to make it more extensive, more stable, and reusable. Moreover, it just feels good because you are doing a good thing.

We have covered the most important things. Still, there are more things to check out. Try existing extensions before writing your own. If an extension almost fits, try contacting the extension author and contributing ideas you have. Reviewing existing code helps you find out useful tricks, dos, and don’ts. Also, check wiki articles and the official forum from time to time; there is a lot of useful information about creating extensions and developing using Yii in general.

See also

• For modern information about PHP coding standards, refer to [http://www.php-fig. org/psr/](http://www.php-fig.org/psr/)

• To learn more about semantic versioning, refer to <http://semver.org>