How Yii framework is developed

Alexander Makarov, Yii core team

What is Yii?

- ✓ PHP5 MVC framework.
- ✓ Easy and fast.
- ✓ Powerful and flexible.
- ✓ Nice API.
- ✓ Took the best from Prado, Rails, Symfony and… Joomla (good parts).
- ✓ Own ActiveRecord.
- ✓ CLDR-based i18n.
- ✓ Cache with dependency suport and a several backends.
- ✓ RBAC.
- ✓ Console.
- ✓ Code generation.
- ✓ …

Prado (2004) → Yii 1.0 (2008) → Yii 1.1 (2010) → Yii2 (?)

Typical controller action

```
public function actionView($id)
{
	$post = Post::model()->findByPk($id);
	if(!$post)
		throw new CHttpException(404);

	$this->render('view', array(
		'post' => $post,
	));
}
```

And a bit more complex one

```
$posts = Post::model()->taggedWith(array('yii',
'DevConf', 'PHP'))->published()->with('comments')-
>findAll();

foreach($posts as $post){
    echo $post->title;
    foreach($post->comments as $comment){
        echo $comment->text;
    }
}
```

Why it's called like that?!

- Yes It Is (2010).
- Easy Efficient Extensible (2009).
- Chinese Yi:
  - Easy and simple.
  - Flexibility, ability to change.
  - Persistence. Core principles are always the same.

"Yi ching", "Book of changes".

Why reinventing the wheel and why MVC?

- When Prado was developed there were at least no competitors.
- Prado took ASP.NET ideas that were “not from this world”.
- MVC is popular and convenient.
- Frameworks of 2008 weren’t that good.

Sometimes it’s better to invent a better wheel!

Borrowing ideas

- Should we check “competitors”?
- Copy-paste is evil. Think.
- Analyze. Take the best and leave the rest.

Who “owns” framework?

Community

- ✓ Framework isn’t developed for a single project or group of similar projects.
- ✓ Community gains from framework improvements and is willing to contribute.
- ✓ Always willing to discuss problems and find solution.
- ✓ Works for reputation.
- ✓ Relatively fast at making decisions.

Company

- × Mainly interested in profit.
- × Framework is often just a side product of commercial products.
- × Works for money. The rest isn’t a priority.
- × Slow at making decisions.

Why BSD?

- Framework is a tool.
- Code should be used to gain real feedback and contributions.
- “Used” means used in real commercial applications.
- BSD allows to use framework in such applications.

Building community

Yii

- Got a lot of talented developers from Prado community.
- Community is not a helpdesk.
- Repeated questions → wiki, search.
- If there aren't that many questions it's not bad, maybe documentation is good.

In general

- Interesting and unique solutions and recipes.
- Get a strong loyal developers core.
- Show what's good by example.
- Give a change to others.

Yii team

- **6** developers.
- Tech. writer.
- **~50** translators.

- Priority:
  - Code stability.
  - Code reliability.
  - Consistent style.

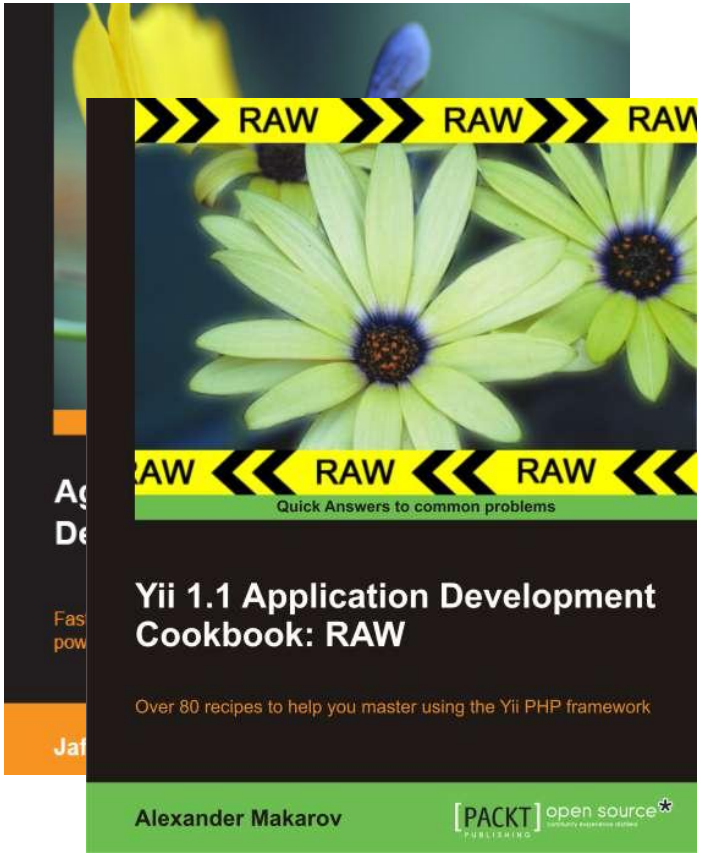
- We’re not expanding team rapidly because:
  - It will require more time for communication. Less time will be left for getting things done.
  - More people, more conflicts.

Use the force and read the source!

No thanks!

Documentation

- Code without any documentation is useless.
- Project success depends on good documentation.
- Documenting when implementing.
- We’re always documenting methods, classes and properties.
- Examples are important but it’s important to know when to stop.



Why code should look good

- To make developer as happy as end users.
- Not to spend too much time documenting things.

- Motivation, time, money.

What API is good?

- **Consistent**. Overall style should be consistent.
- **Not too complex**. Complexity = many things to look for.
  - Magic should be interpreted in a single way. It it's magical it should be simple.
- **Flexible**. Easy to change default behavior.
- **Documented**.

- **Flexibility vs Simplicity**.
  - Edge cases: God object, dependency injection for everything.
- **Solution**: facade to hide details, adapter to make things consistent.

There is no silver bullet

- When developing relatively big project you'll face the need to change defaults.
  - How to update then?
  - How not break it?
- Framework can't implement everything.
  - So I need to change third-party code also?
  - What it if will not work?

Versions and backwards compatibility

1.1.x — backwards compatible releases.

- Small documented changes.
- New features (these aren't breaking anything).
- Good for updating existing projects.

1.x.0 — partial incompatibility.

- Can't update without some significant changes in the project.

x.0.0 — tasty new features. No backwards compatibility is guaranteed.

Supporting legacy versions

Project

- Can be long-term. Several framework versions will be released including incompatible ones.
- Immediate updating can lead to losing time and money.
- Maybe it's better not to update… but how about bugs and security holes?

Framework

- We can't afford supporting all legacy code but understand that it's needed → limit support time to make end of support expected.
- Don't bring new features into old branches. Just fix bugs.

Tools

- «One time» data migration in most cases will be needed multiple times.
- Code should be tested.
- Writing same code again and again is boring.
- IDE should assist when developing with framework.
- Debugging process is important.

Performance and optimizations

- Yii is fast because it does only what needs to be done.
- SPL __autoload.
- Don’t connect to DB until performing a query. Don’t parse data until it’s required etc.
- If code is executed all the time performance is more important than features.
- If code isn’t executed that often we can implement some extra features and don’t worry about performance.

Yii is good!

Thanks

- http://yiiframework.com/
- http://yiiframework.ru/
- http://rmcreative.ru/
- sam@rmcreative.ru

- Will be glad to receive
  - Feedback
  - Ideas
  - Proposals