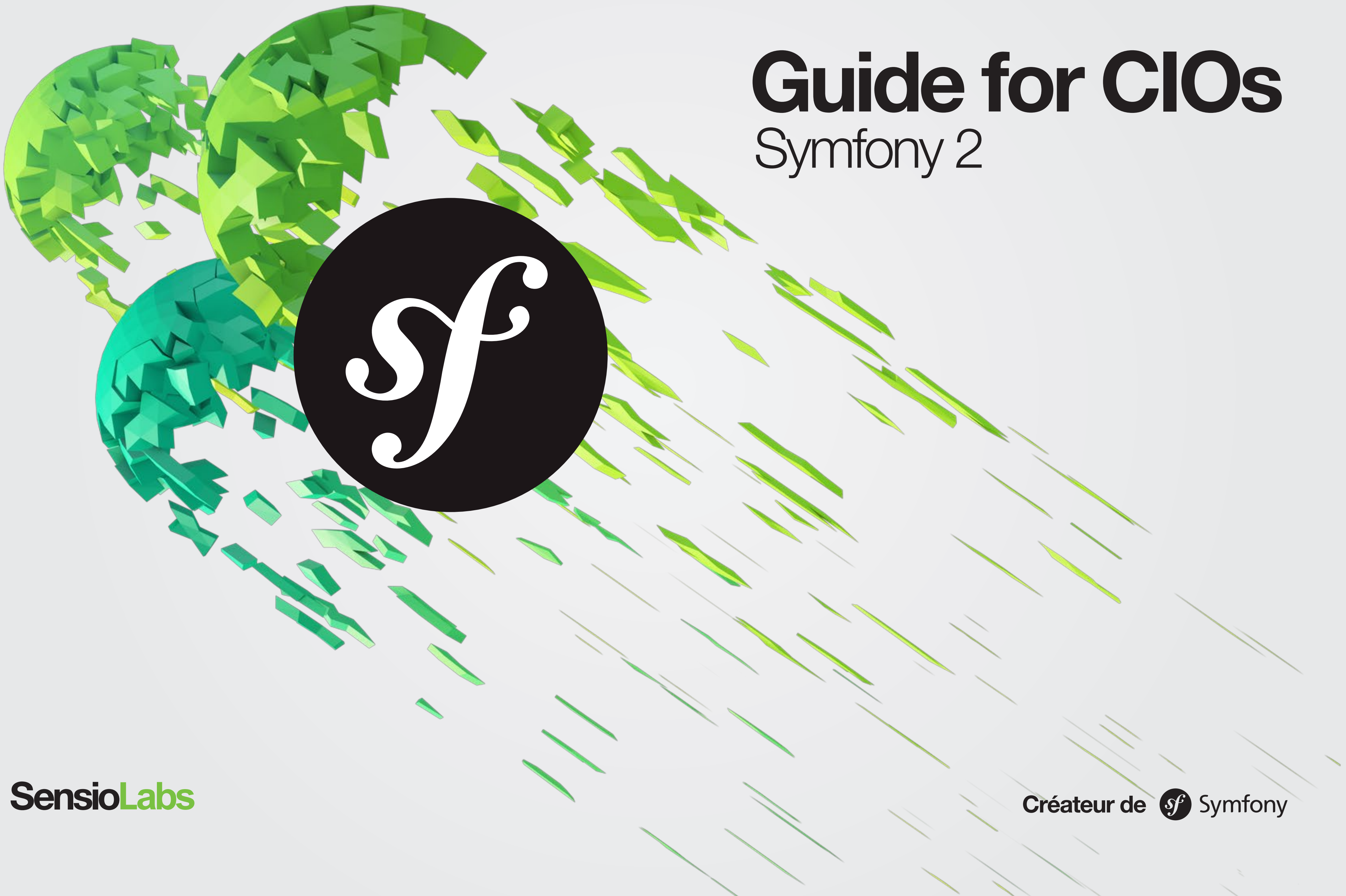
Guide for CIOs

Symfony 2



SensioLabs

Créateur de Symfony

Table of contents

7 common beliefs about Symfony

(alright, 6½)

Symfony is an Open Source framework for building PHP web applications. That's all fine and good. And…

1 Symfony? Never heard of it ... did it just come out?

Hard to sort things out when there are so many frameworks on the web development market! Yet Symfony already has quite a few good years of experience under its belt – the first version was released in 2005, and the second in 2011. Today, it's a stable and internationally recognized product.

Just look at the diverse quantity of companies and projects who endorse it: Dailymotion, Yahoo!, LaFourchette, ezPublish, phpBB, Drupal, Pepsi, LVMH … even YouPorn!

Not to mention the extensive community of contributors who rally behind this Open Source framework. But we'll come back to that later....

2 Aren't frameworks a bit over-rated?

You might ask yourself, "What good is a framework if I have a team of highly skilled PHP developers who can easily do without?" You're right ... but only a little bit right! Frankly, it would be a shame to miss out on a tool that could help you program faster and better, no matter how many *rockstars* you have on your team.

The basic idea behind Symfony? To save time by structuring code and re-using standard modules. In other words, Symfony does half of the work by natively providing pre-packaged, interoperable modules that can be used again and again, any way you like. That means less code to write, less chance for errors, more flexibility, higher quality, and above all, greater productivity. Not absolutely essential, perhaps, but extremely practical!

3 It's only worth using Symfony for large projects! I just want a small app that runs well.

You know the saying: it's not the size that counts.... It just depends on what you need! Selecting the right development solution (framework, CMS, e-commerce solution, etc.) should be based on identifying your needs and the way in which each solution meets those needs.

Of course, Symfony works well for tiny projects as well! Why? Because it has unlimited flexibility since all of its software components are totally independent from each other, Symfony is fully configurable like a toolbox from which you can pick and choose depending on what you need.

So Symfony can be used to develop an entire application, or like Lego blocks, only selecting the functionalities you want.

4 When we eventually upgrade our app, we'll have to start from scratch!

Not so! We'll say it again, FLE-XI-BI-LI-TY!

The fact that everything works in "*bundles*" (those Lego block components) means that you can not only add functionality at any time, but you can also change everything in order to change the behavior of the framework as needed, without having to reconfigure everything. That's because Symfony is based on a system of interface contracts between components.

And don't even think that we're trying to lock you into Symfony (that's not what we're about)! Symfony is consistent with PHP conventions: PHPUnit, class naming conventions, etc. That's what makes it inherently interoperable with industry standards.

In fact, Symfony is so interoperable that it, itself, uses external software components in its kernel (e.g. Doctrine, Swiftmailer).

5 Learning how to code in Symfony is like relearning how to code all over again!

We're not going to lie: even for a PHP developer, tackling Symfony might seem like ... not knowing how to code anymore! Just don't forget that Symfony is still PHP. So it's not that you have to relearn how to write code, but rather learn to understand code in a new way. And even though old habits may be hard to break, the hard work quickly pays off.

The guiding, restrictive, and methodical nature of Symfony ultimately makes it possible to:

- ensure that applications are stable, easy to maintain, and scalable, as a result of consistently using programming best practices;

- allow your developers to focus on the more complex and strategic aspects of the project by freeing them from developing generic functionality.

So much so that, believe it or not, developers end up enjoying it! Everything is set up to make their life easier: speed, reliability, flexibility, security.... And as they say, "A happy developer means a successful project!"

6 Open Source is all well and good, but what about support?

Ah, Open Source! Freedom, openness, sharing! That's all great, but once something goes wrong, who comes to your rescue? Don't worry! With Symfony, your teams won't be staring at their computer screens feeling alone in the world.

Sure, it's Open Source, but also created and supported by a software company: SensioLabs. By offering technical support, advice, training, etc., SensioLabs' experts can guide you and help you get through it!

Backing up SensioLabs is an international community of developers, integrators, users, and companies all over the world who help further develop the framework through their experience and knowledge, sharing their pearls of wisdom on mailing lists, in forums, and in chat rooms. You'll always find an answer to your questions.

And because "*an undocumented line of code is a line of code that doesn't exist*", there are also plenty of books devoted to Symfony to assist you during your project's entire development cycle.

6,5 What else? ... Does Symfony make coffee too?

Sadly, no! But you can come by SensioLabs and talk to us about your project, and we'll gladly offer you one!

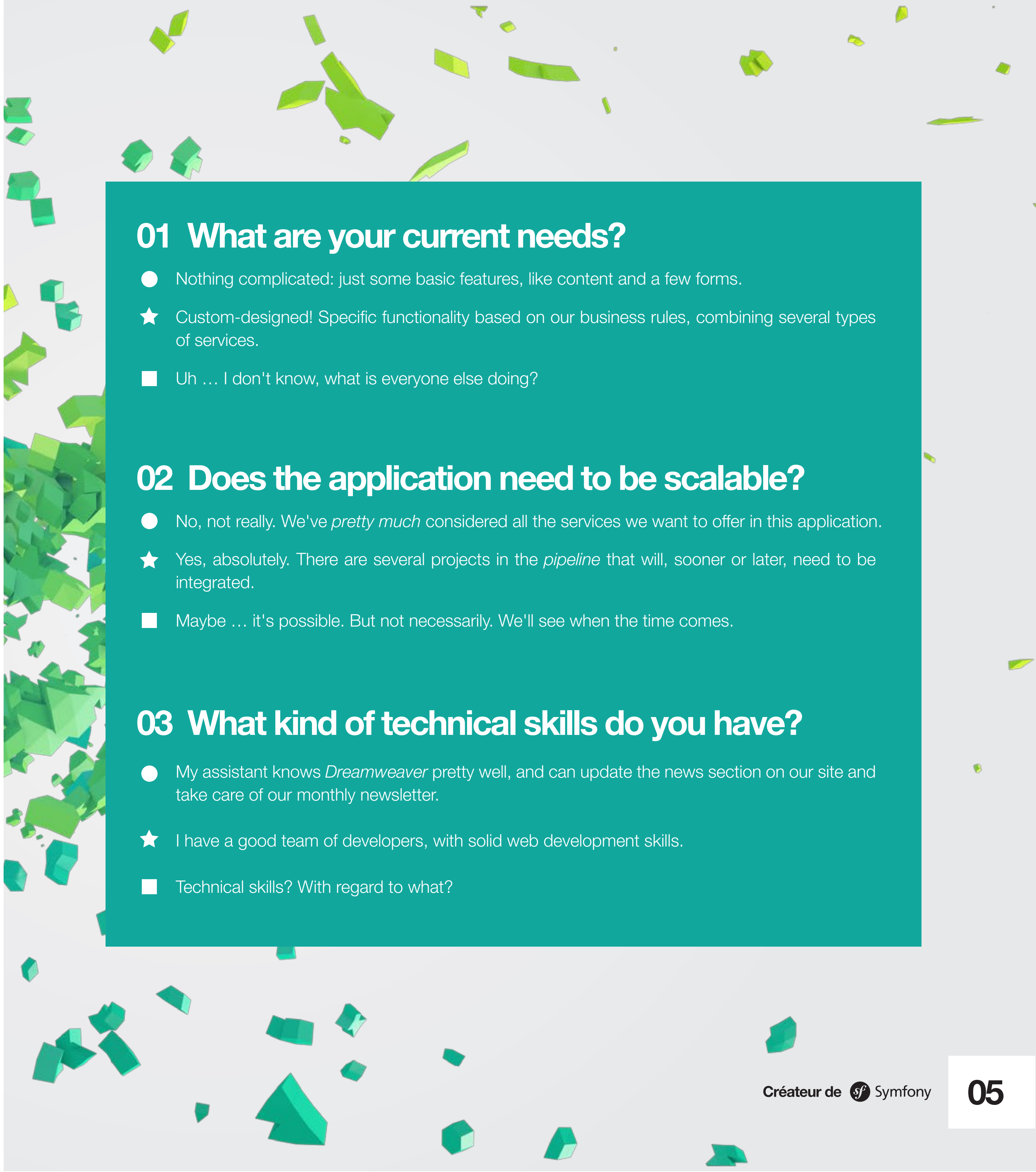
A framework: for whom and for what purpose?

When should you use a framework? For a sprawling website? Or for a small application?

You know the saying, it's not the size that counts.... It just depends on what you need! Selecting the right development solution (framework, CMS, etc.) should, first and foremost, be based on identifying your needs and how each solution is able to satisfy those needs.

But there is a fierce, on-going debate between framework enthusiasts and CMS aficionados, with one side advocating modularity and interoperability, and the other touting ease and speed of deployment. So, how do you make the right choice?

To find out, just ask yourself these 3 questions:



01 What are your current needs?

- ● Nothing complicated: just some basic features, like content and a few forms.
- ★ Custom-designed! Specific functionality based on our business rules, combining several types of services.
- ■ Uh … I don't know, what is everyone else doing?

02 Does the application need to be scalable?

- ● No, not really. We've *pretty much* considered all the services we want to offer in this application.
- ★ Yes, absolutely. There are several projects in the *pipeline* that will, sooner or later, need to be integrated.
- ■ Maybe … it's possible. But not necessarily. We'll see when the time comes.

03 What kind of technical skills do you have?

- ● My assistant knows *Dreamweaver* pretty well, and can update the news section on our site and take care of our monthly newsletter.
- ★ I have a good team of developers, with solid web development skills.
- ■ Technical skills? With regard to what?

How would you describe yourself?

★ You are "framework-oriented"

You have specific needs, certain business rules, need to combine functional components (content + e-commerce, for example), plan to build onto the application: the best choice, in your case, would be a framework. Because its components are independent and interoperable, you can build a custom application that is scalable and easy to maintain over the long term. And a web developer will have no problem familiarizing himself with the tool. It will even help him produce cleaner and more structured code that complies with development standards.

● You are "CMS-oriented"

In general, a CMS – or any other packaged solution (CRM, e-commerce solution, etc.) – can be set up and deployed quickly, doesn't require any special skills, but still offers a certain degree of flexibility. It's perfectly suitable to meet a number of needs, so if its built-in features are enough to meet your immediate and future needs, go right ahead!

■ You are "Not so sure-oriented"

Your requirements don't seem to be well-defined yet. Take the time to bring clarity to your project, while trying to define your current and future needs, in as much detail as possible. This is the first step in helping you decide which development solution makes the most sense for your project.

So: a framework is the best solution for...

- ✓ meeting specific needs, and implementing and combining a complex set of features;
- ✓ a custom-designed and scalable application that is easy to maintain and offers long-term performance;
- ✓ up-skilling your development teams while introducing them to a powerful development tool.

Be sure you're making the right choice, talk to us!

sales@sensiolabs.com | +33(0)1 40 99 80 80
For more information, visit http://www.symfony.com

Myth #23

« Aren't frameworks a bit over-rated? »

You might ask yourself, "What good is a framework if I have a team of highly skilled PHP developers who can easily do without?". You're right ... but only a little bit right! Frankly, it would be a shame to miss out on a tool that could help you program faster and better, no matter how many experts you have on your team.

The basic idea behind a framework? To save time by structuring code and re-using standard modules. Kind of a master toolbox with everything you could ever need. Less code to write, less chance for errors, more flexibility, higher quality, and above all, greater productivity. Not absolutely essential, perhaps, but still extremely practical!

Plus, by using a standards-based framework (as opposed to a "homegrown" one), you are ensuring that the necessary structure is in place so that any developer, whether or not he was involved with the application's development, can easily "take over".

Why choose Symfony ?
(vs. another framework)

You've decided to use a framework to develop your site or application. Which one is yet to be determined: hard to decide with the plethora of frameworks on the web development market.

Here are 8 things to take into consideration when deciding on a framework ... with multiple reasons for choosing Symfony over another!

1. Its reputation and community size

If popularity isn't everything (Justin Bieber did sell out Madison Square Garden...) when it comes to development tools, it's a pretty good indicator of quality and performance, just like the size of the fan club usually signifies the potential for innovation.

Symfony checklist

- ✓ An international community of thousands and thousands of contributors (integrators, Symfony developers, as well as eZPublish and Drupal community members) actively participate in helping Symfony grow (producing documentation, creating *bundles*, improving code, providing support, etc.)
- ✓ Many Open Source tools are built based on Symfony and use some of its components (e.g., Doctrine, Propel, Monolog, Composer, Assetic, Swiftmailer)
- ✓ Flexibility and on-going enhancement
- ✓ One of the leading frameworks for PHP developers

2. Its customers

What better proof of a solution's reputation than the number, diversity, and quality of its customers?

Symfony checklist

- ✓ Thousands of sites and applications developed all over the world
- ✓ Intranets, public websites, social networks, community sites, management and workflow applications, etc.
- ✓ High-profile customers and projects such as Dailymotion, Yahoo!, LaFourchette, ezPublish, phpBB, Drupal, Pepsi, LVMH, American Express, New York Times, M6Web, TF1, Canal +… and even YouPorn!

3. Its philosophy

At the heart of any project lies, above all, the fulfillment of an initial need. Capitalize on knowledge and tools over the long term, so that you can focus on your real business issues without re-inventing the wheel with every new project or change.

Symfony checklist

- ✓ A tool that was first designed to meet the needs of the software company who created it – SensioLabs – and who continues to use it on a daily basis to work on projects for its customers
- ✓ A tool developed by professionals for professionals to address a very specific need (*hence proved*!)

4. Its software license

Having a good understanding of software licensing can potentially save you from some unpleasant surprises! It can actually have a considerable impact on your application: runtime restrictions, terms of use, right to modify source code, etc.

Symfony checklist

- ✓ Distributed through an MIT Open Source license
- ✓ Any or all portions of the source code may be modified
- ✓ No viral license requirement, i.e., re-distributing the code as open-sourced
- ✓ Can be used in proprietary commercial software applications

5. Its stability and future-proof technology

It's good to choose a solution that can be put into immediate use. It's even better if it can stand up to the test of time! A framework must ensure that your application can be maintained and enhanced for many years to come.

Symfony checklist

- ✓ Structured code that complies with development best practices
- ✓ True interoperability with PHP standards
- ✓ Modular structure to make it easy to add, modify, or remove features
- ✓ Clear roadmap (2 minor releases a year, 1 LTS version released every 2 years)
- ✓ 4 years of support for an LTS (Long Term Support) version (4 years for security vulnerabilities and 3 years for bugs correction)
- ✓ Guaranteed compatibility with new releases

6. Essential human and technical resources

A web application must be backed by competent, productive humans, and a powerful, robust infrastructure. Better be sure of it before the end of the project!

Symfony checklist

- ✓ Quick-to-learn tool with extensive documentation and available support
- ✓ Developers acquire skills easily by promoting the methodical use of development best practices and adapting proven concepts from other technologies (e.g., Java, Python, Ruby)
- ✓ Developers focus on the more strategic aspects of the project by eliminating the need to develop generic functionality
- ✓ Ever-increasing number of Symfony developers, including certified developers
- ✓ Developing with Symfony is actually enjoyable

7. Support and documentation

We're not going to lie: even for a seasoned developer, tackling a new framework might seem like not knowing how to code anymore, and feeling totally alone staring at the computer screen....

Having access to the right talent and the appropriate expertise can save the day.

Symfony checklist

- ✓ A motto: « *an undocumented line of code is a line of code that doesn't exist* »
- ✓ Extensive literature is devoted to Symfony, translated into several languages (French version available soon) and updated every day by the entire community: hundreds of blogs, tutorials for all levels, etc.
- ✓ Constant support provided by SensioLabs (consulting, training, technical support, etc.) and by the community (mailing lists, IRC channels, etc.)

8. Security

Any application is potentially vulnerable to malicious attacks, even for the "biggest" names get hit (Sony, still can't forget that one…)

To minimize the risk, it is always best to choose a framework that can account for and deal with major threats.

Symfony checklist

- ✓ « Built-in » protection, in which a precautionary approach prevails (everything is locked, except the elements that need to be unlocked)
- ✓ Faster detection and correction of security holes than for proprietary code
- ✓ Management of the most common vulnerabilities (XSS, CSRF, and SQL injection)
- ✓ Access to the free SensioLabs Security Checker tool which checks whether your application uses dependencies with known security issues.

Get started!
Download Symfony now on
http://www.symfony.com

Or contact us:
sales@sensiolabs.com
+33(0)1 40 99 80 80

Myth #11

« Learning how to code in Symfony is like relearning how to code all over again! »

Let's not forget that Symfony is still PHP. So it's not that you have to relearn how to write code, but rather understand code in a new way. And even though old habits are hard to break, the hard work quickly pays off.

The methodical nature of Symfony ultimately makes it possible to:

- ensure that applications are stable, easy to maintain, and scalable, as a result of consistently using development best practices;

- allow your developers to focus on the more complex and strategic aspects of the project by freeing them from developing generic functionality.

Believe it or not, developers end up actually enjoying it! Everything is set up to make their life easier: speed, reliability, flexibility, security.... And as they say, « A happy developer makes a project successful! »

The 6 commandments of Symfony

Simple, powerful, stable, flexible, scalable.... Symfony is more than just a tool. It's a solution that is, above all, geared toward developers and the applications they build. Let's go back to the 6 commandments upon which the framework was founded.

Rule #1: Thou shall be boosted by speed and performance

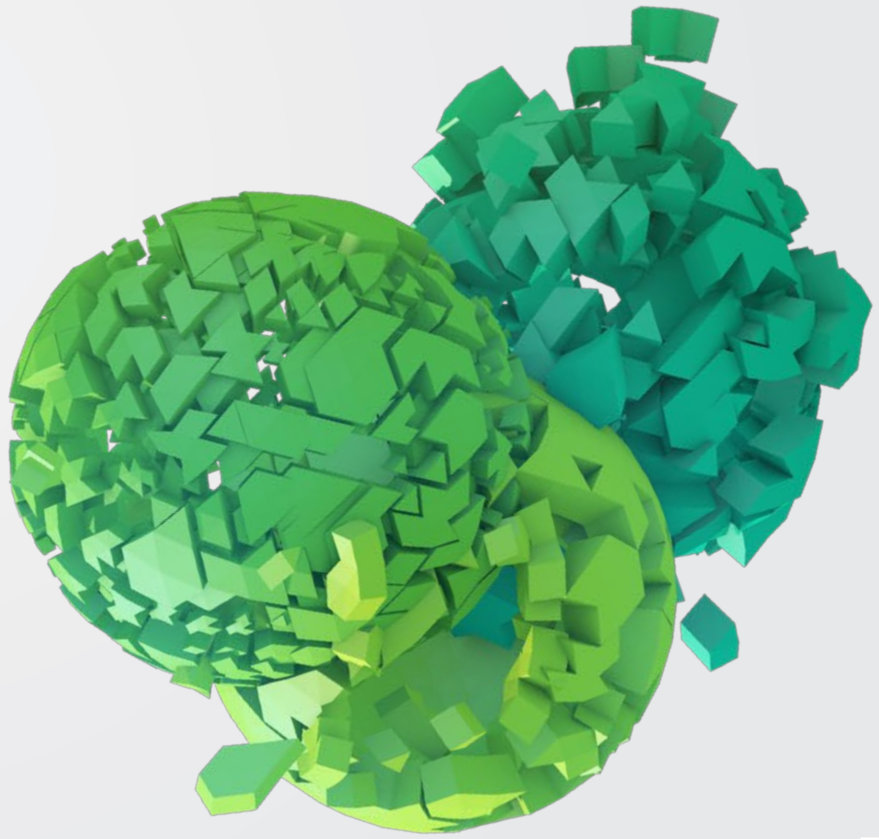
From the beginning, Symfony includes a number of features and tools to create an application that performs at its best. Depending on the specific project and performance goals, developers can take advantage of the Profiler, for example, which collects technical data and metrics on every query (e.g., amount of memory used, page generation times). Or even the dependency injection container, which makes it possible to optimize the use of memory in PHP. In addition, Symfony's architecture is such that it can adapt to the restrictions of any technical environment.

Rule #2: Thou shall be given flexibility and adaptability

What makes Symfony so flexible? Its ability to be freely and fully configurable, over-ridable, and scalable. Because each of Symfony's components is independent, you can:

- develop a complex, multi-feature application in its entirety, using the *Full Stack*
- build a framework "*brick by brick*", based on the features you need
- use a microframework to develop a specific application, without needing to install the entire framework

Symfony is perfectly suited for developing in Agile.



Rule #3: Thou shall be able to make enhancements

From the very smallest component to the Symfony kernel itself, everything exists in *bundles* (or plug-ins). As of today, the Symfony community has contributed over 2,000 *bundles* to meet a variety of needs: content management, user management, image manipulation, content indexing, interfacing with NoSQL databases, streamlined integration with social networks, etc. As a result, you can add whatever functionality you need to the framework, whenever you need it. And you can completely change the behavior of the framework by focusing on the kernel, without having to reconfigure everything – made possible by a system of interface contracts between components. Not to mention that the *bundles* can also be re-used on another project.

Rule #4: Thou shall be guaranteed stability and future viability

What happens when a new version of Symfony is available? It's simple: you won't be abandoned! SensioLabs is committed to supporting major releases of Symfony for 3 years, and potential security issues during 4 years. Furthermore, the contract and Symfony2 interface is guaranteed to work with minor releases: compatibility between all minor releases is supported for the API through public interfaces. This guarantees compatibility between all of the *bundles*.

Rule #5: Thou shall keep thy developers in mind

Because Symfony was first developed for its own tool, SensioLabs thought about the level of ease (and productivity) of its developers when it created a development environment particularly conducive to working efficiently.
First and foremost, this was achieved by eliminating grunt work (developing minor functionality, for example) so they could concentrate on more strategic issues. Next, they were given development tools, such as the now famous Web Debug Toolbar that automatically lists some particularly useful information – database queries, response times, etc. Finally, support was integrated (either from the original program or via community plug-ins) for the IDEs and text editors on the market, such as Eclipse, Netbeans, phpStorm, TextMate, Sublime, and Vim.

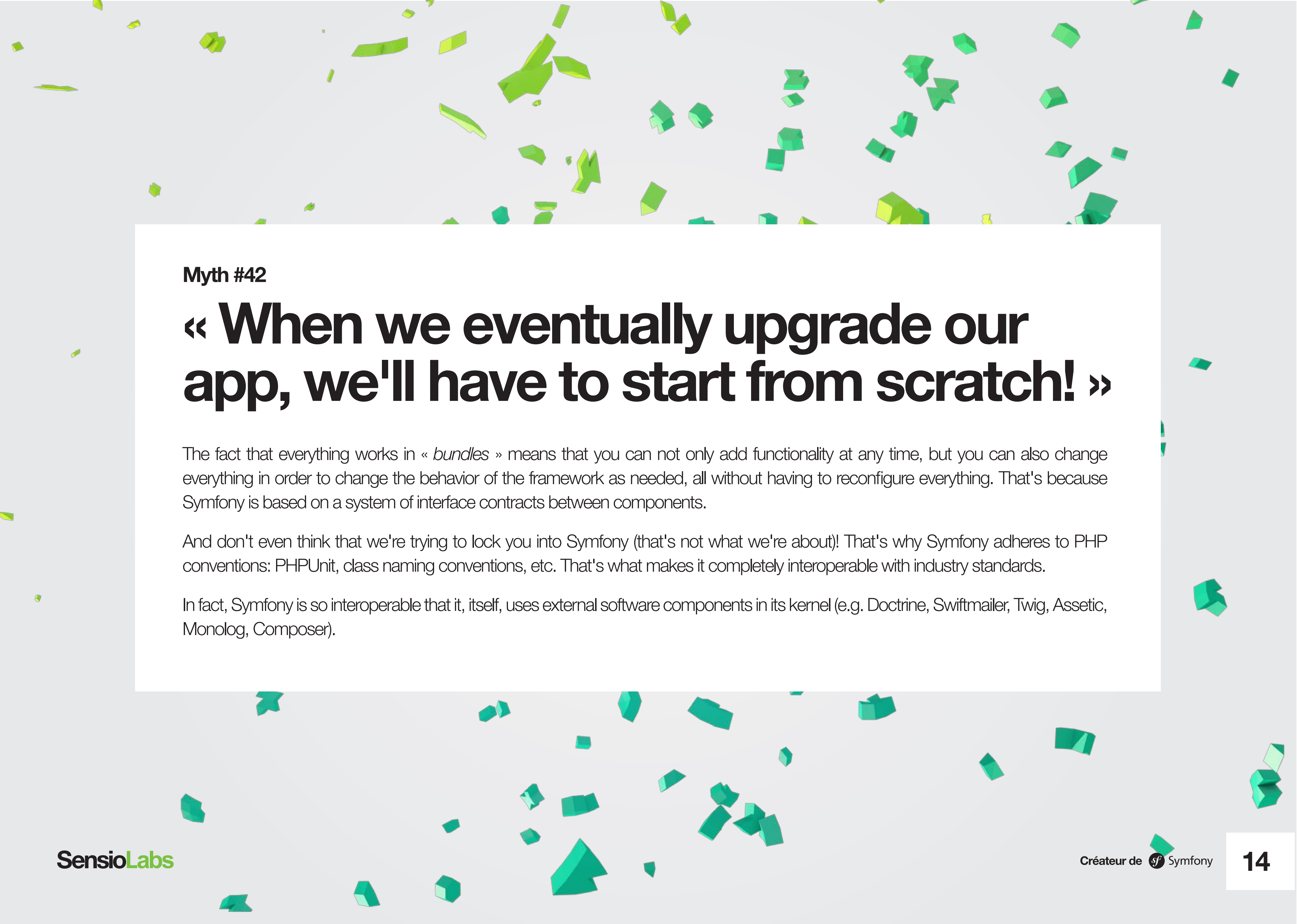
Rule #6: Thou shall make things easy and accessible for thyself

Backed by an active community, along with extensive and clear documentation that is reinforced through professional support, Symfony is one of the most accessible frameworks on the market.
Even for beginners, who are provided all of the necessary resources to quickly and easily feel comfortable.

Shall we talk about your project?

sales@sensiolabs.com
+33(0)1 40 99 80 80

For more information, visit http://www.symfony.com



Myth #42

« When we eventually upgrade our app, we'll have to start from scratch! »

The fact that everything works in « *bundles* » means that you can not only add functionality at any time, but you can also change everything in order to change the behavior of the framework as needed, all without having to reconfigure everything. That's because Symfony is based on a system of interface contracts between components.

And don't even think that we're trying to lock you into Symfony (that's not what we're about)! That's why Symfony adheres to PHP conventions: PHPUnit, class naming conventions, etc. That's what makes it completely interoperable with industry standards.

In fact, Symfony is so interoperable that it, itself, uses external software components in its kernel (e.g. Doctrine, Swiftmailer, Twig, Assetic, Monolog, Composer).

Is a PHP developer a Symfony developer ?

(and vice-versa)

Selecting which framework to use for your development project also involves determining the human resources you will need to devote to it. Yes, Symfony may be based on PHP, but people often say that learning how to code in Symfony is like relearning how to code all over again.... Is it really worth all the hard work?

If those most directly involved are to be believed, the effort – if there really is any – quickly pays off. To illustrate our case, let's observe a Symfony developer's behavior in his natural environment...

He never feels alone behind his computer screen

Don't be deceived by his apparent isolation in the face of adversity! All of the Symfony resources available (extensive documentation, support from SensioLabs, and an international community of thousands of contributors to lean on) make it possible for the developer, even a beginner, to always find the answer to his questions. In this sense, Symfony is not all that complicated to learn....

He is fast, efficient, and calm

Symfony was designed to be a sort of toolbox, made up of independent software components that are configurable, flexible, and re-usable. With less code to write, fewer tedious tasks to deal with, and above all, less chance for errors, the developer is not only more productive, but also has more peace of mind. Not to mention the best practices "embedded" in the framework that are natively applied, without needing to be aware of them or understand them.... Another truth – « the Symfony developer can cut down on his addiction to caffeine ».

He is conscious of his fellow peers

An old application to maintain or update? A developer is all too familiar with the *legacy code* trap, embroiling its herd in the agony of the time-consuming and counter-productive task of analyzing code. A Symfony developer knows how to properly structure an application, avoid this pitfall and ensure that any developer, involved or not with its development, can easily "take over". And in doing so, make it quicker to maintain and update over time.

He does not abandon his roots

Just because a developer is introduced to Symfony doesn't mean that he will forget the basics of PHP, quite the contrary. For one, PHP is like riding a bike – you never forget how to do it. Secondly, Symfony is still PHP, a 100% object-oriented programming language. So it's not that he has to relearn how to write code, but rather understand code in a new way: structuring code, using best development practices each and every time, etc. He is therefore able not only to develop in PHP, but also to write code in a cleaner, more efficient way.

He adds value to his clan

A Symfony developer, assuming he is certified as such, is beyond a major asset within a development team. By validating his working knowledge of the framework, SensioLabs' Symfony Certification gives the developer a legitimate and credible leg up with potential recruiters. And it's also the guarantee for recruiters to not only have an expert on their team, but to showcase the skills and well-known qualifications to their own clients as well.

He is happy

First and foremost, because Symfony frees him from having to develop minor functionality, he is able to focus on the real challenges of the project at hand and realize the full potential of his work. Additionally, because Symfony makes his life much easier: the legendary Web Debug Toolbar, the native support for development environments, detailed error pages, or even built-in security support. Finally, because best practices are promoted (testing, documentation, automation), Symfony allows a project to run smoothly. In other words, an un-paralleled ease of development benefiting everyone involved with the project.

Training, support, resources: learn how our services can help you get started with Symfony.

sales@sensiolabs.com | +33(0)1 40 99 80 80
For more information, visit http://www.symfony.com

Myth #7

« Open Source is all well and good, but what about support? »

Ah, Open Source! Freedom, openness, sharing! That's all great, but once something goes wrong, who comes to your rescue? Don't worry… with Symfony your teams won't be staring at their computer screens alone.

Sure, it's Open Source, but also created and supported by a software company: SensioLabs. By offering technical support, advice, training, etc., SensioLabs' experts can guide you and help you get through the rough patches!

And backing up SensioLabs is an international community of developers, integrators, users, and companies all over the world who help further develop the framework through their experience and knowledge, sharing their pearls of wisdom on mailing lists, in forums, and through IRC channels. You'll always find an answer to your questions.

And because "*an undocumented line of code is a line of code that doesn't exist*", there are also plenty of books devoted to Symfony to assist you during your project's entire development cycle.

Contact SensioLabs

92-98, Boulevard Victor Hugo
92115 Clichy Cedex · France

+33 1 40 99 80 80
sales@sensiolabs.com