

Perl, Outsourcing and China

Outline

- 1 Introduce myself
- 2 Thanks to the Perl community
- 3 Thanks to the sponsors of this event
- 4 The current state of the outsourcing industry in China
 - 4.1 Give some figures about overall growth of the industry
- 5 The current state of Perl education in China
 - 5.1 Discuss the lack of universities teaching Perl
 - 5.2 Discuss attitudes towards Perl (old, dead language)
 - 5.3 Change is coming
- 6 A success story
 - 6.1 Talk about my team and our relationship with Ticketmaster, a company with a strong Perl culture.
- 7 The Future
 - 7.1 Discuss preparedness for the future (increase in outsourcing projects means higher demand for Perl programmers within China)
 - 7.2 Perl 6, a fresh start for Perl

1

Hello, my name is Chris Davaz and I am the Team Leader for the LAMP team at Freeborders. If you haven't heard of Freeborders, it is a U.S. offshore development company with its largest development center in Shenzhen. I've come here today first, because I love Perl! It's really fun to code with and it helps keep me close to Linux as a development platform. Secondly I came to talk a little bit about outsourcing in China and the need for more Perl people to take all the new work coming into the country.

2.

Before I go on, I would just like to thank the Perl community. Without the community, there would be no Perl! Thanks to the contributions made by all of you, we have a thriving community and a living language. Many organizations couldn't thrive without your work, as Perl code has become an integral part of their operations. Also, thanks to Jesse Vincent for managing the Perl 6 project and for coming all the way to China to discuss it with us.

3.

I would also like to thank the sponsors for this event. Again, since Perl is a community project, we do need sponsors like the Thunip Union, Asiacom Technology, and O'Reilly to help hold events like this. Your contribution is greatly appreciated.

4

The advent of outsourcing has greatly impacted the face of IT. As you know, India has taken the leading role in IT outsourcing and China is quickly catching up. In 2007, China's software industry achieved revenue of RMB 583.43 billion and 21.5% YoY (year on year) growth; approximately 5 times more than half a decade ago, and CAGR was 39.4%.

As a result of this explosive growth in IT outsourcing, Chinese software engineers working in this

sector are increasingly exposed to technical areas they may be unfamiliar with. One of these areas is LAMP, especially when the P stands for “Perl” and not “PHP.” This presents a unique set of challenges to the Chinese software engineer and to the Perl community. This can also be a problem for companies who wish to outsource their LAMP projects but are coming up against a wall when it comes to finding the skilled resources they need.

5

One of the reasons there is such a lack of skilled LAMP engineers is that it is simply not taught in colleges and universities. These schools seem bent on pumping out people with rudimentary Java and C# programming skills, glossing over general computer science applicable to all programming languages. Moreover, they seem to be avoiding altogether languages and paradigms not found within the (perceived) absolute mainstream of corporate software development. This fosters an attitude among Chinese job seekers that Perl is an 'old' or even a 'dead' language, making it difficult to draw decent engineers from the talent pool to join LAMP projects. If China is to surpass India in terms of IT expertise, this trend needs to change.

Enough lamenting, what can be done to resolve this predicament? The onus is on us, the Perl community within China, to combat the negative perceptions that have tarnished Perl's name. In order to do that we need to do several things:

1. Evangelize! We need to **tell** people about Perl and why it is the Right Choice for many applications, as well as how **fun** it is to write in Perl!
2. Support! We need to **support** our community, especially the newcomers. We need to **help** beginners, even with simple questions, even when the answer is in page 1 of the manual!
3. Contribute! We need more members of the Chinese Perl community to **join** in open source projects and **make contributions**. Something that desperately needs contributors is **documentation**. We need the standard set of Perl documentation in Chinese.

What can companies do? If your company is taking on LAMP projects but you lack the skilled resources for the job, obviously you need to put together a serious training program. Ideally you would have these LAMP trainees work on one of the many open-source projects out there first and let them build up their LAMP skill-set, then once they are suitably trained, let them loose on your customer's code. Don't just rush through the training! You don't want to make a bad impression on your customer by having people with little or no Perl skills hacking on their code. One big advantage you get with letting your LAMP trainees work on open-source projects is that they get code-reviewed by experienced Perl engineers. Also, your company gets some street-credit for making contributions to open-source projects and you are more likely to attract better talent in the future.

6

Now I would like to talk about an offshore LAMP success story. As I mentioned earlier in the introduction, I work as the Team Lead for the LAMP team at my company, Freeborders. Currently the only client we are engaged with that is offshoring LAMP work is Ticketmaster. You may have heard of this company before. They handle all the ticketing for major sports events, concerts, and the like, including the 2008 Beijing Olympics. Their customer facing website, ticketmaster.com, is all LAMP.

We put together a team of both fresh university graduates and senior engineers. None of these people have worked with Perl in the past, however given a rigorous training program, they were all able to come up to speed fairly quickly. I would like to talk a little bit about the training here, as it may be used by any members of the audience who also need to implement a similar training program.

First we used “Learning Perl, 4th Edition”, the Chinese version. I chose this book because it was what I initially used to learn Perl myself (not the 4th edition of course, but an earlier one). I had each of them go through all the exercises in the book. During their study I got a daily status from each of them so I could keep track of where each one was at and I would field questions as they came up. Also, I would pair up people who were further ahead with those who were lagging behind to try to keep everyone “on the same page.”

In addition to having the book I also taught them how to use the `perldoc` command, gave them several web resources such as <http://perldoc.perl.org> and <http://search.cpan.org>, and setup an IRC chat room we could all use to ask questions to each other.

Once they had the basics down, I started them on an internal project. This allowed them to use Perl in a practical way and to touch upon many of the latest Perl libraries and frameworks. Our project was a simple performance appraisal tracking tool with a web-based UI and MySQL backend. Accordingly, we trained in `mod_perl`, Catalyst and `DBIx::Class`, solving real-world problems with these widely-deployed technologies.

Keep in mind during this time I was holding regular meetings with my team and further elaborating on the documentation. We wound up completing our internal project on time and the team learned a great deal about using Perl in the real world, not just in a textbook scenario.

While our internal training was a success, everyone is now at a level where they can write their own classes in Perl, there were still a few issues I would like to bring up. There simply wasn't enough Chinese documentation. There is lots of great English documentation, but if Perl is to be accessible to the average Chinese coder, there really needs to be a full translation of the standard set of documentation into Chinese. There are books translated from English to Chinese but these tend to be poorly translated, maybe by people who are not very familiar with the subject matter.

7

Before I finish I would like to talk a little bit about Perl 6 and the future of Perl as a general purpose programming language. Perl 6, like Java, has a formal language specification. This is great because it says what Perl **should** do even if a particular implementation doesn't do it (yet). This should help us in many areas, first and foremost in the implementation of the language itself. Secondly, in the area of documentation. We don't have to go back and update the Perl 6 specification because something in its implementation changed as was done in Perl 5!

If you actually take a look at the Perl 6 Synopsis documents you will see Perl 6 allows for much, much more flexibility than Perl 5. Now you can choose to have strong typing, implementation hiding, roles (interfaces), multimethods, metaobject programming, and much more. Perl 6 is a 'real' OO language as oppose to Perl 5 which just had OO capabilities “tacked on” somewhere along its development. This is one of the new things in Perl 6 that I am most excited about and which, once implemented, will certainly keep me away from Java for a long time to come.

Speaking of Java, I believe Perl 6 will finally be able to challenge Java as the platform-agnostic high-level language of choice for rapid development of corporate software applications. In order to do this however, we need a few things. Here are some of my own suggestions:

- Documentation of APIs linked to but developed **independently** from the codebase of any given

module (framework, etc). I envision something like Wikipedia, but for Perl 6 modules. Anyone can go in and edit or add documentation. They don't need to be a module developer with a special account somewhere to allow for commits. Just a normal module user with access to the Wiki. Also, just as in Wikipedia, there will be a list of languages to choose from or add to. This way people can easily maintain non-English versions of the docs side-by-side with the canonical English versions.

- Standardized frameworks. Let's take a look at the most successful frameworks in use today (such as Catalyst or DBIC in Perl 5 land or even Servlets or Hibernate in Java land) and let's come up with a standard for this framework. Then, implement to the standard, the same way Perl 6 itself was done. I feel this will go a far way towards making Perl 6 a viable option for long-term, large-scale corporate development efforts.
- A vibrant and **friendly** community! Let us try to be a bit more friendly to new comers and help them as much as we can, even if the answer to their questions is documented. Let's try to give detailed helpful answers, even if the questions are sometimes a little vague. Let's try to build a reputation for friendliness in our community. We can take a look at #Ubuntu on Freenode. I see many people completely new to Linux asking very basic questions that are widely documented however the people in that room always seem to answer without arrogance or a patronizing attitude. Let's try to do the same.

Anyway, I hope some of what I said today has given you food for thought. I thank you for your time and look forward to discussing with you in more details some of the things mentioned in today's speech.

-Chris Davaz