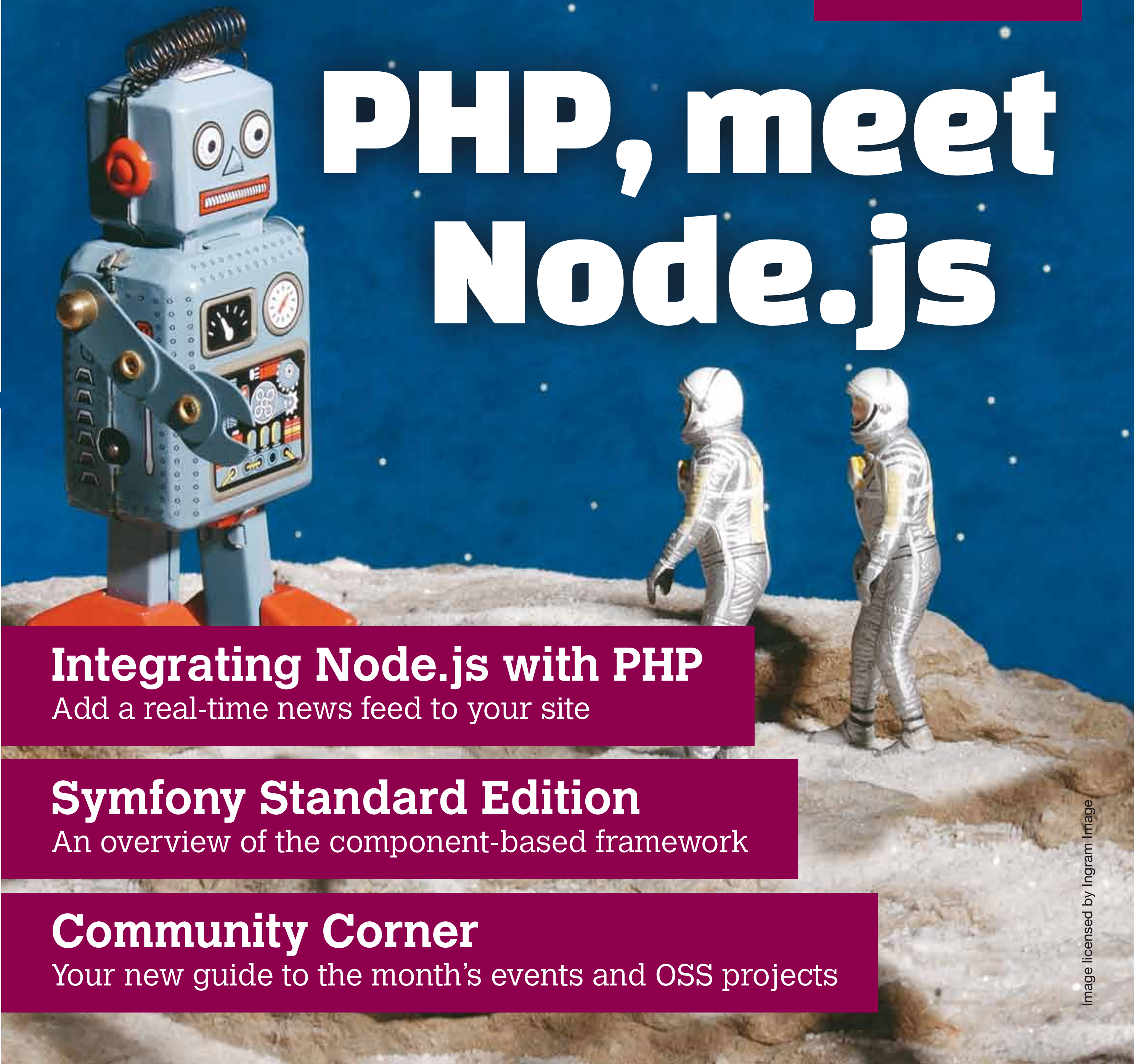
Published by S&S Media Group

# Web & PHP magazine

July 2013
Issue 16



## PHP, meet Node.js

## ALSO IN THIS ISSUE

www.webandphp.com

# Fight to the death!

… is how people often see the interaction between different web technologies. Sometimes, though, these very different languages, frameworks and platforms can live together within the same app and play off each others' strengths.

For example, Node.js may still be young, but it can still show an old dog like PHP a few tricks! By integrating it into your PHP app, you can take advantage of its event-driven model – perfect for real-time features. Lee Boynton explains how on page 7.

Hopefully your daily scrum meetings aren't a battle, either – but if this is the case, our Agile expert Steffan Surdek writes on page 26, you might want to take a step back and reconsider their real purpose.

As well as regular writers Cory Isaacson and Arne Blankets of thePHP.cc, this month we're also saying hello to new columnist Ben Greenaway, who asks: should we try to get our non-technical partners to understand our code?

And if you turn the page, you'll find another new feature, the Web & PHP Community Page. Each month, we'll try to highlight some of the most exciting events and champion community-led meetup groups and open source projects. It's all a bit experimental, so let me know what you think!

Speaking of new things: A few months ago we announced our first ever event, Web & PHP Conference, taking place in San Jose, California in September, and now we've announced our initial speaker lineup! Picking just over 20 sessions from the huge number of submissions we received was incredibly difficult, but we couldn't be happier with our lineup.

PayPal developer evangelist Jonathan LeBlanc is giving a keynote on personalising your website for each user (even tracking their emotional state!), and our very own columnist Frédéric Harper, who has just joined Mozilla as a Senior Technical Evangelist, will speak on developing for Mobile First. Plus, Evan Coury explains how he scaled his startup, described as "a story of luck, risks, mistakes, and eventually, success"

For the rest of the lineup, check out webandphp.com and register now (it's free after all!) We'll soon be announcing details on how you – yes, you! – can help pick the remaining sessions. Until then, though – enjoy the issue!

**Elliot Bentley, Editor**

## Contents

# THIS MONTH'S TOP EVENTS

## July 2 – August 5 // A handpicked selection of some of the most exciting PHP events from around the world

**July 4 // London, UK // Free**

### London Google Glass Creative Kickoff Meet

http://www.meetup.com/London-Google-Glass-Creative/events/123722102/

The UK capital is the latest to get a meet-up group dedicated to Google's new wear-able computing gadget.

**July 5–6 // Madrid, Spain // € 200**

### Spain.js

http://spainjs.org/

A non-profit, volunteer-driven event, Spain.js sees talks from Robert Nyman, editor of the Mozilla Hacks blog and Douglas Crock-ford on "Syntaxation".

**July 16–19 // Orlando, FL, USA // $ 1495**

### SenchaCon 2013

http://senchacon.com/

Sencha Touch users may be interested in the company's annual event, which includes over 60 technical sessions and opportunities to speak to Sencha engineers.

**July 6 // Istanbul, Turkey // Free**

### php-ist: Istanbul PHP Conference

http://2013.phpist.org

A one-day conference taking place in Yildiz Technical University. Talks on Symfony, PHPUnit and Scaling PHP in Public Offices.

**July 24 // Carlsbad, CA, USA // Free**

### San Diego PHP Meetup

http://www.meetup.com/SanDiegoPHP/events/123504062/

At this month's meetup, ScrumMaster Tim Sorweid goes through the basics of agile development in a talk called "It's the Features, Stupid: How scrum helps deliver features on time".

To suggest an event for our calendar, email elliotb@webandphp.com.

# Open source spotlight: PHPCI

Meeting the people behind the most exciting homegrown OSS projects. This month, we speak to **Dan Cryer** about PHPCI, a continuous integration tool "specifically designed for PHP".

**Web & PHP Magazine: Who's behind PHPCI?**
**Dan Cryer**: Before "going public" with the project, all of the development was done by me, funded by the company I run, Block 8 (http://www.block8.co.uk). However since releasing the Alpha version a few weeks ago, we've had some significant contributions from a few other developers, most notably from Github users gabriel403, kamermans and meadsteve.

**WPM: Why did you decide to start it?**
**Cryer:** PHPCI started out as just a pet project of mine, as I've never found a good CI tool for PHP. Sure, you can use Jenkins, but it really is not an enjoyable tool to look at or use, and it's a royal pain to set up!

**WPM: How is it designed specifically for PHP?**
**Cryer:** PHPCI is designed from the ground up to be specifically for PHP – It integrates PHP testing tools like PHP Unit, PHPMD and PHP Spec as first class citizens, rather than as add-ons that you can install manually. It was built by PHP developers, for PHP developers and it was built in PHP to boot.

**WPM: How close to being production-ready is it?**
**Cryer:** Realistically, it is production ready now, it is being used by hundreds of developers already. We're very close to releasing a beta version, there are just a few more features we want to get in place before we do so, such as success/failure notification support. From there, once we're happy that the beta is stable, we'll call it 1.0.

**WPM: How can people get involved, and what can they do to help out?**
**Cryer:** From the beginning, I wanted to make sure that the PHPCI project was one that people could feel comfortable contributing to. All too often, open source projects make it unnecessarily difficult or unpleasant for people to get involved. The easiest way to "do your bit" with PHPCI is to run it, test it and report any bugs you come across – we really value that feedback. If you want to get a bit more deeply involved, you could:

- Try and fix a bug/implement a feature you've found in the Github issue tracker.
- Create a new plugin to support a testing tool we haven't covered yet.
- Work on improving the UI or add additional reporting functionality to builds.

We have a mailing list (https://groups.google.com/forum/#!forum/php-ci) and an IRC channel (irc.freenode.net#phpci) where you're more than welcome to ask questions, talk about your ideas and so on.

**WPM: Where do you hope to see the project in a year's time?**
**Cryer:** Naturally the first goal is to get a stable 1.0 release out and see how developers take to it. I'd really like to see PHPCI used as extensively as the tools that it supports (PHPUnit, PHPMD, PHPCS and so on.).

From there, I have a few key features I'd like to implement, including the ability to run builds on a set of servers (so for example to test PHP 5.3, 5.4 and 5.5 simultaneously,) and to be able to launch Vagrant VMs to run tests in.

Features like that would be optional of course, as the primary goal is to have a testing tool so easy to set up and use that PHP developers have no excuse but to test their code!

Homepage: http://www.phptesting.org/
GitHub: https://github.com/Block8/PHPCI

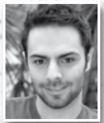
Web & PHP conference

# KEYNOTES

## Mobile First

**Frédéric Harper**

As the industry of mobility is exploding right now, and the support for new technology is now available on all the principal platforms, we must stop thinking about mobile phones as secondary devices. The mobile first philosophy forces us to think, design, and create for the mobile phones first.



## A startup story: Sending a billion text messages

**Evan Coury**

How does a high school student turn a laptop, two prepaid cell phones, and a cease and desist from Amazon into one of the largest text messaging gateways in the U.S? Join Evan Coury as he shares his story of luck, risks, mistakes, and eventually, success.

## The Future of Identification: Personalization Through Interaction

**Jonathan LeBlanc**

The future of user identification and personalization will be in mining the interaction of users with those sites, creating personality graphs and tracking the emotional state of each person based on their actions, delivering customized content, recommendations, and a fully personalized user experience.

September 16 – 18, 2013 San Jose, CA

HEY! HO! LET'S CODE!

SESSIONS . WORKSHOPS . KEYNOTES

FREE ENTRY

@webandphp

www.webandphp.com

Web & PHP conference

There's a new event in town, and it's designed with you in mind. Web & PHP Magazine invites you to its first ever conference. This 3 day event will immerse you in a world of continuously evolving technologies. And sticking to our ethos of open knowledge sharing, conference sessions, keynote presentations, hacks and Expo will be FREE!

For more INFO GO ONLINE

www.webandphp.com

## SESSIONS ANNOUNCED

### Scaling PHP in the real world!
**Dustin Whittle**

How do the largest internet companies scale PHP to meet demand? Find out about the latest tools for developing high performance applications.

### Bootstrap and Foundation Compared
**Jen Kramer**

When discussing responsive design frameworks, Bootstrap and Foundation are bound to come up. What are these HTML5/CSS3/Javascript/jQuery frameworks and what problems do they solve?

### Automated Deployment: From Jenkins To Production
**Sebastian Bergmann, Arne Blankerts**

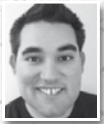
Learn how to make Jenkins and friends push your deployment into the next level, easily maintaining any set of servers.

### Git (and GitHub) for Ninjas
**Ben Straub**

You're a git user, and you love it. Learn how to take it to the next level, straight from the experts at GitHub.

### Reliable Memcached
**Ilia Alshanetsky**

This session outlines various approaches for working with Memcached: making it High-Availability, replicated or fail-over configuration from the context of PHP applications.



### Git Educated About Git – 20 Essential Commands
**Jeremy Lindblom**

Learn 20 essential Git commands that will help you work with your next project, as well as some common conventions and workflows.

### Four web technologies you should be looking at now!
**John Mertic**

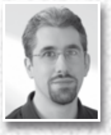
OpenGraph, Shadow DOM, Websockets, and Webhooks: 4 technologies we see as key to driving a whole new class of web apps.

### Alice & Bob: Public key cryptography 101
**Joshua Thijssen**

HTTPS, SSL, SSH, PGP are terms most people know are somehow related to encryption. But how does it work? And who are Alice and Bob?

### DOs and DON'Ts of MongoDB
**Jeremy Mikola**

Tips and caveats will be sprinkled throughout the session as we look at DOs and DON'Ts realized over three years of using MongoDB. (23)



### Building high performance websites
**Arne Blankerts, Stefan Priebsch**

This session will acquaint you with ideas for a high performance software architecture which let you scale from single server to cloud.

### Faster Websites
**Marco Emrich**

This talk will introduce you to the discipline of Web Performance Optimization (WPO) and present you a selection of basic and advanced patterns and techniques.

September 16 – 18, 2013 San Jose, CA

**Working in perfect harmony**

# Integrating Node.js with PHP

©iStockphoto.com/alkir

**It may be in vogue, but is there a practical reason to adopt Node.js? Lee Boynton shows how it can be used to add a real-time news feed to your PHP site.**

by Lee Boynton

Node.js is a server-side solution for building applications in JavaScript which, in the few years it has been around, has already become quite popular. It is built on Chrome's V8 JavaScript runtime, and it is especially well suited to building real-time websites with push capabilities thanks to its event-driven architecture whereby I/O calls are asynchronous.

This article aims to show you how you can start using Node to add real-time features to your PHP-based website. First, we shall look a bit more at what makes Node a good fit for real-time apps, before going on to demonstrate how to build a real-time news feed and incorporate it into your PHP website.

## Thread-based vs Event-based

Traditionally PHP is served with Apache and the mod_php module. If you run the 'top' command on your Unix-based web server you will probably see a large number of Apache processes serving web clients. In this setup, each client request typically spawns a new Apache process until all of the available RAM is used up. Recently, nginx and php-fpm has emerged as the most efficient method of serving PHP websites, but even in this setup each client is served by a different PHP process. The key point here is that, from start to finish, a client request is using up a PHP process for the entire duration. If it takes a long time to process each request, the server's resources can be used up very quickly.

In Node, a single Node process typically serves every client in an event loop. For long-running, expensive processes like accessing the file system, a database or a remote API, it is advocated to use asynchronous method calls instead of blocking ones. This is achieved through the use of callbacks which are triggered when an action like accessing the file system has finished. This means that a single Node process can continue to process new requests whilst the expensive operation is run in the background. When the expensive operation is complete, it goes back into the event loop queue to be processed further by Node.

In essence, Node can be viewed as a similar environment for building applications such as Python's Twisted or EventMachine in Ruby. Node also has a built-in

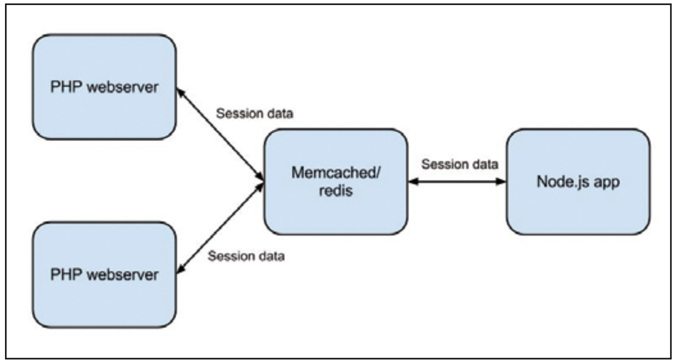


**Figure 1: Shared session architecture**

production-ready HTTP server so it does not need a separate server to run it such as Apache or Nginx, further enhancing its lean resource requirements (barring any memory leaks).

The code example in Listing 1 shows how you can write an obligatory "hello world" web server in just a few lines of code. The first line demonstrates usage of the module system known as CommonJS which Node uses to include separate modules. The *require* function is built-in, and in this case imports Node's HTTP module for use in the application. The second line creates a new web server object. Notice that the first parameter to the *createServer* method is an anonymous function. Most methods in Node accept a callback function as a parameter, and this is the key to building event-driven applications.

The next line of execution is line 5, which uses method chaining to call the *listen* method on the return value of the *createServer* method (the value returned is the HTTP module instance). The *listen* method causes the server to start accepting HTTP requests on port 1337 on localhost. The last line writes a message to the console to say that the server has started. Only when a request to the server is made is the anonymous function called, which sets the HTTP status code to 200 OK and sets the Content-Type header. The 'Hello World' message is finally written to the HTTP response body on line 4.

### Why Should I Use Node.js Then?

Node's event-driven model is especially suited to real-time applications such as games, news feeds and chat applications. In addition, it also allows you to use the same language on the frontend and backend. JavaScript is only becoming more popular as more rich client-side applications are built and web browsers get faster at executing JavaScript. Having to switch between languages can be frustrating.

Second, it has good support for WebSockets. Whilst it is possible to support WebSockets in PHP, Node's asynchronous nature and its built-in HTTP server makes it a better fit. WebSockets are a way of maintaining a persistent connection to the browser in order to push data to the client quickly. Compared to previous solutions such as long polling or comet, WebSockets entail much lower latency, as there is no overhead of instantiating an HTTP connection each time some data needs to be sent. The downside to WebSockets is that it is an HTML5 feature, and as such is not as well-supported in browsers as plain old Ajax is. However, it is possible to gracefully fall back to alternative techniques such as long polling in browsers which do not support WebSockets.

Bear in mind though, Node is an immature platform compared to PHP. Originally created in 2009, it is still in its infancy and has not reached version 1.0 yet – if that matters to you. You may find that APIs you use change in the future, or be unable to find a framework which has the same feature set as your favourite PHP framework. Indeed, in my experience the third party libraries and frameworks available tend to consist of much smaller bundles of functionality that you need to piece together.

There is also a greater risk of memory leaks bringing your application to a grinding halt. Node processes typically run continually, whereas PHP processes tend to be respawned periodically to negate the effect of memory leaks.

### The Integrating Part

The news feed will be integrated with a basic PHP website which handles user logins and sessions, using a common setup of php-fpm and nginx. We will be using JavaScript to communicate with a Node application on the server side and dynamically update the news feed without reloading the page. First though, a quick aside on sessions.

If you aren't already doing so, you should be using a centralised storage area for your sessions (Figure 1). Memcached can easily be used for this task by using the built in session save handler in the PECL memcached extension. If you want the ability to restart the

**Listing 1**

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');
console.log('Server running at http://127.0.0.1:1337/');
```

**Listing 2**

```
<?php
ini_set('session.serialize_handler', 'msgpack');
ini_set('session.save_handler', 'memcached');
ini_set('session.save_path', 'sess1:11211');
?>
```

### Listing 3

```
<?php
require 'vendor/autoload.php';
$memcached = new Memcached();
$memcached->addServer('localhost', 11211);
$handler = new Lboy\Session\SaveHandler\Memcached($memcached);
session_set_save_handler(
    array($handler, 'open'),
    array($handler, 'close'),
    array($handler, 'read'),
    array($handler, 'write'),
    array($handler, 'destroy'),
    array($handler, 'gc')
);
register_shutdown_function('session_write_close');
session_start();

$_SESSION['user']['username'] = $_GET['username'];
?>

<!-- Javascript -->
<script src="/components/jquery/jquery.js"></script>
<script src="/components/underscore/underscore.js"></script>
<script src="/socket.io/socket.io.js"></script>
<script>
  $(document).ready(function() {
    var socket = io.connect();
    var template = _.template($('#js-news-template').html());
    socket.on('news', function (news) {
      var element = template({news: news});
      $(element).hide().prependTo('#js-news-container').slideDown();
    });
  });
</script>
<script type="text/template" id="js-news-template">
  <p class="well"><%- news %></p>
</script>
```

server storing your sessions without losing data, then Redis [1] is a good bet. Either way, a centralised session storage enables you to load balance your application across multiple webservers. It also enables you to share session data with applications built with other programming languages.

However, there is a small problem of parsing session data. Below shows the default serialisation format of a PHP session:

```
not|a:2:{i:0;s:4:"easy";i:1;a:1:{s:2:"to";s:5:"parse";}}
```

It may look like you can use string manipulation to parse it, but there could be edge cases which are tricky to solve. It would be nice if the session was serialised in the much-loved JSON format:

```
{"this":{"is": "easier", "to": "parse"}}
```

Much better. It is fairly easy to write your own session serialiser which will convert whatever you store in $_SESSION to JSON format, see my version [2].

Alternatively, you might want to consider msgpack [3] which can be configured to serialise sessions, as in Listing 2. It also demonstrates how to use memcached as the session save handler.

There is a third-party library available for Node which can serialise and deserialise msgpack [4], available in npm [5]. We shall now look at building the news feed using Node and incorporating this into a PHP application.

## PHP App

The PHP application simply handles user logins and sessions. Sessions are stored in memcached, but could quite easily be stored in redis. Listing 3 shows snippets of the simple one page application (for the full source go to [6]).

The first line of the script includes the Composer [7] autoloader file, meaning that all dependencies are autoloaded automatically, obviating the need for separate include or require lines *[Editor's note: for more on Composer, see Jefersson Nathan de O. Chaves' article in the May 2013 issue]*. In this case, the session save handler is the only piece of external code which is required. However, it is very easy to include any third

### Listing 4: The nginx configuration file

```
upstream node {
  server localhost:3000;
}

server {
  listen 8080;
  server_name php-node-demo.localhost;
  root /home/lee/public_html/php-node-demo;
  index index.php;

  location / {
    try_files $uri $uri/ /index.php?$args;
  }

  location ~ \.php$ {
    include fastcgi_params;
    fastcgi_index index.php;
    fastcgi_pass  unix:/var/run/php5-fpm.sock;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
  }

  location ~ /socket.io {
    proxy_pass http://node;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
  }
}
```

party packages available on packagist or elsewhere at a later date by using Composer which will also be automatically added to autoloading. Lines 3 and 4 set up a connection to memcached, whilst the session save handler is initialised and registered with PHP in lines 5-14. Users are logged in simply by specifying the username as a GET parameter, though in a real system this should be replaced with a more functional authentication system.

On the JavaScript side, we use some third party libraries: jQuery, underscore.js and the Socket.IO client, which is the browser part of the Node module we will be using to push data to the browser. The *io.connect* method call creates a connection to the Node application, which could use one of a number of transports such as websockets or Ajax depending on browser support. The *socket.on* method sets up an event handler which will render a news item on the page whenever a 'news' event is triggered by the server side.

Note: The application uses Composer for installing the session save handler and Bower, a package manager for installing client-side assets [8]. To install Bower, run 'npm -g install bower'. Then to install the assets, do 'bower install'. Alternatively, update the script tags and stylesheets with local versions.

The application is served using nginx (see Listing 4 for the nginx configuration). Lines 1-3 define an upstream server which runs the Node app. Any request ending in .php will be sent to php-fpm (lines 15-20), whilst any requests with /socket.io in the URL will be passed to the Node app (lines 22-27). Lines 25 and 26 tell nginx to support protocol switching and enable websockets to be proxied to Node. This means that the PHP app and Node app are both run on the same port, as far as the client is concerned.

### Listing 5

```
io.set('authorization', function(handshake, callback) {
  var cookies = cookie.parse(handshake.headers.cookie);
  mcClient.get('sessions/' + cookies.PHPSESSID, function(error, result) {
    if (error) {
      callback(error, false);
    } else if (result) {
      handshake.session = JSON.parse(result);
      callback(null, true);
    } else {
      callback('Could not find session ID ' + cookies.PHPSESSID + ' in memcached',
                                                                          false);
    }
  });
});

io.sockets.on('connection', function(socket) {
  var session = socket.handshake.session;
  sockets[session.user.username] = socket;
});

function getNews() {
  redis.blpop('news', 0, function(err, data) {
    news = JSON.parse(data[1]);
    if (typeof news.to !== 'undefined') {
      if (typeof sockets[news.to] !== 'undefined') {
        sockets[news.to].emit('news', news.content);
      }
    } else {
      io.sockets.emit('news', news.content);
    }
    process.nextTick(getNews);
  });
}
```

## Node.js App

Snippets of the Node.js app are displayed in Listing 5 (see [9] for the full source), and simply handles any client requests for the news feed. Socket.io is a library which supplies a single API for performing WebSocket communication between client and server [10]. It also supports graceful fallback when WebSockets aren't available in the browser, and other useful features in a messaging protocol such as heartbeats, timeouts and disconnection support which is not supported out of the box with the HTML5 WebSocket API.

The first snippetconfigures authorization for requests that Socket.io receives. When a new request comes in, it will parse the cookies in the request and attempt to retrieve a memcached key which corresponds to the value of the PHPSESSID cookie (the default name for the PHP session cookie). If found, it will store the parsed value of the memcached key in data.session which can be accessed later on.

The next snippet configures what should happen when Socket.io triggers the 'connection' event. This event is triggered during the initial connection from the client, once authorised. The session data that was previously retrieved from memcached can now be referenced via the socket.handshake variable. When the client connects, the socket instance is associated with the client's username so that messages can be sent to individual users.

The last snippet contains a function to check for new news items in a redis queue. It uses the BLPOP command in redis which means if the queue is empty it will block until some data is appended to the queue before popping it. When some data can be popped off the queue, it is parsed as JSON before determining if the content should be sent to every connected client or just a single user. The news item content is sent to the correct socket by calling the emit() method on the socket which was associated with the user previously in the connection event handler.

Finally, Node's process.nextTick() method is called with the getNews function as an argument. This means that the getNews function will be called the next time the event loop runs and will continue to check the redis queue for data until the application is stopped.

To install, use npm to download the required dependencies. Then run the application with 'node app.js'. You should now be able to open a web browser and navigate to http://localhost:8080/?username=bob and see the news feed application. Now open a second browser tab or window with a different username, for example http://localhost:8080/?username=sally.

## Updating the Feed

Updating the feed is simply a case of pushing a new news item into the queue. Connect to the redis CLI interface using the command 'redis-cli'. This is an interactive shell for redis, allowing you to send commands to the server directly. The code sample below shows how to push to the queue:

```
rpush news '{"content": "Testy test", "to": "bob"}'
```

In the web browser window you opened for user bob you should see the news item slide down from the top of the page. Alternatively, you can push news out to both bob, sally and any otherall connected clients by excluding the "to" parameter, as follows:

```
rpush news '{"content": "Everyone should see this"}'
```

In a real application you would push data into the queue from PHP, using for example the php-redis extension [11] or predis [12].

## Conclusion

This is just a simple example to demonstrate how Node can be integrated with your PHP application. There are a few limitations in the implementation. For example, it will only remember one connection from each user. Therefore if a user has multiple tabs or windows open to the news feed, only one page will update. This can be resolved by storing an array of sockets per user, and keeping track of each connection and disconnection to add or remove the sockets from the array. It is left to the reader to implement a better solution. It also showed off tools such as Composer and Bower which I highly recommend you consider using in your applications.

**Lee Boynton** is a developer based in Hampshire, UK, with a particular interest in real-time applications such as instant messaging and activity streams. He has knowledge of server side development and administration as well as frontend development. He works at local company Symbios Group and is also a member of PHP Hampshire for who he helps organise events.

## References

[1] http://redis.io/

[2] https://github.com/lboynton/memcached-json-session-save-handler

[3] Alternative session serialiser, msgpack: https://code.google.com/p/php-msgpack/

[4] Node msgpack encoder/decoder: https://npmjs.org/package/msgpack-js

[5] Node Package Manager (npm): https://npmjs.org/, Contains a large number of modules which can be used in your application with just a simple 'npm install <package>'

[6] https://github.com/lboynton/phphants-march-php

[7] http://getcomposer.org/

[8] http://bower.io/

[9] https://github.com/lboynton/phphants-march-node

[10] http://socket.io/

[11] https://github.com/nicolasff/phpredis

[12] https://github.com/nrk/predis

More than just a "second screen"

# HTML5 Now: Mobile First

As pixel-crafters, we now have to design and create for not only many types of devices, but (more importantly) many different device sizes. While there are many approaches to giving a good experience on different devices, most of the time, we start to design on the screen we know: the desktop. What about starting with mobile first?

by Frédéric Harper

The Problem: Too often, when we are building websites or web applications for our customers, the mobile part comes last. Sometimes, it's not even in consideration, or merely an afterthought. With mobile usage continuing to grow, we cannot let the mobile experience be the poor child. As Luke Wobleski wrote in his book, Mobile First [1]: "Fundamentally, there's just one World Wide Web, but it can be experienced in different ways on different devices." How often is it that you go to a website on your mobile, as on your desktop, and can clearly see that the effort was put on the desktop version first. Perhaps you even have the impression that you visited two totally different sites? In my own opinion, it happens far too often! One solution to give an amazing experience on all devices is to start with the mobile experience – that is, Mobile First.

## Mobile First

Mobile First is a simple but powerful idea that Luke Wobleski thought up nearly three years ago and shared on his blog [2]. As you may already understand, the philosophy behind mobile first is to start ... with mobile platforms first. Why not begin our design and development process with the smallest device our audience will use. There are four pillars that we'll consider when doing this:

1. The growth of mobility
2. Minimizing the constraints
3. The capacity of mobile devices
4. The context of usage

Let's explore each of those, after which you should see that the idea of starting with the mobile site first – even if it's hard at the beginning, – make total sense.

## The growth of mobility

There is nothing you can do about it: by 2014, mobile internet users will have outgrown desktop users [3]. In the USA, 25 % of Internet users are mobile only [4]: that means they don't use the desktop at all! To provide a more tangible example, 40 % YouTube views now come from mobile, an increase of 300 % in the past year [5]. This is amazing, especially coming from one of the biggest sites on the Internet ...

Just to be sure you understand that you can't ignore mobile anymore, here are some numbers from 2012: there were 371,000 babies born per day, 378,000 iPhones sold per day, and 700 000 Android devices activated at the same time [6]. Crazy, I know!

## Minimizing the constraints

As much as we love using our mobile devices, there do come with constraints. As developers, we need to keep these in mind, so why not think about them right from the beginning?

The first, and more frustrating one: low signal, or worse, no signal at all. It's a reality that there are still places where you can't get a phone signal, or can't find a wifi connection. You need to think about those users, so if it's possible, let them use your application or site offline.

Let's talk about data plans. Depending on where you are living, it can be really cheap. Personally, I'm in Canada, and let's just say that we aren't as lucky as our neighbors in the USA. Do I really need to download the high resolution picture you added to your website that you will resize within the browser anyway? How about this oh so wonderful background music?

Don't forget pressing buttons and "clicking" links with your fingers: Smartphones are touch devices, and you need to design your application accordingly. It could be really frustrating to always have to zoom in, zoom out, scroll up, and down to finally touch the wrong option ...

### The capacity of mobile devices

Mobile devices may have their constraints, but they also have advantages, like capacities that traditional devices don't have. Think about the GPS: with the geolocation, you are able to find the device's position, and save the user some time. How frustrating is it for a lazy user like me to have to find the address where I am right now (hey, I may be on the corner of a street I don't know well), type it, and see if the site can give me accurate information related to my position. Of course you can do this on the desktop, but unless you have a GPS on your laptop, it may not be as precise (as it will use your Internet connection).

The fact that most mobile devices are touchscreen opens up to a lot of scenarios of creative ideas you can have. Add this to the fact that you can detect the orientation of the device with the accelerometer, and you can add new features or save time for users. Things you wouldn't be able to do on most desktops or laptops.

There are many features with smartphones that you don't have on the desktop, and it's just the beginning. You should think about using those hardware capabilities to make the experience even better.

### The context of usage

I forget where, but I once read that there are three motivations to use your smartphone:

1. I'm multitasking
2. I'm bored
3. I'm local

Taking into consideration those three motivations, it's important to remember that use of a mobile device doesn't always mean the *owner* is mobile. I don't know about you, but when I'm watching television, I quite often have my smartphone with me, and I like to use it (for example) to find information on a specific actor. Does that mean that I need to have a minimal experience? In this case, I may need the same information as if I was in front of my 27 inch screen.

On the other hand, that also means it's possible your mobile user will be there because he needs to do something now, like modifying a document on the fly; or because he is bored, like playing games; or because he needs local information to find a restaurant near the convention center, as an example. Too often, we think about mobile experience only for the local purpose.

### So, how does it work?

By starting the process with mobile in mind, you can approach all those pillars easily. You can minimize the constraints by thinking about them right now: as for the desktop version, no worries, your site will already be bulletproof. Instead of giving the same experience on mobile, you will take into consideration the unique features of a phone: the desktop user will have the same experience he had in the past, and the mobile one will be blessed with new features. As for the context of usage, does that really mean that the desktop users don't focus on those three we mentioned before? Of course not, but your site will be ready to take care of all these. By starting with mobile, you'll become more experienced, save costs to your customers (as it will be more than just an afterthought) and, most importantly, provide a great mobile experience.

More important than all of these points is the content. Since you have less screen space, you'll need to focus on what is the most important: both for the users and your business. It's easy to add a lot of text, pictures or animations when you have plenty of space, but what about on a smaller device? It will force you, and the site owner, to really focus on what is important – and for the rest, no worries, you'll be able to add the extra on the big-screen version. That will also improve your non-mobile design too. 'Content first' is the key to a successful mobile approach.

Of course, this approach has some disadvantages. You'll have to rethink how you are working, and it won't be easy at the beginning. You'll need to understand a lot more the needs of your customers (and we know they may not always really know what they really need) or the audience you are targeting with your website, as you'll need to focus on the most important aspects.

Ultimately, you won't fail just because you don't do the mobile experience first, but by starting with the smallest device, you have more chance of making a successful one. It's all about redefining the mobile experience, creating innovative experiences, responding to users' needs, prioritizing content, and preparing our site for future opportunities.

**Frédéric Harper** is Senior Technical Evangelist at Mozilla, he share his passion about the Open Web, and help developers be successful with Firefox OS. Experienced speaker, t-shirts wearer, long-time blogger, passionate hugger, and HTML5 lover, Fred lives in Montréal, and speak Frenglish. Always conscious about the importance of unicorns, and gnomes, you can read about these topics, and other thoughts at outofcomfortzone.net.

### References

[1] http://www.abookapart.com/products/mobile-first

[2] http://www.lukew.com/ff/entry.asp?933

[3] http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/

[4] http://www.trinitydigitalmarketing.com/the-rise-of-mobile-infographic

[5] http://crave.cnet.co.uk/software/google-is-now-a-mobile-first-company-execs-say-50009727/

[6] http://www.lukew.com/ff/entry.asp?1506

Bundles of fun

# Getting started with Symfony Standard Edition

What can you do with Symfony2? Tom van Looy gives an overview of the component-based framework.

by Tom van Looy

What is Symfony? The first important thing to say about Symfony [1] is that it's not just another framework. Symfony is a project that tries to provide a sort of PHP middleware by developing a reusable set of standalone components. There is a lot to say about the Symfony2 components and the ecosystem around them. Over the last few years, big projects like Drupal, ezPublish, phpBB, Laravel and many more have started adopting these components to solve their common web development problems.

Based on these components, Symfony2 can also be a full-stack web framework on its own. As you probably don't want to spend time installing and configuring components and devising a good directory structure yourself, Symfony2 came up with distributions: simple, easy-to-install and packaged solutions designed to meet specific user needs. A distribution is made up of Symfony2 components, a selection of bundles that build upon the components, and has a certain directory structure with a default configuration.

For example, distributions are being built around content management, like the Symfony CMF [1], and e-commerce, like Sylius [2] and Vespolina [3]. However, the recommended distribution when starting a new project is the Symfony Standard Edition, which contains the most common bundles and comes with a simple configuration system.

## What's inside the Standard Edition?

The fundamental principles of Symfony2 are centred around the HTTP specification. It is a Request/Response framework. If you think in Model-View-Controller (MVC) terms, Symfony2 is only providing the tools for the Controller and the View part, although tight integration does exist for popular object-relational mappers (ORM's). Hence, the Standard Edition comes with Doctrine2 [4] for the Model part. Whether you want to exploit the full ORM or just keep yourself to the DBAL is up to you.

Doctrine is plugged into Symfony via a bundle. Bundles are first class citizens in Symfony, by which I mean that in Symfony everything is a bundle. Even the core framework functionality is a bundle. When you start a new project, the first thing you will do is create a bundle to harbour your own custom code.

Besides Symfony2 and Doctrine2, the Standard Edition comes with Twig [5] as the only configured templating engine. It also comes with Swiftmailer [6] as a library for sending emails, uses Monolog [7] as a logging library, has Assetic [8] for asset processing, and has an excellent Profiler bundle that enables profiling functionality and a web debug toolbar.

Mailing, logging, database access, and anything that accomplishes a specific task is a typical service. To make services available inside your application, Symfony2 uses a service container. The container standardizes and centralizes the way these services are constructed in an application. Through the service container, you will find yourself using these services very easily. The service container abstracts away how a service is constructed and configured for you.

The Standard Edition also enables annotations for everything. It relies on Doctrine Common as a PHP annotation implementation. Since Doctrine ORM and DBAL also rely on Doctrine Common, this is rather convenient. For those that don't know annotations, they are just classes that allow you to extend the functionality of your code from within the doc block comments of your class files, on properties and methods. Code is annotated by adding metadata inside the doc blocs. This metadata will be used for configuration purposes. It is easy to use and allows you to define everything in the same file. If you don't like this concept, you will be happy to hear that annotations are completely optional. Symfony2 is flexible enough to let you accomplish the same thing with XML, YAML, or even plain PHP. But, there are some very powerful annotations available that can make your life as a Symfony2 developer a lot easier.

In versions prior to Symfony Standard Editon 2.3, the bundles JMSDiExtraBundle, JMSSecurityExtraBundle and JMSAopBundle enabled dependency injection, security configuration etc. with annotations. These bundles were dropped from the Standard Edition 2.3 because they are not MIT/BSD licensed. They are covered by the open-source Apache 2.0 license, which allows you to use them freely and they can be enabled again very easily. Since these bundles demonstrate the power and simplicity of annotations, let's see some examples anyway (Listings 1 and 2).

### Listing 1: Defining a service with the @Service annotation

```
use JMS\DiExtraBundle\Annotation\Service;
use JMS\DiExtraBundle\Annotation\InjectParams;
use JMS\DiExtraBundle\Annotation\Inject;

/**
 * @Service("some.service.id")
 */
class SomeService
{
  /**
   * @InjectParams({"dependency" = @Inject("other.service.id")})
   */
  public function __construct($dependency)
  {
    // ...
  }
}
```

The code in Listing 1 allows you to use the service by requesting it from the service container like so:

```
$container->get("some.service.id");
```

In Listing 2, when you submit a new blogpost, you have to use a POST method on */blogpost/create*, and you have to be authenticated with a user who belongs to *"ROLE_WRITER"*. The template that will be displayed on success is the Blogpost's *show.html.twig* template. This is just an example. The default template that would be used when we omit the *@Template* annotation would be *create.html.twig*.

Symfony2 also has a very powerful Event Dispatcher component. This component basically implements the Observer pattern. You can hook into existing

### Listing 2: Routing, templating and security

```
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route,
    Sensio\Bundle\FrameworkExtraBundle\Configuration\Method,
    Sensio\Bundle\FrameworkExtraBundle\Configuration\Template;
use JMS\SecurityExtraBundle\Annotation\Secure;

class BlogpostController
{
  /**
   * @Route("/blogpost/create")
   * @Method("post")
   * @Template("AcmeDemoBundle:Blogpost:show.html.twig")
   * @Secure(roles="ROLE_WRITER")
   */
  public action createAction()
  {
    // ...
  }
}
```

events, for example HTTP kernel events like "kernel.response" and "kernel.terminate", or you can create events for your own application: for example a *"blog-post.create"* event, which you can extend later on.

For working with user input, the Form component provides tools for defining HTML forms and rendering and mapping request data to related object models. It's one of the most complex Symfony components, but a very powerful one when you learn to master it. Furthermore, the Form component integrates nicely with the Validation component, for validation of object models.

The form component is also well integrated with Twig, for the rendering of forms. There is a lot to say about Twig. With features like template inheritance and automatic output escaping, it's one of the most modern and flexible templating engines around. Like a lot of things in Symfony, Twig builds upon the experiences learned from other, even non-PHP, web development frameworks.

Trying to cover every bundle included with the Symfony Standard Edition and explaining every aspect of it is far beyond the scope of this article. It might seem hard to get started with all of this. But, to make it all very easy for beginners, Symfony Standard Edition comes with an example project and a code generator bundle that helps you get started. Starting a project with Symfony Standard Edition is also very easy, just run the command:

```
php composer.phar create-project symfony/framework-standard-edition
                                                      myproject 2.3.0
```

### Hello Symfony!

After running the create-project command, you are ready to create a bundle for your own project:

```
php app/console generate:bundle --namespace=Wpm/HelloBundle --format=yml
```

This command will create a bundle for you in *src/Wpm/HelloBundle* and generate your first controller in *Controller/DefaultController.php* of the bundle directory. The controller has an *indexAction()* method. In this example, I chose YAML as the default configuration format. But remember that this also works with XML, PHP or annotation. Because I chose YAML, the routes to the controller actions will be defined in the file *Resources/config/routing.yml*. A route */hello/{name}* was already provided for the *indexAction()*.

The *{name}* parameter in the route is passed as an argument to the controller action. After doing some work in the controller, for example fetching database records, a Twig template is rendered. In this case, the template *Resources/views/Default/index.html.twig* will be used.

In the Twig file we can print the *$name* parameter that we passed to the render function. Twig has the ability to loop over arrays, has support for if then else constructions, and has filters for things like date formatting:

```
Hello {{ name }}!
```

When pointing a browser to */hello/Symfony*, the new *WpmHelloBundle* will print "Hello Symfony!". This is a very basic example to give you an impression of how things work. The Symfony documentation builds upon this example and shows you a lot more that you can do.

**Listing 3**

```
namespace Wpm\HelloBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;

class DefaultController extends Controller
{
  public function indexAction($name)
  {
    return $this->render(
      'WpmHelloBundle:Default:index.html.twig',
      array('name' => $name)
    );
  }
}
```

### Going further

Symfony2 dependencies are managed with Composer [9], the new PHP packaging system. Packages live inside a package repository. You can make Composer work with your own private repository, or just use the main Composer repository called Packagist. Packagist aggregates all sorts of PHP packages that are installable with Composer. These packages can be searched from Packagist. Packages can contain anything, and are definitely not limited to Symfony2. If you want to extend your Symfony2 project with specific functionality, chances are that someone already created a package for it that fits your needs.

**"Twig has the ability to loop over arrays, support for if then else constructions and filters for things like date formatting."**

Most of Symfony2 specific bundles are also packaged for Composer and searchable on Packagist. A useful resource for finding Symfony2 specific bundles is the website Knp Bundles [11]. It searches the web for bundles and assigns scores to them, based on parameters like the number of followers on GitHub, if there is a README file present, etc. If bundles are packaged, they can be installed just like any other Composer package. Two of the most popular bundles are FOSUserBundle [12] and SonataAdminBundle [13].

The FOSUserBundle adds support for a database-backed user management system in Symfony2. Users can be stored with Doctrine ORM, MongoDB/CouchDB ODM or Propel [14]. The bundle also provides features like registration support, with an optional confirmation per mail, and password reset support. It does not provide an authentication system of its own but relies on the core SecurityBundle for this instead.

"It is easy to override any part of a bundle, a process well covered in the Symfony2 documentation."

SonataAdminBundle is a bundle to generate robust user-friendly administration interfaces. You basically tell it about your model, which can be Doctrine ORM, MongoDB ODM [15] or PHPCR [16]. Sonata will activate an entire create-read-update-delete (CRUD) flow with forms, lists and everything you need to manage your model. You can override any of the defaults to make it fit your specific needs. The bundle relies on some other bundles like SonataBlockBundle [17] for a block system and KnpMenuBundle [18] for menu management.

These bundles are very useful, so if you need a feature for your website, consider using and extending them before rolling your own custom solution. It is easy to override any part of a bundle, and this is a process that is well covered in the Symfony2 documentation.

## Getting help

The first version of Symfony2 was released on July 28, 2011. Last month, version 2.3 was released. This is also the first Long Term Support (LTS) version, which means it will get a support period of three years, plus one extra year for security fixes. Since the release of 2.3, backward compatibility will be kept at all cost. If not possible, the change will be scheduled for the next major version. If licenses are important for you, you may be happy to read that the 2.3 LTS only depends on MIT/BSD [19] licensed code.

A lot of effort is put into writing and maintaining documentation. There is a reference book with documentation that covers the basics of Symfony2, ideally for getting started. If you want to delve deep into the components, you will find that each of them has their own set of documentation. There is also a cookbook where you can find specific solutions for specific needs. All this documentation can be found on the Symfony website [20].

Also worth mentioning is that Symfony has a conference called "Symfony Live" [21] that is held multiple times a year, all over the world. The talks that are given at Symfony Live are published afterwards on the SensioLabs Youtube channel [22].

SensioLabs [23] is the commercial company behind Symfony. If you are interested in training, paid support or certification, this is one of the companies that you can contact. The founder of SensioLabs and project lead of Symfony, Fabien Potencier, regularly writes on the Symfony blog [24] and on his own blog [25].

**Tom van Looy** is a support and maintenance worker at Intracto Digital Agency (http://www.intracto.com). We mainly focus on Drupal, Symfony2 and Magento. But we don't shy away from working in other codebases as well. Tom tweets as @tvlooy and can be contacted by mail at tom@ctors.net.

## References

[1] http://symfony.com

[2] http://cmf.symfony.com

[3] http://www.sylius.com

[4] http://vespolina.org

[5] http://www.doctrine-project.org

[6] http://twig.sensiolabs.org

[7] http://swiftmailer.org

[8] http://github.com/Seldaek/monolog

[9] http://github.com/kriswallsmith/assetic

[10] http://getcomposer.org

[11] http://www.packagist.org

[12] http://knpbundles.com

[13] Packaged as friendsofsymfony/user-bundle at https://github.com/FriendsOfSymfony/FOSUserBundle

[14] Packaged as sonata-project/admin-bundle at https://github.com/sonata-project/SonataAdminBundle

[15] http://propelorm.org

[16] http://docs.doctrine-project.org/projects/doctrine-mongodb-odm

[17] http://phpcr.github.io

[18] https://github.com/sonata-project/SonataBlockBundle

[19] https://github.com/KnpLabs/KnpMenuBundle

[20] http://en.wikipedia.org/wiki/MIT_license, http://en.wikipedia.org/wiki/BSD_license

[21] http://symfony.com/doc

[22] http://live.symfony.com

[23] http://www.youtube.com/sensiolabs

[24] http://sensiolabs.com

[25] http://symfony.com/blog

[26] http://fabien.potencier.org

# Trust us, our ad-servers are secure

by Arne Blankerts

**Online adverts aren't just annoying: They open up a number of technical issues too, says thePHP.cc's security expert.**

Many people wholeheartedly hate them: Advertisements. Be it a commercial break in their favorite TV show, in-game ads or classical print campaigns in newspapers and magazines. Despite the serious dislikes, advertising works in many occasions, be it by directly sparking an interest in the shown product or by shaping a new image for the brand in question. Bigger posters, interactive bus-stop commercials or split-screen spots during sport events – there seems to be no end to new and more aggressive types of marketing which sometimes even push the actual main content into the background. And of course what worked for the so-called real world was soon adapted to the internet.

There is just one tiny problem: The users found a way to fight back! Almost all modern browsers now come with popup suppression enabled by default, support the blocking of cookies or have plugins like Adblock or Ghostery to fully cleanse the website from any type of tracking or advertising. And it seems to work, as a recent campaign by various german online publishers reveals. These publishers asked their readers to disable the adblocking software, arguing they would cause more harm than good and result in serious losses of income.

Feedback to that campaign was, as to be expected, mixed. Many people explained they merely use adblocking as a means of self defense: adverts, they said, were too aggressive in color, distracting, badly placed or too fat for mobile devices on a low bandwidth connection. Quite valid reasons to my taste – and even the publishers admitted they made sense. A shockingly small amount of people replied that they do not explicitly block ads, but merely do not allow JavaScript to be executed by default; and, as a side-effect, are killing pretty much all those tags that require JavaScript to adjust the HTML DOM within the browser to display the ad or do the statistical tracking.

So why would those users – originally being only a relatively small group anyway – disable JavaScript? The easy answer would be to simply name them paranoid geeks. But then, as another side-effect of the campaign, adblock plugin downloads doubled or even tripled during said campaign, and donations to the developers went up too. So maybe some of the people are just fed up with commercials taking over their browser, and the lack of privacy because of all the tracking and data mining? Or perhaps, the truth is that they are – like me to some extent – paranoid enough to not trust these ads to execute JavaScript (or worse, fire up the Flash plugin). Because, what many don't know is that ad serving is quite often a recursive pro-

## Bio

Arne Blankerts consults for thePHP.cc, solving IT problems long before many companies realize that they even exist. IT security is his passion, which he pursues with almost magical intuition, creating solutions that always bear his hallmark. Companies around the world rely on his concepts and Linux-based system architectures.

cess: While the webmaster may have only embedded the ad-tag from his adserver of choice, the actual content delivered may have been served by a completely unrelated machine to download additional code from or after the spot on the website had been re-sold various times. The latter happens because, in case there is no current booking for a slot, an adserver is not expected to deliver a blank image, but merely fall back to some other ad used instead.

While this might still create some small revenue for the website owner, the recursive lookup of hostnames as well as downloads of additional JavaScript and media files is quite a performance drain for the enduser's device. And it reduces the trustworthiness of the actual content, as nobody can really tell where the script has come from by the time it made its way to the browser. So even if the company running the adserver you are embedding into your website can claim – hopefully correctly – that their servers are safe, they hardly have any say in the content they deliver.

A related – though not necessarily obvious – problem is the actual embedding of the ad tag. While it simply won't work with JavaScript disabled on the browser level, there are some other technical issues as well. People claim the fact that the XML version of HTML (aka XHTML) was not exactly successful in terms of market share as well as browser support is partly related to the way advertisements had to be embedded into websites. Most services require adding a *<script>* element into the *<head>* section of the document as well as an inline function call wherever an ad should appear on the page. What seems convenient from a development perspective, however, is technically a really bad choice: First of all, inline scripting makes assumptions on how the browser is processing the html, and secondly it forces the browser to stop the parsing of the markup and switch into executing JavaScript – just to switch back to parsing afterwards. The problem is, when using an XML-Parser rather than an SGML-Parser, many things JavaScript allows for, like document.write(), cannot be used and thus do not work. As a result, the inline script might not work as expected, or even not be executed at all.

“Instead of fixing the real problem the adserver vendors merely changed the scriptcode to work without the problematic functions.”

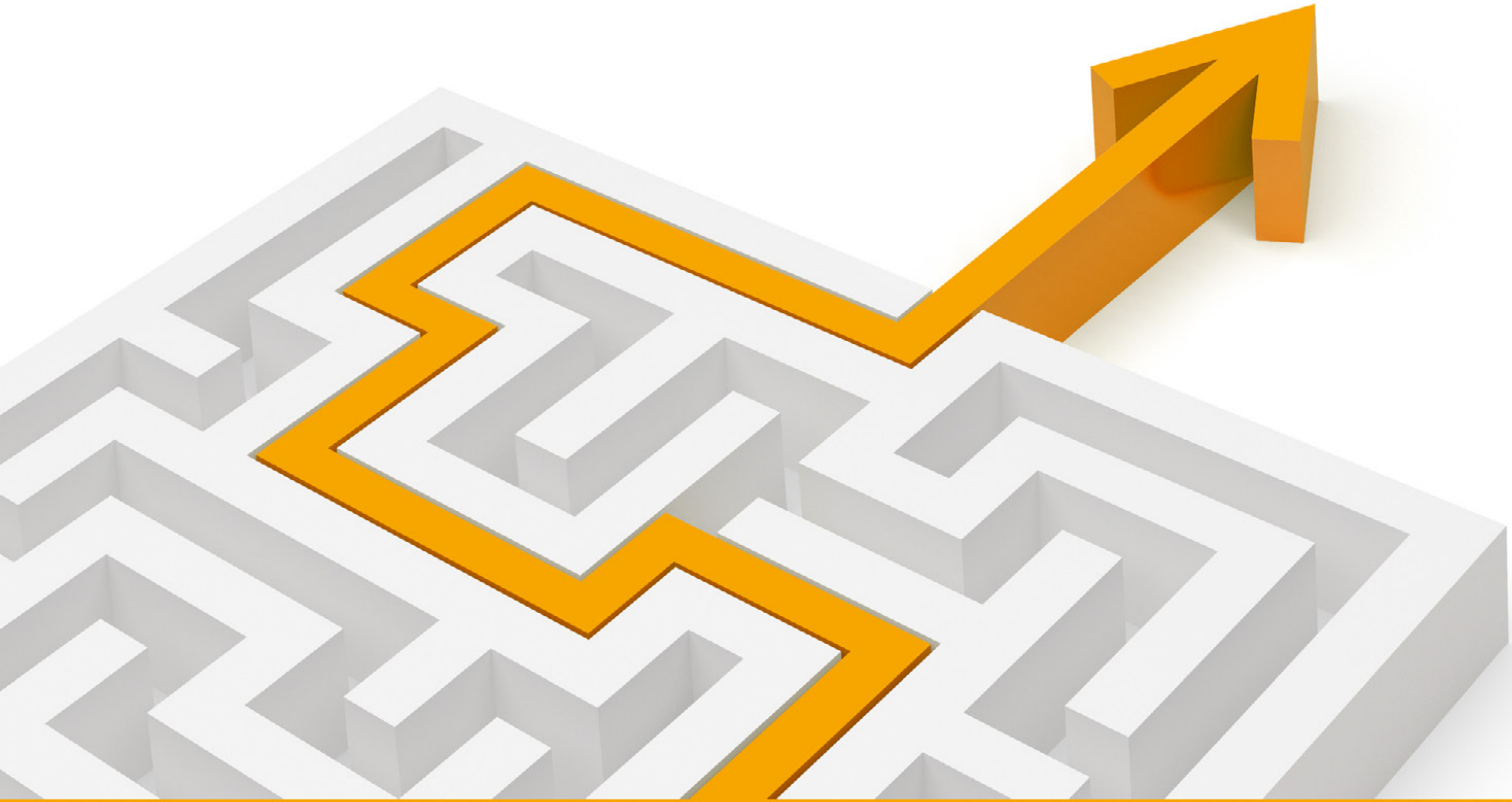
Back in the days when XHTML was defined, the adserver companies at first ignored these problems, telling everyone who complained to switch back to HTML 4 or force the browser into compat- or quirksmode to stop it from using the XML parser. After a while, instead of fixing the real problem – the use of inline scripting – the adserver vendors merely changed the scriptcode to work without the problematic functions, making it work in HTML and XHTML variants. And while I guess it seems like a "good enough" solution at first, I already back than was waiting for the problem to return. And, voila, in 2013 the problem is back.

In an attempt to block XSS (Cross Site Scripting) attacks from being successful, Mozilla, Apple and Google implemented a new security mechanism named "Content Security Policy", abbreviated CSP, into their respective browsers. The idea behind the CSP is simple but highly effective: Only allow external JavaScript sources from whitelisted domains (individually defined by the webmaster) and disable all inline scripting. Assuming an attacker won't be able to alter the list of allowed domains at the same time as he's injecting code into the content, it would no longer be possible to inject active scripting into the HTML.

While that approach is technically fixing the XSS issues on the wrong end, it feels a bit like self-defense again: Since the backend developers obviously fail at escaping the output properly – despite the fact that XSS is decades old and every developer should know how to protect his or her website – the browser seems to be the last line of defense.

Of course, as with pretty much all solutions that try to fix things at the wrong place, the protection won't be 100 %: I can come up with multiple ways to work around these "limitations" already and I bet the bad guys will come up with even more in almost no time. But at least the adserver vendors are going to be forced to finally fix their adtags to work without inline scripting and without downloading additional code from unknown sources. Better than nothing.

sharing experience



**thePHP.cc • The PHP Consulting Company**

Poignring 24 • 82515 Wolfratshausen • Germany • +49.8171.428058 • team@thePHP.cc • http://thePHP.cc

# Data Modeling 104: De-normalization

by Cory Isaacson

**For web-scale performance, it's necessary to disrupt your beautifully normalized data model.**

This article continues the series on basic data modeling techniques. I will be referencing data modeling often in the future, as it applies to various types of DBMS engines and specific approaches for scaling your database. If it's not obvious already, I consider data modeling a fundamental for many aspects of making your database perform and scale the way you need it to; therefore consider these articles a reference foundation for many articles yet to come.

In the previous column, I covered *database normalization* techniques and examples, illustrating how you can improve the integrity and simplicity with this skill. Now that you have spent time learning about (and hopefully applying) this approach, let's go through the first detailed examples on *database de-normalization*.

**Why do you need to "de-normalize" your data model?**
After all the stress I put on proper *normalization* of your model, why would you want to *de-normalize* it? The answer is simple. Using database *normalization* you have eliminated redundancy and inappropriate relationships in your model. This is vital to ensure that your model design is correct , easy to understand, and as flexible as possible in meeting the needs of your application.

However, a fully *normalized* model does not always perform as fast as you would like, and sometimes is inconvenient to use given the number of joins that may be required for common functions. Therefore, as I touched upon in the last article, you *de-normalize* your model when required for *performance* and *convenience*. This will give you an even better data model, one that will perform the way you expect and be as natural as possible for your developers.

In general, *database de-normalization* means adding some redundant data elements or structures in order to match specific requirements of your application. There is no hard and fast rule you can apply as to when to *de-normalize* your data model, but I will cover some examples with the *Angry Shards* game data model to give you a feel for when and how to use this technique.

Bio

Cory Isaacson is CEO/CTO of CodeFutures Corporation. Cory has authored numerous articles in a variety of publications including SOA Magazine, Database Trends and Applications, and recently authored the book Software Pipelines and SOA. Cory has more than twenty years experience with advanced software architectures, and has worked with many of the world's brightest innovators in the field of high-performance computing. Cory has spoken at hundreds of public events and seminars, and assisting numerous organizations address the real-world challenges of application performance and scalability. In his prior position as president of Rogue Wave Software, he actively led the company back to a position of profitable growth, culminating in a successful acquisition by a leading private equity firm. Cory can be reached at: cory.isaacson@codefutures.com

## The Angry Shards Game Database

In the previous article, I extended the *Angry Shards* example database [1], resulting in **Figure 1**.

The model so far supports the ability to have a *player* play a *game*, resulting in a *player_game* for each instance of a game that the player plays. In addition I added the *player_game_round* to support one or more rounds for a single game, enriching the game to support multiple *round_level* values to make the game more challenging and interesting.

Now I'll extend the model in some new ways, features that can benefit from *database de-normalization*.

## A Game Leaderboard

A Leader Board is an extremely common, yet challenging feature to add to a database in an efficient manner. The purpose of a leaderboard is to provide the top scoring *player* list so any *player* can see how they compare to others. Usually the list is limited to a specific number of *player* entries, but in some more advanced game databases, it is desirable to have a Leader Board allow any *player* to see where they rank. For this first example, our leaderboard will be limited to the top 10 player scores in the database.

A typical approach to accomplishing a leaderboard is to utilize *aggregation* functions on the database. For example, in SQL we can do it with a query that looks like this:

```
SELECT player_id, MAX(player_score) AS TOP_SCORE
FROM player_game
GROUP BY player_id
ORDER BY TOP_SCORE
LIMiT 10
```

This certainly will do the trick, giving you a list of the Top 10 *player_id* key values and the related *player_score*. The output will look something like this:

```
player_id     TOP_SCORE
---------     ---------
101           4795
203           3850
557           3297
...
```

This is somewhat useful, but I doubt that users will know who is who when given just a *player_id* in the result. Let's modify the query like this to provide the *player screen_name*:

```
SELECT pg.player_id, p.screen_name,
  MAX(player_score) AS TOP_SCORE
FROM player_game pg, player p
WHERE pg.player_id = p.player_id
GROUP BY player_id, p.screen_name
ORDER BY TOP_SCORE
LIMiT 10
```

Now our output will look like this:

```
player_id     screen_name     TOP_SCORE
---------     -----------
101           joe             4795
203           mary            3850
557           sam             3297
....
```

Now this is more useful, we can easily display the *screen_name* with the results, far more meaningful. So what is wrong with this approach, and how could *de-normalization* help?

As the *player_game* table grows, this query will get *slower and slower*. In an earlier article, Why Databases Slow Down I covered the main *enemies* of database performance. The most important *enemy* was the *table scan*. In a *table scan* the table must be searched from top to bottom – every single row – in order to satisfy the query. This query would certainly cause a *table scan*, and further it must compare every *player_score* value and sort the values in order to return the result. This is required to identify the *MAX(player_score)* value for each *player_id*. While an index on *player_score* might help, by itself it won't lower the cost of the search much, since every row must be compared. These are extremely intensive operations for the DBMS engine to perform, and over time will certainly degrade in performance. Further, a query like this, if frequently run, could degrade the entire application.

There is a simple answer in this case, one that is already allowed for in our data model. We will use the *high_score* column on the *player* table, a simple

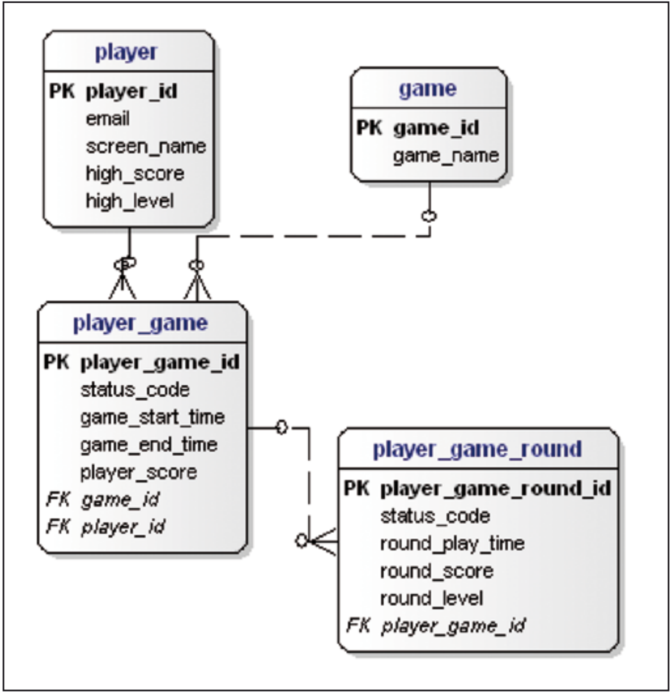


**Figure 1: The "Angry Shards" database as we left it last issue**

*de-normalization* to solve the problem. Here is the logic that is required:

- Whenever a *player* completes a *player_game_round*, increment the *player_score* in the *player_game* table. This can be done with a simple UPDATE statement in a relational database.
- At the same time, increment the *high_score* on the *player* table, again with an UPDATE for that purpose.

Listing 1 shows what the SQL statements look like (assuming the *player* got a score of 1857 in the current *player_game_round*, and a total *player_game* score of 3277), but you can use the language of any DBMS engine to do something similar.

### Listing 1

```
-- Insert the player_game_round row
INSERT INTO player_game_round (status_code,... round_score,...,
                                                    player_game_id)
VALUES
(10,..., 1857,..., 6093)
;

-- Update player_game, incrementing the player_score
UPDATE player_game
SET player_score = player_score + 1857
WHERE player_game_id = 6093
;
-- Update player, incrementing the high_score, only if this
-- score is higher than the last high_score recorded
UPDATE player
SET high_score = 3277
WHERE player_id = 7699
  AND high_score < 3277
;
```

Using this simple *de-normalization*, now we can add an index on *high_score*, and make a very efficient query to generate the leaderboard result:

```
SELECT screen_name, high_score
FROM player
ORDER BY high_score
LIMIT 10
;
```

This query, as long as the *high_score* index is in place, will avoid the *table scan*, and can easily retrieve the leaderboard with very little processing work. Further, it avoids the join and aggregate function used in the *normalize* example above. The main cost of this approach is with the extra UPDATE statements, but since they only are done for one *player* and *player_game* at a time, these operations are isolated and should not cause excessive locking on other concurrent *player* activity. This should show how *de-normalization* helps with both *performance* and *convenience*, making the job easier all the way around.

## Other Reasons to De-Normalize

There are other reasons to de-normalize, and involve more advanced techniques that I will cover in future articles. Here are some of the many problems that can be solved with judicious use of *database de-normalization*:

- Resolving so-called *self-joins*, where a data table has an implicit *parent-child* relationship. This is very common, especially in social network type applications (which of course includes game applications, making it a natural follow-on to this series of articles).
- Resolving *complex relationships*, especially across many table structures or lists.
- How and why to integrate *multiple types* of DBMS engines into your application. With the plethora of great DBMS engines today, there are often major advantages to utilizing the right combination of tools. At first it may not appear that using multiple DBMS engines is *de-normalization*, but in fact this solution fits well into this category.
- Modeling structures for NoSQL/NewSQL DBMS engines, where *de-normalization* is often far more prevalent as a requirement.
- Scaling your database efficiently, using *de-normalization* to minimize *distributed* operations. Definitely the motto of database scalability is a *shard-nothing* environment, with a minimum of *distributed* reads and writes. There are specific *de-normalization* techniques developed for accomplishing this.

## Wrapping it up

As you can see, *database de-normalization* is an important topic, with wide-reaching consequences both for performance and accessibility (i. e., *convenience*) of data access. This article covered one aspect of this technique, there are many more to come in future entries to the Scaling Big Data column.

## References

[1] The Angry Shards game, data model and graphics are copyright © 2012-2013 by CodeFutures Corporation and is used with its permission

# Scope and Purpose

by Ben Greenaway

**As developers, understanding our non-technical partners' needs is the best way to develop great code.**

It might be a brochure, a tool or a fully-fledged application that you last created with PHP. Whatever it was, chances are that it had a fairly broad spec. You probably extended the requirements again with repurposing and reuse in mind. And if you've been following recent trends, added a social media requirement, a mash-up opportunity or exposed a new, public API.

Over the past eight years the growth of application for PHP projects has similarly extended the design vocabulary of our clients. And at the same time, public end-users have equally grown in sophistication and understanding, allowing in turn the development of richer and more deeply engaging experiences across all supported browsers, device clients and APIs. It is a good time to be a web developer.

This story speaks to a simple fact, and it is one you should proudly be telling many times over. Our industry has matured, its tools and techniques now forming University degree programs and being written about far beyond our trade journals. A dozen years of Agile development has done us the world of good, and a new professionalism has bought – perhaps more than anything else – a much needed critical distance and appreciation of our output. Though sometimes it is not so obvious that we benefit from it.

Consider the following situation; an enthusiastic client or disruptive startup employs you to develop their application. They plan to 'buck the trend' and do something utterly unique for their field. Having spent the last year refining the plan and raising capital they begin by building a development team that you will lead. It is a brilliant idea and you love it. Your bosses are new to the web and bring an entirely new concept with them from their own industry, but are unfamiliar with our development protocols. The holders of the purse need their risks mitigated and to know that everything will work safely. To satisfy them, you begin to explain how your solution will work, reducing each complex process to simpler, familiar problems. Time spent explaining how things don't break if built a particular way consumes much of your early research and definition stage and larger project milestones begin to approach all too rapidly. Your enthusiastic team turns their attention to other matters, leaving you to lay down the code and begin commissioning hosts. You start spending that precious purse without having iterated the design even once. And from this point on you're back-hoofing it.

## Bio

Ben Greenaway has been a software developer for 18 years. After pioneering internet narrowcasting in the 90's and collaborations with Cyberia and Virtual Futures on web installation projects he led development on A/V installation projects for Pixar and Sony Entertainment and web applications for SMEs in Southern California before moving to the San Francisco Bay in 2010 and developing performance solutions for Dyn DNS and e-commerce APIs for Tzolkin TZO. He now lives in South London where he balances time writing and developing with the demands of an ever growing collection of retro home computers.

Eventually, the day does come when your broader appreciation of the project allows you to offer up new features, but they are likely met by your purse-holder's objections to changing the plan just now, and you are told to include them in the next version.

And the scenario applies just as much for a team lead on existing product as it does for project leads of legacy teams, too. In collaboration with your client or your executive, or even the lead on your current project, try to be spending your time on "how it could work" stories rather than "why it won't fail" ones. When you do, you are much more likely to develop additional features soon enough to include them and identify smart extensions to create additional value in your products. The vast majority of successful projects began this way, including PHP itself. And conversely, the best way to find out which code you should be writing for yourself is through discovering a purpose for its development that a client currently may have.

When you do next sit down to begin your next creation, remember that our new maturity is best embodied by taking this more understated approach to championing the brilliance of our coding. We need to be generous with our features and the time we will spend on their refinement, and no less outgoing with our design and the communication of it and its goals and potential for our collaborators. Doing so gives our code a shine all the more brilliant from its deeper reflections of our client's conceptual intent than any trendy torch shone blindly into eyes of startled users can.

We write great code when we have something great to build with it. Economists agree that economic recovery will be most sustainably led online and through our tech sectors. So when working with your new client or on a new project, advantage yourself and the rest of us of the full opportunity it presents. Engage with its intention and purpose, not just the tools and technologies you need to build it. When you do you will find the killer-apps and the richer, more engaging experiences which can move us all forward.

# Daily Scrums explained

by Steffan Surdek

## Your team's daily synchronization tool

Coaching agile teams, I get to see many daily scrum meetings and I hear many complaints from team about how ineffective and pointless their meetings are. This article will introduce you to the Scrum daily meeting and will present you with various ways to make them more useful to your team. Before I tell you what the meeting is about, let me begin by telling you that, contrary to popular belief, the daily scrum meeting is not a status report meeting. If your team merely stands around in a circle, each reciting what they did yesterday and what they will do today while looking straight at the Scrum Master, then you are the kind of team I am talking about!

The daily scrum is in fact about allowing team members to synchronize on a daily basis to coordinate their efforts for a sprint. Notice that when you build a sprint backlog, you identify the tasks and the number of hours for each of these tasks but you do not identify target end dates or build Gantt charts for dependencies. The daily scrum is how teams synchronize and communicate throughout the sprint.

Why should the meeting occur every day, at the same time and in the same location? Because sometimes routine can be your friend. What you really want is for John or Julie to walk into the office in their zombie-like state, automatically grab their coffee and find their way to the meeting. You want the meeting to become a habit for everyone.

### Who should attend?

Teams often ask me who should attend the daily scrum meeting. Team members are mandatory attendees, while the Scrum Master and Product Owners are optional attendees. Ideally team members will self-organize and run the meeting on their own, but initially the Scrum Master facilitates the meeting to make sure the team gets on track.

One team I collaborated with decided to name a different meeting facilitator every day of the week. This allowed the team to take full ownership of the meeting instead of the Scrum Master.

### Focus on synchronization

Sprint backlogs do not contain due dates so teams need some form of mechanism to work on their dependencies during a sprint. The most effective and useful way to run your daily scrum meeting is to focus the team on answering three daily scrum questions with a goal of synchronizing their efforts and making sure they will meet their sprint goal.

The not so subtle distinction between the daily scrum as status report instead of a synchronization meeting begins with how the team answers the following three questions:

Bio

Steffan Surdek is a senior consultant and agile coach at Pyxis Technologies. Steffan has worked in IT for over eighteen years in collaboration with many distributed teams around the world. In the last few years, Steffan was an agile trainer and coach in large companies such as IBM and the TD Bank Group. He speaks at many conferences and user groups about agility with distributed teams. Steffan is co-author of the book "A Practical Guide to Distributed Scrum" written in collaboration with the IBM Scrum Community. He blogs on his websites http://www.surdek.ca and http://www.provokingleadership.com.

What did I do yesterday?
What did I do today?
What are my blockers?

When team members talk about what they did yesterday, they provide cues to others with a dependency on their tasks. For a simple case, a developer pointing out all coding tasks related to a specific feature are complete signals to the tester that they can begin testing it.

When a team member talks about what they will do today, they are essentially making a verbal commitment to the team. Other team members will know what this person is working on and if they are focusing on work relevant to meeting the sprint goal. Similar to the previous example, knowing ahead of time what another team member is working on may help you better plan your day. For example, if a developer expects to finish the work on a feature today, a tester may decide to focus on finishing the relevant test cases for this feature.

## Task boards increase visibility

Just answering the three questions may not be useful if the team cannot picture their progress during the sprint. Looking at the sprint task board during the daily scrum meetings is a simple way to get this visibility.

Some teams manually build and uphold their sprint task boards (**Figure 1**) using sticky notes and tape on a wall or a rolling whiteboard they can move around, while other teams use electronic agile planning tools. Using the task board view of an agile planning tool may be more suitable for distributed teams.

I find that answering the three questions in front of a whiteboard while moving tasks and updating hours in real time helps make the progress come alive. As an alternative, some teams I know have a rule stating that team members need to update the task board at least five minutes before the start of the meeting.

In either case, I usually ask team members to go up to the task board and point to the tasks they are alluding to so the team understand what they are discussing. Having people point also helps the team make sure everyone is focusing on the most important user stories and shows people are doing work relevant to the sprint.

## Notice the unsaid blockers

When team members are answering the three questions, it is good practice to listen for the blockers people are **not** directly mentioning. I often see situations where team members say they have no blockers, but their answers to the two other questions indirectly mention many blockers.

For example, someone discussing what they did yesterday may talk about the unexpected support issues that keep coming up. Unmanaged, these support issues become blockers. Someone else may talk about a weekly internal committee meeting they need to attend that is eating away some of their time.

A more subtle form of unsaid blocker occurs when people keep putting new tasks in progress because they are waiting for answers on other tasks. To notice these, the team also needs to pay attention to the work already in progress. These unsaid blockers should produce questions and become conversation points for the team.

## Getting long Daily Scrums under control

Effective daily scrum meetings should last no longer than fifteen minutes. For many teams, respecting this time limit is a big challenge but it is important to respect the time box or team members will lose interest in the meeting. Teams larger than five to nine peoplemay decide that fifteen minutes is not suitable for them and may opt for a slightly longer time box.

When you see your team having design discussions or trying to problem solve, the best approach is to call a separate meeting after the daily meeting. Invite only key people to this discussion to not waste the time of all team members.

If the meetings are running long because people are talking too much, one solution is to use a timer. I have two variants on this technique: the first method is to time the meeting and stop at fifteen minutes, no matter what. You can be sure the people who did not have time to speak will get upset and will want to speak first the next day. The other method is to divide the meeting time equally between all participants; when I use this, I interrupt people when their time is up.

## The importance of useful answers

Another challenge of daily scrums is when people provide answers that are useless to the three questions. You can see one of my favorite examples below. As you can see, the answers are completely useless to the team. The team would get much more value if the person would take the time specify which defects he is talking about.

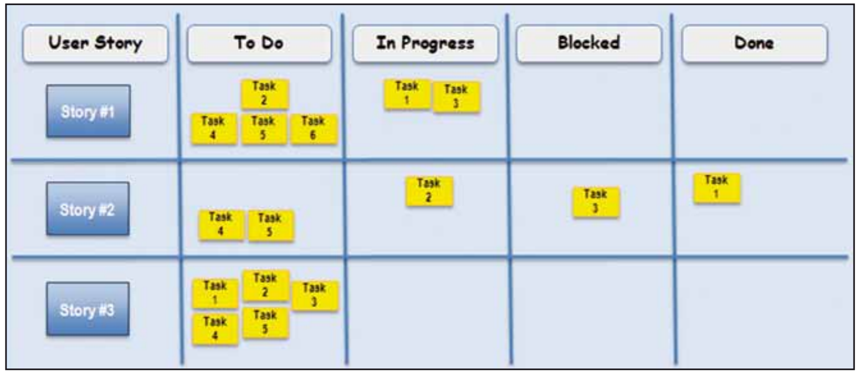


**Figure 1: Sprint task board**

*"Hi Gang … What did I do yesterday? Well, I fixed some defects … Today, I think I am going to fix more defects … What is blocking me? Well … If only I did not have so many defects to fix!"*

When team members (not just the Scrum Master) feel the answers are useless, they should not just let it slide, but ask more pointed questions. In these situations, I find humor works well, and I will make a joke such as "Joe, that sounds both nice and mysterious … Could you please tell us more?" to get more information.

Peer pressure and visibility are important parts of the daily scrum. When you tell the team what you will work on today, you are making a verbal commitment to the team. Team members that depend on your work expect you to complete what you said you would do today so their own work can move forward tomorrow.

## Keeping an eye on the goal

Just blindly answering the three questions, even with the sprint task board in front of you, is rather useless if the team is not keeping an eye on delivering on their sprint goal. One of the last questions I like to ask at the end of the meeting is whether the team feels they will meet their sprint goal.

When the team does not feel confident, members may need to reorganize their work to allow them to complete the high priority pieces while letting others drop to the next sprint. The team needs to discuss such decisions with the Product Owner.

Learning in the last days of a sprint that the team will not deliver the planned work is a powerful sign the team is not keeping track of the spring goal.

## Preserving transparency and focus

One common issue that I see is when certain team members refuse to update their estimates of time remaining. This causes situations where someone has an everlasting thirty-hour task, even though they work on it every day. Ideally, tasks should be no longer than a day or two to allow the team to see progress.

In such situations, the best approach is to encourage that person to break up tasks in smaller blocks at the sprint planning meeting to allow delegation to others. During a sprint you may want to ask the person to list the remaining work and create tasks for each work item. Once you have these tasks work on getting proper estimates.

Be aware some people refuse to break down their work into smaller tasks because they are uncomfortable with the visibility it provides. For example, if they estimate five hours and it takes seven or eight instead, they may look bad. When facing these people, you may need to have some separate one-on-one conversations to hear their concerns and address the issue.

Another focus-related challenge is of team members with five tasks already "in progress" on the task board that feel an overwhelming need to add yet another task in that column. During the meeting, the team needs to ask this member about the status of the other tasks currently in progress. Are some of these blocked? Are some of these done but not yet moved to the "done" column? If they are still in progress, they need to focus on completing their work before starting something new.

## Making sure everyone pays attention

When many people talk at once or side conversations occur, it can cause serious communication challenges. When side conversations occur, my preferred approach is stopping the person currently speaking to the group so that everyone can focus on listening to the side conversation. I always find it interesting how putting people in the spotlight encourages them to self-discipline and stop their conversation.

For teams where many team members chronically speak at once, I occasionally use a talking stick, where only the person holding stick may speak. This forces the team to have only one speaker at a time. In one company I worked with, I used a plastic microphone as a talking stick with the teams I coached. I saw other teams use a ball they toss to one another or a dog squeaky toy as their preferred speaking permission tool.

Other ways people stop paying attention is by fiddling around on their cell phones or staring into nothingness. The best way to get their attention is by calling on them to speak next or using humor to draw them back in. The most direct way to address this is literally calling them out for not paying attention.

## Conclusion

Remember the goal of the daily scrum meeting is to allow the team to coordinate their efforts and ensure they are on track to meet their sprint goal. The meeting is not meant or feel like a daily status report meeting.

When team members answer the question "what am I doing today?" they are implicitly making a verbal commitment to their team. Peer pressure (i. e. the team asking why someone did not meet a commitment) is another important aspect of the daily scrum. Team members should feel comfortable challenging one another in a healthy way about their deliverables.

The team should be able to do a daily scrum meeting with or without the Scrum Master present. This is part of the team taking ownership of their software development process. The Scrum Master should actively encourage the team to step up and own this meeting.

Daily scrums are an important part of the sprint cycle because they help the team collaborate and plan on a daily basis. Keeping them interesting and effective will enable your team to stay focused and regularly deliver on their sprint goals.

Coming soon to Cambridge, Massachusetts

# Conference preview: PHP Northeast

**In August, four user groups will join forces to once again host some of the world's top speakers.**

Photo by Anna Filina

by Bradley Holt

The Northeast PHP Conference was born out of a humble conversation between organizers of the Boston PHP, Atlantic Canada PHP, and Burlington, Vermont PHP user groups. Boston PHP was (and still is) the largest and most active PHP community in the world and had long dreamed of starting its own conference. After almost a year of planning and collaboration between organizers throughout the northeast region, this dream was finally realized with the inaugural Northeast PHP Conference in August of 2012.

Starting with no budget and no conference experience, the organizational team overcame many challenges to create an event on par with the big technology conferences. With the event space generously donated by Microsoft's New England Research and Development (NERD) Center and sponsors such as Wayfair (for t-shirts and a Saturday party for attendees) and Engine Yard (for a speakers' dinner), the NEPHP organizers were able to check many large ticket items off their list. Many other sponsors, too numerous to mention here, provided additional resources making it possible to keep the ticket price to an absurdly low $ 100 per person. Most profoundly, all the organizers or speakers not only volunteered their time, but also paid their own expenses to participate in the conference.

Last year's conference featured NASA's J.J. Toothman as the opening keynote, highlighting how NASA uses PHP in their everyday work. User experience guru Jared Spool brought down the house with his closing keynote on mobile and user experience. As one attendee shared, "This talk alone was worth the entire conference ticket price and more." [1].

Despite the lack of payment, over 30 speakers came from as far as Canada and Norway to give over 40 talks for the 2012 conference. Speakers included Alan Seiden on relaxing at the keyboard and IBM i; Anna Filina on documentation and motivating developers; Ben Ramsey on HTTP; Dave Stokes on MySQL; Heather O'Neill on testing and usability; Ilia Alshanetsky on Memcached and PHP 5.4; Jonathan Klein on high-performance PHP; Michael Stowe on PHP security, mobile development, and WordPress; Michelangelo van Dam on community and quality assurance; Mike Willbanks on Gearman and Varnish; Pek Pongpaet on user experience; and Ross Tuck on Redis.

Tickets sold out for the 2012 event, surpassing the organizers' expectations, and the conference was voted #1 for 2012 at the Microsoft NERD Center [2]. One attendee said that the Northeast PHP Conference 2012 "had to be one of the most enjoyable weekends that did not include close friends or family" [3]. Another attendee said, "I've attended a lot of conferences over the years, but I found this one to be one of the most enjoyable ones that I've ever attended" [4].

After a successful first year, the organizing team enthusiastically decided to do it again in 2013. This year's conference is scheduled for Friday, August 16th through Sunday, August 18th, again at the Microsoft NERD Center in Cambridge, Massachusetts. Not content to just repeat last year's success, the organizers are including a third day fully focused on workshops, to add a more hands-on element to the conference this year. The workshop day is a separate registration from the two days of talks, and users can opt in, though workshop space is more limited than general tickets. Both the workshops and regular sessions will be focused around PHP, Web Technology, and User Experience tracks.

"The conference is organized entirely by volunteers and backed by Boston PHP, a nonprofit."

While grounded in PHP, the Northeast PHP Conference is not just about PHP. As the organizers know, most PHP developers are working on websites and applications, which means that they need more than just PHP skills to get ahead. The Web Technology and User Experience tracks help these developers broaden their skill sets and expand their knowledge & experience.

Photo by Joe Baz

Photo by Anna Filina

Photo by Anna Filina

As with the 2012 conference, this year's conference is organized entirely by volunteers and backed by Boston PHP, a nonprofit organization. The organizers are keeping costs low through the generous support of Microsoft NERD Center, Engine Yard, GitHub, and other sponsors[5]. Tickets for this year's conference are $ 200 per person for the regular sessions (Saturday and Sunday). Friday's workshops can be added for an additional $100. While still very affordable, the increased ticket price will allow the Northeast PHP Conference to provide a small travel stipend to speakers. This is huge, as speakers at the Northeast PHP Conference 2012 spent their own money on airfare and hotel in order to help make the event possible.

Speakers at this year's conference will include Terry Chay on features engineering at Wikipedia; Anthony Ferrara on SOLID (Single responsibility, Open-closed, Liskov substitution, Interface segregation & Dependency inversion); Sheeri Cabral on InnoDB/Memcached; Anna Filina on jQuery Mobile; Larry Ullman on teaching PHP and AJAX; Eryn O'Neil on avoiding the programmer's user interface; Jonathan Barronville on Laravel; Andrew Curioso on API development; Michael Stowe on REST and agile; Peter MacIntyre on preparing for Zend 5.3 certification; Adam Culp on clean application development; Heather O'Neill on honest empathy and usability testing for the common man; Ross Tuck on building JavaScript applications with vector graphics; and Jonathan Klein on scaling PHP and responsive design [6].

As the Northeast PHP team looks to the future there is a strong desire to take the event on the road, traveling throughout the northeast region. With many developers in the northeast region, and not many conferences serving these developers, Northeast PHP can help developers grow and learn, becoming confident and even sharing their own talks. Initially, the team plans to have satellite events in Prince Edward Island and Burlington, Vermont, with additional locations to be considered in future years. Follow @NEPHP [7] on Twitter for updates and don't miss out on the Northeast PHP Conference 2013 in Boston!

## References

[1] https://joind.in/6848
[2] http://blog.microsoftcambridge.com/2013/01/04/top12of2012/
[3] http://www.josephcrawford.com/2012/08/northeast-php-conference-2012/
[4] http://blog.ircmaxell.com/2012/08/the-anatomy-of-great-conference.html
[5] http://northeastphp.org/sponsors
[6] http://www.northeastphp.org/speakers
[7] https://twitter.com/NEPHP

## About Web & PHP Magazine

**Publisher**
Software & Support Media Ltd

**Editorial Office Address**
86 Great Suffolk St
London SE1 0BE
www.sandsmedia.com

**Editor:** Elliot Bentley (elliotb@sandsmedia.com)
**Authors:** Arne Blankerts, Lee Boynton, Ben Greenaway, Frédéric Harper, Bradley Holt, Cory Isaacson, Steffan Surdek, Tom van Looy
**Creative Director:** Jens Mainz
**Layout:** Tobias Dorn, Petra Rüth

**Sales:**
Ellen May
+44 207 199 6234
ellenm@sandsmedia.com