The Art Of Application Logging

PHPNW 2012

Ben Waine - @bwaine

http://github.com/benwaine/ApplicationLoggingTalk

Ben Waine

Contractor / Co Founder
Recensus

PHP Developer

Logs in spare time....


php

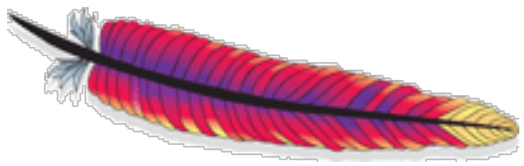
Roadmap

- Introduction - Why Is Logging Good?
- The Mechanics - Code / Examples / Live Demos (*GULP*)
- Logging Strategy - What to do with your logs
- Conclusion

Roadmap

- **Introduction - Why Is Logging Good?**
- The Mechanics - Code / Examples / Live Demos (*GULP*)
- Logging Strategy - What to do with your logs
- Conclusion

Name Some Logs.....

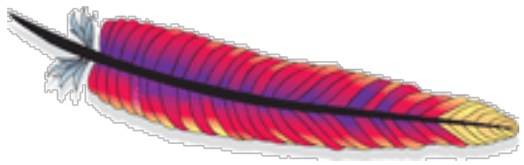


Name Some Logs.....

The Benefits Logs Provide

tail -f

- Insight into how a process is being executed
- Ability to analyse execution after the fact
- Ability to catch information about execution errors
- Look for patterns and trends



These are all services.....

What about application logging?

I

SKYBET's Log


N-59
0284
3
1


WAITING

Benefits For Beginners

- Provide direction when picking up an application for the first time.
- Well sign posted layers increases understanding of the request.
- Quick error isolation and context
- Times in logs keep performance prominent

Benefits For Everyone


DEVELOPERS
Y U NO LOG??

Roadmap

- Introduction - Why Is Logging Good?

- **The Mechanics - Code / Examples / Live Demos (*GULP*)**

- Logging Strategy - What to do with your logs

- Conclusion

Our Logging Goals

Log information about a script which creates users.

User - Represents a user

UserMapper - A class which saves users

Scripts - Control User and UserMapper

1) The Basics

https://github.com/benwaine/ApplicationLoggingTalk/tree/master/examples/01-NativeLogging

error_log()

```php
require_once __DIR__ . '/app/User.php';
require_once __DIR__ . '/app/UserMapper.php';
require_once __DIR__ . '/../../vendor/autoload.php';

// Create a mapper
$mapper = new UserMapper();

// Create Users and persist them
error_log('Beginning User Creation');

while(true)
{
    $user = new User(rand(1, 10000), 'Betty Boo');

    $mapper->save($user);

    sleep(1);

    unset($user);
}
```

01-basic.php

```php
class UserMapper {

    public function save(User $user)
    {

        error_log('Saving User: ' . $user->getId());

        // Code For Saving User

    }

}
```

UserMapper.php

DEMO

```php
require_once __DIR__ . '/app/User.php';
require_once __DIR__ . '/app/UserMapper.php';
require_once __DIR__ . '/../../vendor/autoload.php';

ini_set('error_log', '/tmp/user.log');

// Create a mapper
$mapper = new UserMapper();

// Create Users and persist them
error_log('Beginning User Creation');

while(true)
{
    $user = new User(rand(1, 10000), 'Betty Boo');

    $mapper->save($user);

    sleep(1);

    unset($user);
}
```

02-ToFile.php

Pros

- Native Functionality

Cons

- error_log, semantically correct?
- PHP errors mixed in with application logging
- Not very powerful compared to other tools

2) Logging Library

https://github.com/benwaine/ApplicationLoggingTalk/tree/master/examples/02-LoggingAsADependency

Log4PHP

PEARLog

Monolog

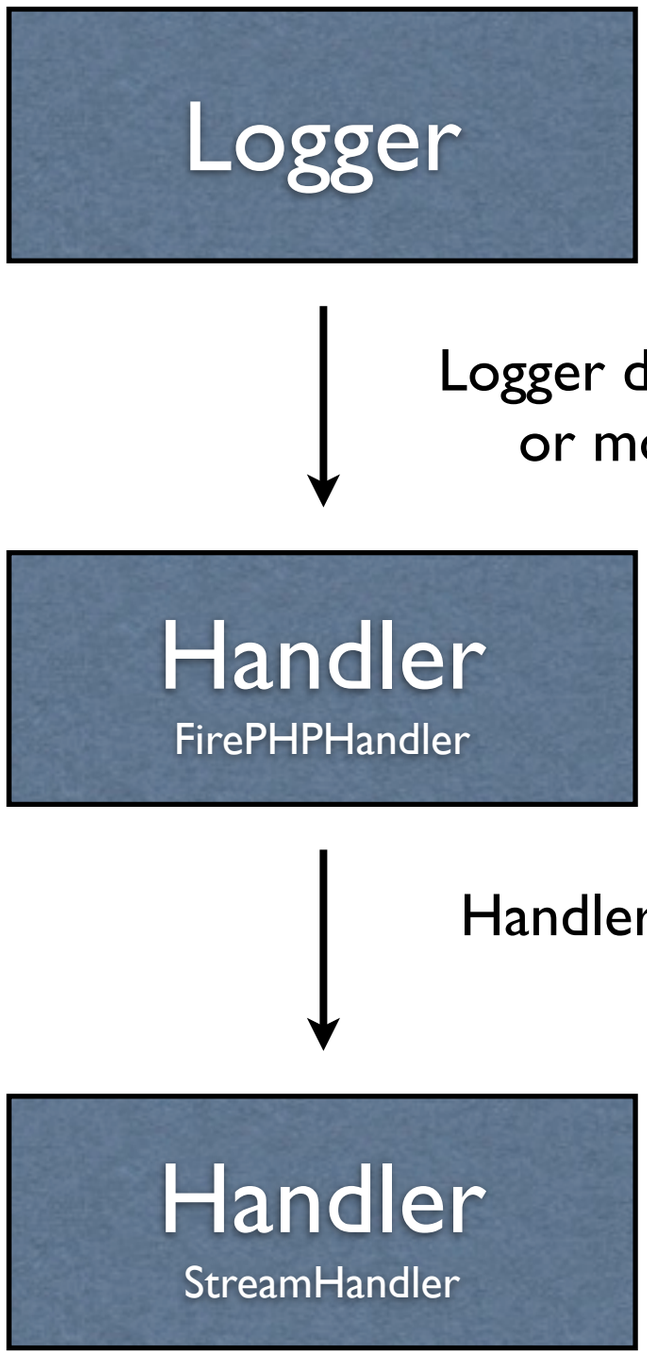
ZendLog

KLogger

Monolog

https://github.com/Seldaek/monolog

Monolog Architecture




```php
use Monolog\Logger;
use Monolog\Handler\StreamHandler;
use Monolog\Handler\FirePHPHandler;

// Create the logger
$logger = new Logger('my_logger');
// Now add some handlers
$logger->pushHandler(new
StreamHandler(__DIR__.'/my_app.log',
Logger::DEBUG));
$logger->pushHandler(new FirePHPHandler());

// You can now use your logger
$logger->addInfo('My logger is now ready');
```

Log formatting is changed using supplied or custom formatters.

Handlers can be used by multiple loggers. Formatters can be used by multiple handlers.

DEMO

```php
// ... Require Statements ...
use Monolog\Logger;
use Monolog\Handler\StreamHandler;

// Create a New MonoLog
$logger = new Monolog\Logger('Application Log');
// Add a handler (writer)
$logger->pushHandler(new StreamHandler('/tmp/user.log',
Logger::DEBUG));

// Create a mapper
$mapper = new UserMapper($logger);

// Create Users and persist them
$logger->notice('Beginning User Creation');

while(true)
{
    $user = new User(rand(1, 10000), 'Betty Boo');
    $mapper->save($user);
    sleep(1);
    unset($user);
}
```

01-basic.php

```php
use Monolog\Logger;

class UserMapper {

    protected $logger;

    public function __construct(Logger $logger)
    {
        $this->logger = $logger;
    }

    public function save(User $user)
    {

        $this->logger->info('Saving User: ' . $user->getID());

        // Code For Saving User

    }

}
```

UserMapper.php

Pros

- Object Oriented Logging
- Multiple Log Writers (File, Email, Firebug, StdOut)
- Log Levels (See Example)

Cons

- ‘Unwieldy’ instantiation
- Logging control code permeates application layer

Log Levels

RFC: 5434
(The Syslog Protocol)

```
0         Emergency: system is unusable
1         Alert: action must be taken immediately
2         Critical: critical conditions
3         Error: error conditions
4         Warning: warning conditions
5         Notice: normal but significant condition
6         Informational: informational messages
7         Debug: debug-level messages
```

DEMO

```php
use Monolog\Logger;
use Monolog\Handler\StreamHandler;

$environment = getenv('ENVIRONMENT');
$level = ($environment == "live") ? Logger::NOTICE : Logger::DEBUG;

// Create a New MonoLog
$logger = new Monolog\Logger('Application Log');
// Add a handler (writer)
$logger->pushHandler(new StreamHandler('/tmp/user.log', $level));
// Create a mapper
$mapper = new UserMapper($logger);

// Create Users and persist them
$logger->notice('Beginning User Creation');

while(true)
{
    $user = new User(rand(1, 10000), 'Betty Boo');

    $mapper->save($user);

    sleep(1);

    unset($user);
}
```

02-LogLevels.php

```php
use Monolog\Logger;

class UserMapper {

    protected $logger;

    public function __construct(Logger $logger)
    {
        $this->logger = $logger;
    }

    public function save(User $user)
    {

        $this->logger->info('Saving User: ' . $user->getID());

        // Code For Saving User

    }

}
```

UserMapper.php

3) Your Application Logger

https://github.com/benwaine/ApplicationLoggingTalk/tree/master/examples/03-YourApplicationLogClass

```php
public function __construct(MLogger $application,
                            MLogger $security,
                            array $requestParams) {

    $this->application = $application;
    $this->security = $security;
    $this->requestParams = $requestParams;

    $this->requestId =
       md5(microtime() .
       $this->requestParams['REMOTE_ADDR']);
}
```

Logger.php - __constructor

```php
/**
 * Adds a record to the application log.
 *
 * @param string $message
 * @param int $level
 *
 * @return void
 */
public function applicationLog($message, $level) {

    // Including a request identifier allows us to track a
    // users progress throughout the application by grepping the ID.

    $context = array('RID' => $this->requestId);

    $this->application->addRecord($level, $message, $context);
}
```

Logger.php - applicationLog

```php
// Create first logger (application)
$applicationLogger = new MonoLogger('Application');
// Add a handler (writer)
$applicationLogger->pushHandler(new StreamHandler('/tmp/
application.log', MonoLogger::DEBUG));

// Create second logger (security)
$securityLogger = new MonoLogger('Security');
$securityLogger->pushHandler(new StreamHandler('/tmp/security.log',
MonoLogger::DEBUG));

$logger = new Logger($applicationLogger, $securityLogger,
$requestParams);

// Create a mapper
$mapper = new UserMapper($logger);
```

01-basic.php 1/2

```php
// Create Users and persist them
$logger->applicationLog('Beginning User Creation', Logger::INFO);

for($x = 0; $x <= 4; $x++)
{
    $user = new User(rand(1, 10000), 'Betty Boo');

    // The mapper generates some informational logs when the
    // the user record is 'saved'
    $mapper->save($user);
    unset($user);

    if($x % 2) {

        // Simulate some kind of security warning when creating some
        // of the users.
        $logger->securityLog('UNSAFE USER!', Logger::WARNING);
    }
}

$logger->applicationLog('Ending User Creation', Logger::INFO);
```

01-basic.php 2/2

Pros

- Consolidated logging control logic
- Encapsulate common logging actions

Cons

- Still ‘Unwieldy’ instantiation
- Can be a little inflexible

4) Your Application Logger (DI'ed)

https://github.com/benwaine/ApplicationLoggingTalk/tree/master/examples/04-DiConfiguredLogging

```yaml
parameters:
    logger.class:       Logger
    logger.security.path: /tmp/security.log
    logger.security.name: security
    logger.security.level: 200
    logger.application.path: /tmp/application.log
    logger.application.name: application
    logger.application.level: 200
    logger.monolog.class: Monolog\Logger
    logger.monolog.handlerClass: Monolog\Handler\StreamHandler
    user.mapper.class: UserMapper
    request.params:
      REMOTE_ADDR:
        - 127.0.0.1
```

services.yml 1/2

```yaml
services:
    logger:
        class: %logger.class%
        arguments: [@applog, @seclog, %request.params%]
    seclog:
        class: %logger.monolog.class%
        arguments: [%logger.security.name%]
        calls:
            - [ pushHandler, [ @seclogHandler ] ]
    applog:
        class: %logger.monolog.class%
        arguments: [%logger.application.name%]
        calls:
            - [ pushHandler, [ @applogHandler ] ]
    seclogHandler:
        class: %logger.monolog.handlerClass%
        arguments: [%logger.security.path%, %logger.security.level%]
    applogHandler:
        class: %logger.monolog.handlerClass%
        arguments: [%logger.application.path%, %logger.application.level%]
    UserMapper:
        class: %user.mapper.class%
        arguments: [@logger]
```

services.yml 2/2

```php
// Set Up Container
$container = new ContainerBuilder;
$loader = new YamlFileLoader($container, new FileLocator(__DIR__));
$loader->load("services.yml");

// Get Mapper and Logger
$logger = $container->get('logger');
```

01-basic.php 1/2

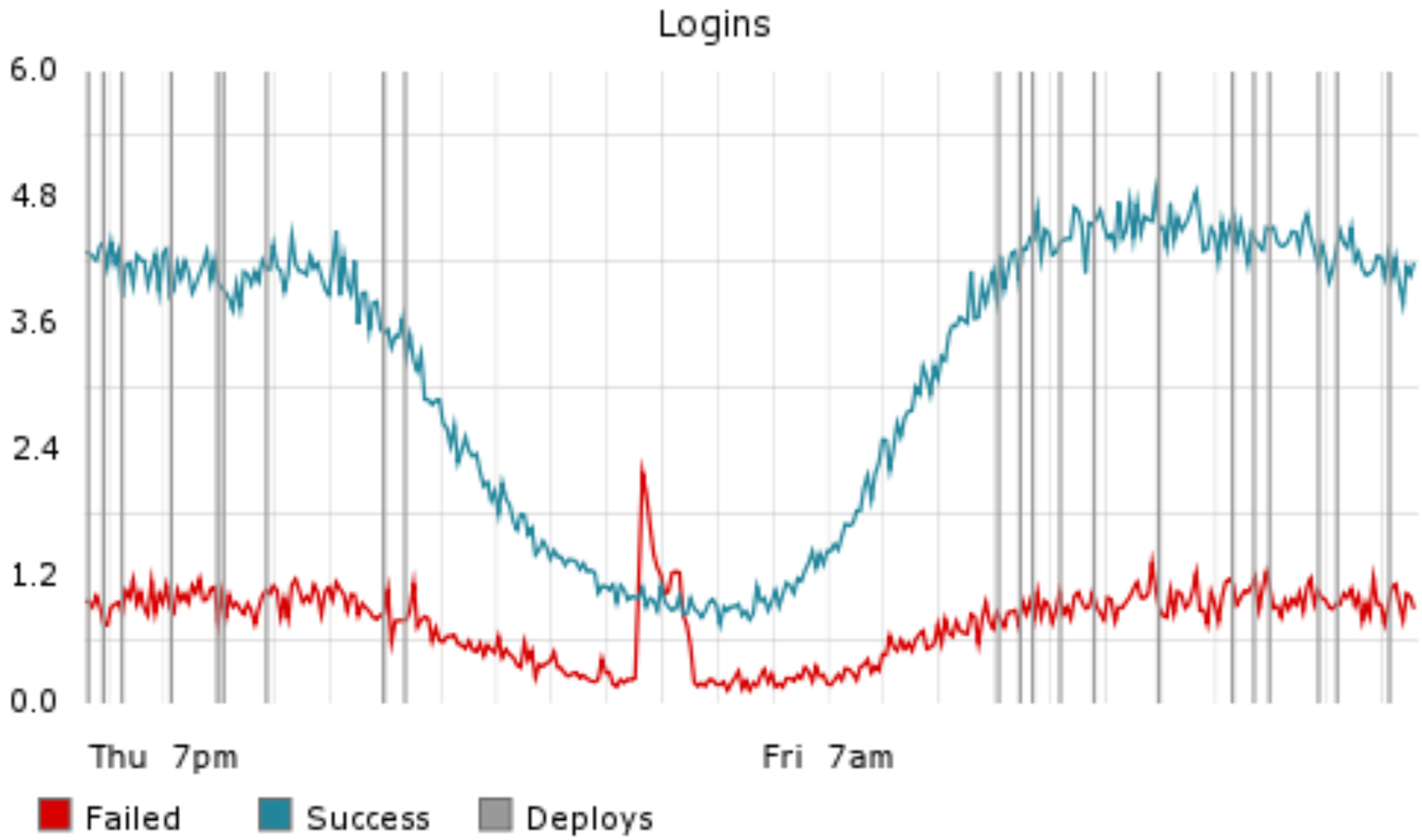
Me

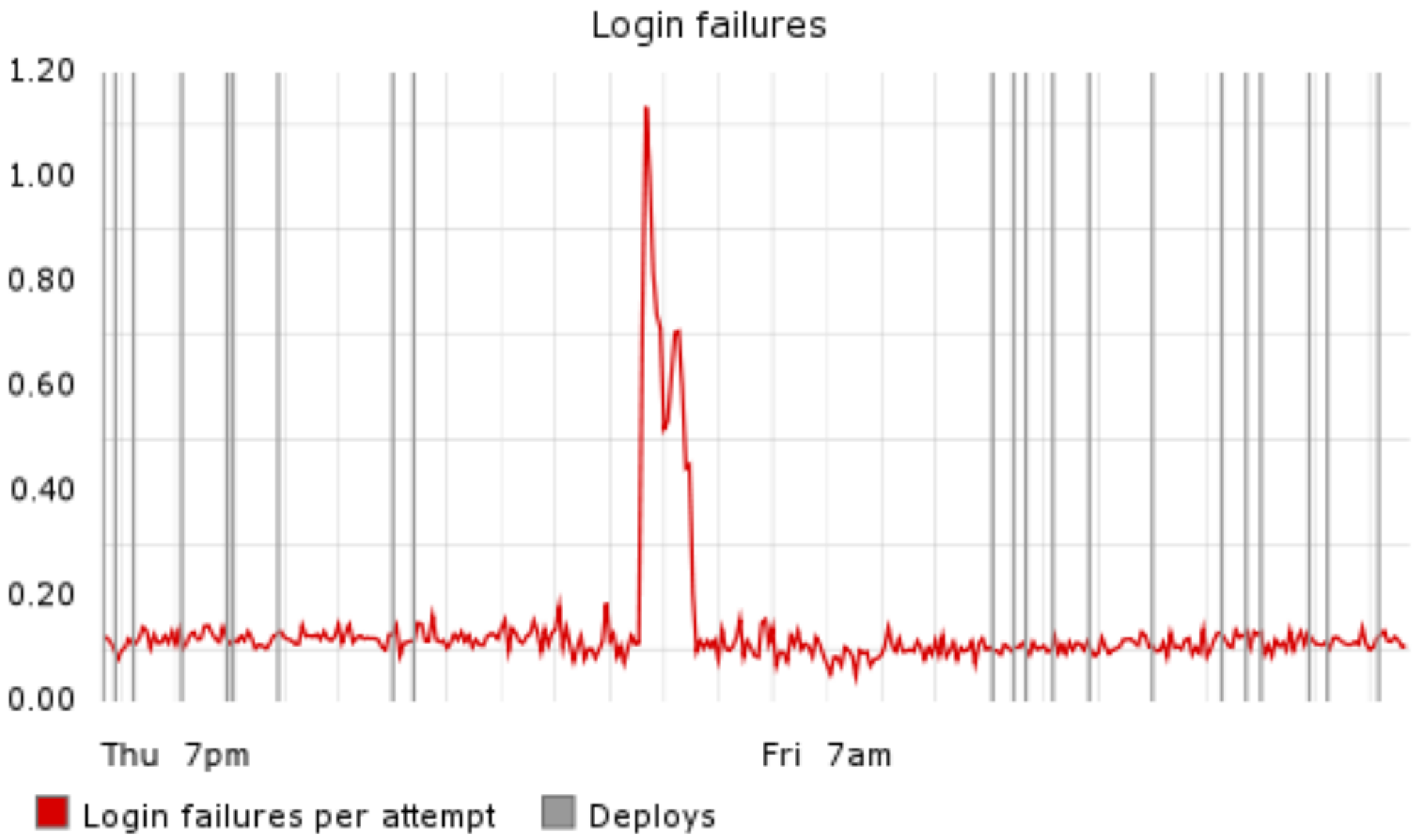

INJECTS DEPENDENCIES
LIKE A BOSS
memegenerator.net

5) Event Logging

https://github.com/benwaine/ApplicationLoggingTalk/tree/master/examples/05-StatsDGraphite

Log Entries As Events That Occur In Your System


Logins
6.0
4.8
3.6
2.4
1.2
0.0
Thu 7pm
Fri 7am
Failed
Success
Deploys


Login failures
1.20
1.00
0.80
0.60
0.40
0.20
0.00
Thu 7pm
Fri 7am
Login failures per attempt
Deploys

Graphite + StatsD

StatsD

- Buckets
- Values
- Flush

```php
// Creating StatsD Client
$connection = new \Domnikl\Statsd\Connection\Socket('localhost', 8125);
$statsd = new \Domnikl\Statsd\Client($connection, "test.namespace");

// simple count
$statsd->increment("request");
```

01-basic.php (extract)

DEMO

```php
public function logBusinessEvent($event) {

    if(!isset($this->events[$event]))
    {
        throw new \Exception('Invalid Logging Event');
    }

    $this->statsd->increment($this->events[$event]);

}
```

```php
public function logBusinessTime($event, $time)
{
    if(!isset($this->events[$event]))
    {
        throw new \Exception('Invalid Logging Event');
    }

    $this->statsd->timing($this->events[$event], $time);
}
```

Logger.php (extract)

```php
while(true) {

    $number = rand(1, 10);

    for($x = 0; $x <= $number; $x++)
    {

        $user = new User(rand(1, 10000), 'Betty Boo');
        $mapper->save($user);
        $logger->logBusinessEvent(Logger::EVENT_USERCREATED);

        unset($user);

        if($x%2 === 0) {
            // Simulate some kind of security warning when
            // creating some of the users.
            $logger->logBusinessEvent(Logger::EVENT_USERWARNING);
        }
    }

     sleep(rand(1, 15));

}
```

Logger.php (extract)

5) Putting It All Together

https://github.com/benwaine/ApplicationLoggingTalk/tree/master/examples/06-AllTogether

Roadmap

- Introduction - Why Is Logging Good?
- The Mechanics - Code / Examples / Live Demos (*GULP*)
- **Logging Strategy - What to do with your logs**
- Conclusion

Logging Strategy
Content

1) Choose a log format and be consistent

2) Make sure common tasks like formatting and filtering are dealt with centrally.

3) Choose a set of rules / style around when to log and be consistent.

Logging Strategy

Volume

1) Request Start / End (with times and memory) **Notice**

2) Transition between layers
Eg - Control > Model > DB < Model < Control > View **Info**

3) Database Queries **Info / Debug**

4) Business Events **Separate Log**

5) Message Queue Activity **Debug**

6) PHP Errors / PHP Notices **Error**

Logging Strategy
Audience

1) Make it easy to switch log levels (environment variables).

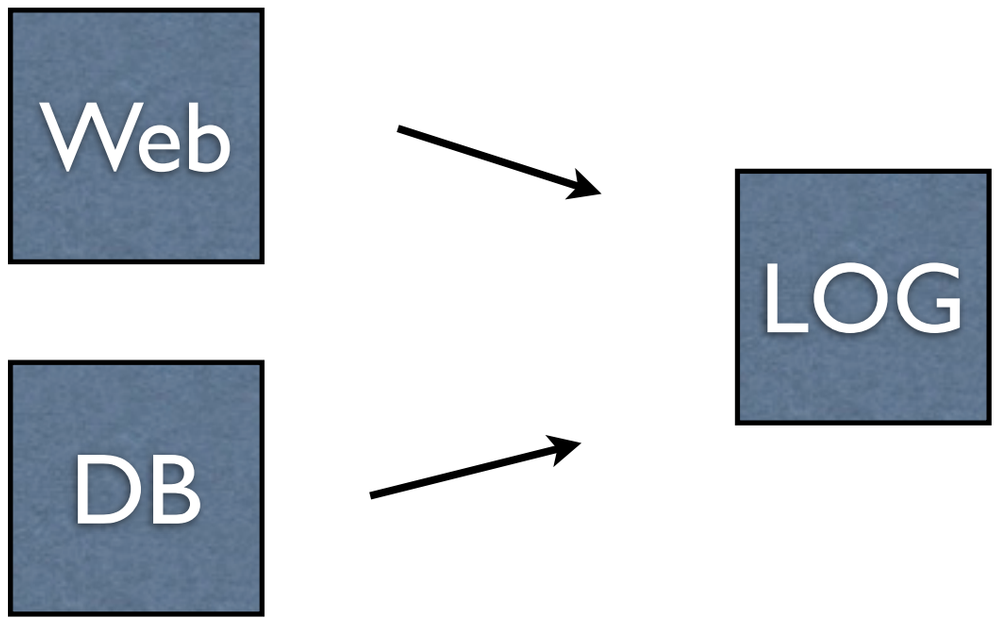
2) Always think of your log consumers when adding log lines.

Log Aggregation & Presentation

RSYSLOG + LOGSTASH + STATSD + GRAPHITE

RSYSLOG

- Send Logs Via TCP / UDP To A Central Location
- Use Filters & Templates To Write Logs To Files / Database



/var/log/web/%ip-address%/access.log
/var/log/web/%ip-address%/error.log
/var/log/db/%ip-address%/query.log
/var/log/db/%ip-address%/slow-query.log

LOGSTASH

```
input {
  file {
    path => "/var/log/apache/access.log"
    type => "apache-access"
  }
}

filter {
  grok {
    type => "apache-access"
    pattern => "%{COMBINEDAPACHELOG}"
  }
}

output {
  statsd {
    # Count one hit every event by response
    increment => "apache.response.%{response}"
  }
}
```

http://logstash.net/docs/1.1.1/tutorials/metrics-from-logs

LOGSTASH

Inputs

1. amqp
2. eventlog
3. exec
4. file
5. ganglia
6. gelf
7. generator
8. heroku
9. irc
10. log4j
11. pipe
12. redis
13. stdin
14. stomp
15. syslog
16. tcp
17. twitter
18. udp
19. xmpp
20. zeromq

Filters

1. checksum
2. csv
3. date
4. dns
5. environment
6. gelfify
7. grep
8. grok
9. grokdiscovery
10. json
11. multiline
12. mutate
13. noop
14. split
15. syslog_pri
16. xml
17. zeromq

Outputs

1. amqp
2. boundary
3. circonus
4. datadog
5. elasticsearch
6. elasticsearch_http
7. elasticsearch_river
8. email
9. file
10. ganglia
11. gelf
12. graphite
13. graphtastic
14. http
15. internal
16. irc
17. juggernaut
18. librato
19. loggly
20. metriccatcher
21. mongodb
22. stomp
23. nagios
24. nagios_nsca
25. null
26. opentsdb
27. pagerduty
28. pipe
29. redis
30. riak
31. riemann
32. sns
33. statsd
34. stdout
35. tcp
36. websocket
37. xmpp
38. zabbix
39. zeromq

DEMO

splunk>

Hosted Services

New Relic.

splunk™

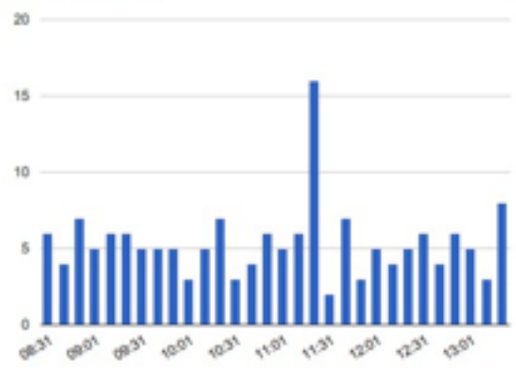

splunk> Application Management v2.0
Administrator | App | Manager | Alerts | Jobs | Logout
About | IT Operations | Executive Dashboards | Reports
Help | About
CEO | Actions
RT Transaction Volume
real-time
Real-time transaction volume across all tiers.
19
$151.00
average purchase
RT Visitor Count
real-time
Real-time concurrent users browsing website.
62
Visitor Location
< 1m ago
Splunk
Canada
United States
North Atlantic Ocean
Iceland
Sweden
United Kingdom
Poland
Ukraine
Kazakhstan
Spain
Turkey
Afghanistan
Pakistan
Algeria
Libya
Egypt
Saudi Arabia
Iraq
Iran
Mauritania
Mali
Niger
Chad
Sudan
Nigeria
Ethiopia
Venezuela
Colombia
Google
Top Items Sold
5m ago
200
100
2:30 PM Fri Dec 23 2011
2:40 PM
2:50 PM
3:00 PM
3:10 PM
3:20 PM
_time
Birthday Bouquet
Cake Serving Set
Chocolate ...nfections
Day Spa Certificate
Dozen Red Roses
Greetings Fruit Basket
Mixed Rose Bouquet
Sweet Dre...s Bouquet
Sweet Spl...r Bouquet
Tulip Bouquet
Abandoned Baskets
5m ago
Average value of abandoned shopping carts on website via logout or expired sessions over the last 60 minutes.
value in $
100
50
2:30 PM Fri Dec 23 2011
2:50 PM
3:10 PM
% Coupon Usage
real-time
Real-time percentage of shoppers using coupons at checkout.
Top Promotions
1s ago
Promotional program popularity over the past 24 hours.
promo
Groupon
Weekly Flyer
Email
0
100
200
300

New Relic®

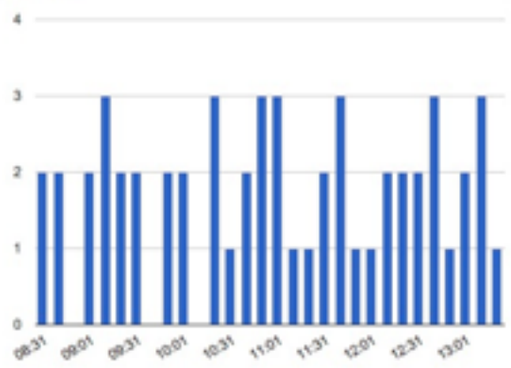

Average response time, broken down by tier (ms)
300
250
200
150
100
50
0
16:00
16:05
Ruby
Database
Worldwide response times
20 sec
15 sec
10 sec
5 sec
0 sec
15:50
United States
Canada
Worldwide Application Performance
Throughput (rpm)
1750
1500
1250
16:00
16:10
16:20

Loggly Dashboard: jgame

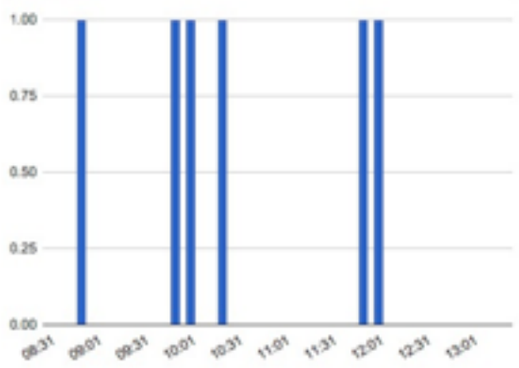
Page Views Today



Redirects



Not Found



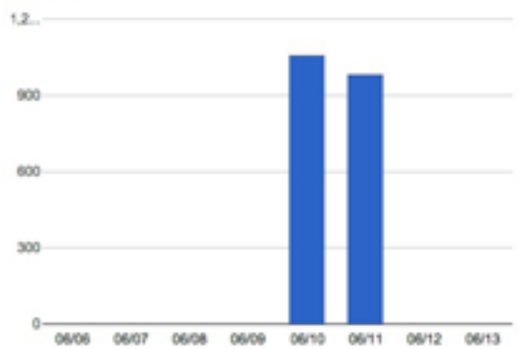
Recent Pages

/	06/14 1:29 PM
/blog/feed	06/14 1:29 PM
/	06/14 1:26 PM
/rss/	06/14 1:23 PM
/blog/feed	06/14 1:23 PM
/	06/14 1:21 PM
/	06/14 1:21 PM
/tagged/MilkMaid/rss	06/14 1:21 PM
/	06/14 1:20 PM
/	06/14 1:18 PM

Referrals

http://www.google.com/url?	06/14 7:44 AM
http://mac.softpedia.com/developer/G	06/14 6:04 AM
http://jga.me/	06/14 5:59 AM
http://owanotifier.en.softonic.com/mac	06/14 5:59 AM
http://jga.me/apps/	06/14 5:58 AM
http://jga.me/	06/14 5:58 AM
http://owanotifier.en.softonic.com/mac	06/14 5:58 AM
http://www.productivelife.cn/2009/03/	06/14 5:51 AM
http://www.google.de/search?	06/14 3:01 AM
http://www.google.de/search?	06/14 3:01 AM

Daily Page Views



Roadmap

- Introduction - Why Is Logging Good?
- The Mechanics - Code / Examples / Live Demos (*GULP*)
- Logging Strategy - What to do with your logs
- **Conclusion**


php

Fin - Questions?

Orange Digital

Hiring Developers & Testers (Contract / Perm)

ray.fawcett@orange.com