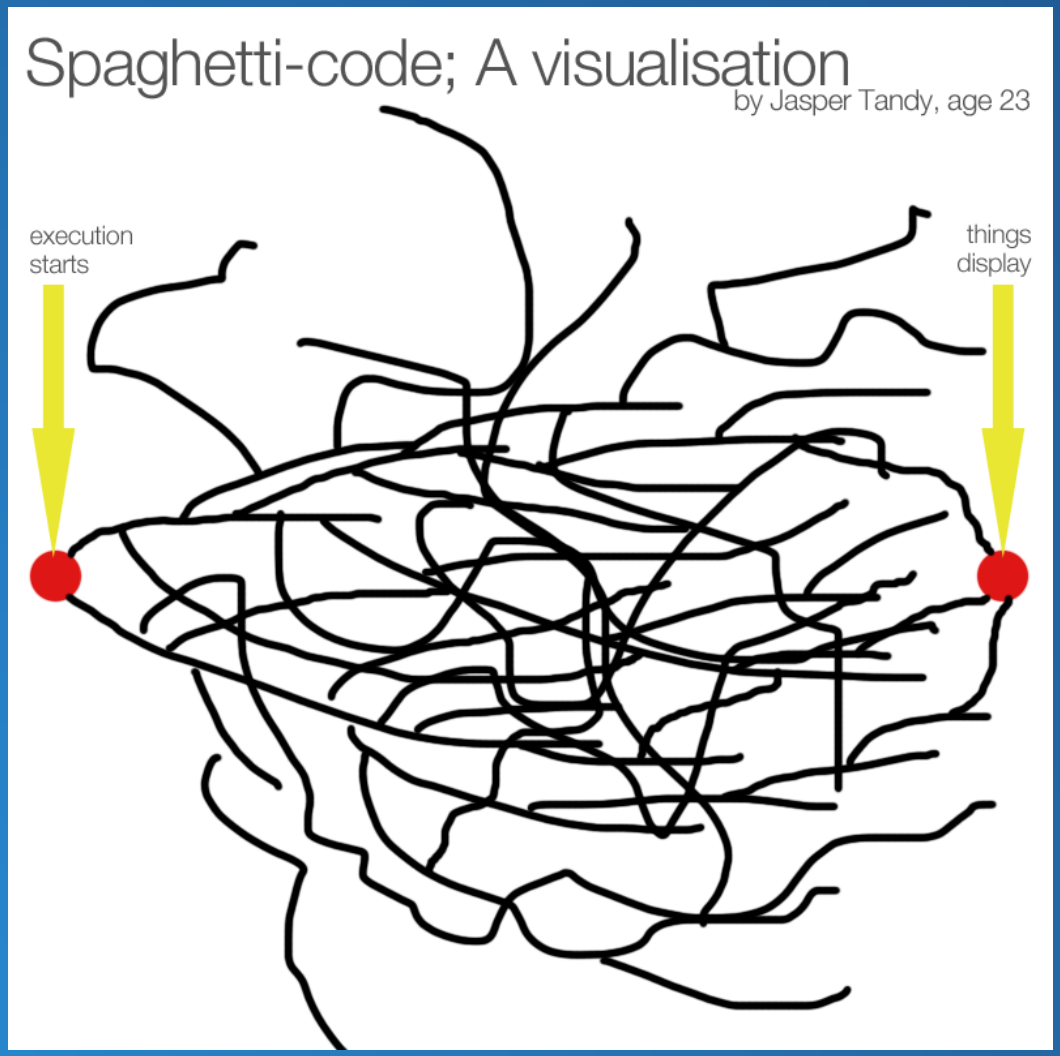
# Modernize your 10-15 years old PHP applications

Legacy PHP applications are everywhere.

History of PHP applications:

- 79%  websites writen in PHP among the top 1 million, according to W3Techs.com
- 1994: PHP was created.  Start of the Spaghetti age
- 2004: PHP5 was released
- 2007: ZF 1.0.0 and SF 1.0 were released.

# Spaghetti Code



# Business rely on features.

- Maintaining two versions of applications need two developers teams.
- New developments don't makes money until the new version is released.
- Releasing everything new at once is very dangerous and can cost you big amount of clients - money.

# Business must react really fast on market changes

*Before the big used to eat the small,*
*nowadays it is the fast who eat the slow*

# Progressive rewrite

Only 1 app to maintain!

- Release rewritten features one by one.
- Don't forget about existing features in rewrite process
- You can always stop that process and moves resources to implementations.

# Testing!

- Create functionally tests for that what could harm your business (registrations, payments etc.)
- Spaghetti code is deeply coupled - touching one part breaks something other.

Create functional test on the most critical scenarios - use Mink + ZombieJs (https://github.com/Behat/Mink).

# Migrate to modern solutions like Symfony2

## Use Symfony Components:

- **HttpFoundation** - Defines an object-oriented layer for the HTTP specification.
- **Routing** - Maps an HTTP request to a set of configuration variables.
- **Console** - Eases the creation of beautiful and testable command line interfaces.
- **DependencyInjection** - Allows you to standardize and centralize the way objects are constructed in your application.
- **EventDispatcher** - Implements a lightweight version of the Observer design pattern.

# Legacy handling

```
$kernel = new AppKernel('dev', true);
$request = Request::createFromGlobals();
$kernel->boot();
try {
    // TRY SYMFONY
    $response = $kernel->handle(
        $request,
        HttpKernelInterface::MASTER_REQUEST,
        false
    );
    $response->send();
    $kernel->terminate($request, $response);
} catch (NotFoundHttpException $e) {
    // TRY LEGACY APP
    require_once __DIR__ . '/../application.php';
    $application->bootstrap();
    $application->run();
}
```

# Other useful things

- Try to share dependency container between legacy and modern apps.
- Try to merge applications directories
- Make legacy code compatible with PHP 5.3 and 5.4 (use Phpcs CodeSniffs)
- Run tests often!
- Release new rewritten features often - one after one.

# Thank You!


## Paweł Mikołajczuk
Newscoop Developer

pawel.mikolajczuk@sourcefabric.org
Github & Twitter: ahilles107
www.sourcefabric.org