Modernizing with Symfony

Alexander M. Turek

about:me

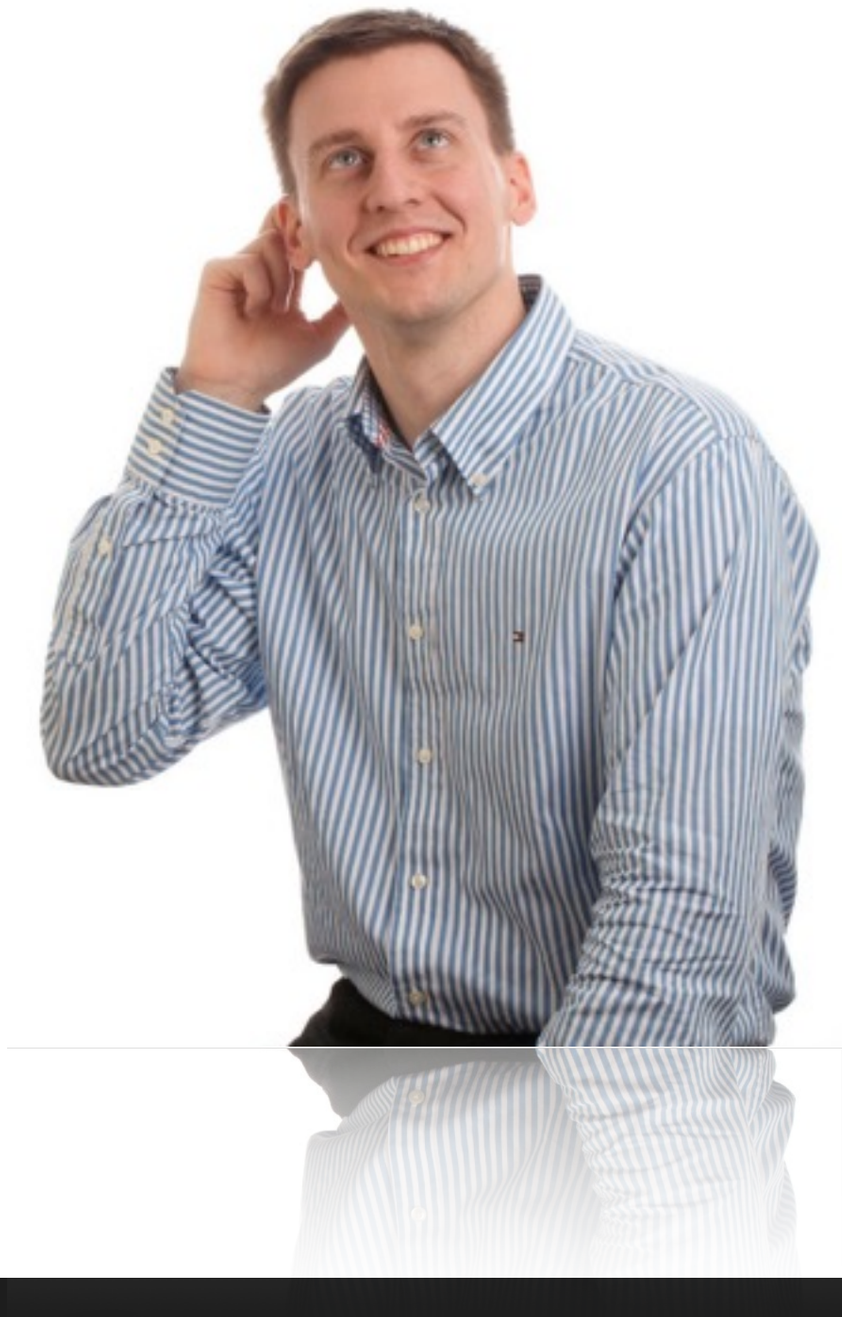
Freelance software developer.

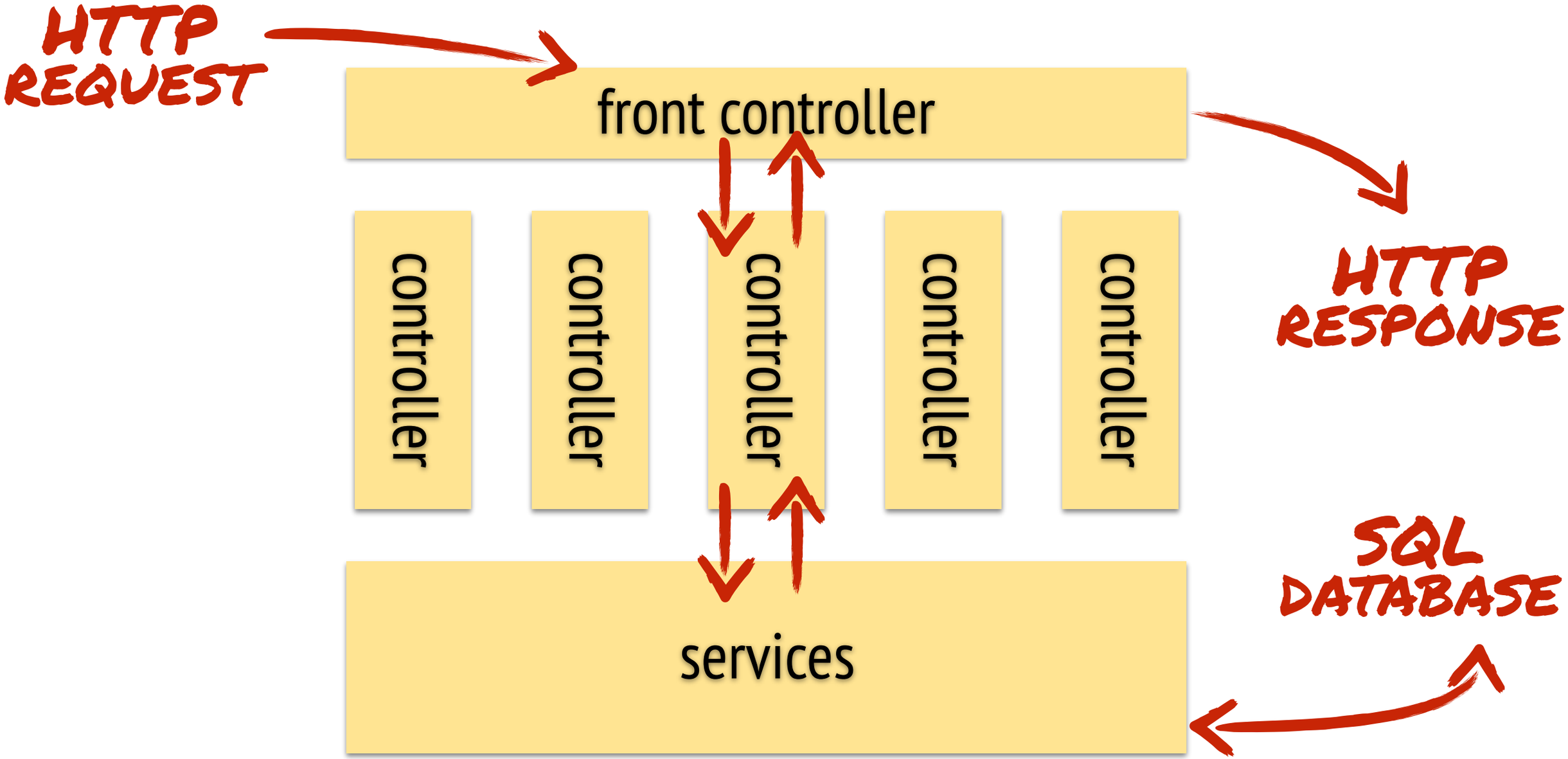
Builds web applications with php.

Lives in ~~Frankfurt~~ ~~München~~ Berlin.

Likes Symfony.

Likes digging into legacy applications.





A Symfony Application

ROUTER

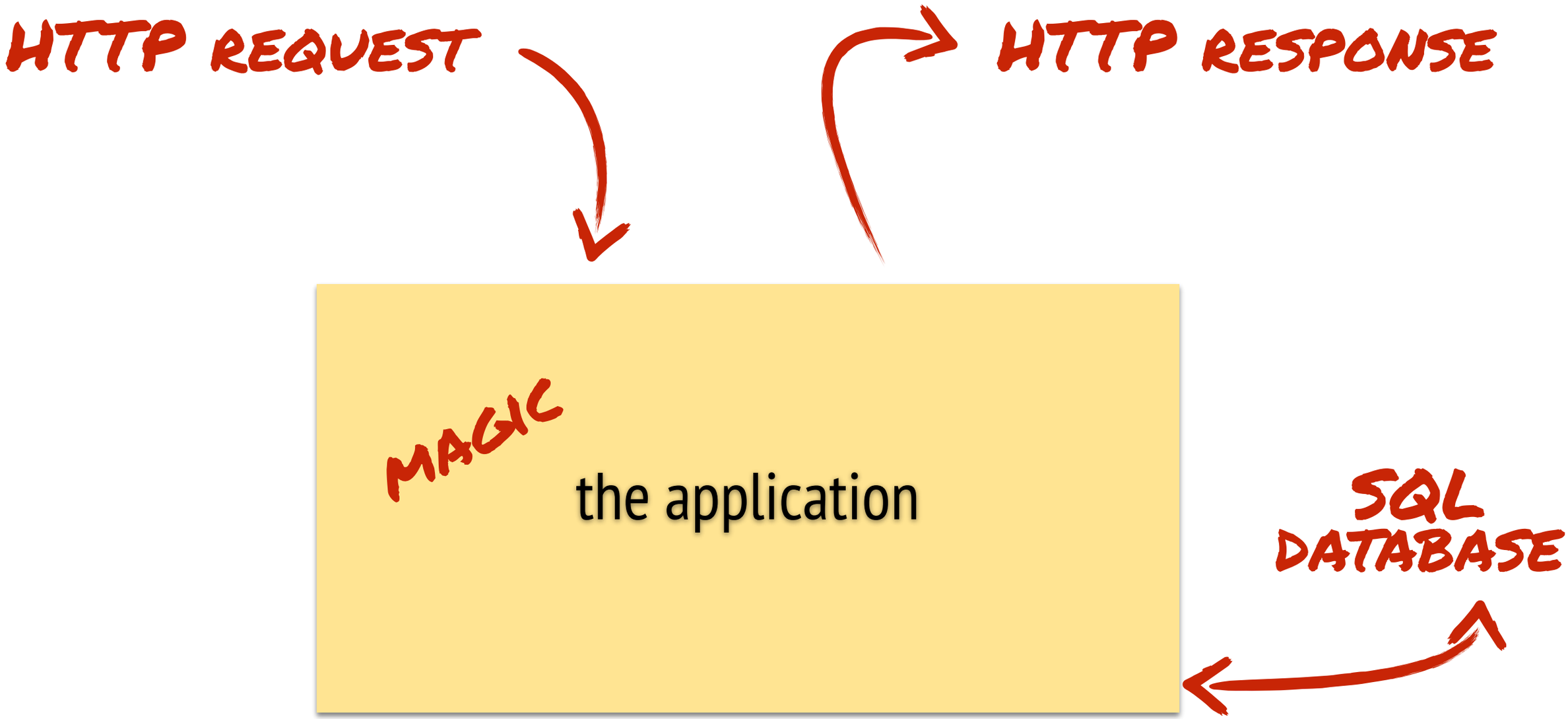
SERVICE CONTAINER

front controller

controller controller controller controller controller

services

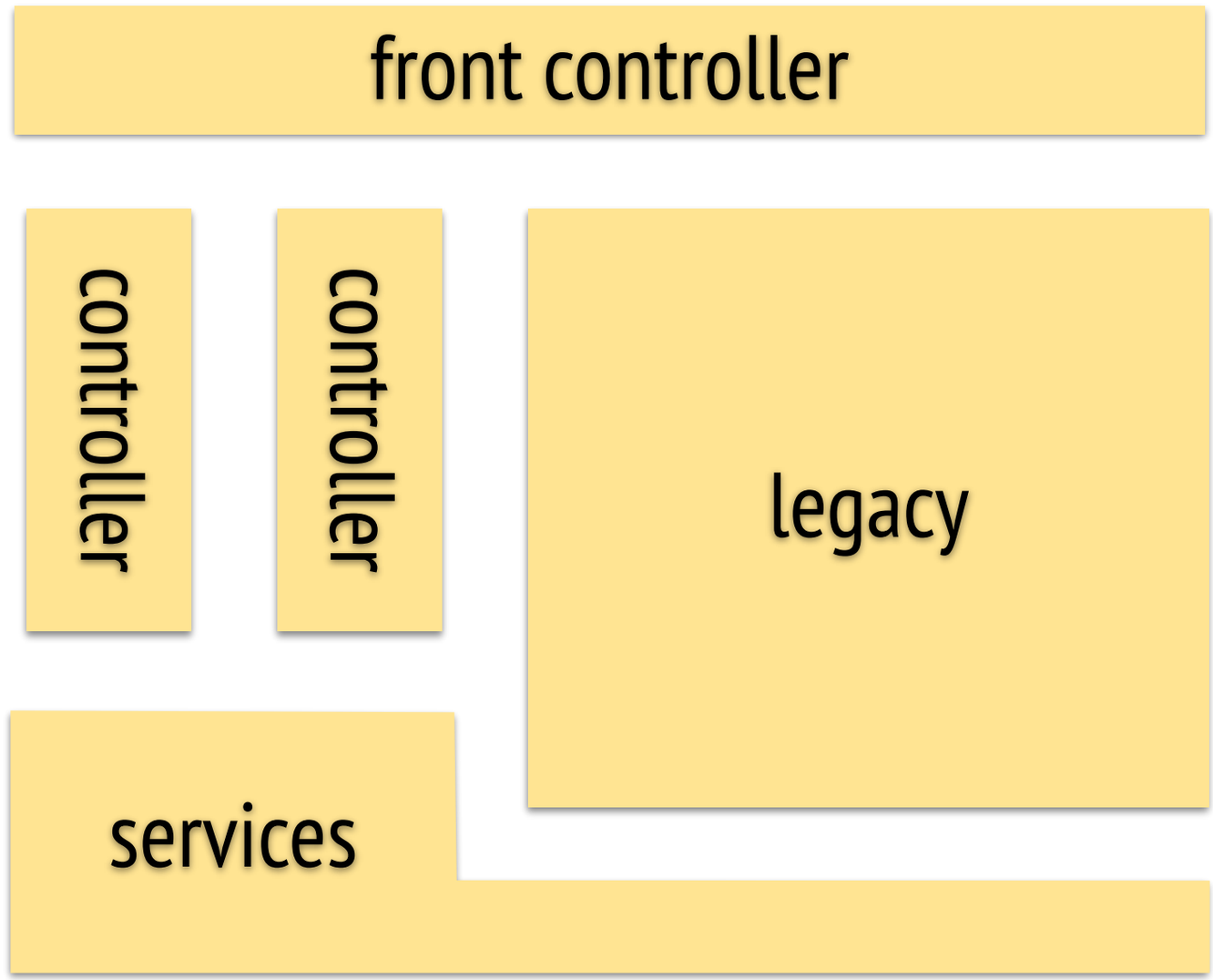
Extension Points



A Legacy Application

Why Symfony?

- Modern application framework.
- Enables to develop applications that are decoupled from the framework itself.
- Highly extensible.



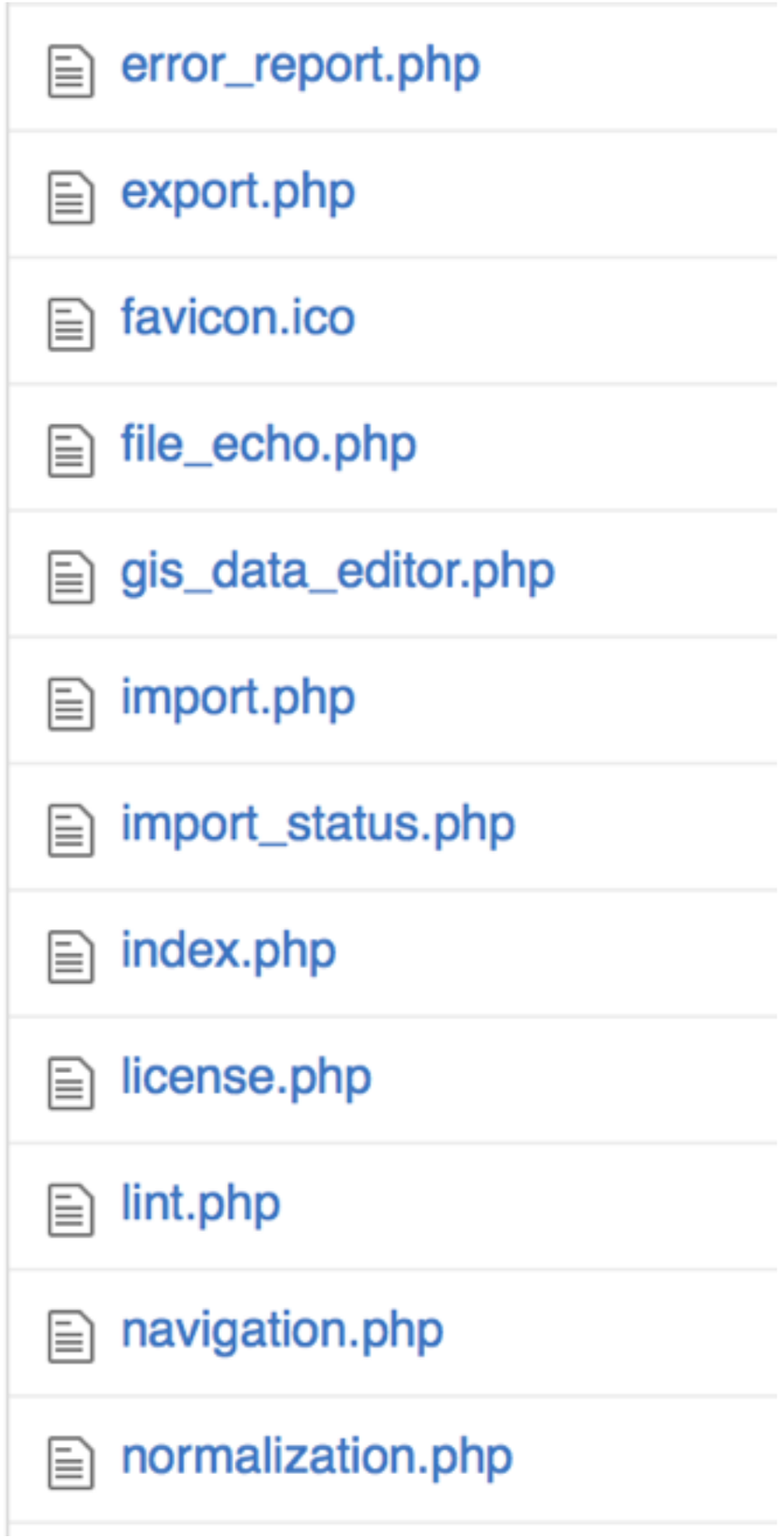
Symfony before Legacy

Symfony before Legacy

- All requests are directed to a front controller.
- A Symfony kernel is being bootet.
- In case Symfony does not recognize the route, the legacy application is run.

Routing via File System

Instead of a front controller, many legacy applications use many php scripts that are called directly.



Catching Requests with mod_rewrite

```
RewriteEngine On

RewriteCond %{REQUEST_URI}::$1 ^(/.+)/(.*)::\2$
RewriteRule ^(.*) - [E=BASE:%1]

RewriteCond %{ENV:REDIRECT_STATUS} ^$
RewriteRule ^app\.php(?:/(.*)|$) %{ENV:BASE}/$1 [R=301,L]

RewriteRule ^app\.php - [L]

RewriteCond %{REQUEST_FILENAME} -f
RewriteCond %{REQUEST_FILENAME} !^.+\.php$
RewriteRule ^ - [L]

RewriteRule ^ %{ENV:BASE}/app.php [L]
```

Emulate the Old Call

Global State: Direct call ≠ front controller

- PHP_SELF, SCRIPT_NAME and SCRIPT_FILENAME point to the front controller.
- Working directory is the directory of the front controller.
- PATH_INFO needs to be recalculated.

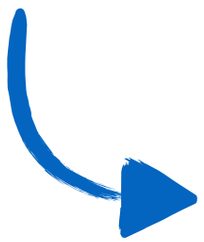
Front Controller with Legacy Bridge

```php
global $kernel;

require __DIR__.'/../app/autoload.php';

$kernel = new AppKernel(
    getenv('MYAPP_RUNTIME_ENV') === 'dev' ? 'dev' : 'prod',
    getenv('MYAPP_RUNTIME_ENV') === 'dev'
);

$request = Request::createFromGlobals();
$response = $kernel->handle($request);

$scriptFile = LegacyBridge::prepareLegacyScript($request, $response, __DIR__);
if (null !== $scriptFile) {
    require $scriptFile;
} else {
    $response->send();
}

$kernel->terminate($request, $response);
```

Kernel as Service Locator

The legacy application has access
to a fully booted Symfony kernel.

```php
$connection = mysql_connect($config['dbhost'], $config['dbuser'], $config['dbpassword']);
mysql_select_db($config['database'], $connection);
$res = mysql_query("SELECT * FROM my_table WHERE foo='$bar';", $connection);
```



```php
$connection = $kernel->getContainer()->get('doctrine.dbal.default_connection');
$res = $connection->executeQuery('SELECT * FROM my_table WHERE foo=?;', [$bar]);
```

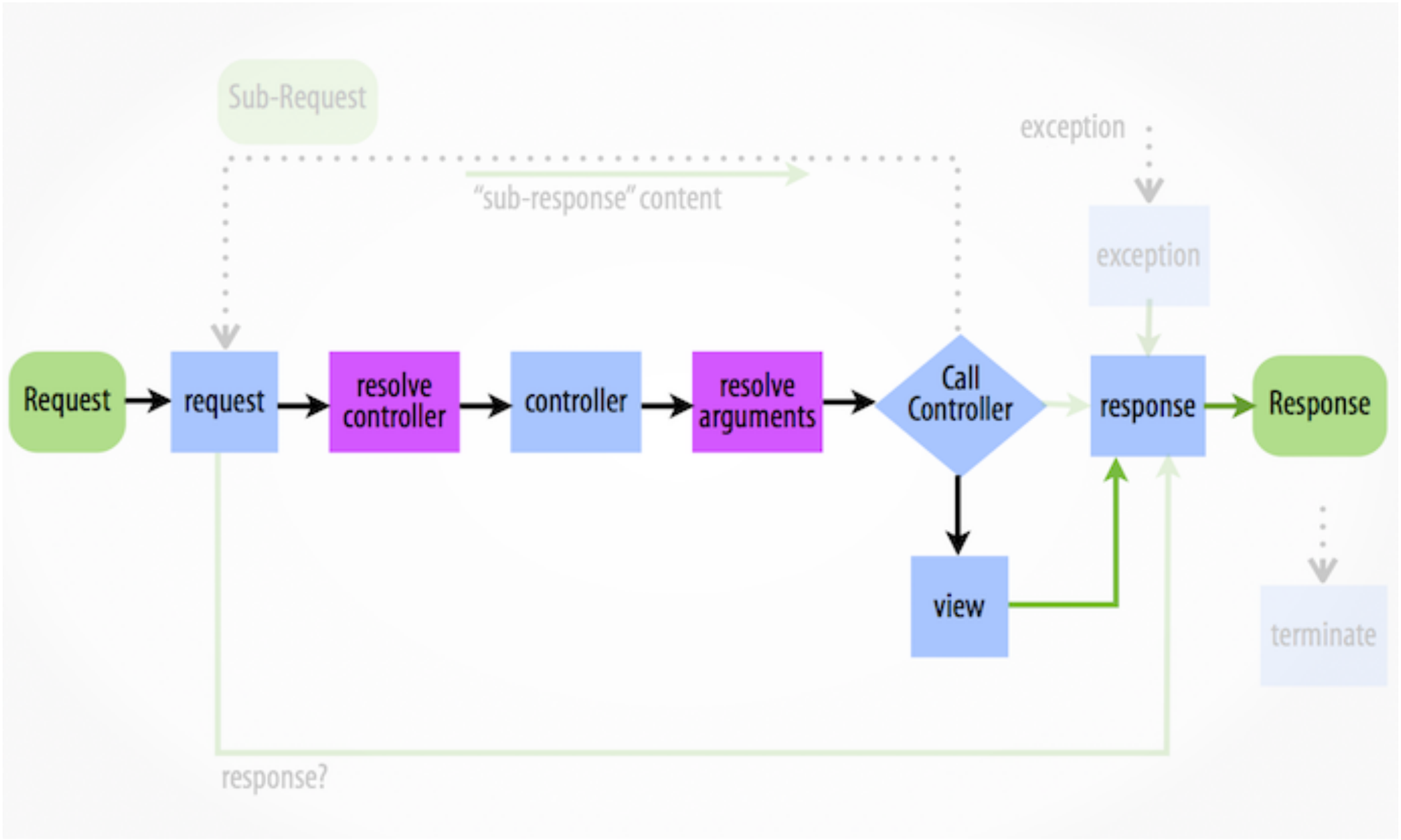
Authentication

After a successful login, many legacy applications write some representation of the user into the php session.

```php
if (
    isset($_POST['user'])
    && isset($_POST['password'])
    && check_login(
        $_POST['user'],
        $_POST['password']
    )
) {
    $_SESSION['username']
        = $_POST['user'];
}
```

Detect Authenticated Users

```php
class LegacySessionAuthenticator extends AbstractGuardAuthenticator
{
    public function getCredentials(Request $request)
    {
        return isset($_SESSION['username'])
            ? $_SESSION['username']
            : null;
    }

    public function getUser($credentials, UserProviderInterface $userProvider)
    {
        return $userProvider->loadUserByUsername($credentials);
    }

    public function checkCredentials($credentials, UserInterface $user)
    {
        return $user->getUsername() === $credentials;
    }

    // start(), onAuthenticationFailure()
    // onAuthenticationSuccess(), supportsRememberMe()
}
```



Symfony Event Cycle

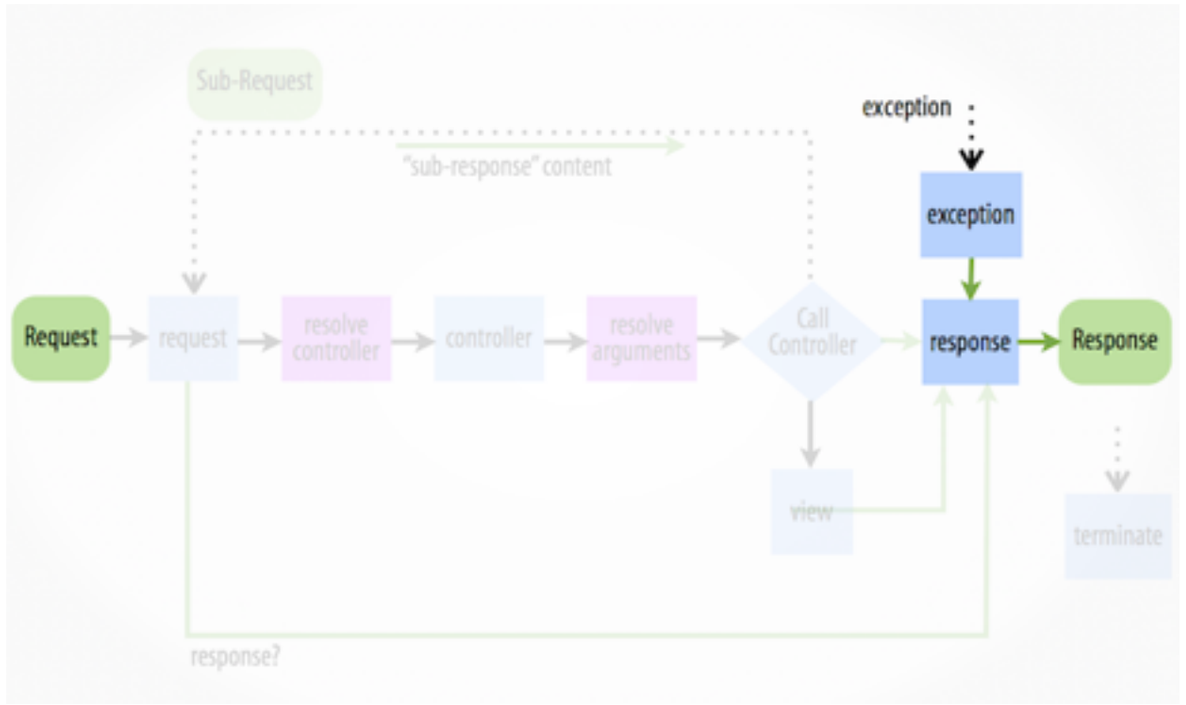
Priority	Listener
kernel.request	
2048	Symfony\Component\HttpKernel\EventListener**DebugHandlersListener**::configure() /Users/rabus/PhpstormProjects/dummy-app/vendor/symfony/symfony/src/Symfony/Component/HttpKernel/EventListener/DebugHandlersListener.php (line 63)
256	Symfony\Component\HttpKernel\EventListener**ValidateRequestListener**::onKernelRequest() /Users/rabus/PhpstormProjects/dummy-app/vendor/symfony/symfony/src/Symfony/Component/HttpKernel/EventListener/ValidateRequestListener.php (line 31)
128	Symfony\Bundle\FrameworkBundle\EventListener**SessionListener**::onKernelRequest() /Users/rabus/PhpstormProjects/dummy-app/var/cache/dev/classes.php (line 16)
48	Symfony\Component\HttpKernel\EventListener**FragmentListener**::onKernelRequest() /Users/rabus/PhpstormProjects/dummy-app/vendor/symfony/symfony/src/Symfony/Component/HttpKernel/EventListener/FragmentListener.php (line 56)
32	Symfony\Component\HttpKernel\EventListener**RouterListener**::onKernelRequest() /Users/rabus/PhpstormProjects/dummy-app/var/cache/dev/classes.php (line 2355)
16	Symfony\Component\HttpKernel\EventListener**LocaleListener**::onKernelRequest() /Users/rabus/PhpstormProjects/dummy-app/vendor/symfony/symfony/src/Symfony/Component/HttpKernel/EventListener/LocaleListener.php (line 47)
10	Symfony\Component\HttpKernel\EventListener**TranslatorListener**::onKernelRequest() /Users/rabus/PhpstormProjects/dummy-app/vendor/symfony/symfony/src/Symfony/Component/HttpKernel/EventListener/TranslatorListener.php (line 38)
8	Symfony\Component\Security\Http**Firewall**::onKernelRequest() /Users/rabus/PhpstormProjects/dummy-app/var/cache/dev/classes.php (line 3037)

The kernel.request Event

Symfony Firewall

Symfony authenticates <u>after</u>
the routing.

If the router throws an Exception, our firewall rules are not checked anymore.



LegacyRouteProvider

```php
public function getLegacyRoutes()
{
    $finder = new Finder();
    $finder->files()->name('*.php');
    $collection = new RouteCollection();

    /** @var SplFileInfo $file */
    foreach ($finder->in($this->webDir) as $file) {
        $collection->add(
            'my_app.legacy.' . str_replace('/', '__', substr($file->getRelativePathname(), 0, -4)),
            new Route(
                $file->getRelativePathname(),
                [
                    'legacyScript' => $file->getPathname(),
                    'requestPath' => '/' . $file->getRelativePathname()
                ]
            )
        );
    }

    return $collection;
}
```

Refactoring to Symfony Security

```php
if (!empty($_SESSION['is_admin'])) {
    // Display Admin stuff
}
```

```php
$authorizationChecker = $kernel->getContainer()
    ->get('security.authorization_checker');
if ($authorizationChecker->isGranted('ROLE_ADMIN')) {
    // Display Admin stuff
}
```

Wrapping the Legacy Application into a Controller

```php
class LegacyScriptController
{
    public function loadLegacyScriptAction($requestPath, $legacyScript)
     {
         return StreamedResponse::create(function () use ($requestPath, $legacyScript) {
             $_SERVER['PHP_SELF'] = $requestPath;
             $_SERVER['SCRIPT_NAME'] = $requestPath;
             $_SERVER['SCRIPT_FILENAME'] = $legacyScript;

             chdir(dirname($legacyScript));

             require $legacyScript;
         });
     }
}
```

Warning: The old script is not run inside the global variable scope anymore!

Wrapping Existing Controllers

```
{
    "class": "Acme\\MyApplication\\MyController",
    "method": "myControllerAction",
    "parameters": [
        "some parameter",
        42
    ]
}
```

```
class MyController
{
    public function myControllerAction(
        $firstParameter,
        $theAnswerToLife
    ) {
        // …
    }
}
```

Wrapping Existing Controllers

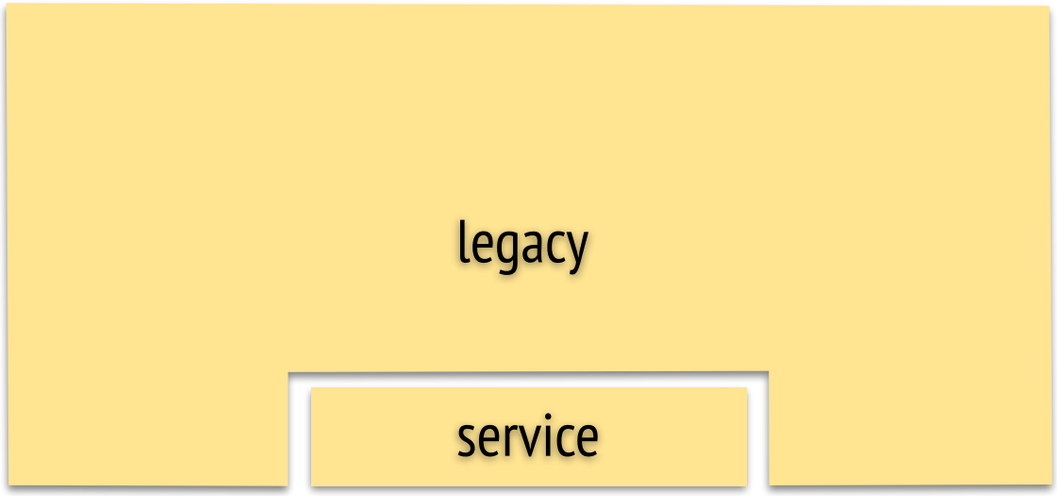
```php
class RequestMatcher implements RequestMatcherInterface
{
    public function matchRequest(Request $request)
    {
       $rpcCall = json_decode($request->getContent(), true);

        $controllerClass = $rpcCall['class'];
        $method = $rpcCall['method'];

        $reflectionMethod = new ReflectionMethod($controllerClass, $method);

        $parameters = [
            '_route' => 'rpc',
            '_controller' => $controllerClass . '::' . $method,
            '_format' => 'json',
        ];

        foreach ($reflectionMethod->getParameters() as $parameter) {
           $parameters[$parameter->getName()] = $rpcCall['data'][$parameter->getPosition()];
        }

        return $parameters;
    }
}
```

Wrapping Existing Controllers

```php
class ResponseListener implements EventSubscriberInterface
{
    public static function getSubscribedEvents()
    {
        return [
            KernelEvents::VIEW => 'onKernelView',
        ];
    }

    public function onKernelView(GetResponseForControllerResultEvent $event)
    {
        if ($event->getRequest()->attributes->get('_route') !== 'rpc') {
            return;
        }

        $result = $event->getControllerResult();
        $response = new Response();
        // …

        $event->setResponse($response);
    }
}
```

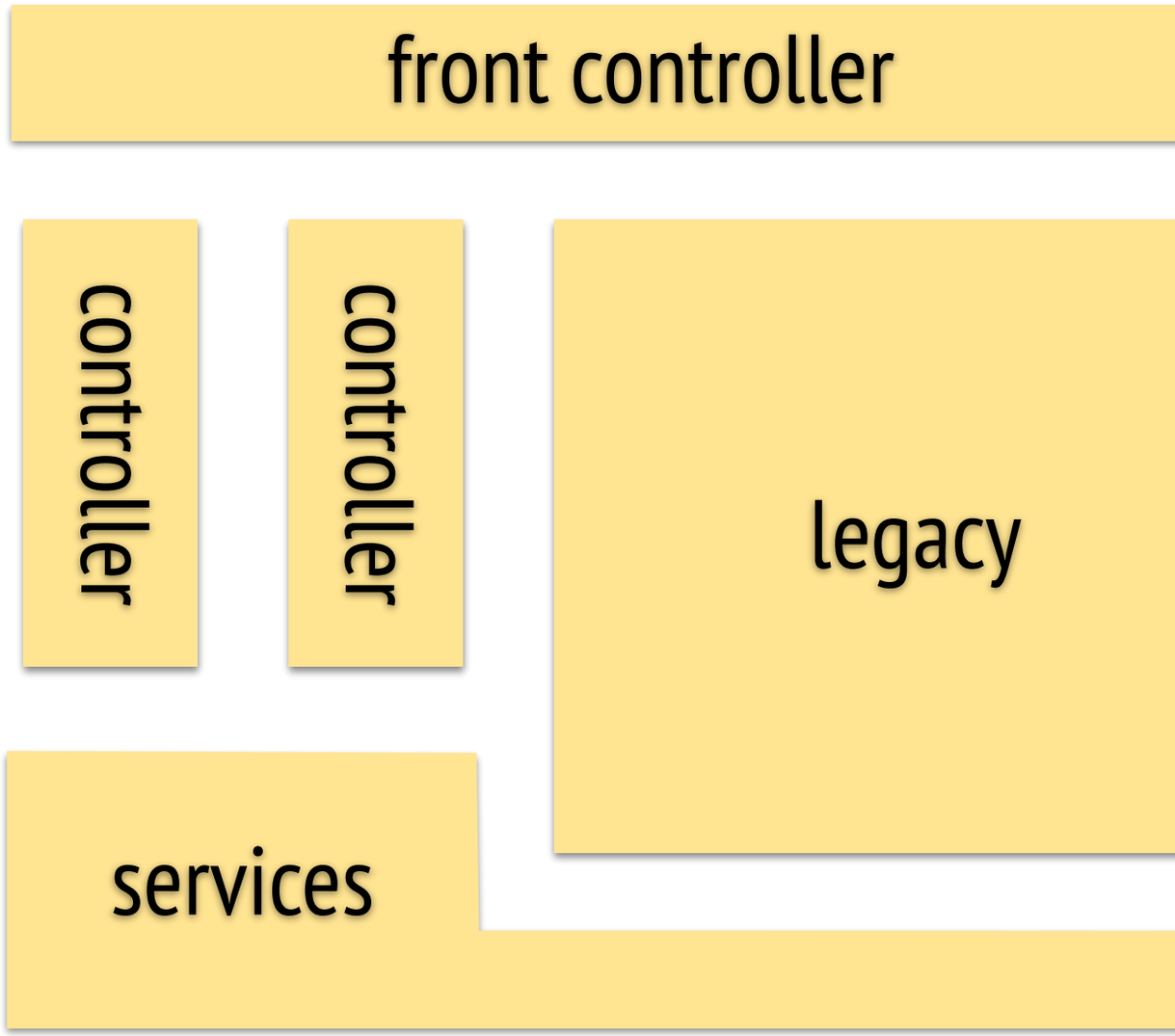
Refactoring to Services

- Refactor a legacy component into a Symfony service.
- Change all calling code, so the new service is used.
- Delete the legacy component.
- Repeat.



Refactoring to Controllers

- Turn a legacy script into a controller.
- Create a new route for the new controller.
- Delete the legacy script.
- Repeat.



Modernizing with Symfony

- Symfony as the framework for new and modernized application parts.
- Refactoring in small steps.
- Refactoring before changes.
- Avoid blocking refactorings.
- No never-ending rewrite.

Thank You

spam me:

me@derrabus.de

follow me:

@derrabus

hire me:

https://derrabus.de