Symfony2 for Beginners

Hi, my name is ...

Julius Beckmann

and i work at

... and i do:

- PHP, Symfony2
- Erlang, Elixir
- NodeJS
- DevOps + Continuous Everything!

(github|twitter).com/h4cc

Agenda

Agenda 1/2

- What is Symfony?

Agenda 1/2

- What is Symfony?
- Howto
  - Install
  - Develop
  - Deploy

Agenda 2/2

- Plain PHP to Symfony
  - Starting Point
  - Dispatching
  - Routing
  - Symfony HTTP Foundation
  - Symfony HTTP Kernel
  - Symfony Routing
  - Symfony Controller

Agenda 2/2

- Plain PHP to Symfony
  - Starting Point
  - Dispatching
  - Routing
  - Symfony HTTP Foundation
  - Symfony HTTP Kernel
  - Symfony Routing
  - Symfony Controller
- Extending Symfony Kernel
  - Kernel Events
  - Request/Response Flow
  - (Subrequests)

What is Symfony?

What is Symfony?

- Abstract:
  - PHP Web Application Framework
  - At least PHP 5.3.3
  - By Fabien Potencier *(fabpot)*

What is Symfony?

- Abstract:
  - PHP Web Application Framework
  - At least PHP 5.3.3
  - By Fabien Potencier *(fabpot)*
- Versions:
  - v1 since 2005 *(legacy)*
  - v2 since 2011
  - v3 in November 2015

- Patterns:
  - Framework Components
  - Modular design (Bundles)
  - Dependency Injection
  - Event Dispatcher

- Patterns:
  - Framework Components
  - Modular design (Bundles)
  - Dependency Injection
  - Event Dispatcher

- Used by:
  - Drupal 8
  - eZPublish 5
  - Contao 4
  - phpBB 3
  - …

Installing Symfony

Symfony Installer

A tool for bootstrapping new Symfony projects.

Symfony Installer

A tool for bootstrapping new Symfony projects.

Installing the installer

```
curl -LsS http://symfony.com/installer -o symfony.phar

chmod a+x symfony.phar
```

Symfony Installer

A tool for bootstrapping new Symfony projects.

Installing the installer

```
curl -LsS http://symfony.com/installer -o symfony.phar

chmod a+x symfony.phar
```

Using the installer

```
php symfony.phar new blog

php symfony.phar new blog 2.3

php symfony.phar new blog 2.5.2
```

Composer

Manual way of bootstrapping symfony projects.

```
php composer.phar create-project \
    symfony/framework-standard-edition \
    blog "2.3.*"
```

Composer

Manual way of bootstrapping symfony projects.

```
php composer.phar create-project \
    symfony/framework-standard-edition \
    blog "2.3.*"
```

This will do:

1. `git clone https://github.com/symfony/symfony-standard blog`
2. `cd blog`
3. `git checkout v2.3.X`
4. `rm -rf .git`
5. `composer install`

Development

Development

Using the PHP >=5.4 internal Webserver

```
php app/console server:run
# or
php -S 127.0.0.1:8080 -t web/
```

Development

Using the PHP >=5.4 internal Webserver

```
php app/console server:run
# or
php -S 127.0.0.1:8080 -t web/
```

Frontend Tools

- `php app/console assets:install`
- `php app/console assetic:dump`
- Grunt, Gulp, Brunch

Deployment

Deployment

Poor-Man example:

Initial

```
git clone git@github.com:your/project.git /var/www
cd /var/www
# ensure correct permissions on app/cache and app/logs
```

Deployment

Poor-Man example:

Initial

```
git clone git@github.com:your/project.git /var/www
cd /var/www
# ensure correct permissions on app/cache and app/logs
```

Deploy

```
git pull
rm -f web/app_*.php
composer install
php app/console cache:clear --env=prod
# ... more needed commands like "doctrine"
```

Tools: Capifony, Ansible, SaltStack, Puppet, Chef.
Automation with: Jenkins.

From plain PHP to Symfony

Starting point

GET /post.php?id=1

```php
<?php // post.php

$postId = (int)$_GET['id'];
$posts = array(1 => 'My first post');

if(!isset($posts[$postId])) {
    header("HTTP/1.0 404 Not Found");
    die('Post not found');
}

$post = $posts[$postId];
echo '<h1>Post #', $postId, '</h1>
      <p>', $post, '</p>';
```

Webserver Rewrites

Using rewrites

GET /post/1

.htaccess Configuration

```
RewriteEngine on
RewriteRule ^post/([0-9]+)$ post.php?id=$1
```

Using rewrites

GET /post/1

.htaccess Configuration

```
RewriteEngine on
RewriteRule ^post/([0-9]+)$ post.php?id=$1
```

Problems

- Depends on Webserver
- Static configuration
- *Outside* of Application

Dispatcher and Routing

Using dispatcher

.htaccess Configuration

```
RewriteEngine On
RewriteRule ^(.*)$  /index.php [QSA]
```

Using dispatcher

.htaccess Configuration

```
RewriteEngine On
RewriteRule ^(.*)$  /index.php [QSA]
```

index.php with Routing

```php
<?php // index.php

$path = $_SERVER['PATH_INFO'];

// Routing
if(preg_match("@^/post/(?P<id>[0-9]+)@", $path, $m)){
    $_GET['id'] = $m['id'];
    require('controller/post.php');
}else{
    require('controller/index.php');
}
```

Symfony HTTP Foundation

1. Using HTTP as a **Interface**

Symfony HTTP Foundation

1. Using HTTP as a **Interface**
2. **Hiding** PHP-ServerAPI

   → $_GET, $_POST, $_SERVER, echo(), header(), die() ...

Symfony HTTP Foundation

1. Using HTTP as a **Interface**
2. **Hiding** PHP-ServerAPI

   → $_GET, $_POST, $_SERVER, echo(), header(), die() ...
3. **Providing** Request/Response Objects

Symfony Request Class

```php
<?php
use Symfony\Component\HttpFoundation\Request;

$_GET['foo'] = 42;
$_POST['bar'] = 1337;

$request = Request::createFromGlobals();

$request->query->get('foo'); // from _GET
$request->request->get('bar'); // from _POST

$request->get('foo'); // from any source.
```

Encapsulates `$_GET`, `$_POST`, `$_COOKIE`, `$_FILES` and `$_SERVER`

Symfony Response Class

```php
<?php
use Symfony\Component\HttpFoundation\Response;

return new Response('Hello World', 200);
```

Symfony Response Class

```php
<?php
use Symfony\Component\HttpFoundation\Response;

return new Response('Hello World', 200);
```

- Capsules response content, status and headers.
- Some Helper:
  - `JsonResponse(array('answer' => 42))`
  - `RedirectResponse('http://example.com/', 302)`
  - `StreamedResponse(function() {...})`
  - `BinaryFileResponse('images/42.jpg')`

Benefits

1. Reproducible → **Testable**

Benefits

1. Reproducible → **Testable**
2. Common Interface → **Reuseable**

Symfony HTTP Kernel

Http Kernel Interface

```php
<?php
interface Symfony\Component\HttpKernel\HttpKernelInterface {
    const MASTER_REQUEST = 1; // External request (from browser).
    const SUB_REQUEST = 2;    // Internal request.

    /** @return Response */
    public function handle(Request $request, $type=1, $catch=true);
}
```

A HttpKernel **has** to transform `Request` to `Response`.

HTTP Kernel

```php
<?php // index.php

class ExampleKernel implements HttpKernelInterface {
    public function handle(Request $req) {

        // Step 1: Routing
        $controller = $this->routeRequestToController($req);

        // Step 2: ?
        $response = $this->callRequestOnController($controller, $req);

        // PROFIT!!!
        return $response;
    }
    // ...
}

$kernel = new ExampleKernel();
$response = $kernel->handle(Request::createFromGlobals());
$response->send();
```

Symfony Routing

Symfony Routing

Goal

Find Controller and Action for given Request.

Symfony Routing

Goal

Find Controller and Action for given Request.

Example Blog Post Route

```
# app/config/routing.yml

blog_post_show:                 # Name of route
    path: /post/{id}                # Pattern with placeholders
    defaults:
        _controller: BlogBundle:Post:show
    requirements:
        id: \d+                     # Regex possible
        _method: GET                # Ensure used method
```

Symfony Routing - Generate URLs

From PHP using “router”

```php
<?php
$router = $this->get('router');

$path = $router->generate('blog_post_show', ['id' => 42]);
$url = $router->generate('blog_post_show', ['id' => 42], true);
```

Symfony Routing - Generate URLs

From PHP using “router”

```php
<?php
$router = $this->get('router');

$path = $router->generate('blog_post_show', ['id' => 42]);
$url = $router->generate('blog_post_show', ['id' => 42], true);
```

From Twig using helper

```
<a href="{{ path('blog_post_show', {'id': 42}) }}">
    Relative URL
</a>

<a href="{{ url('blog_post_show', {'id': 42}) }}">
    Absolute URL
</a>
```

Symfony Routing - Tipps

Routing to static page

```
imprint:
    path: /imprint
    defaults:
        _controller: FrameworkBundle:Template:template
        template:    static/imprint.html.twig
```

Symfony Routing - Tipps

Routing to static page

```
imprint:
    path: /imprint
    defaults:
        _controller: FrameworkBundle:Template:template
        template:    static/imprint.html.twig
```

Routing to static redirect

```
admin_shortcut:
    path: /admin
    defaults:
        _controller: FrameworkBundle:Redirect:urlRedirect
        path:        /backend/administration/login
        permanent:   false
```

Symfony Controller

Symfony Controller

$$\textit{Request} \rightarrow \text{Controller} \rightarrow \textit{Response}$$

Symfony Controller

$$Request \rightarrow \text{Controller} \rightarrow Response$$

```php
<?php
class HelloController {
    public function worldAction(Request $request) {

        // Translate Request to application logic here ...

        $content = 'Hello '.$request->get('name', 'World');

        // ... and create Response for result.

        return new Response($content, 200);
    }
}
```

post.php as Symfony Controller

PostController

```php
<?php // src/BlogBundle/Controller/PostController.php

class PostController extends Controller {
    public function showAction(Request $request) {

        $id = $request->get('id');

        $post = $this->getDoctrine()
            ->getRepository('BlogBundle:Post')
            ->find($id);

        if(!$post) {
            // Using exceptions here.
            throw $this->createNotFoundException('Post not found');
        }

        return new Response('<h1>Post #'. $post->getId(). '</h1>
                            <p>'. $post->getContent(). '</p>', 200);
    }
}
```

Parameter injection

Parameter injection

```php
<?php // src/BlogBundle/Controller/PostController.php

class PostController extends Controller {
    public function showAction($id) { // <-- Parameter from route

        $post = $this->getDoctrine()
            ->getRepository('BlogBundle:Post')
            ->find($id);

        if(!$post) {
            throw $this->createNotFoundException('Post not found');
        }

        return new Response('<h1>Post #'. $post->getId(). '</h1>
                            <p>'. $post->getContent(). '</p>', 200);
    }
}
```

Templating

Templating

```php
<?php // src/BlogBundle/Controller/PostController.php

class PostController extends Controller {
    public function showAction($id) {

        $post = $this->getDoctrine()
            ->getRepository('BlogBundle:Post')
            ->find($id);

        if(!$post) {
            throw $this->createNotFoundException('Post not found');
        }

        // Using template system
        return $this->render('post/show.html.twig',  ['post' => $post]);
    }
}
```

ParamConverter Annotation

ParamConverter Annotation

```php
<?php // src/BlogBundle/Controller/PostController.php

use Sensio\Bundle\FrameworkExtraBundle\Configuration\ParamConverter;

class PostController extends Controller {
    /**
     * @ParamConverter()
     */
    public function showAction(Post $post) { // Post from DB or 404

        return $this->render('post/show.html.twig',  ['post' => $post]);
    }
}
```

Template Annotation

Template Annotation

```php
<?php // src/BlogBundle/Controller/PostController.php

use Sensio\Bundle\FrameworkExtraBundle\Configuration\ParamConverter;

class PostController extends Controller {
    /**
     * @ParamConverter()
     * @Template("post/show.html.twig")
     */
    public function showAction(Post $post) {
        return ['post' => $post];
    }
}
```

Template Annotation

```php
<?php // src/BlogBundle/Controller/PostController.php

use Sensio\Bundle\FrameworkExtraBundle\Configuration\ParamConverter;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Template;

class PostController extends Controller {
    /**
     * @ParamConverter()
     * @Template("post/show.html.twig", vars={"post"})
     */
    public function showAction(Post $post) {}
}
```

Route Annotation

Route Annotation

```php
<?php // src/BlogBundle/Controller/PostController.php

use Sensio\Bundle\FrameworkExtraBundle\Configuration\ParamConverter;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Template;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;

class PostController extends Controller {
    /**
     * @Route("/show/{id}", requirements={"id"="\d+", "_method"="GET"})
     * @ParamConverter()
     * @Template(vars={"post"})
     */
    public function showAction(Post $post) {}
}
```

Extending the Symfony Kernel

Kernel Events

Symfony is using Events to process Requests.

Kernel Events

Symfony is using Events to process Requests.

Every Request

- **kernel.request**
- **kernel.controller**
- **kernel.response**

Kernel Events

Symfony is using Events to process Requests.

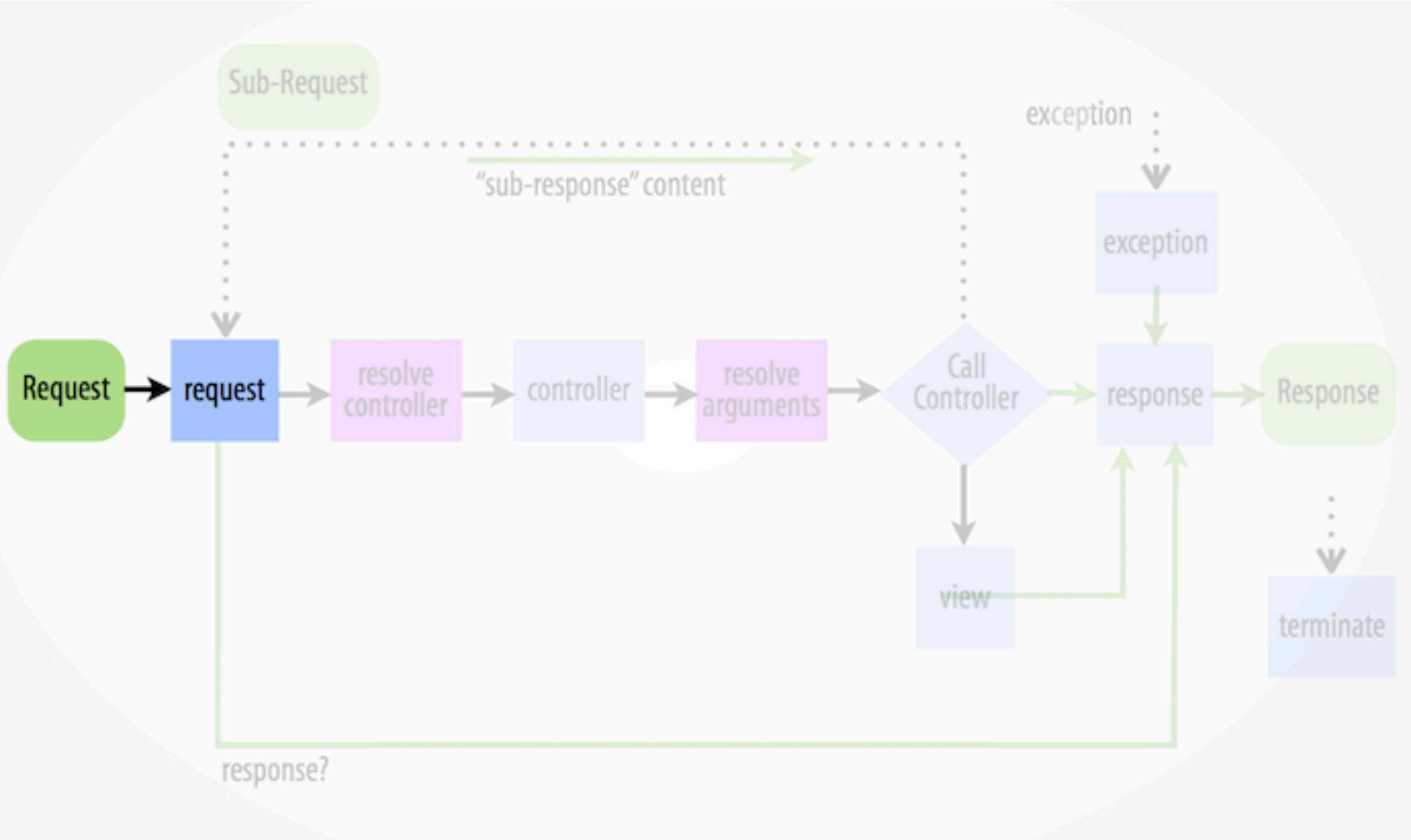
Every Request

- **kernel.request**
- **kernel.controller**
- **kernel.response**

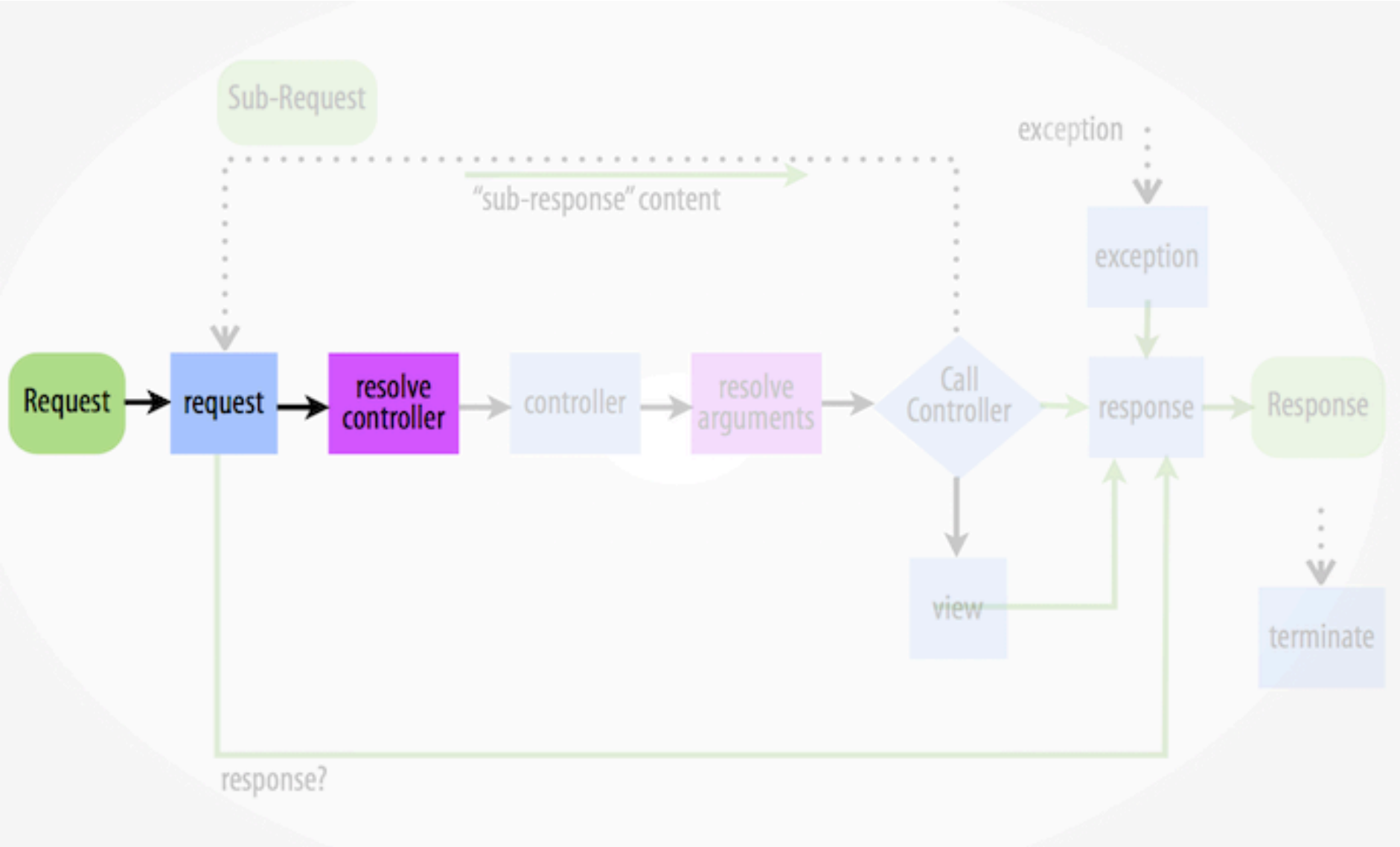
Optional

- **kernel.view**
- **kernel.exception**
- **kernel.terminate**

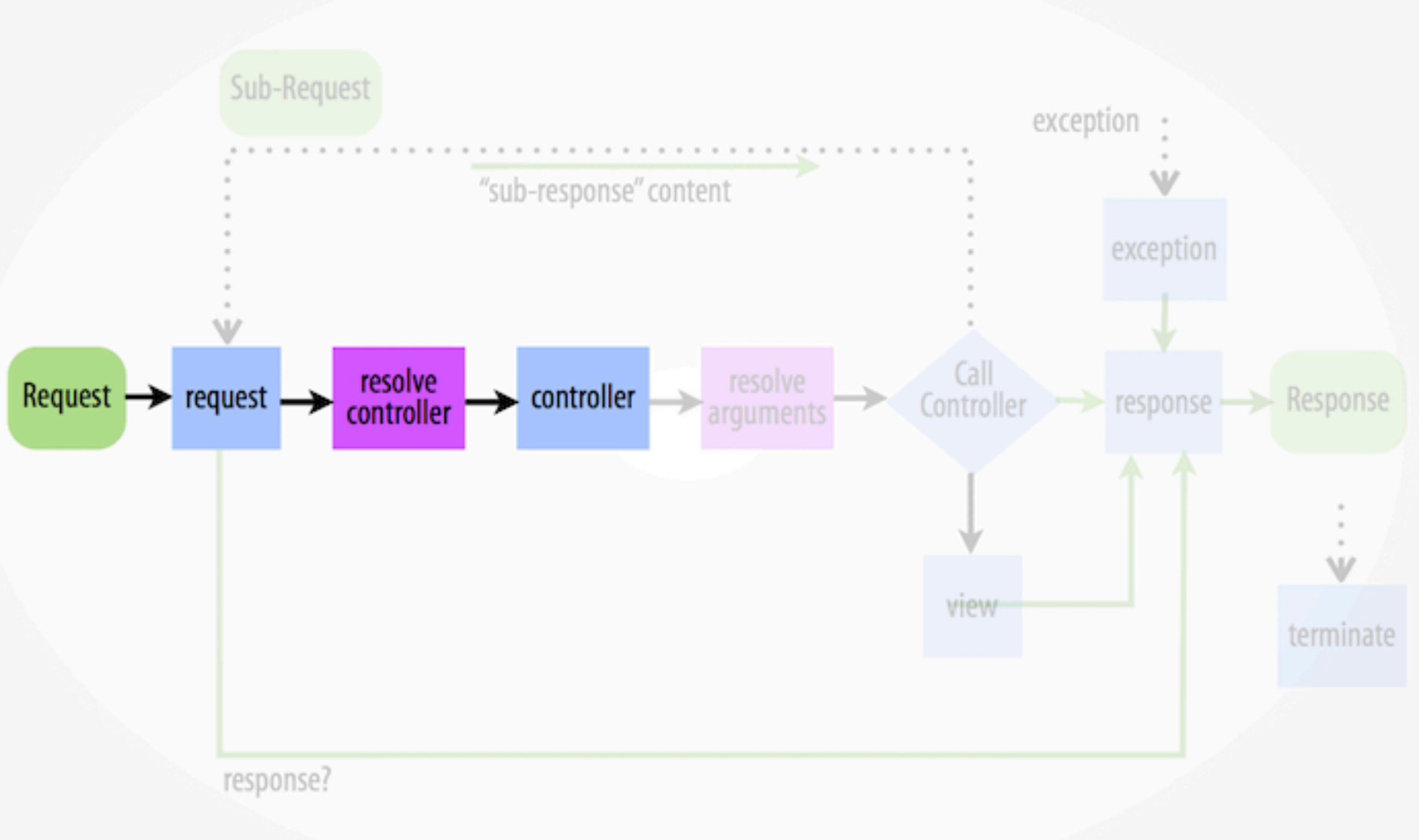
Event kernel.request



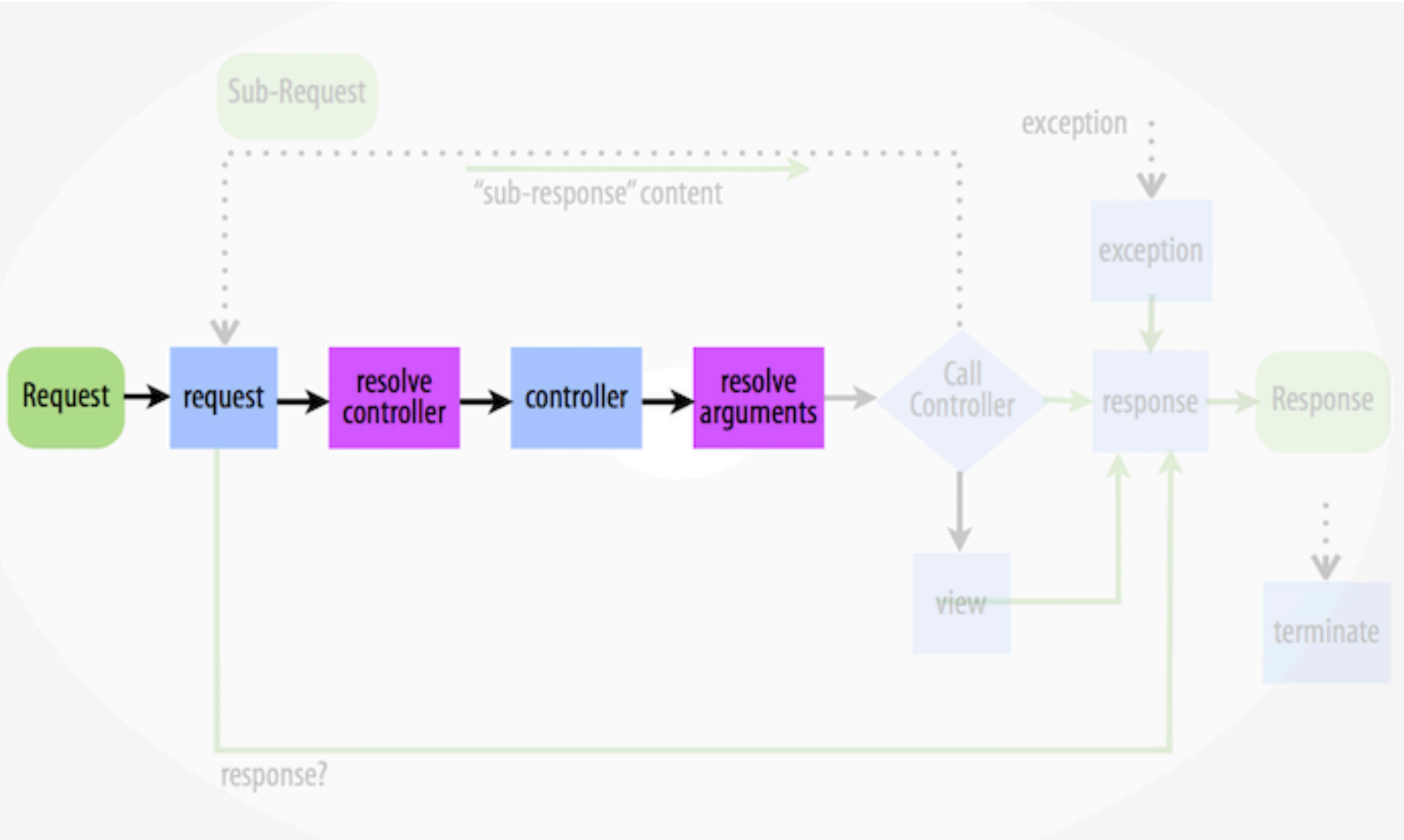
Resolve Controller



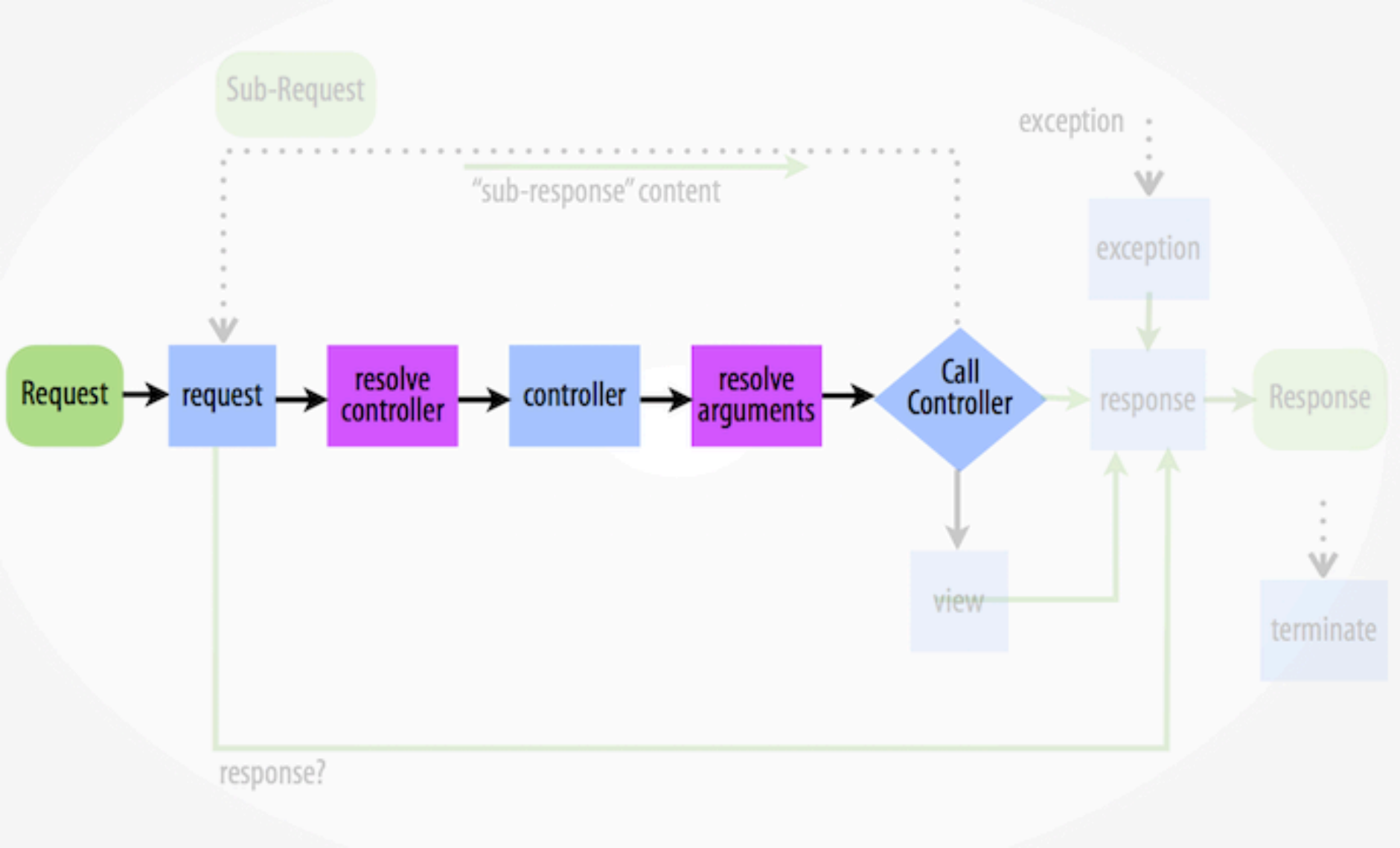
Event kernel.controller



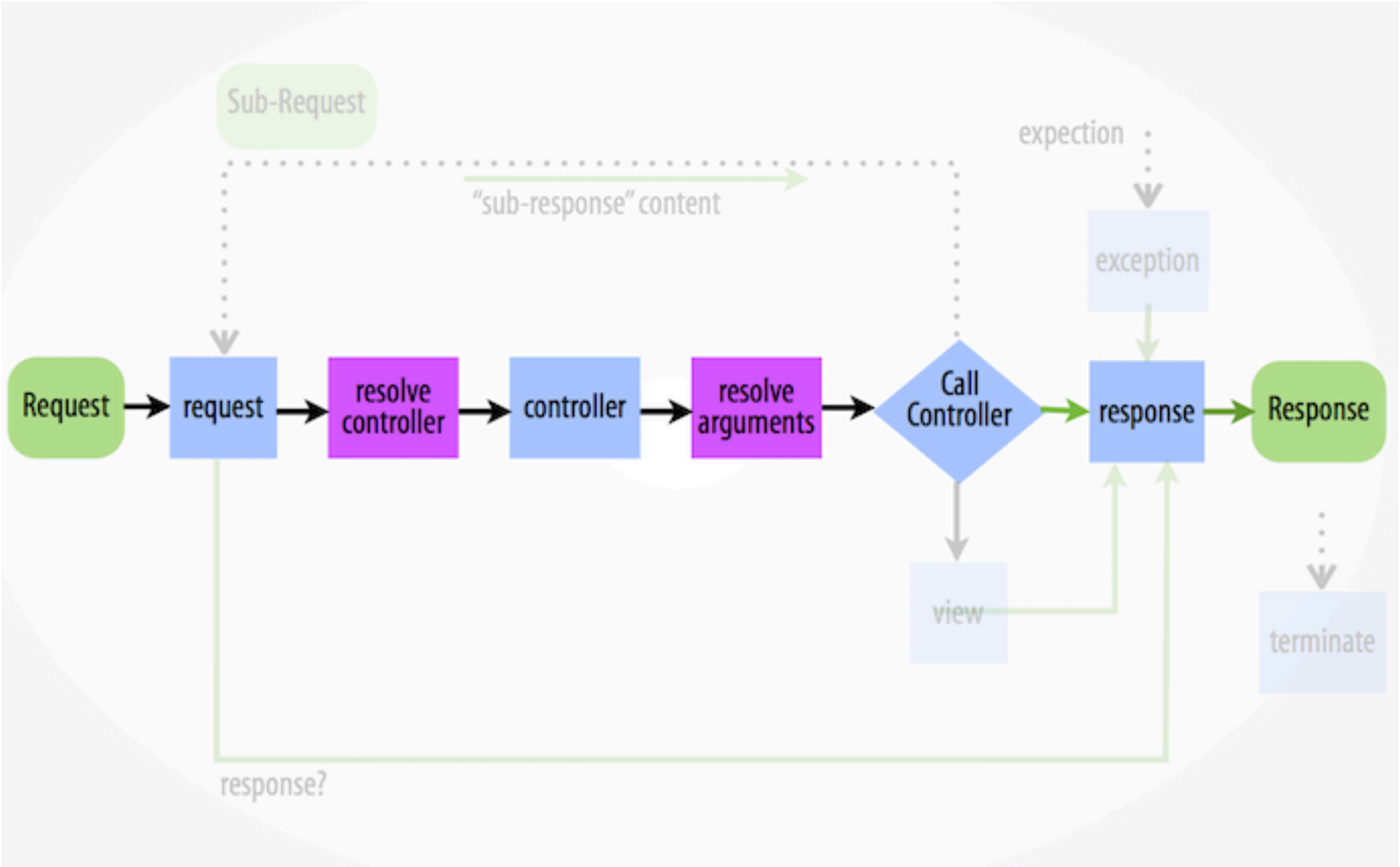
Controller Arguments



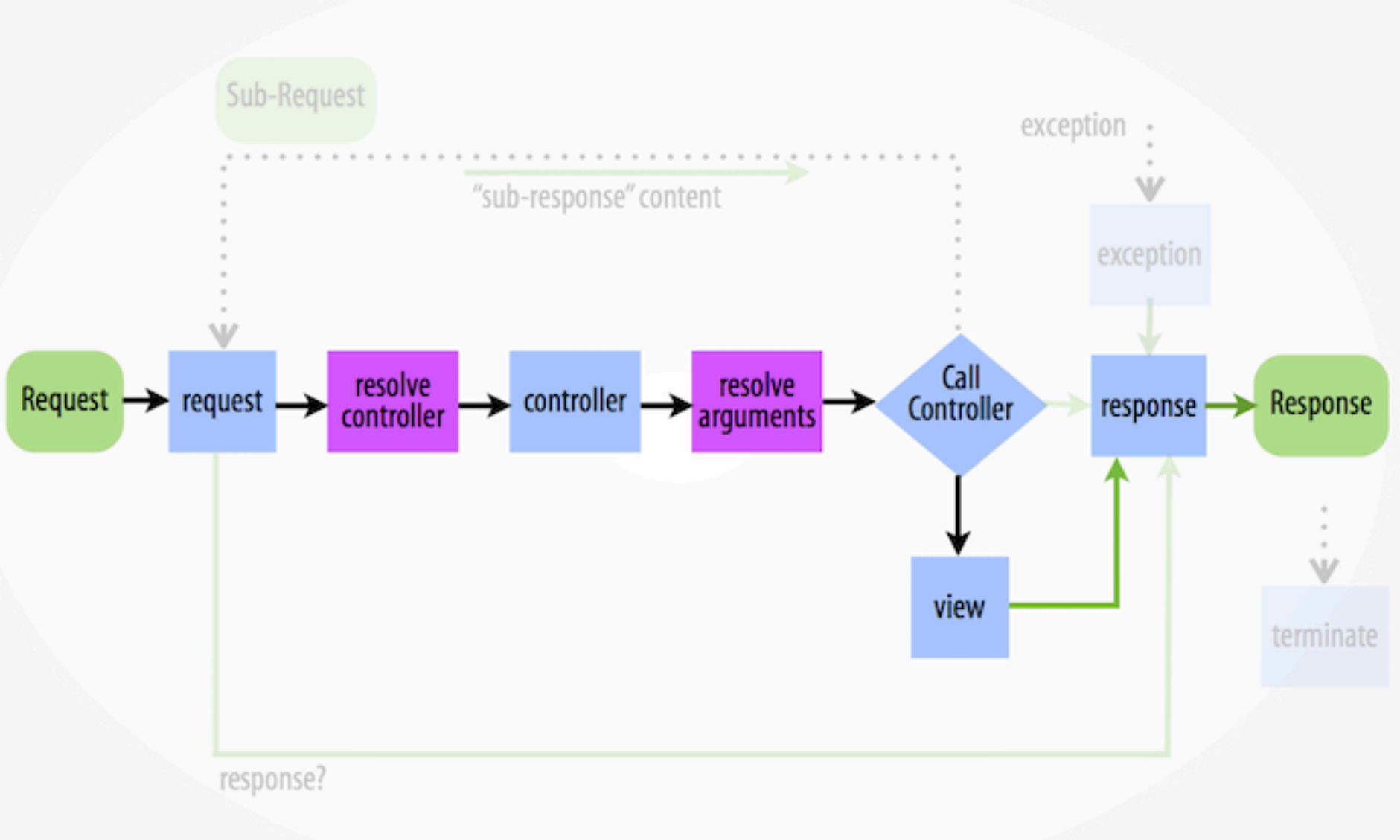
Call Controller



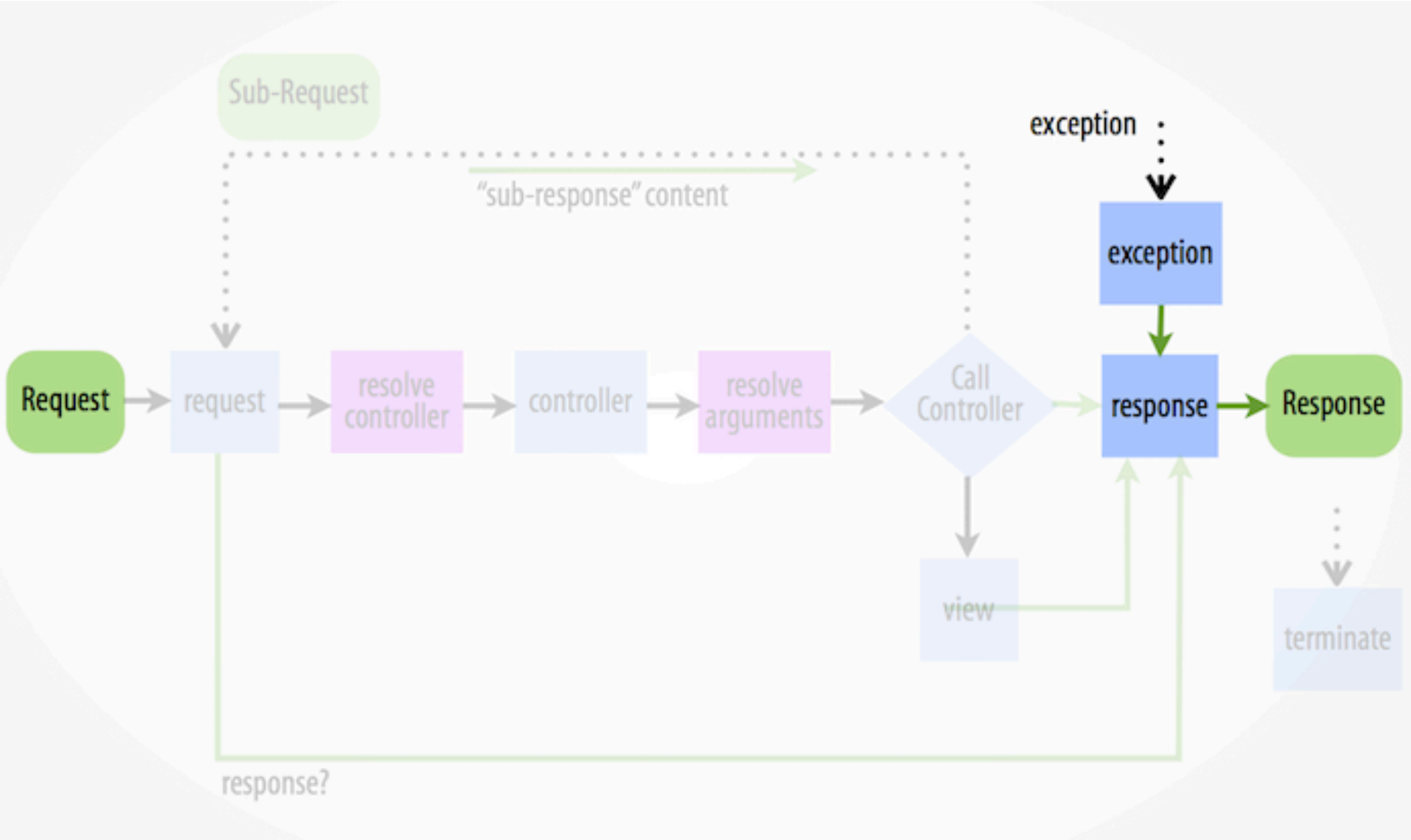
Call Controller



Event kernel.view



Event kernel.exception



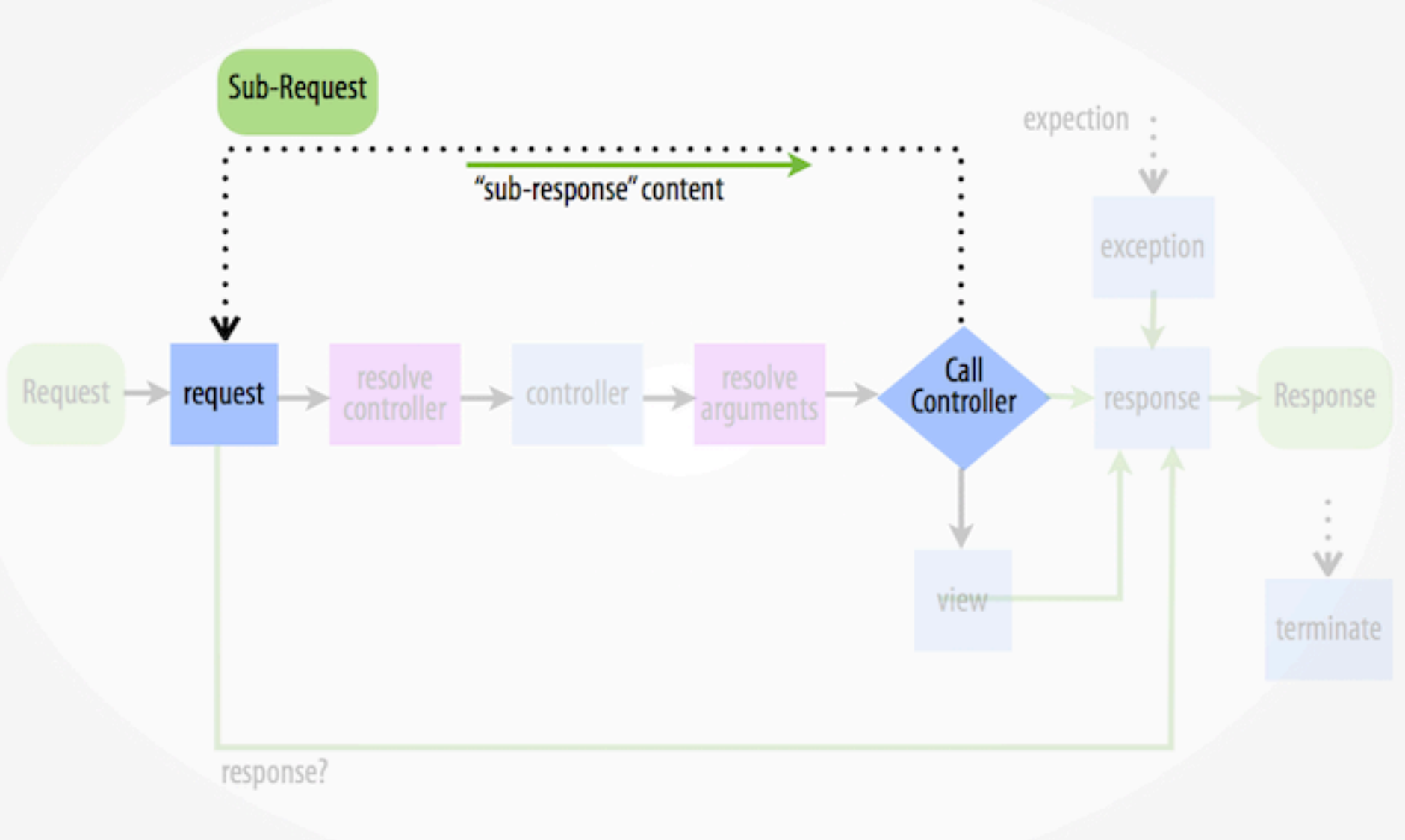
Symfony HttpKernel Events

```php
<?php // Example implementation of symfony kernel-events.
class HttpKernel implements HttpKernelInterface, TerminableInterface {
  public function handle(Request $req) {
    try {
      $gre = $this->dispatch('kernel.request', new GetResponseEvent($req));
      if($gre->hasResponse()) {
        $resp = $gre->getResponse();
        goto response;
      }

      $this->resolveController($req);
      $controller = $this->dispatch('kernel.controller', new FilterControllerEvent($req));
      $this->resolveArguments($req);

      $resp = $this->callController($controller, $req);

      if(!$resp instanceof Response) {
        $grfcre = new GetResponseForControllerResultEvent($req, $resp);
        $resp = $this->dispatch('kernel.view', $grfcre)->getResponse();
      }

    }catch(Exception $e) {
      $resp = $this->dispatch('kernel.exception', new GetResponseForExceptionEvent($req, $e))->getResponse();
    }

    response:
      return $this->dispatch('kernel.response', new FilterResponseEvent($req, $resp))->getResponse();
  }

  public function terminate(Request $req, Response $resp) {
    $this->dispatch('kernel.terminate', new PostResponseEvent($req, $resp));
  }
}
```

Questions?

Symfony Sub-Requests

Sub-Requests: Flow



Sub-Requests: Example

Forwarding

```php
<?php
public function impressumAction($name) {
    $response = $this->forward('ExampleContentBundle:Static:imprint');
}
```

Sub-Requests: Example

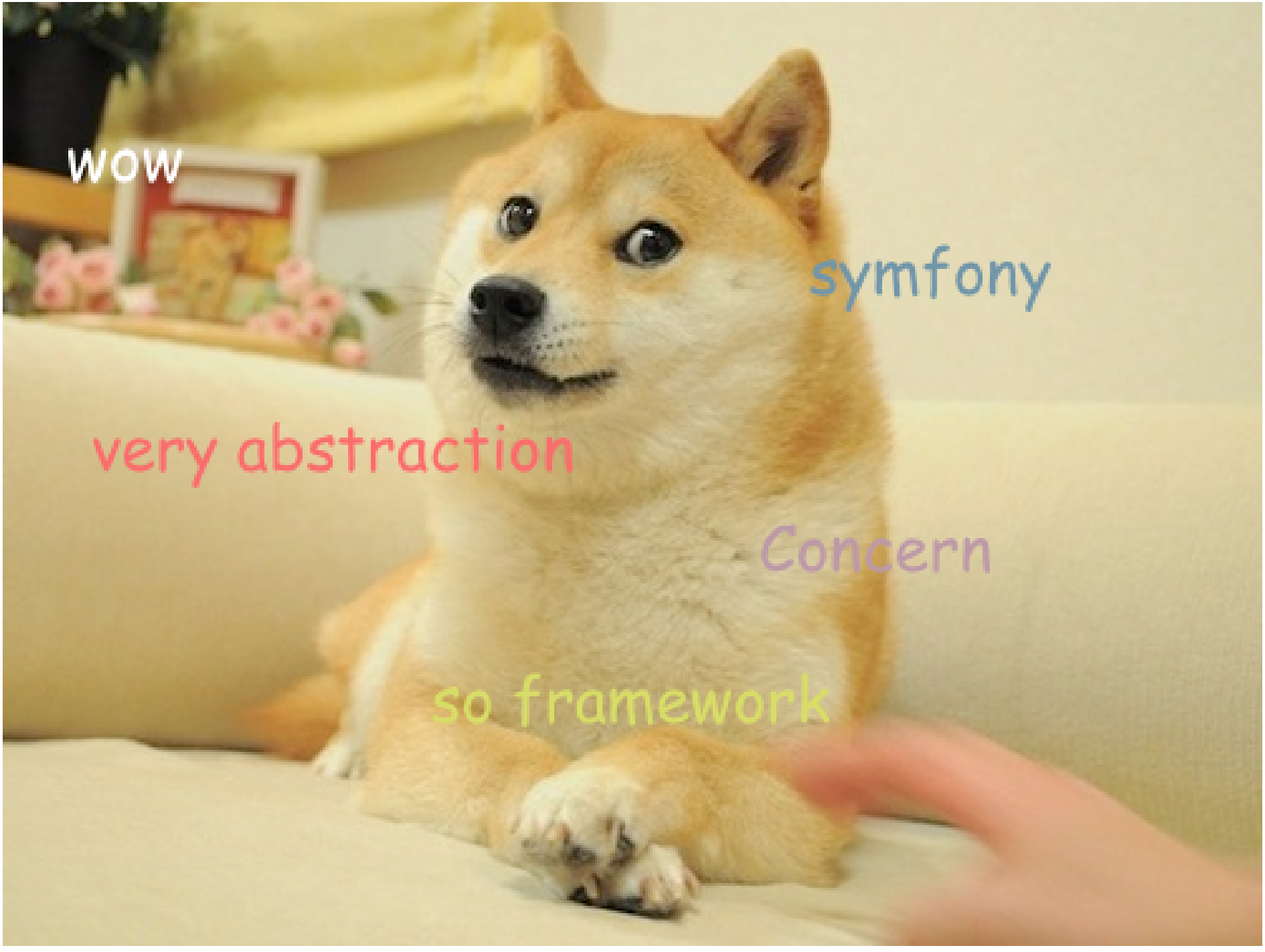
Forwarding

```php
<?php
public function impressumAction($name) {
    $response = $this->forward('ExampleContentBundle:Static:imprint');
}
```

Rendering Templates

```
{{ render(url('/partial/footer.html')) }}

{# ... or ... #}

{{ render(controller('ExampleContentBundle:Partial:footer')) }}
```

The end!


wow
symfony
very abstraction
Concern
so framework