Hugo Hamon
@hhamon

Silex meets REST and SOAP

What is Silex?

ELLIX
GEHÄRTETER STAHL


return $this->handleRaw($
try {
return $this->handleRaw($
} catch (\Exception $e) {
if (false === $catch) {
throw $e;
}
return $this->handleException($e, $request, $type);
}
}
}
request to convert it to a response.
* Exceptions are not caught.
*
* @param Request $request A Request instance
* @param integer $type The type of the request (one of H
*
* @return Response A Response instance
*
* @throws \LogicException If one of the listener does not
Kernel.php
Soft Tabs: 4

Symfony

http://silex.sensiolabs.org

Why choosing Silex?

What's inside?

The Silex Philosophy

silex.phar

cache/

logs/

src/

tests/

vendor/

web/

Silex Mantra

```
namespace Symfony\Component\HttpKernel;

interface HttpKernelInterface
{
    (Response) function handle(Request $request);
}
```

Request Handling

```php
require_once __DIR__.'/silex.phar';

$app = new Silex\Application();

$app->get('/hello/{name}', function($name) use($app) {
    return 'Hello '. $app->escape($name);
});

$app->run();
```

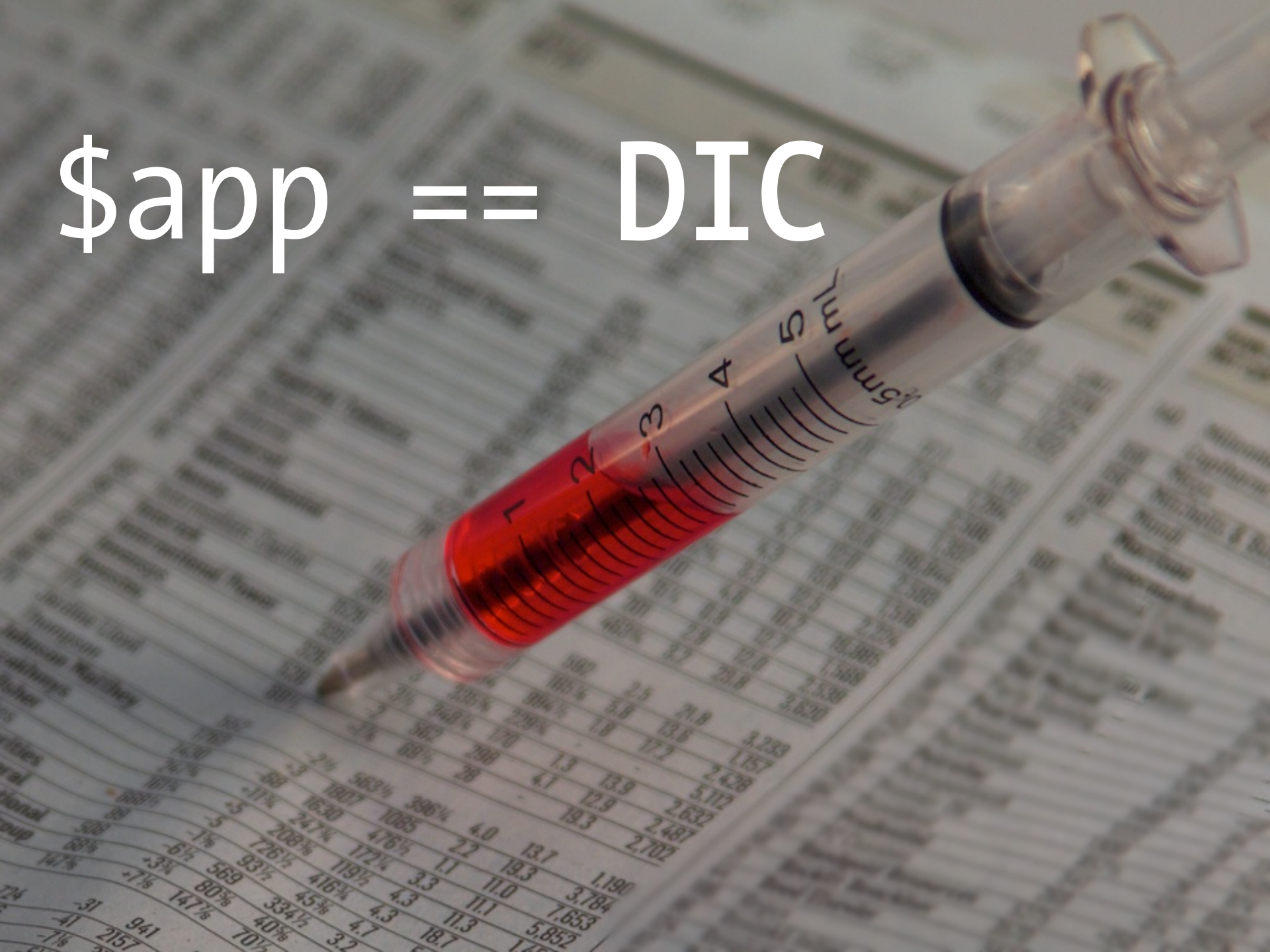
Request Handling

```php
require_once __DIR__.'/silex.phar';

use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;

$app = new Silex\Application();

$app->get('/hello/{name}', function(Request $request) use($app) {
    $name = $request->attributes->get('name');
    return new Response('Hello '. $app->escape($name));
});
```


$app == DIC

Deploying REST Web services

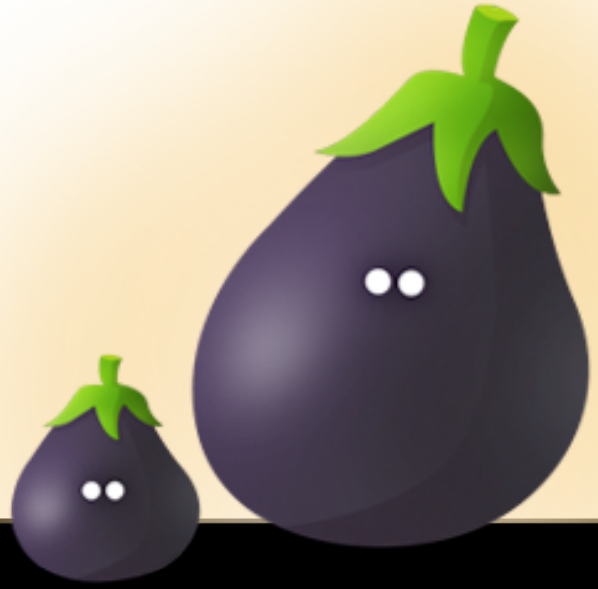
REpresentational

State

Transfer

Architecture style

Designing REST APIs



```php
$app->get('/events', function (Request $request) {

    $events = array(
        array('name' => 'OSIDays', 'venue' => 'Bangalore'),
        array('name' => 'PHP Tour', 'venue' => 'Lille'),
        array('name' => 'Confoo', 'venue' => 'Montreal'),
        // ...
    );

    return new Response(json_encode($events), 200, array(
        'Content-Type' => 'application/json'
    ));
});
```

```php
$app->post('/event', function (Request $request) use ($app) {

    // Get POST data or 400 HTTP response
    if (!$data = $request->get('event')) {
        return new Response('Missing parameters.', 400);
    }

    // Persist data to the database
    $event = new Event()
    $event->title = $data['title'];
    $event->venue = $data['venue'];
    $event->save();

    // Trigger redirect to the newly created record URI
    return $app->redirect('/event/'. $event->id, 201);
});
```

```php
$app->put('/event/{id}', function ($id) use ($app) {

    if (!$data = $request->get('event')) {
        return new Response('Missing parameters.', 400);
    }

    if (!$event = $app['event_manager']->find($id)) {
        return new Response('Event not found.', 404);
    }

    $event->title = $data['title'];
    $event->venue = $data['venue'];
    $event->save();

    return new Response('Event updated.', 200);
});
```

```php
$app->delete('/event/{id}', function ($id) use ($app) {

    $event = $app['event_manager']->find($id);

    if (!$event) {
        return new Response('Event not found.', 404);
    }

    $event->delete();

    return new Response('Event deleted.', 200);
});
```

Advanced routing

```php
$app->get('/archive/{year}/{month}', function ($month, $year) {

    // ...

})

->bind('archives')                  // Route name


->value('year', date('Y'))          // Default parameter value
->value('month', date('m'))


->assert('year', '\d{4}')           // Parameter format
->assert('month', '\d{2}');
```

Events management

```php
$app->before(function (Request $request) use ($app) {

    $user = $request->server->get('PHP_AUTH_USER');
    $pwd  = $request->server->get('PHP_AUTH_PW');

    if (
        $app['api_user'] !== $user
        ||
        $app['api_pwd'] !== $pwd
    ) {
        return new Response('Unauthorized', 403);
    }
});
```

```php
$app->after(function (Request $request, Response $response) {

    // Get URI parameter to determine requested output format
    $format = $request->attributes->get('format');

    switch ($format) {
        case 'xml':
            $response->headers->set('Content-Type', 'text/xml');
            break;
        case 'json':
            $response->headers->set('Content-Type', 'text/json');
            break;
        default:
            $response->headers->set('Content-Type', 'text/plain');
            break;
    }
});
```

Exception and error handling

```php
$app->error(function (\Exception $e, $code) {

    switch ($code) {
        case 400:
            $message = 'Bad request.';
            break;
        case 404:
            $message = 'Page not found.';
            break;
        default:
            $message = 'Internal Server Error.';
    }

    return new Response($message, $code);
});
```

```php
$app['debug'] = true;
```

```php
$app->post('/event', function (Request $request) use ($app) {

    if (!$event = $request->get('event')) {
        $app->abort(400, 'Missing parameters.');
    }

    // ...

    return $app->redirect('/event/'. $event->id, 201);
});
```

Logging with Monolog

```php
use Silex\Provider\MonologServiceProvider;

$app->register(new MonologServiceProvider(), array(
    'monolog.logfile'    => __DIR__.'/../logs/app.log',
    'monolog.class_path' => __DIR__.'/../vendor/monolog/src',
));
```

```php
if ($app['debug']) {

    $app['monolog']->addInfo('Testing the Monolog logging.');

    $app['monolog']->addDebug('Method foo() was called.');

    $app['monolog']->addWarning('Missing parameter "bar".');

    $app['monolog']->addError('Class Foo does not exist.');
}
```

Database interactions with Doctrine

```php
use Silex\Provider\DoctrineServiceProvider;

$app->register(new DoctrineServiceProvider(), array(
    'db.options'  => array(
        'driver'  => 'pdo_mysql',
        'host'    => 'localhost',
        'user'    => 'root',
        'dbname'  => 'event_demo',
    ),
    'db.dbal.class_path' => __DIR__.'/../vendor/doctrine-dbal/lib',
    'db.common.class_path' => __DIR__.'/../vendor/doctrine-common/lib',
));
```

```php
$app->get('/events', function () use ($app) {

    $query = 'SELECT id, title, venue FROM events';

    $events = $app['db']->fetchAll($query);

    return new Response(json_encode($events));
});
```

Input validation

```
use Silex\Provider\ValidatorServiceProvider;

$app->register(new ValidatorServiceProvider());
```

```
$app['validator']->validate($object);
```

```php
namespace Confeet\Model;

use Symfony\Component\Validator\Mapping\ClassMetadata;
use Symfony\Component\Validator\Constraints\NotBlank;
use Symfony\Component\Validator\Constraints\MaxLength;

class Event extends Model
{
  private $title;
  private $venue;

  static public function loadValidatorMetadata(ClassMetadata $metadata)
  {
    $metadata->addPropertyConstraint('title', new NotBlank());
    $metadata->addPropertyConstraint('title', new MaxLength(array('limit' => 50)));
    $metadata->addPropertyConstraint('venue', new NotBlank());
  }
}
```

```php
$app->post('/event', function (Request $request) use ($app) {

    if (!$data = $request->get('event')) {
        $app->abort(400, 'Missing parameters.');
    }

    $event = new Event()
    $event->setTitle($data['title']);
    $event->setVenue($data['venue']);

    if (count($app['validator']->validate($event)) > 0) {
        $app->abort(400, 'Invalid parameters.');
    }

    $event->save();

    return $app->redirect('/event/'. $event->id, 201);
});
```

Template engine

Twig

- Fast
- Concise and rich syntax
- Automatic output escaping
- Modern features
- Extensible

```php
use Silex\Provider\TwigServiceProvider;

$app->register(new TwigServiceProvider(), array(
    'twig.path'       => __DIR__.'/../views',
    'twig.class_path' => __DIR__.'/../vendor/twig/lib',
));
```

```php
$app->get('/events.{format}', function ($format) use ($app) {

    $events = array(
        array('name' => 'OSIDays', 'venue' => 'Bangalore'),
        array('name' => 'PHP Tour', 'venue' => 'Lille'),
        array('name' => 'Confoo', 'venue' => 'Montreal'),
        // ...
    );

    return $app['twig']->render('events.'.$format.'.twig', array(
        'events' => $events,
    ));
})
->assert('format', 'xml|json');
```

```
<!-- views/events.xml.twig -->
<?xml version="1.0" encoding="utf-8" ?>
<events>
    {% for event in events %}
        <event>
            <title>{{ event.title }}</title>
            <venue>{{ event.venue }}</venue>
            <begin>{{ event.startAt }}</begin>
            <end>{{ event.endAt }}</end>
        </event>
    {% endfor %}
<events>
```

HTTP Caching & ESI

Reverse Proxy Caching

```php
use Silex\Provider\HttpCacheServiceProvider;

$app->register(new HttpCacheServiceProvider(), array(
    'http_cache.cache_dir' => __DIR__.'/../cache',
));
```

```php
$app->get('/events', function () use ($app) {

    $events = array(
        array('name' => 'OSIDays', 'venue' => 'Bangalore'),
        // ...
    );

    $content = $app['twig']->render('events.twig', array(
        'events' => $events,
    ));

    return new Response($content, 200, array(
        'Cache-Control' => 'public, s-maxage=3600',
        'Surrogate-Control' => 'content="ESI/1.0"',
    ));
});
```

Edge Side Includes

```
<!-- views/events.twig -->
<?xml version="1.0" encoding="utf-8" ?>
<events>

    <esi:include src="/metadata" />

    <!-- ... -->

<events>
```

```php
$app->get('/metadata', function () use ($app) {

    return new Response('<meta>...</meta>', 200, array(
        'Cache-Control' => 'public, s-maxage=600',
    ));
});
```

Functional testing

Client
Crawler
PHPUnit

```php
class EventApiTest extends Silex\WebTestCase
{
    public function testRecentEvents()
    {
        $client = $this->createClient();
        $crawler = $client->request('GET', '/events.xml');

        $response = $client->getResponse();

        $this->assertTrue($response->isOk());
        $this->assertEquals(5, count($crawler->filter('event')));
        $this->assertRegExp('/OSIDays/', $response->getContent());
        ...
    }
}
```

Integrating Zend Soap

```php
$path = __DIR__.'/../vendor/zend/library';

$app['autoloader']->registerNamespace('Zend', $path);
```

```php
use Zend\Soap\Server;
use Confeet\Model\EventService;

$app->get('/gateway', function (Request $request) use ($app) {

    $server = new Server();
    $server->setObject(new EventService($app));
    $server->setReturnResponse(true);

    return new Response($server->handle(), 200, array(
        'Content-Type' => 'text/xml'
    ));
});
```

Questions?

92-98, boulevard Victor Hugo

92 115 Clichy Cedex, France

trainings@sensio.com (+33 (0)140 998 211)

sensiolabs.com - symfony.com – trainings.sensiolabs.com

symfony DAY

october 21st,