The Routing Component

Before we start diving into the Routing component, let's refactor our current framework just a little to make templates even more readable::

```
<?php

// example.com/web/front.php

require_once __DIR__.'/../vendor/autoload.php';

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

$request = Request::createFromGlobals();

$map = array(
    '/hello' => 'hello',
    '/bye'   => 'bye',
);

$path = $request->getPathInfo();
if (isset($map[$path])) {
    ob_start();
    extract($request->query->all(), EXTR_SKIP);
    include sprintf(__DIR__.'/../src/pages/%s.php', $map[$path]);
    $response = new Response(ob_get_clean());
} else {
    $response = new Response('Not Found', 404);
}

$response->send();
```

As we now extract the request query parameters, simplify the`hello.php` template as follows::

```
<!-- example.com/src/pages/hello.php -->

Hello <?php echo htmlspecialchars($name, ENT_QUOTES, 'UTF-8') ?>
```

Now, we are in good shape to add new features.

One very important aspect of any website is the form of its URLs. Thanks to the URL map, we have decoupled the URL from the code that generates the associated response, but it is not yet flexible enough. For instance, we might want to support dynamic paths to allow embedding data directly into the URL instead of relying on a query string:

```
# Before
/hello?name=Fabien

# After
/hello/Fabien
```

To support this feature, add the Symfony2 Routing component as a dependency:

.. code-block:: sh

```
$ php composer.phar require symfony/routing
```

Instead of an array for the URL map, the Routing component relies on a `RouteCollection` instance::

```
use Symfony\Component\Routing\RouteCollection;

$routes = new RouteCollection();
```

Let's add a route that describe the `/hello/SOMETHING` URL and add another one for the simple `/bye` one::

```
use Symfony\Component\Routing\Route;

$routes->add('hello', new Route('/hello/{name}', array('name' => 'World')));
$routes->add('bye', new Route('/bye'));
```

Each entry in the collection is defined by a name (`hello`) and a `Route` instance, which is defined by a route pattern (`/hello/{name}`) and an array of default values for route attributes (`array('name' => 'World')`).

.. note::

```
Read the official `documentation`_ for the Routing component to learn more
about its many features like URL generation, attribute requirements, HTTP
method enforcements, loaders for YAML or XML files, dumpers to PHP or
Apache rewrite rules for enhanced performance, and much more.
```

Based on the information stored in the `RouteCollection` instance, a `UrlMatcher` instance can match URL paths::

```
use Symfony\Component\Routing\RequestContext;
use Symfony\Component\Routing\Matcher\UrlMatcher;

$context = new RequestContext();
$context->fromRequest($request);
$matcher = new UrlMatcher($routes, $context);

$attributes = $matcher->match($request->getPathInfo());
```

The `match()` method takes a request path and returns an array of attributes (notice that the matched route is automatically stored under the special `_route` attribute)::

```
print_r($matcher->match('/bye'));
array (
  '_route' => 'bye',
);

print_r($matcher->match('/hello/Fabien'));
array (
  'name' => 'Fabien',
  '_route' => 'hello',
);

print_r($matcher->match('/hello'));
array (
  'name' => 'World',
  '_route' => 'hello',
);
```

.. note::

Even if we don't strictly need the request context in our examples, it is
used in real-world applications to enforce method requirements and more.

The URL matcher throws an exception when none of the routes match::

```
$matcher->match('/not-found');

// throws a Symfony\Component\Routing\Exception\ResourceNotFoundException
```

With this knowledge in mind, let's write the new version of our framework::

```
<?php

// example.com/web/front.php

require_once __DIR__.'/../vendor/autoload.php';

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

$request = Request::createFromGlobals();
$routes = include __DIR__.'/../src/app.php';

$context = new Routing\RequestContext();
$context->fromRequest($request);
$matcher = new Routing\Matcher\UrlMatcher($routes, $context);

try {
    extract($matcher->match($request->getPathInfo()), EXTR_SKIP);
    ob_start();
    include sprintf(__DIR__.'/../src/pages/%s.php', $_route);

    $response = new Response(ob_get_clean());
} catch (Routing\Exception\ResourceNotFoundException $e) {
    $response = new Response('Not Found', 404);
} catch (Exception $e) {
    $response = new Response('An error occurred', 500);
}

$response->send();
```

There are a few new things in the code::

- Route names are used for template names;
- 500 errors are now managed correctly;
- Request attributes are extracted to keep our templates simple::

  Hello

- Route configuration has been moved to its own file:

  .. code-block:: php

```
<?php

// example.com/src/app.php

use Symfony\Component\Routing;

$routes = new Routing\RouteCollection(); $routes->add('hello', new
Routing\Route('/hello/{name}', array('name' => 'World'))); $routes->add('bye', new
Routing\Route('/bye'));

return $routes;
```

We now have a clear separation between the configuration (everything specific to our application in app.php) and the framework (the generic code that powers our application in front.php).

With less than 30 lines of code, we have a new framework, more powerful and more flexible than the previous one. Enjoy!

Using the Routing component has one big additional benefit: the ability to generate URLs based on Route definitions. When using both URL matching and URL generation in your code, changing the URL patterns should have no other impact. Want to know how to use the generator? Insanely easy::

```
use Symfony\Component\Routing;

$generator = new Routing\Generator\UrlGenerator($routes, $context);

echo $generator->generate('hello', array('name' => 'Fabien'));
// outputs /hello/Fabien
```

The code should be self-explanatory; and thanks to the context, you can even generate absolute URLs::

```
echo $generator->generate('hello', array('name' => 'Fabien'), true);
// outputs something like http://example.com/somewhere/hello/Fabien
```

.. tip::

Concerned about performance? Based on your route definitions, create a highly optimized URL matcher class that can replace the default ``UrlMatcher``::

```
    $dumper = new Routing\Matcher\Dumper\PhpMatcherDumper($routes);

    echo $dumper->dump();
```

.. _documentation: http://symfony.com/doc/current/components/routing.html