



symfony [camp]

Be RESTful

Fabien Potencier

symfony 1.2 is
REST compliant
out of the box

symfony 1.2 supports

GET

POST

HEAD

PUT

DELETE

To simulate PUT or DELETE
from a browser, symfony
uses a special parameter

`sf_method`

```
<form action="#" method="POST">  
  <input type="hidden" name="sf_method" value="PUT" />  
  
  <!-- // ... -->  
</form>
```

```
form_tag('#', array('method' => 'PUT'))
```

```
link_to('@article_delete?id=1', array('method' => 'DELETE'))
```

```
$b = new sfBrowser();
```

```
$b->  
    click('/delete', array(), array('method' => 'delete'))->  
    // ...  
;
```

```
$form->renderFormTag( '@article_update',  
    array( 'method' => 'PUT' )  
)
```

Adds a `sf_method` hidden field if method is PUT or DELETE

Automatically choose PUT or POST method for a Propel form

Adds the enctype automatically

```
public function executeUpdate($request)
{
    $this->forward404Unless($request->isMethod('PUT'));

    // or

    throw new LogicException('You must call this page with
the PUT HTTP method. ');

    // ...
}
```

```
# apps/frontend/config/routing.yml
```

```
article_update:
```

```
  url:           /article/:id/update
```

```
  param:        { module: article, action: update }
```

```
  class:        sfRequestRoute
```

```
  requirements: { sf_method: PUT }
```

```
# apps/frontend/config/routing.yml
```

```
article_update:
```

```
  url:           /article/:id
```

```
  param:        { module: article, action: update }
```

```
  class:        sfRequestRoute
```

```
  requirements: { sf_method: PUT }
```

```
article_show:
```

```
  url:           /article/:id
```

```
  param:        { module: article, action: show }
```

```
  class:        sfRequestRoute
```

```
  requirements: { sf_method: GET }
```

Routes are
first-class objects
in symfony 1.2

sfRoute

The route object
embarks all the logic

to match a URL

to generate a URL

The route context

`method`: PUT, POST, GET, ...

`format`: html, json, css, ...

`host`: The host name

`is_secure`: https?

`request_uri`: The requested URI

`prefix`: The prefix to add to each generated route

Let's create a new route class

sfDomainRoute

Some websites have URLs like

http://USERNAME.example.com/...

```
user_homepage:
```

```
  url:      /user
```

```
  class:    sfDomainRoute
```

```
  params:  { module: user, action: index }
```

```
public function executeIndex($request)
```

```
{
```

```
  $username = $request->getAttribute( 'username' );
```

```
}
```

```
class sfDomainRoute extends sfRequestRoute
{
    public function matchesUrl($url, $context = array())
    {
        if (false === $retval = parent::matchesUrl($url, $context))
        {
            return false;
        }

        $retval[1]['username'] = $this->getSubdomain($context);

        return $retval;
    }

    protected function getSubdomain($context)
    {
        $parts = explode('.', $context['host']);

        return $parts[0];
    }
}
```

```
echo url_for( '@user_homepage' );
```

```
// generates /user
```

```
echo url_for( '@user_homepage?username=foo' );
```

```
// must generate http://foo.example.com/user
```

```
public function matchesParameters($params, $context = array())
{
    unset($params['username']);

    return parent::matchesParameters($params, $context);
}

public function generate($params, $context = array(), $absolute = false)
{
    $subdomain = isset($params['username']) ? $params['username'] : false;
    unset($params['username']);

    if ($subdomain && $subdomain != $this->getSubdomain($context))
    {
        $url = parent::generate($params, $context, false);

        return $this->getHostForSubdomain($context, $subdomain).$url;
    }

    return parent::generate($params, $context, $absolute);
}
```

```
protected function getHostForSubdomain($context, $subdomain)
{
    $parts = explode('.', $context['host']);
    $parts[0] = $subdomain;
    $host = implode('.', $parts);
    $protocol = 'http'.(isset($context['is_secure']) &&
    $context['is_secure'] ? 's' : '');

    return $protocol.'://'.$host;
}
```

```
$routing = $this->getContext()->getRouting();
```

```
echo $routing->generate('user_homepage');
```

```
echo $routing->generate('user_homepage', array(), true);
```

```
echo $routing->generate('user_homepage', array('username' => 'foo'));
```

```
user_homepage:  
  url:    /user  
  class:  sfObjectDomainRoute  
  params: { module: user, action: index }
```

```
public function executeIndex($request)  
{  
  $username = $request->getAttribute('username');  
  $user = $request->getAttribute('user');  
}
```

```
class sfObjectDomainRoute extends sfDomainRoute
{
    public function matchesUrl($url, $context = array())
    {
        if (false === $retval = parent::matchesUrl($url, $context))
        {
            return false;
        }

        $subdomain = $this->getSubdomain($context);
        if (is_null($object = $this->getObject($subdomain)))
        {
            $message = sprintf('Unable to find the "user" object with the following
subdomain "%s".', $subdomain);
            throw new sfError404Exception($message);
        }

        $retval[1]['user'] = $object;

        return $retval;
    }

    protected function getObject($subdomain)
    {
        return sfGuardUserPeer::retrieveByUsername($subdomain);
    }
}
```

```
user_homepage:
  url:    /user
  class:  sfObjectDomainRoute
  params: { module: user, action: index }
  options:
    model: sfGuardUserPeer
    method: retrieveByUsername
```

```
public function executeIndex($request)
{
    $username = $request->getAttribute('username');
    $user = $request->getAttribute('user');
}
```

```
protected function getObject($subdomain)
{
    if (!isset($this->options['model']))
    {
        throw new InvalidArgumentException('You must pass a "model".');
    }

    if (!isset($this->options['method']))
    {
        throw new InvalidArgumentException('You must pass a "method".');
    }

    return call_user_func(
        array($this->options['model'], $this->options['method']),
        $subdomain
    );
}
```

A URL is the representation
of a resource

A route analyzes a URL to convert it
to parameters that will call a controller

A route describes a resource

symfony 1.2 introduces
resources description
in the route definitions

Built-in route classes

sfRoute

sfRequestRoute

sfObjectRoute

sfPropelRoute

sfPropelRoute
binds a URL
to a Propel resource

```
articles:
```

```
  url:      /articles
```

```
  params:  { module: article, action: list }
```

```
  class:   sfPropelRoute
```

```
  options: { model: Article, list: articles }
```

```
public function executeList($request)
```

```
{
```

```
  $this->articles = $request->getAttribute('articles');
```

```
}
```

```
articles:
  url:      /articles
  params:  { module: article, action: list }
  class:   sfPropelRoute
  options:
    model:      Article
    method:     getRecentArticles
    list:       articles
    allow_empty: false
```

```
article:
```

```
  url:      /article/:id
```

```
  params:  { module: article, action: show }
```

```
  class:   sfPropelRoute
```

```
  options: { model: Article, object: article }
```

```
public function executeShow($request)
```

```
{
```

```
  $this->article = $request->getAttribute('article');
```

```
}
```

article:

url: /article/:slug

params: { module: article, action: show }

class: sfPropelRoute

options:

model: Article

object: article

```
post:
```

```
  url:      /post/:id/:year/:month/:day/:slug
```

```
  param:    { module: article, action: show }
```

```
  class:    sfPropelRoute
```

```
  options: { model: Article, object: article, method:  
getObjectForRoute }
```

```
class ArticlePeer extends BaseArticlePeer
```

```
{
```

```
  static public function getObjectForRoute($parameters)
```

```
{
```

```
  $criteria = new Criteria();
```

```
  $criteria->add(self::ID, $parameters['id']);
```

```
  return self::doSelectOne($criteria);
```

```
}
```

```
}
```

article:

url: /article/:id-:slug

params: { module: article, action: show }

class: sfPropelRoute

options:

model: Article

method: retrieveOnlineByPk

object: article

allow_empty: false

segment_separators: [/, ., -]

```
url_for('@article?id=' . $article->getId() . '&slug=' . $article->getSlug())
```

```
url_for('article', array('id' => $article->getId(), 'slug' => $article->getSlug()))
```

```
url_for('article', $article)
```

```
url_for(array('sf_route' => 'article', 'sf_subject' => $article))
```

```
url_for('article', array('sf_subject' => $article, 'sf_method' => 'get'))
```

```
link_to('Link to the same page', 'article', $article)
```

A single entry in routing.yml
can represent a collection of routes

sfRouteCollection

sfObjectRouteCollection

sfPropelRouteCollection

articles:

```
class: sfPropelRouteCollection
```

```
options: { model: Article }
```

```
~/work/tmp $ ./symfony app:routes frontend
```

```
>> app          Current routes for application "frontend"
```

Name	Method	Pattern
homepage	GET	/
articles	GET	/articles.:sf_format
articles_new	GET	/articles/new.:sf_format
articles_create	POST	/articles.:sf_format
articles_show	GET	/articles/:id.:sf_format
articles_edit	GET	/articles/:id/edit.:sf_format
articles_update	PUT	/articles/:id.:sf_format
articles_delete	DELETE	/articles/:id.:sf_format

```
~/work/tmp $ ./symfony app:routes frontend articles_update
```

```
>> app Route "articles_update" for application "frontend"
```

```
Name      articles_update
Pattern   /articles/:id.:sf_format
Class     sfPropelRoute
Defaults  module: 'foo'
          action: 'update'
          sf_format: 'html'
Requirements id: '\\d+'
          sf_method: 'put'
          sf_format: '[^/\\.]+
Options    suffix: ''
          variable_prefixes: array (0 => ':',)
          segment_separators: array (0 => '/',1 => '.',)
          variable_regex: '\\w\\d]+'
          generate_shortest_url: true
          extra_parameters_as_query_string: true
          model: 'AdcArticlePeer'
          object: 'article'
          object_model: 'AdcArticle'
          variable_prefix_regex: '(?:\\:)'
          segment_separators_regex: '(?:/|\\.)'
          variable_content_regex: '[^/\\.]+'
Regex     #^
          /articles
          /(P<id>\\d+)
          (?:\\. (P<sf_format>[^/\\.]+)
          )?
          $#x
Tokens    separator array (0 => '/',)
          text      array (0 => 'articles',)
          separator array (0 => '/',)
          variable  array (0 => ':id',1 => 'id',)
          separator array (0 => '.',)
          variable  array (0 => ':sf_format',1 => 'sf_format',)
```

```
./symfony propel:generate-module-for-route frontend articles
```

```
class articlesActions extends sfActions
{
    public function executeIndex($request) {}

    public function executeShow($request) {}

    public function executeNew($request) {}

    public function executeCreate($request) {}

    public function executeEdit($request) {}

    public function executeUpdate($request) {}

    public function executeDelete($request) {}

    protected function processForm($request, $form) {}
}
```

```
public function executeIndex($request)
{
    $this->articles = $request->getAttribute('articles');
}

public function executeShow($request)
{
    $this->article = $request->getAttribute('article');
}

public function executeNew($request)
{
    $this->form = new AdcArticleForm();
}

public function executeEdit($request)
{
    $this->form = new AdcArticleForm($request->getAttribute('article'));
}

public function executeDelete($request)
{
    $request->getAttribute('article')->delete();

    $this->redirect('@articles');
}
```

```
public function executeCreate($request)
{
    $this->form = new AdcArticleForm();

    $this->processForm($request, $this->form);

    $this->setTemplate('new');
}

public function executeUpdate($request)
{
    $this->form = new AdcArticleForm($request->getAttribute('article'));

    $this->processForm($request, $this->form);

    $this->setTemplate('edit');
}

protected function processForm($request, $form)
{
    $form->bind($request->getParameter('adc_article'));
    if ($form->isValid())
    {
        $article = $form->save();

        $this->redirect('@articles_edit?id='.$article->getId());
    }
}
```

```
articles:
  class: sfPropelRouteCollection
  options:
    model: Article
    plural: articles
    singular: article
    actions: [list, show, edit, update]
    module: articles
    column: slug
  # ...
```

```
form_tag_for($form, '@articles')
```

Automatically choose POST for new objects and PUT for existing ones

Generates the REST URL thanks to the given route prefix

Adds the enctype automatically

When will it be available?

... NOW

Questions?

Sensio S.A.

92-98, boulevard Victor Hugo
92 115 Clichy Cedex
FRANCE

Tél. : +33 1 40 99 80 80

Contact

Fabien Potencier
fabien.potencier@sensio.com

<http://www.sensiolabs.com/>

<http://www.symfony-project.org/>