SUTRIX MEDIA

Symfony 2

Preface

Signature

Originator By: Quang Tran	**Date: 29/01/2014**
Prepared By:	
	Date: ……/……../…….
Approved By:	
Thuan Nguyen	Date: 21/03/2014
Reviewed By:	
Distributed To:	

Preface

Signature Revision History

*A – Added, M – Modified, D - Deleted

Version	Date	A*, M, D	Change Description	Author	Approved By
1.0	29/01/2014	A*		Quang Tran	

Symfony 2

Installation – Option A

http://getcomposer.org/

```
> php -r "eval('?>'.file_get_contents('https://getcomposer.org/installer'));"
```

```
> php composer.phar
> php composer.phar create-project symfony/framework-standard-edition /path/to/webroot/Symfony 2.3.0
```

Installation – Option B

Download, Extract, Start

Home » Get started » Download

Download

English

Symfony Standard Edition

The Symfony Standard Edition is the best distribution to use when starting a new project. It contains the most common bundles, and comes with a simple configuration system.

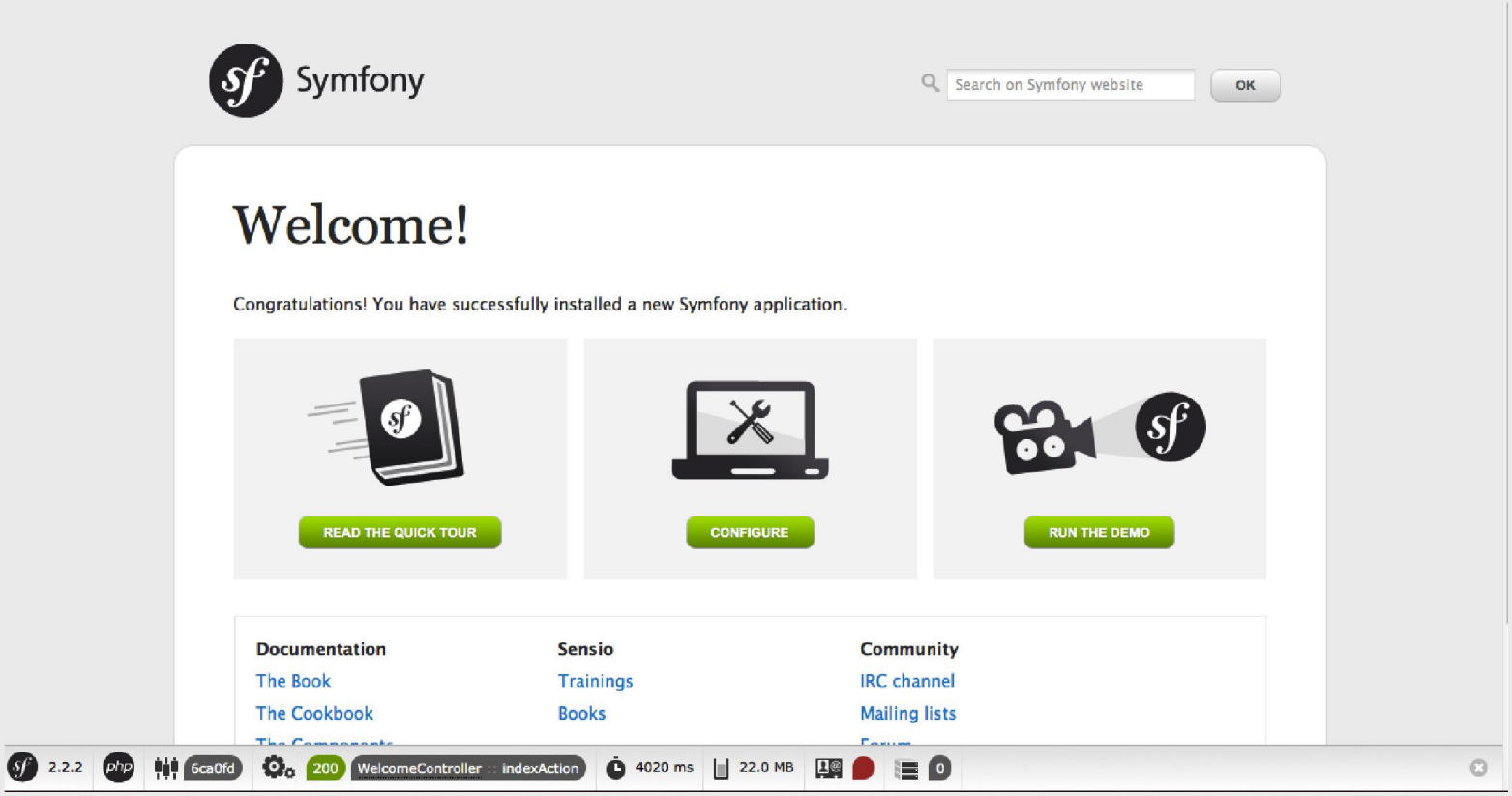
Symfony Standard 2.3.4 (.tgz)

/download?v=Symfony_Standard_Vendors_2.3.4.tgz

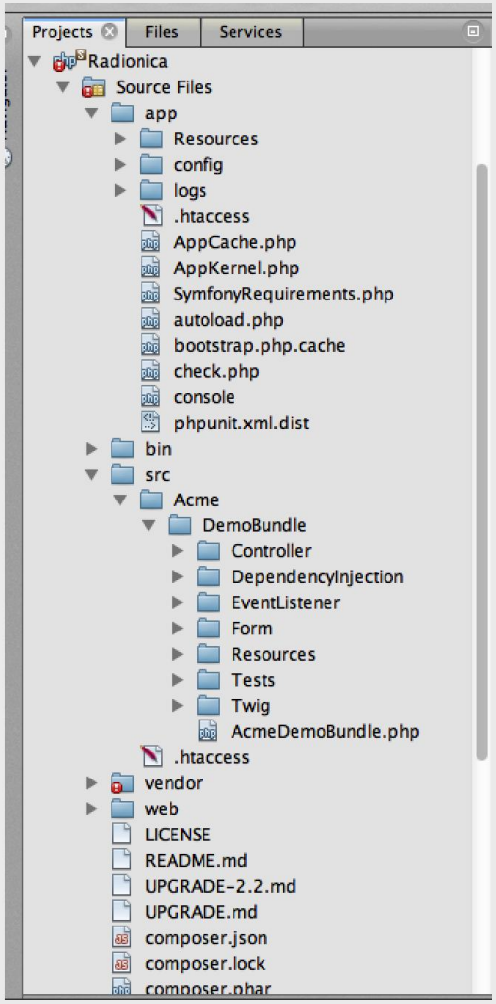
DOWNLOAD NOW

COMFORTABLE WITH THE CONSOLE?

Welcome Symfony



Folder structure



Console

php app/console

```
  doctrine:ensure-production-settings   Verify that Doctrine is properly configured for a productio
n environment.
  doctrine:generate:crud                Generates a CRUD based on a Doctrine entity
  doctrine:generate:entities            Generates entity classes and method stubs from your mapping
 information
  doctrine:generate:entity              Generates a new Doctrine entity inside a bundle
  doctrine:generate:form                Generates a form type class based on a Doctrine entity
  doctrine:mapping:convert              Convert mapping information between supported formats.
  doctrine:mapping:import               Imports mapping information from an existing database
  doctrine:mapping:info                 Shows basic information about all mapped entities
  doctrine:query:dql                    Executes arbitrary DQL directly from the command line.
  doctrine:query:sql                    Executes arbitrary SQL directly from the command line.
  doctrine:schema:create                Executes (or dumps) the SQL needed to generate the database
 schema
  doctrine:schema:drop                  Executes (or dumps) the SQL needed to drop the current data
base schema
  doctrine:schema:update                Executes (or dumps) the SQL needed to update the database s
chema to match the current mapping metadata
  doctrine:schema:validate              Validates the doctrine mapping files
generate
  generate:bundle                       Generates a bundle
  generate:controller                   Generates a controller
  generate:doctrine:crud                Generates a CRUD based on a Doctrine entity
  generate:doctrine:entities            Generates entity classes and method stubs from your mapping
 information
  generate:doctrine:entity              Generates a new Doctrine entity inside a bundle
  generate:doctrine:form                Generates a form type class based on a Doctrine entity
init
  init:acl                              Mounts ACL tables in the database
  init:jms-secure-random
orm
  orm:convert:mapping                   Convert mapping information between supported formats.
router
  router:debug                          Displays current routes for an application
  router:dump-apache                    Dumps all routes as Apache rewrite rules
  router:match                          Helps debug routes by simulating a path info match
swiftmailer
  swiftmailer:spool:send                Sends emails from the spool
translation
  translation:update                    Updates the translation file
twig
  twig:lint                             Lints a template and outputs encountered errors
Antonios-MacBook-Pro:app antonios$
```

The Symfony2 Components

HttpFoundation - Contains the Request and Response classes, as well as other classes for handling sessions and file uploads;

Routing - Powerful and fast routing system that allows you to map a specific URI (e.g. /contact) to some information about how that request should be handled (e.g. execute the contactAction() method);

Form - A full-featured and flexible framework for creating forms and handling form submissions;

Validator A system for creating rules about data and then validating whether or not user-submitted data follows those rules;

ClassLoader An autoloading library that allows PHP classes to be used without needing to manually require the files containing those classes;

Templating A toolkit for rendering templates, handling template inheritance (i.e. a template is decorated with a layout) and performing other common template tasks;

Security - A powerful library for handling all types of security inside an application;

Translation A framework for translating strings in your application.

Creating a Bundle

- **Bundle is everything in Symfony2 :) - first-class citizens**
- Directory that houses everything related to a specific feature (configuration, PHP, JS, CSS...)
- We can compare it with modul or plugin
- Flexible, independent, powerful

> php app/console generate:bundle

Creating page in Symfony2

Creating a new page in Symfony2 is a simple two-step process:

Create a route: A route defines the URL (e.g. /about) to your page and specifies a controller (which is a PHP function) that Symfony2 should execute when the URL of an incoming request matches the route path;

Create a controller: A controller is a PHP function that takes the incoming request and transforms it into the Symfony2 Response object that's returned to the user.

Environment: Prod, dev, test

Creating a Bundle

- **Bundle is everything in Symfony2 :) - first-class citizens**
- Directory that houses everything related to a specific feature (configuration, PHP, JS, CSS...)
- We can compare it with modul or plugin
- Flexible, independent, powerful

> php app/console generate:bundle

Creating page in Symfony2

Step1: Create route

```
# app/config/routing.yml
acme_hello:
    resource: "@AcmeHelloBundle/Resources/config/routing.yml"
    prefix:   /

# src/Acme/HelloBundle/Resources/config/routing.yml
hello:
    path:     /hello/{name}
    defaults: { _controller: AcmeHelloBundle:Hello:index }
```

Creating page in Symfony2

Step2: Create controller

```
// src/Acme/HelloBundle/Controller/HelloController.php
namespace Acme\HelloBundle\Controller;

use Symfony\Component\HttpFoundation\Response;

class HelloController
{
    public function indexAction($name)
    {
        return new Response('<html><body>Hello '.$name.'!</body></html>');
    }
}
```

Creating page in Symfony2

Step3: Create template

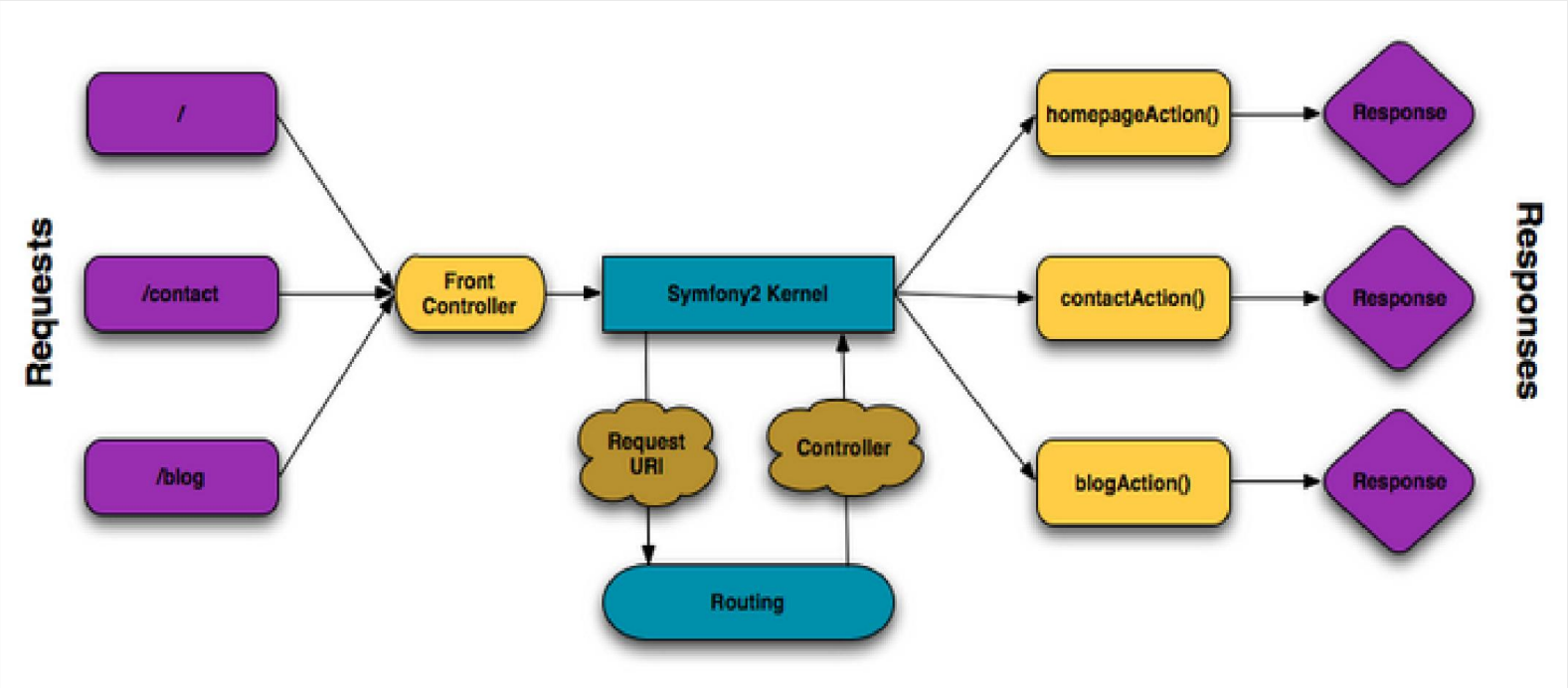
```
// src/Acme/HelloBundle/Controller/HelloController.php
namespace Acme\HelloBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;

class HelloController extends Controller
{
    public function indexAction($name)
    {
        return $this->render(
            'AcmeHelloBundle:Hello:index.html.twig',
            array('name' => $name)
        );
    }
}
```

Controller

- Request -> Response
- The response could be an HTML page, an XML document, a serialized JSON array, an image, a redirect, a 404 error or anything else you can dream up.
- The controller contains whatever arbitrary logic your application needs to render the content of a page.

```
use Symfony\Component\HttpFoundation\Response;

public function helloAction()
{
    return new Response('Hello world!');
}
```

Requests, Controller, Response Lifecycle



Simple Controller

```
// src/Acme/HelloBundle/Controller/HelloController.php
namespace Acme\HelloBundle\Controller;

use Symfony\Component\HttpFoundation\Response;

class HelloController
{
    public function indexAction($name)
    {
        return new Response('<html><body>Hello '.$name.'!</body></html>');
    }
}
```

Mapping url to a Controller

```
# app/config/routing.yml
hello:
    path:      /hello/{first_name}/{last_name}
    defaults:  { _controller: AcmeHelloBundle:Hello:index, color: green }

// order of arguments does not metter
public function indexAction($first_name, $color, $last_name)
{
    // ... do whatever logic and return Response
}
```

The Request as Controller Argument

```
use Symfony\Component\HttpFoundation\Request;

public function updateAction(Request $request)
{
    $form = $this->createForm(...);

    $form->handleRequest($request);
    // ...
}
```

The Base Controller Class

```
// src/Acme/HelloBundle/Controller/HelloController.php
namespace Acme\HelloBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Response;

class HelloController extends Controller
{
    // ... do whatever logic and return Response
}
```

By extending this Controller class, you can take advantage of several helper methods.

Common Controller Tasks

```
// redirecting
public function indexAction()
{
    return $this->redirect($this->generateUrl('homepage'));
}

public function indexAction()
{
    return $this->redirect($this->generateUrl('homepage'), 301);
}
```

Shortcut for:

```
use Symfony\Component\HttpFoundation\RedirectResponse;
return new RedirectResponse($this->generateUrl('homepage'));
```

Common Controller Tasks

```
// forwarding
public function indexAction($name)
{
    $response = $this->forward('AcmeHelloBundle:Hello:fancy', array(
        'name'  => $name,
        'color' => 'green',
    ));

    // ... further modify the response or return it directly

    return $response;
}

public function fancyAction($name, $color)
{
    // ... create and return a Response object
}
```

Common Controller Tasks

Shortcut for:

```
$httpKernel = $this->container->get('http_kernel');
$response = $httpKernel->forward(
    'AcmeHelloBundle:Hello:fancy',
    array(
        'name'  => $name,
        'color' => 'green',
    )
);
```

Common Controller Tasks

```
// rendering templates
use Symfony\Component\HttpFoundation\Response;

$content = $this->renderView(
    'AcmeHelloBundle:Hello:index.html.twig',
    array('name' => $name)
);

return new Response($content);

return $this->render(
    'AcmeHelloBundle:Hello:index.html.twig',
    array('name' => $name)
);
```

Common Controller Tasks

Shortcut for:

```
$templating = $this->get('templating');
$content = $templating->render(
    'AcmeHelloBundle:Hello:index.html.twig',
    array('name' => $name)
);

$templating->render(
    'AcmeHelloBundle:Hello/Greetings:index.html.twig',
    array('name' => $name)
);
```

404 error

```
public function indexAction()
{
    // retrieve the object from database
    $product = ...;
    if (!$product) {
        throw $this->createNotFoundException('The product does not exist');
    }

    return $this->render(...);
}
```

Error 500

```
throw new \Exception('Something went wrong!');
```

Accessing other Services

```
$request = $this->getRequest();

$templating = $this->get('templating');

$router = $this->get('router');

$mailer = $this->get('mailer');
```

Managing the Session

```
$session = $this->getRequest()->getSession();

// store an attribute for reuse during a later user request
$session->set('foo', 'bar');

// in another controller for another request
$foo = $session->get('foo');

// use a default value if the key doesn't exist
$filters = $session->get('filters', array());
```

Flash Messages

```
public function updateAction()
{
    $form = $this->createForm(...);

    $form->bind($this->getRequest());
    if ($form->isValid()) {
        // do some sort of processing

        $this->get('session')->getFlashBag()->add('notice', 'Your changes
were saved!');

        return $this->redirect($this->generateUrl(...));
    }
    return $this->render(...);
}
```

Flash Messages

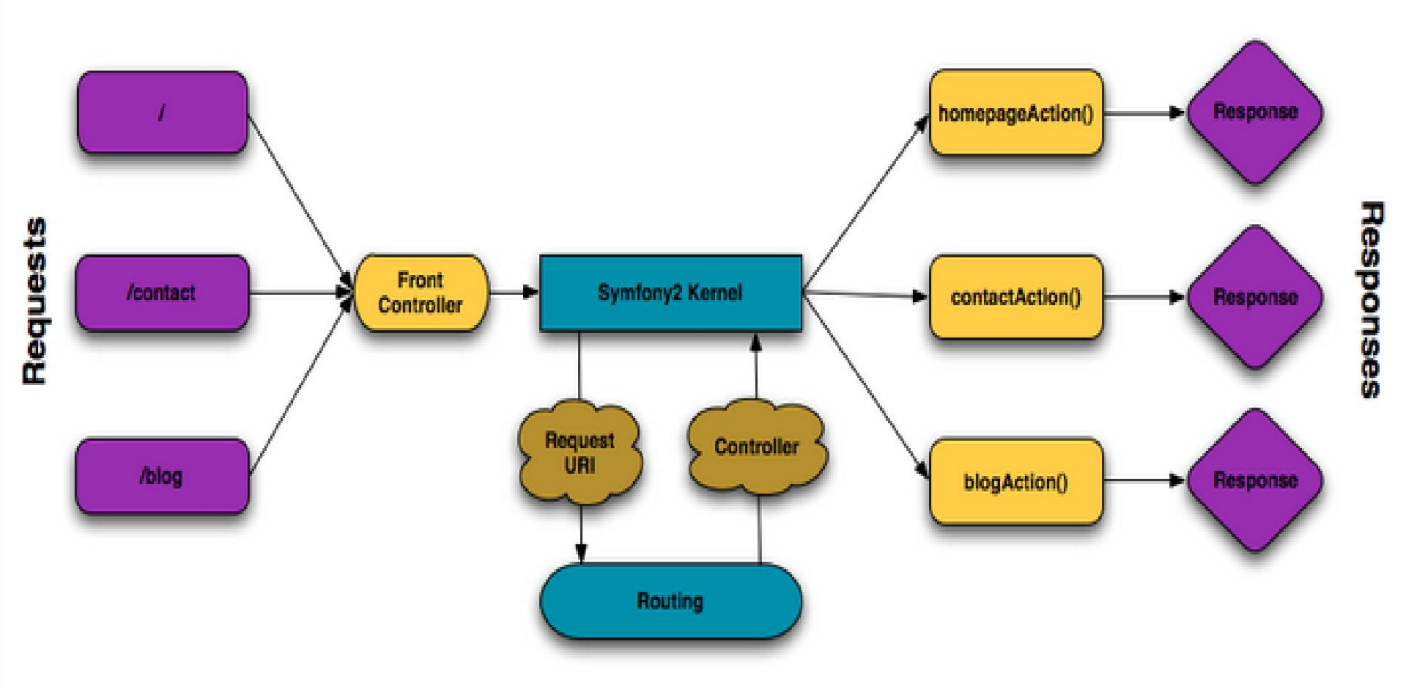
```
{% for flashMessage in app.session.flashbag.get('notice') %}
    <div class="flash-notice">
        {{ flashMessage }}
    </div>
{% endfor %}
```

Render static page

No need for controller!!!

```
acme_privacy:
    path: /privacy
    defaults:
        _controller: FrameworkBundle:Template:template
        template: 'AcmeBundle:Static:privacy.html.twig'
```

Routing


Requests
/
/contact
/blog
Front Controller
Symfony2 Kernel
Request URI
Controller
Routing
homepageAction()
contactAction()
blogAction()
Response
Response
Response
Responses

Basic Route Configuration

```
_welcome:
    path:       /
    defaults:  { _controller: AcmeDemoBundle:Main:homepage }
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<routes xmlns="http://symfony.com/schema/routing"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://symfony.com/schema/routing
        http://symfony.com/schema/routing/routing-1.0.xsd">
    <route id="_welcome" path="/">
        <default
key="_controller">AcmeDemoBundle:Main:homepage</default>
    </route>
</routes>
```

Routing with placeholder

```
blog_show:
    path:       /blog/{slug}
    defaults:  { _controller: AcmeBlogBundle:Blog:show }
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<routes xmlns="http://symfony.com/schema/routing"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://symfony.com/schema/routing
        http://symfony.com/schema/routing/routing-1.0.xsd">
    <route id="blog_show" path="/blog/{slug}">
        <default
key="_controller">AcmeBlogBundle:Blog:show</default>
    </route>
</routes>
```

Adding Requirements

```
blog:
    path:      /blog/{page}
    defaults:  { _controller: AcmeBlogBundle:Blog:index, page: 1 }

blog_show:
    path:      /blog/{slug}
    defaults:  { _controller: AcmeBlogBundle:Blog:show }
```

URL	route	parameters
/blog/2	blog	{page} = 2
/blog/my-blog-post	blog	{page} = my-blog-post

Adding Requirements

```
blog:
    path:       /blog/{page}
    defaults:  { _controller: AcmeBlogBundle:Blog:index, page: 1 }
    requirements:
        page:  \d+
```

URL	route	parameters
/blog/2	blog	{page} = 2
/blog/my-blog-post	blog_show	{slug} = my-blog-post

Adding HTTP Method Requirements

```
contact:
    path:     /contact
    defaults: { _controller: AcmeDemoBundle:Main:contact }
    methods:  [GET]

contact_process:
    path:     /contact
    defaults: { _controller: AcmeDemoBundle:Main:contactProcess }
    methods:  [POST]
```

Generating URLs

```
class MainController extends Controller
{
    public function showAction($slug)
    {
        // ...

        $url = $this->generateUrl(  // helper method
            'blog_show',
            array('slug' => 'my-blog-post')
        );
    }
}
```

Generating URLs

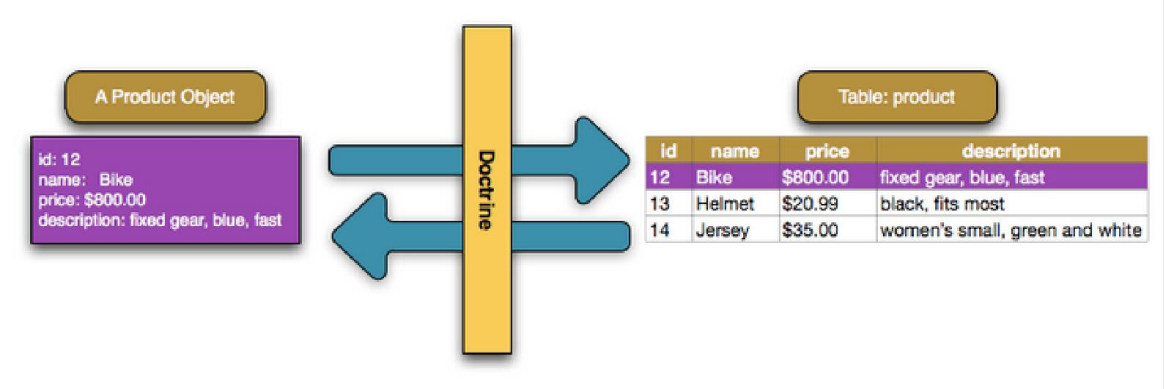
```
$params = $this->get('router')->match('/blog/my-blog-post');
// array(
//     'slug'        => 'my-blog-post',
//     '_controller' => 'AcmeBlogBundle:Blog:show',
// )


$uri = $this->get('router')->generate('blog_show', array('slug' => 'my-blog-post'));
// /blog/my-blog-post


$this->get('router')->generate('blog_show', array('slug' => 'my-blog-post'), true);
// http://www.example.com/blog/my-blog-post
```

Model

ORM Doctrine 2

parameters.yml → Config.yml

> php app/console doctrine:database:create



MODEL - metadata

```
// src/Acme/StoreBundle/Entity/Product.php
namespace Acme\StoreBundle\Entity;
use Doctrine\ORM\Mapping as ORM;
/**
 * @ORM\Entity
 * @ORM\Table(name="product")
 */
class Product
{
    /**
     * @ORM\Id
     * @ORM\Column(type="integer")
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;
```

MODEL - metadata

```
/**
 * @ORM\Column(type="string", length=100)
 */
protected $name;

/**
 * @ORM\Column(type="decimal", scale=2)
 */
protected $price;

/**
 * @ORM\Column(type="text")
 */
protected $description;
}
```

MODEL

A bundle can accept only one metadata definition format. For example, it's not possible to mix YAML metadata definitions with annotated PHP entity class definitions.

Be careful that your class name and properties aren't mapped to a protected SQL keyword (such as group or user).

Create getters and setters

> php app/console doctrine:generate:entities Acme/StoreBundle/Entity/Product

> php app/console doctrine:schema:update –force

Persist object to database

```
// src/Acme/StoreBundle/Controller/DefaultController.php
// ...
use Acme\StoreBundle\Entity\Product;
use Symfony\Component\HttpFoundation\Response;

public function createAction()
{
    $product = new Product();
    $product->setName('A Foo Bar');
    $product->setPrice('19.99');
    $product->setDescription('Lorem ipsum dolor');
    $em = $this->getDoctrine()->getManager();
    $em->persist($product);
    $em->flush();
    return new Response('Created product id '.$product->getId());
}
```

Fetching Objects from the Database

```
public function showAction($id)
{
    $product = $this->getDoctrine()
        ->getRepository('AcmeStoreBundle:Product')
        ->find($id);

    if (!$product) {
        throw $this->createNotFoundException(
            'No product found for id '.$id
        );
    }

    // ... do something, like pass the $product object into a template
}
```

Repository Basic

```
// query by the primary key (usually "id")
$product = $repository->find($id);

// dynamic method names to find based on a column value
$product = $repository->findOneById($id);
$product = $repository->findOneByName('foo');

// find *all* products
$products = $repository->findAll();

// find a group of products based on an arbitrary column value
$products = $repository->findByPrice(19.99);
```

Repository Basic

```
// query for one product matching be name and price
$product = $repository->findOneBy(array('name' => 'foo', 'price' =>
19.99));

// query for all products matching the name, ordered by price
$products = $repository->findBy(
    array('name' => 'foo'),
    array('price' => 'ASC')
);
```

Updating an Object

```
public function updateAction($id)
{
    $em = $this->getDoctrine()->getManager();
    $product = $em->getRepository('AcmeStoreBundle:Product')->find($id);

    if (!$product) {
        throw $this->createNotFoundException(
            'No product found for id '.$id
        );
    }

    $product->setName('New product name!');
    $em->flush();

    return $this->redirect($this->generateUrl('homepage'));
}
```

Deleting on object

```
$em->remove($product);
$em->flush();
```

Querying for Object with DQL

```
$em = $this->getDoctrine()->getManager();
$query = $em->createQuery(
    'SELECT p
    FROM AcmeStoreBundle:Product p
    WHERE p.price > :price
    ORDER BY p.price ASC'
)->setParameter('price', '19.99');

$products = $query->getResult();
```

Using Doctrine's Query Builder

```
$repository = $this->getDoctrine()
    ->getRepository('AcmeStoreBundle:Product');

$query = $repository->createQueryBuilder('p')
    ->where('p.price > :price')
    ->setParameter('price', '19.99')
    ->orderBy('p.price', 'ASC')
    ->getQuery();

$products = $query->getResult();
```

Custom Repository Classes

```
// src/Acme/StoreBundle/Entity/ProductRepository.php
namespace Acme\StoreBundle\Entity;

use Doctrine\ORM\EntityRepository;

class ProductRepository extends EntityRepository
{
    public function findAllOrderedByName()
    {
        return $this->getEntityManager()
            ->createQuery(
                'SELECT p FROM AcmeStoreBundle:Product p ORDER BY
p.name ASC'
            )
            ->getResult();
    }
}
```

Custom Repository Classes

```
$em = $this->getDoctrine()->getManager();
$products = $em->getRepository('AcmeStoreBundle:Product')
            ->findAllOrderedByName();
```

MODEL – custom repository

```
public function updateGallery($photoId, $galleryId)
    {
        $qb = $this->createQueryBuilder('p');

        $query = $qb
                ->update()
                ->set('p.gallery', $galleryId)
                ->where('p.id = :id')
                ->setParameter('id', $photoId);

        return $query->getQuery()->execute();
    }
```

Relationship Mapping Metadata

```
// src/Acme/StoreBundle/Entity/Category.php
// ...
use Doctrine\Common\Collections\ArrayCollection;

class Category
{
    // …
    /**
     * @ORM\OneToMany(targetEntity="Product", mappedBy="category")
     */
    protected $products;

    public function __construct()
    {
        $this->products = new ArrayCollection();
    }
}
```

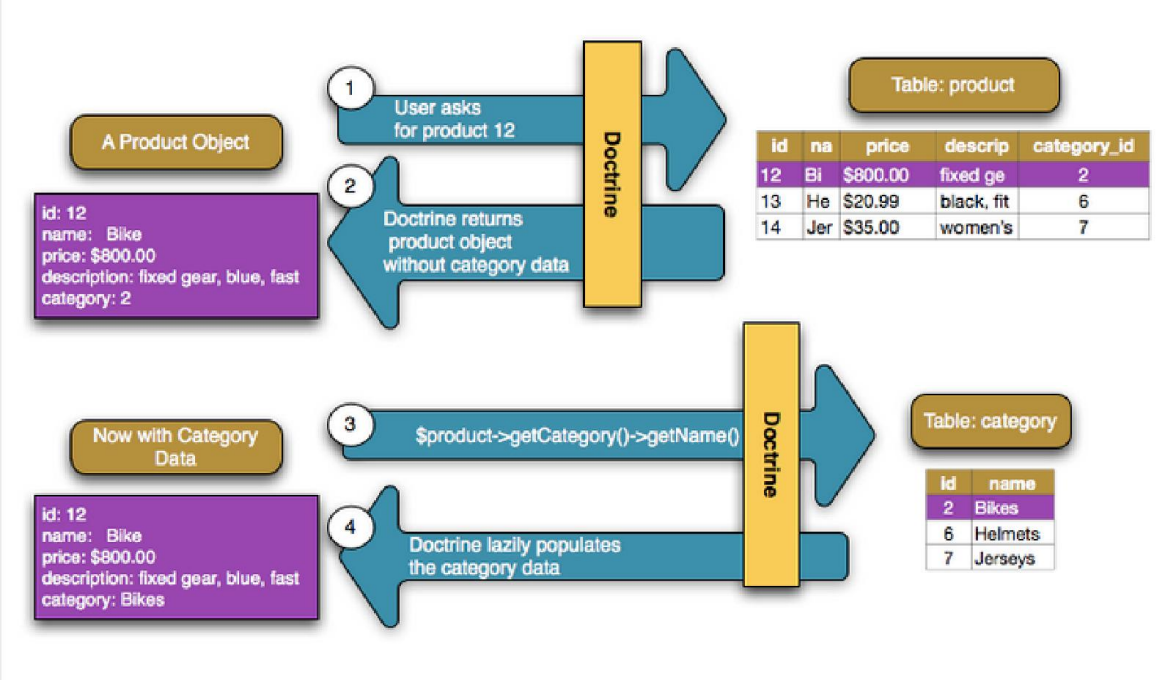
Relationship Mapping Metadata

```
// src/Acme/StoreBundle/Entity/Product.php

// ...
class Product
{
    // ...

    /**
     * @ORM\ManyToOne(targetEntity="Category", inversedBy="products")
     * @ORM\JoinColumn(name="category_id",
referencedColumnName="id")
     */
    protected $category;
}
```

Fetching related Objects

```
public function showAction($id)
{
    $product = $this->getDoctrine()
        ->getRepository('AcmeStoreBundle:Product')
        ->find($id);

    $categoryName = $product->getCategory()->getName();

    // ...
}
```

Fetching related Objects



SUTRIX MEDIA

Thank you

Sutrix Media (Vietnam) JSC.
72/4 Truong Quoc Dung Street, Ward 10,
Phu Nhuan District, Ho Chi Minh City, Vietnam

Sutrix Media HK Limited.
Unit 706, 7/F., South Seas Centre, Tower 2,
75 Mody Road, TsimShaTsui, Hong Kong

LOGO KHACH HANG