Symfony

O que é um framework?

“Um framework, ou arcabouço, em desenvolvimento de software, é uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica.” (Wikipedia)

Porque devo usar um framework?

- Não reinventar a roda
- Focar nas regras de negócios da aplicação
- Manutenibilidade
- Escalabilidade

Como escolher um framework?

- Popularidade
- Comunidade
- Suporte
- Boas práticas de desenvolvimento
- Segurança
- Documentação
- Licensa

Porque escolher o Symfony2?

- Reconhecido internacionalmente
- Comunidade ativa
- Suporte por 3 anos
- Padrões de projeto, ...
- Serviços de segurança (Security)
- Boa documentação (ainda em avanço)
- Licensa MIT

Mas o que são Padrões de Projeto?

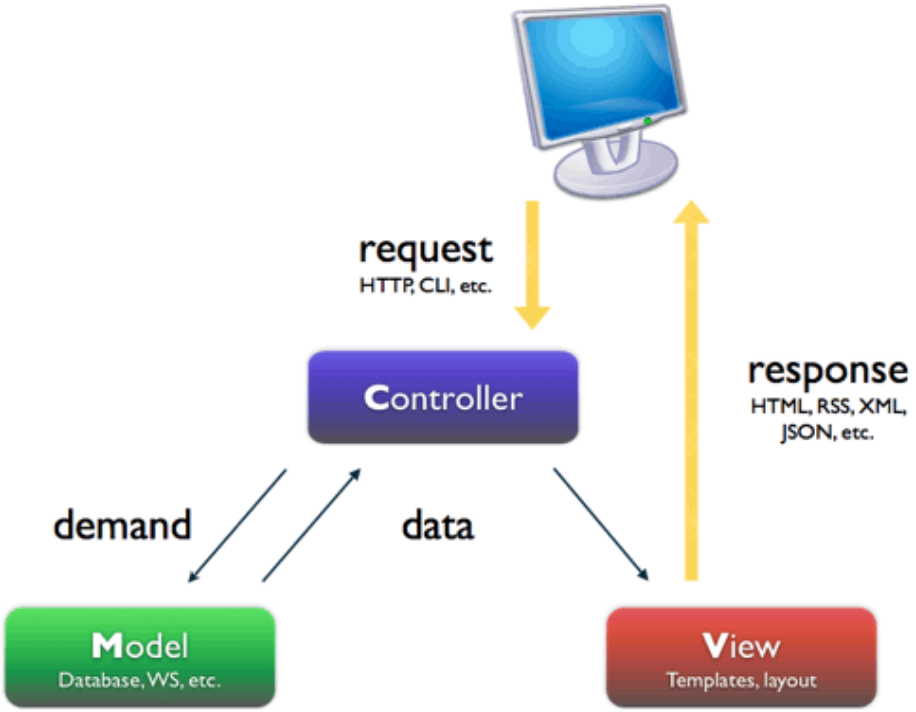
- Soluções para problemas recorrentes
- Solução reutilizável (reuso de idéias)
- Propostos por projetistas OO experientes

Padrões de Projeto presentes no Symfony2

- **MVC**
- **Dependency Injection**
- **Decorator**
- Singleton
- Chains Of Responsability
- …

MVC (Modelo, Visão, Controlador)

- Padrão de arquitetura de software
- Separar regras de negócios da apresentação
- Permite o desenvolvimento isolado de ambos
- Baixo acoplamento entre as partes



(M) Modelo

- Define o dominio da aplicação
- Acesso a dados
- Exemplo
    - Aplicação: Blog
    - Dominio: Post, Tag, Categoria, Usuario

(V) Visão

- Apresentação dos dados
- Formatos
    - HTML
    - RSS
    - XML
    - JSON
- Outros dispositivos, móveis por exemplo

(C) Controlador

- Independente do modelo
- Independente da apresentação
- Responsável por receber uma requisição e responder
- Papel de “cola” entre modelo e apresentação

Injeção de Dependencias

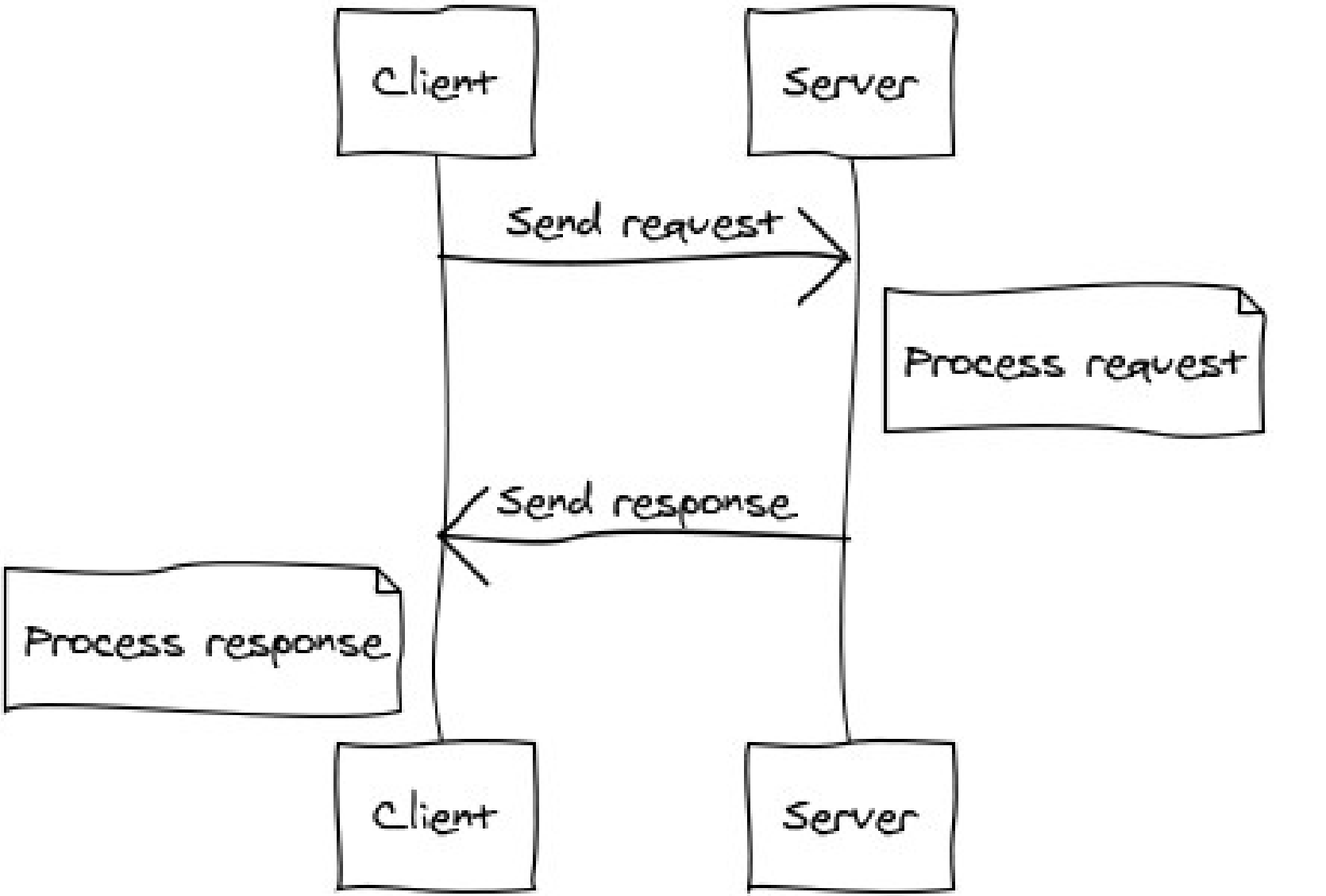
- Reduzir o acoplamento entre componentes de software
- Objetos dependem de outras partes da aplicação
- Retira a responsabilidade dos objetos de saber como outras partes da aplicação funcionam
- As dependencias serão injetadas seguindo um “contrato”
- Exemplo: Criação e obtenção de serviços

Decorator

- Alternativa mais flexível ao uso de herança
- Adicionar funcionalidades a uma classe dinamicamente
- Exemplo: Engine de templates do Symfony2 (Twig)

O protocolo HTTP

- Permite que duas máquinas se comuniquem por meio de mensagens
- Dois atores: cliente e servidor
- Cliente envia uma requisição(mensagem) para o servidor
- Servidor “entende” a requisição e retorna uma resposta(mensagem) para o cliente


Client
Server
Send request
Process request
Send response
Process response
Client
Server
www.websequencediagrams.com

Estrutura de arquivos do Symfony2

```
www/
    Symfony/
        app/
            cache/
            config/
            logs/
            Resources/
        bin/
        src/
            Acme/
                DemoBundle/
                    Controller/
                    Resources/
                    ...
        vendor/
            symfony/
            doctrine/
            ...
        web/
            app.php
            ...
```

app/	Configurações relacionadas a aplicação como um todo, caches, logs, registro de bundles ativos, ...
src/	Contém todo código PHP da aplicação além de templates, configurações, … organizados por bundles
vendor/	Bundles que compõe o core do framework e os desenvolvidos por terceiros.
web/	Todos os arquivos públicos, ou seja, que podem ser acessados diretamente pelo cliente. Além disso, tem os arquivos app.php e app_dev.php que são “*front controllers*” do Symfony2.

Características do Symfony2

- Convenção por configuração
- Ambientes de desenvolvimento
  - Produção (app.php)
  - Desenvolvimento (app_dev.php)
  - Testes

O Sistema de Bundles

- Um bundle é um conjunto estruturado de arquivos que implementam uma funcionalidade única, um blog por exemplo
- Um bundle contém tudo relacionado a funcionalidade que ele implementa como códigos PHP, configurações, templates, testes, ...
- Podem ser compartilhados com outras aplicações, por exemplo um bundle de gerenciamento de usuários

Criando Bundles

- Criados na pasta src/
- Boas práticas para nome
  - Usar apenas caracteres alfanuméricos e underscore
  - CamelCase
  - Descritivo com no máximo 2 palavras
  - Prefixado com o nome do criador
  - Sufixado com Bundle
  - Exemplos: AcmeBlogBundle, AcmeForumBundle
- Comando
  - php app/console generate:bundle --namespace=Acme/BlogBundle

Criando um Controlador

- Nome: UsuarioController
- Herda da classe Controller ou ContainerAware
- Coleção de ações
- **public function** indexAction() { ... }
- **public function** exibirAction(Usuario $usuario) { … }

Exemplo

```
(…)

class UsuarioController extends Controller
{
    public function indexAction()
    {
        return new Response('Hello World!!!');
    }

    public function helloAction($nome)
    {
        return new Response(sprintf('Hello %s!!!', $nome));
    }
}
```

O Sistema de Roteamento

- Associa padrões de URL's a controladores
- Permite configuração por Anotação, YML, XML e PHP
- Vamos utilizar configuração por Anotação

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

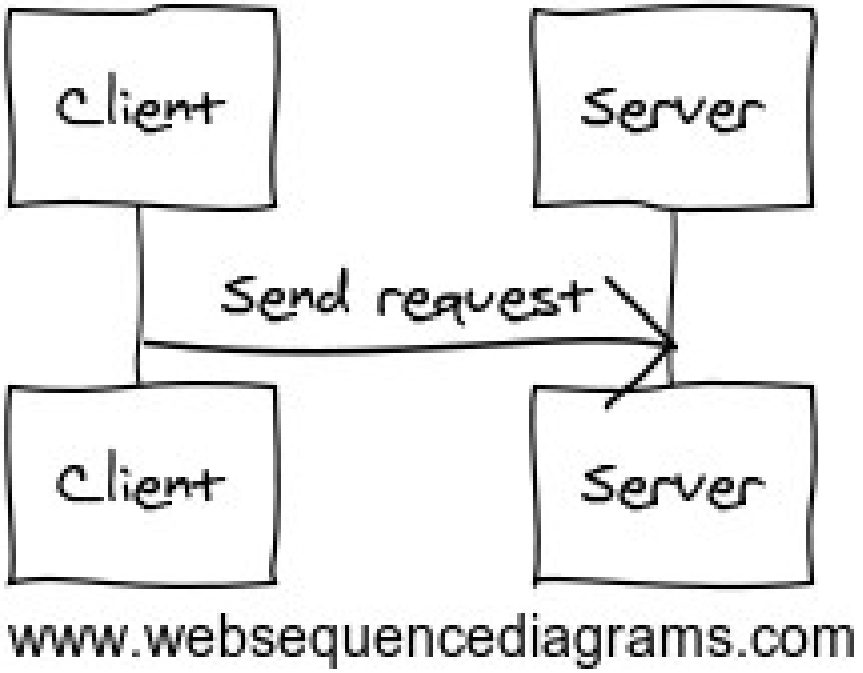
Exemplo

(…)

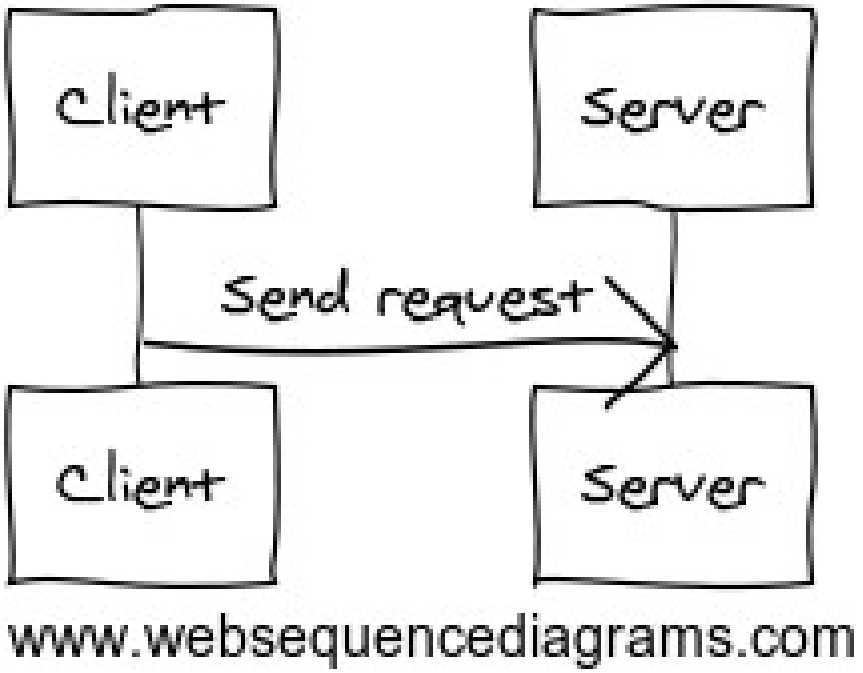
```
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
```

(…)

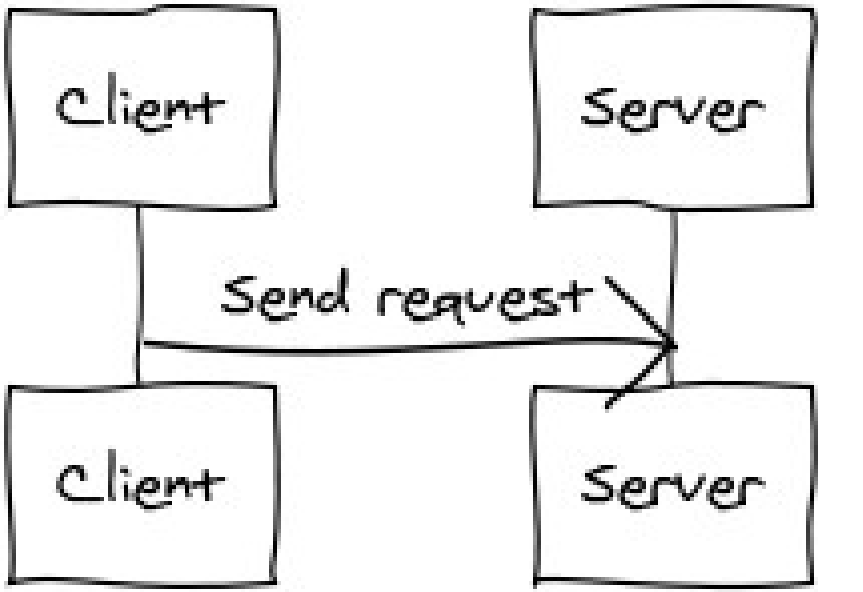
```
/**
 * @Route(“/”, name=”usuario_index”)
 */
public function indexAction() { ... }

/**
 * @Route(“/hello/{nome}”, name=”usuario_hello”,requirements={“nome”=”\w+”})
 */
public function helloAction($nome) { ... }

/**
 * @Route(“/exibir/{id}”, name=”usuario_exibir”, requirements={“id”=”\d+”})
 */
public function exibirAction(Usuario $usuario) { … }
```



GET / HTTP/1.1
Host: blog.com
Accept: text/html
User-Agent: Mozilla/5.0



GET /hello/Marcus Henrique HTTP/1.1
Host: blog.com
Accept: text/html
User-Agent: Mozilla/5.0



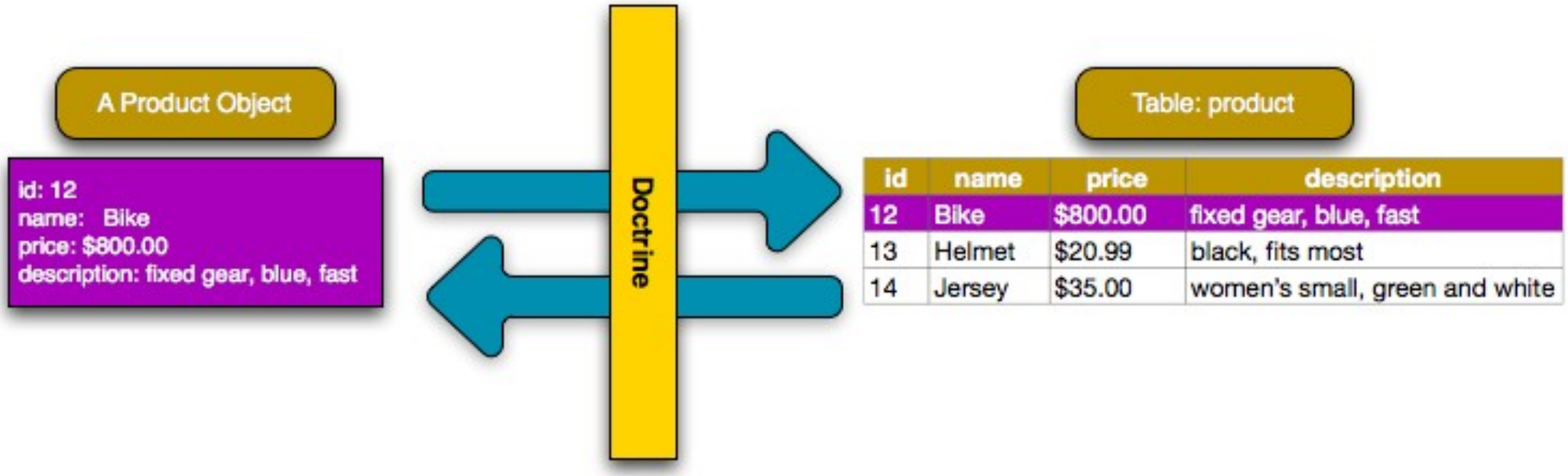
GET /exibir/1 HTTP/1.1
Host: blog.com
Accept: text/html
User-Agent: Mozilla/5.0

Acesso a Dados (Modelo)

- Symfony2 não provê acesso a dados em banco
- Utiliza o framework Doctrine2

Doctrine2

- Mapeamento Objeto-Relacional
- Abstrai informações do SGBD utilizado
- Abstrai os tipos de dados do SGBD
- Utiliza DQL para abstrair a SQL utilizada pelo SGBD


A Product Object
id: 12
name: Bike
price: $800.00
description: fixed gear, blue, fast
Doctrine
Table: product
id
name
price
description
12
Bike
$800.00
fixed gear, blue, fast
13
Helmet
$20.99
black, fits most
14
Jersey
$35.00
women's small, green and white

Entidades

- Localizadas na pasta Entity/ do bundle
- Classes simples que manipulam dados
- Persistidas e obtidas do banco de dados
- Comando para criar entidades
  - php app/console doctrine:generate:entity --entity=AcmeBlogBundle:Usuario
- Comando para gerar GETTERS e SETTERS
  - php app/console doctrine:generate:entities AcmeBlogBundle
  - php app/console doctrine:generate:entities AcmeBlogBundle:Usuario

Exemplo

```
use Doctrine\ORM\Mapping as ORM;

/**
 * Acme\DemoBundle\Entity\Usuario *
 * @ORM\Table()
 * @ORM\Entity
 */
class Usuario
{
    /**
     * @var integer $id
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

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

Exemplo (Relacionamento)

> Entity/Usuario.php (...)

```
/**
 * @var ArrayCollection $posts
 * @OneToMany(targetEntity="Post", mappedBy="autor")
 */
private $posts;

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

> Entity/Post.php (...)

```
/**
 * @var Usuario $autor
 * @ORM\ManyToOne(targetEntity="Usuario", inversedBy="posts")
 * @ORM\JoinColumn(name="autor", referencedColumnName="id", nullable=false)
 */
private $autor;
```

Exemplo (Persistência)

> Controller/UsuarioController (...)

```
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Method;

/**
 * @Route("/salvar", name="usuario_salvar")
 * @Method("POST")
 */
public function salvarAction()
{
	$usuario = new Usuario();
	(…)
	$em = $this->getDoctrine()->getEntityManager();
	$em->persist($usuario);
	$em->flush();
	(...)
}
```

Forms

- Lidar com formulários HTML é muito comum no desenvolvimento de aplicações web
- Tratar dados vindos de um POST de um formulário
- Gerar visualização do formulário
- Reutilização de código
- Comando
  - php app/console generate:doctrine:form AcmeBlogBundle

Exemplo 1

> Contrroller/UsuarioController (…)

```
public function salvarAction()
{
	$usuario = new Usuario();
	$requisicao = $this->getRequest();
	(…)

	$form = $this->createFormBuilder($usuario)
		->add('nome')
		->add('email', 'e-mail')
		->getForm();

	If ($requisicao->getMethod() === 'POST') {
		$form->bindRequest($requisicao);
		If ($form->isValid()) {
			// Persistir entidade.
			// Direcionar para outra página.
		}
	}

	// Exibir o formulário com erros e dados preenchidos se tiverem.
	(...)
}
```

Exemplo 2

> Contrroller/UsuarioController (…)

```
public function salvarAction()
{
	$usuario = new Usuario();
	$requisicao = $this->getRequest();
	(…)

	$form = $this->createForm(new UsuarioType(), $usuario);

	If ($requisicao->getMethod() === 'POST') {
		$form->bindRequest($requisicao);
		If ($form->isValid()) {
			// Persistir entidade.
			// Direcionar para outra página.
		}
	}

	// Exibir o formulário com erros e dados preenchidos se tiverem.
	(...)
}
```

Exemplo 2 (UsuarioType)

> Form/UsuarioType (…)

```
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilder;

class UsuarioType extends AbstractType
{
	public function buildForm(FormBuilder $builder, array $options)
	{
		$builder->add('nome');
		$builder->add('email', 'email');
		$builder->add('login');
		$builder->add('senha', 'repeated', array(
			'type' => 'password',
			'first_name' => 'Digite seu senha,
			'second_name' => 'Confirme sua senha',
			'invalid_message' => 'Por favor, confirme sua senha.'
		));
	}

	public function getName()
	{
		return 'usuario';
	}
}
```

Twig

- Linguagem de templates
- Diversos “filtros” prontos para uso
- Extensível
- Utiliza o padrão Decorator
- Templates em Resources/views/{Controller}/{Action}.twig.html
- {Controller} é o nome do controlador sem “Controller”
- {Action} é o nome da ação sem “Action”

Exemplo

> *Resources/views/Usuario/hello.html.twig*

```
{% extends "AcmeDemoBundle::layout.html.twig" %}

{% block title "Hello " ~ name %}

{% block content %}
    <h1>Hello {{ name }}!</h1>
{% endblock %}
```

Twig (Tipos sintaticos especiais)

- {{ }}
  - “Diz algo”: imprime uma variável ou expressão
- {% %}
  - “Faz algo”: controla a lógica do template
  - Executa alguma instrução

Exemplo (Listagem de Usuários)

> *Resources/views/Usuario/index.html.twig*

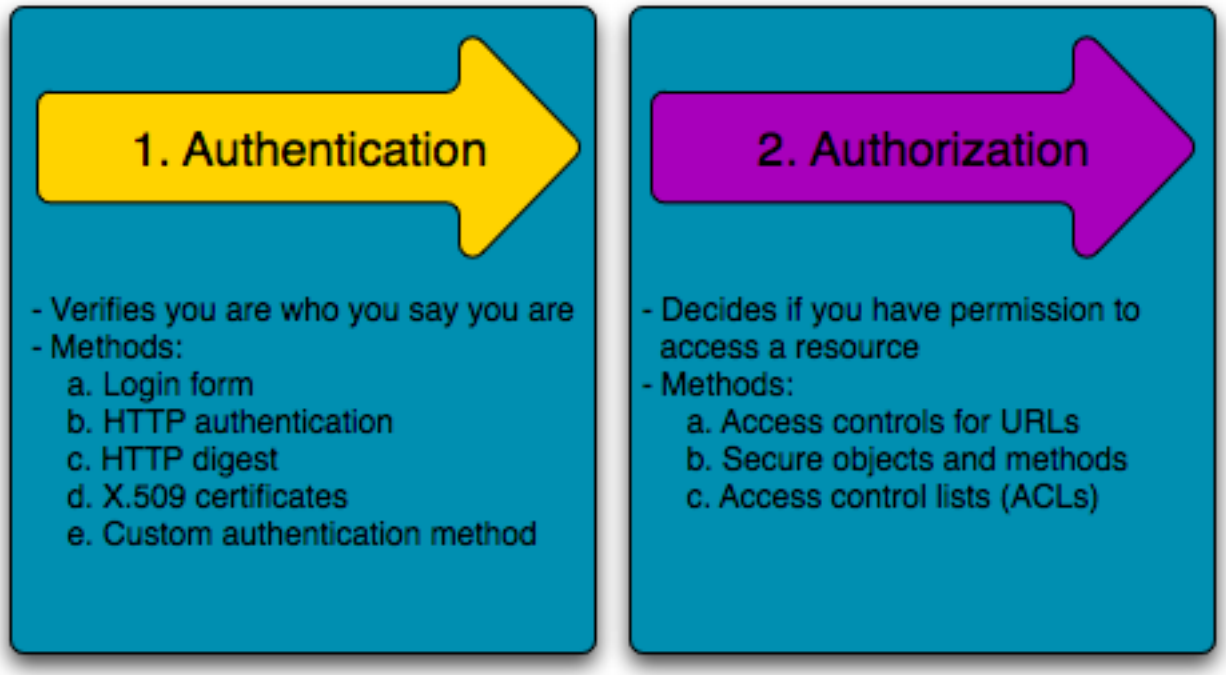
```
{% extends 'AcmeBlogBundle::layout.html.twig' %}

{% block title 'Listagem de Usuários' %}

{% block content %}
    {% for usuario in usuarios %}
        {{ usuario.nome }} - {{ usuario.email }}
        {% if usuario.ativo %}
            - Inativo
        {% endif %}
    {% endfor %}
{% endblock %}
```

CRUD

- Create, Retrieve, Update e Delete
- Operações básicas em bancos de dados relacionais
- Gerador de CRUD do Symfony2
- Bom para prototipação
- Comando
  - php app/console generate:doctrine:crud --entity=AcmeBlogBundle

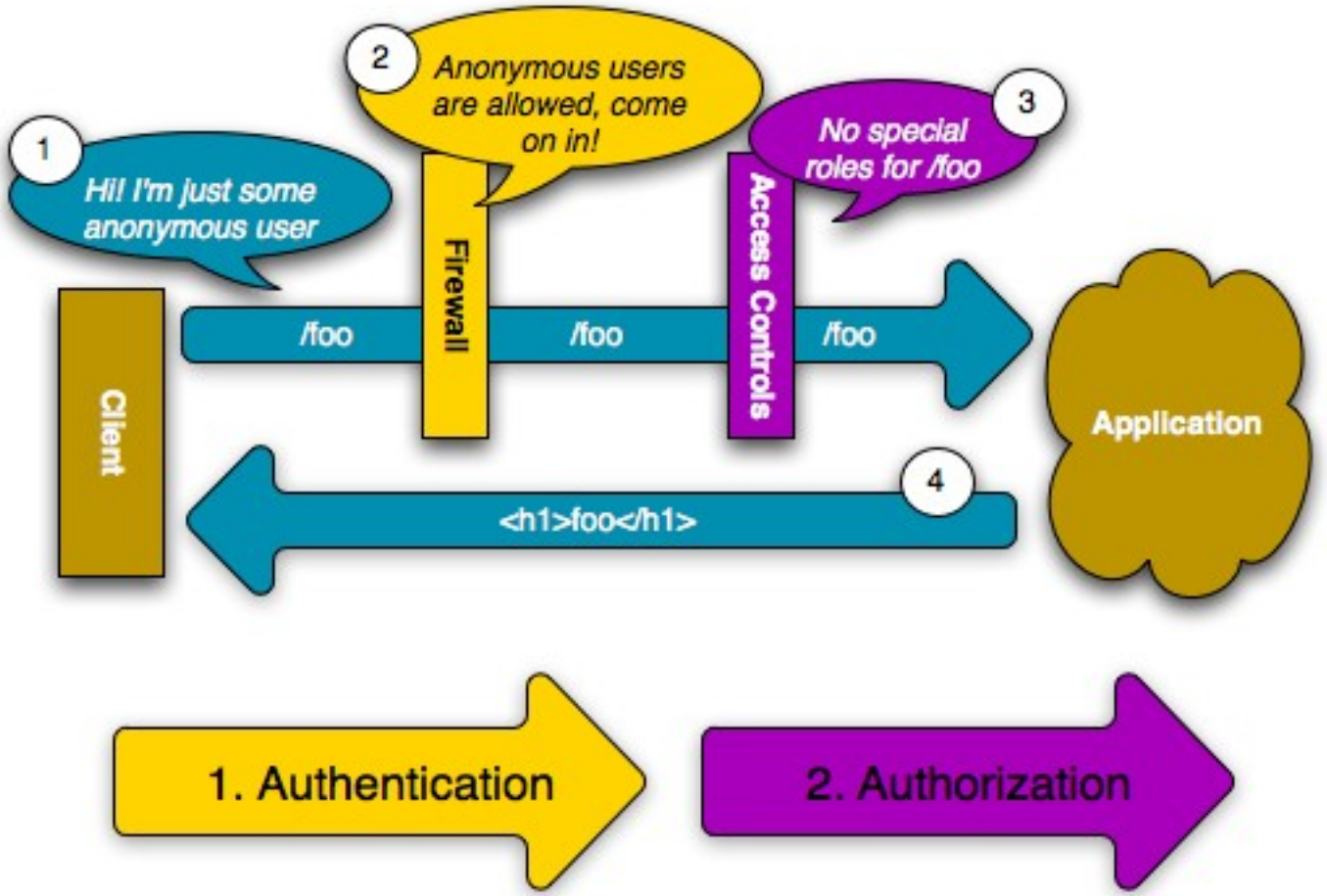
Segurança (Symfony2 Security)

- Processo de 2 passos
  - Autenticação
  - Autorização
- Evitar que usuários acessem recursos que eles não tem acesso



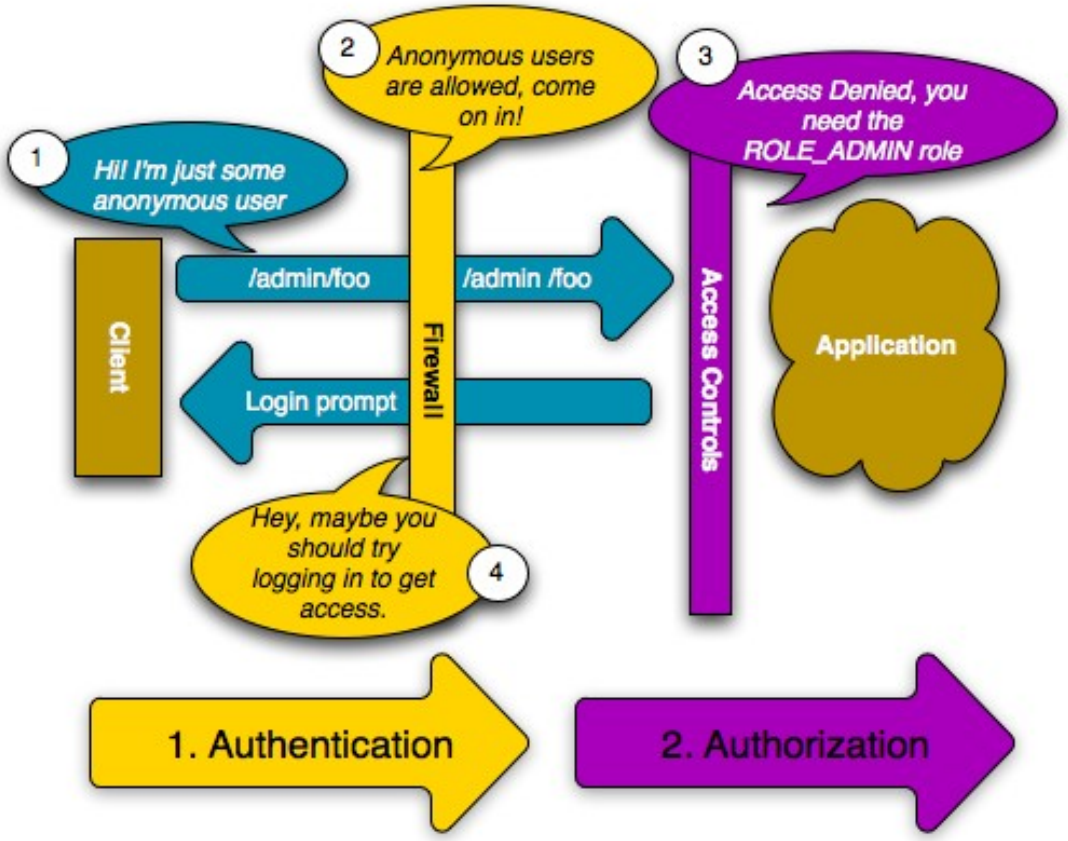
Autenticação

- Primeiro passo do processo
- O sistema está tentando descobrir quem é você



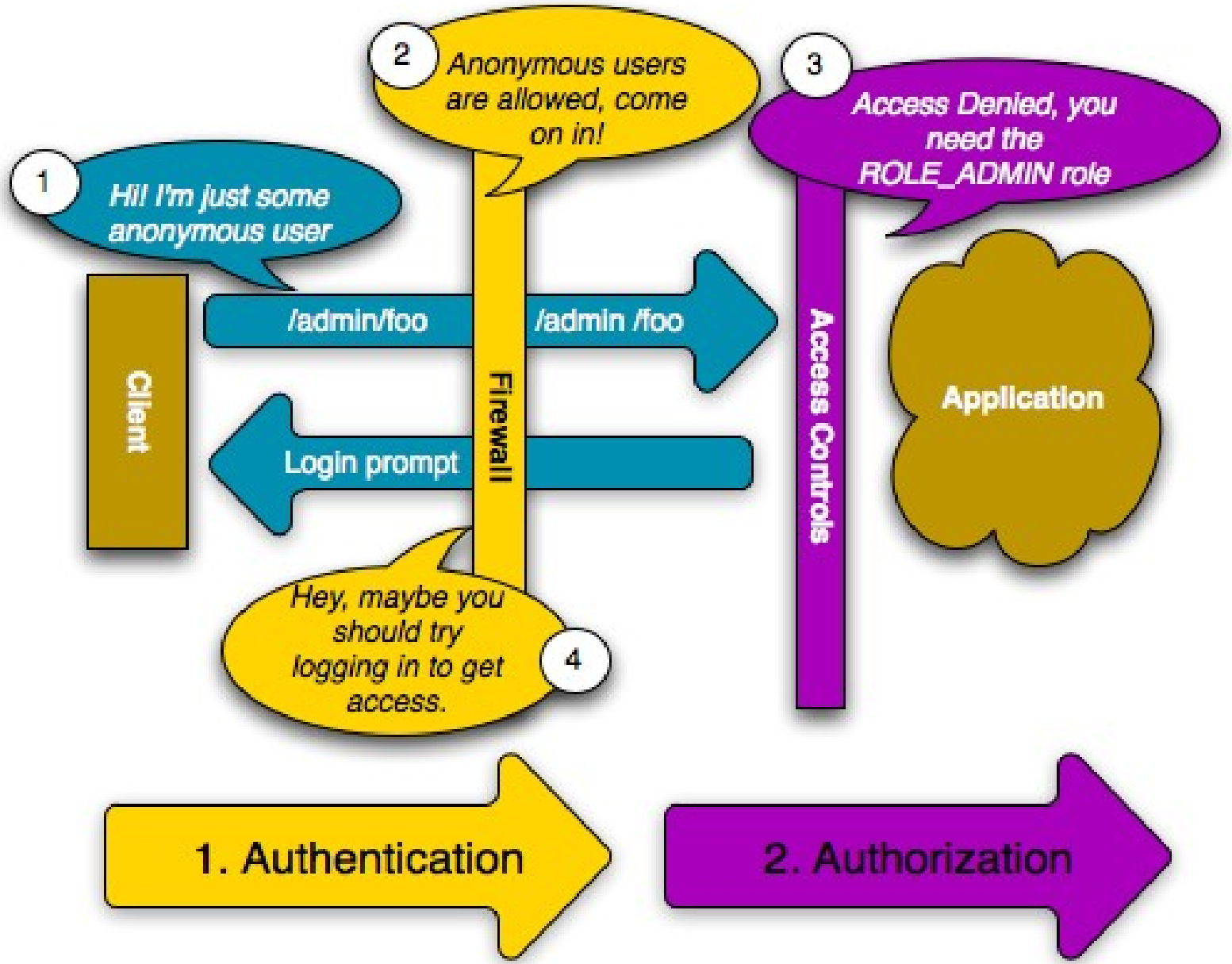
Autorização

- Segundo passo do processo
- Agora o sistema já sabe quem é você
- Próximo passo: determinar se você tem acesso ao recurso



Roles

- Indicam os papéis que um usuário tem na aplicação
- Serão utilizadas como permissões
- Podem ser organizadas hierarquicamente
- Podem ser criadas a qualquer momento seguindo a convenção ROLE_*
- Exemplos: ROLE_USER, ROLE_ADMIN, ROLE_SUPER_ADMIN, ...


1
Hi! I'm just some anonymous user
2
Anonymous users are allowed, come on in!
3
Access Denied, you need the ROLE_ADMIN role
Client
/admin/foo
/admin /foo
Firewall
Access Controls
Application
Login prompt
Hey, maybe you should try logging in to get access.
4
1. Authentication
2. Authorization

Dúvidas?

http://symfony.com/