



**Prof. David Fernandes de Oliveira  
Instituto de Computação  
UFAM**

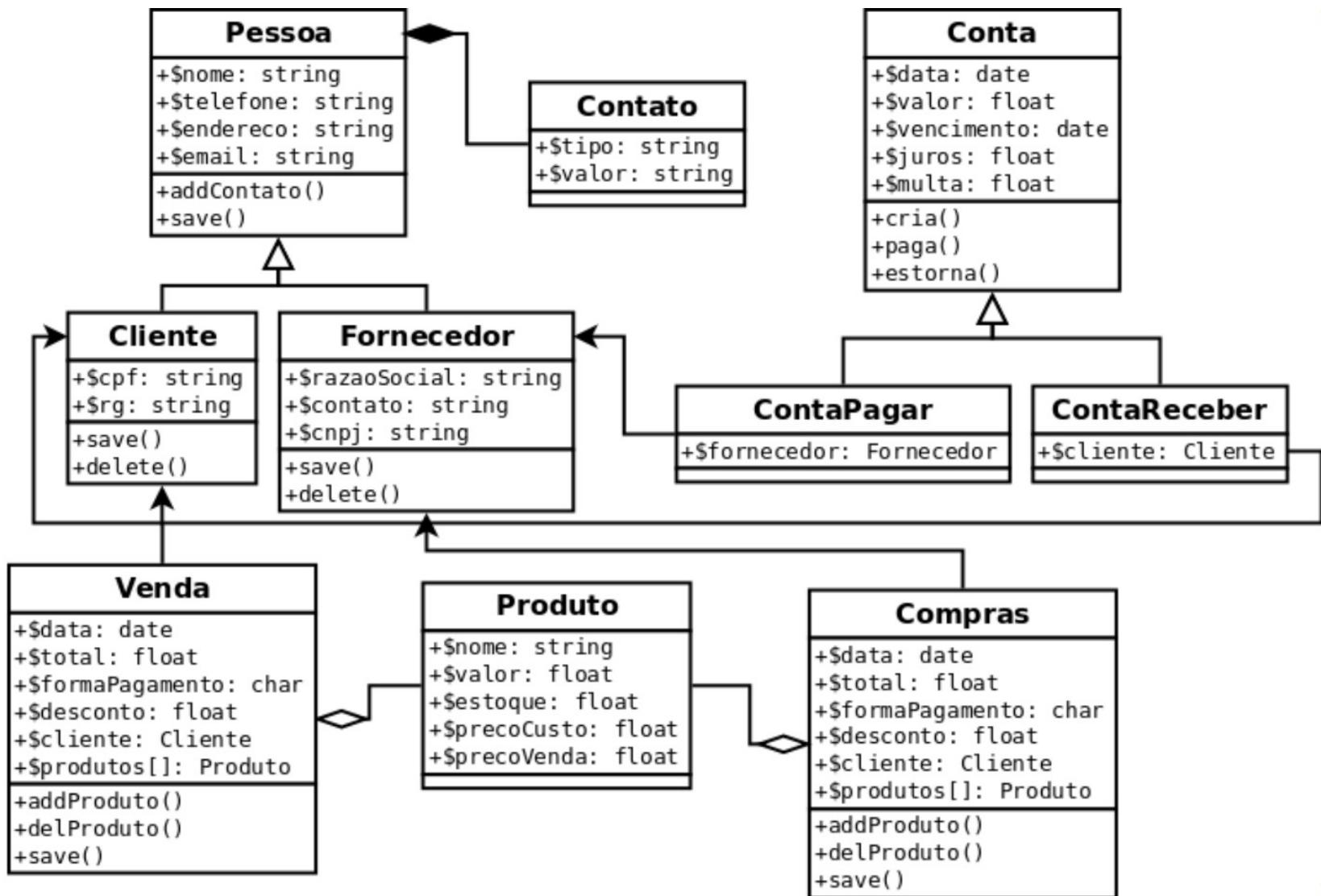
# Programação Orientada a Objetos

- O Yii é um framework orientado a objetos, de forma que para aprender sobre esse framework, é necessário conhecer OOP
- Usar OOP significa organizar sua aplicação como uma coleção de objetos que incorporam estrutura de dados e um conjunto de operações que manipulam estes dados



<b>Pessoa</b>	
+CTPS:	integer
+CPF:	integer
+RG:	integer
+Nome:	string
-Salario:	float
-Cargo:	string
-DtContrata:	date
-DtDemissao:	date
+Admitir(Data)	
+Demitir(Data)	
+Promover(Cargo,Salario)	
+GetSalario()	

# Programação Orientada a Objetos

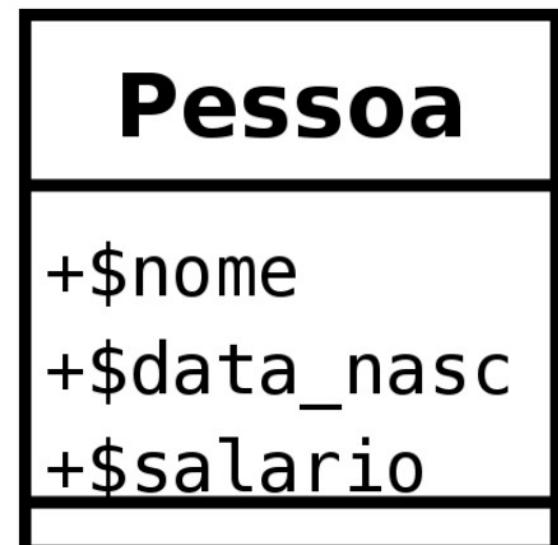


# Programação Orientada a Objetos

- A classe é uma estrutura estática utilizada para descrever objetos
  - A classe é um modelo (template) para criação de objetos
- Um grupo de objetos é descrito por uma classe, e um objeto é uma instância de uma classe
- Exemplo:
  - Classe: Fatura / Objeto: Fatura no. 5470
  - Classe: Pessoa / Objeto: João, Maria, etc...

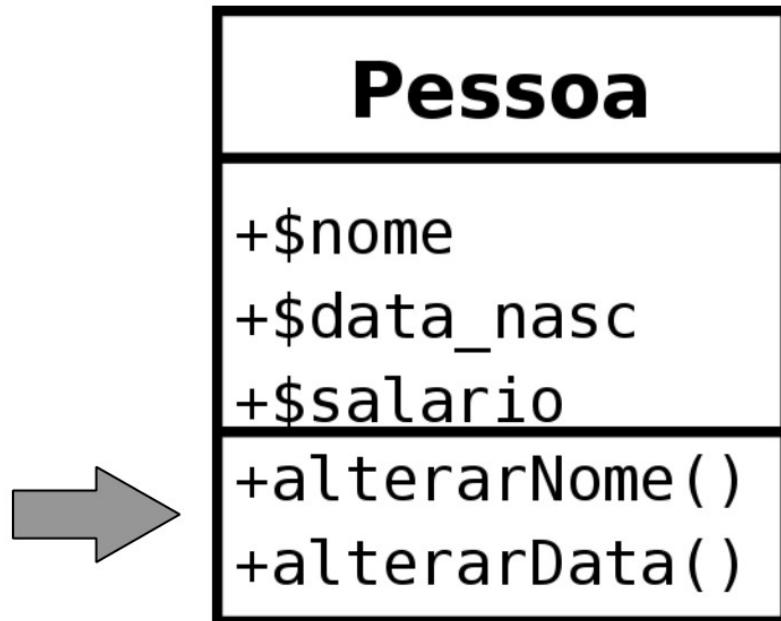
# Propriedades

- Propriedades são **atributos, características** de um objeto, que definem a identidade do objeto
  - Exemplo: Um ser humano é definido por um conjunto de atributos: físicos (altura, cor da pele, algura, cor do cabelo), psicológicos (personalidade, humor, empatia), de trabalho (capacidade, especialização, criatividade), dentre outros



# Métodos

- Os métodos são **operações** que definem o **comportamento** de um objeto
- Os métodos definem como o objeto irá se relacionar com o mundo externo
- É por meio de um método, que solicitamos que um objeto faça algo



# Definindo uma classe

```
<?php
class Pessoa
{
    // define os atributos
    public $Codigo;
    public $Nome;
    public $Altura;
    public $Idade;

    // define os métodos
    function setNome($nome)
    {
        $this->Nome = $nome;
    }

    function Crescer($centimetros)
    {
        $this->Altura += $centimetros;
    }

    function Envelhecer($anos)
    {
        $this->Idade += $anos;
    }
}
?>
```

A diagram illustrating the components of the PHP class. On the left, the code is shown. On the right, three grey arrows point from specific sections of the code to labels: one arrow points to the attribute definitions under the heading 'Atributos'; three other arrows point to the three methods (setNome, Crescer, and Envelhecer) under the heading 'Métodos'.

Atributos

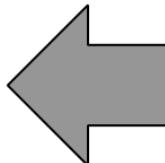
Métodos

Métodos

Métodos

# Instanciando objetos

```
<?php  
// inclui a classe  
include 'pessoa.class.php';  
  
// instancia o objeto  
$maria = new Pessoa;  
  
// define atributos  
$maria->Altura = 1.7;  
$maria->Idade = 28;  
  
// executa métodos  
$maria->setNome('Maria da Silva');  
$maria->Crescer(0.1);  
$maria->Envelhecer(1);  
  
// imprime o objeto  
var_dump($maria);  
?>
```



Instanciando o objeto

output

```
object(Pessoa)#1 (4) {  
    ["Codigo"]=> NULL  
    ["Nome"]=>  
        string(14) "Maria da Silva"  
    ["Altura"]=> float(1,8)  
    ["Idade"]=> int(29)  
}
```

# Construtores e Destruidores

- Um **construtor** é um método especial utilizado para definir o comportamento inicial de um objeto, ou seja, o comportamento no momento de sua criação
  - O método construtor é executado automaticamente no momento em que instanciamos um objeto por meio do operador **new**
- Um método **destrutor** ou **finalizador** é um método especial executado automaticamente quando o objeto é desalocado da memória
  - Ou seja, quando atribuímos o valor **NULL** ao objeto, quando utilizamos a função **unset ()** sobre o mesmo ou, em última instância, quando o programa é finalizado

# Construtores e Destruidores

```
<?php  
class Pessoa  
{  
    public $Codigo;  
    public $Nome;  
  
    // método construtor  
    function __construct($codigo, $nome)  
    {  
        $this->Codigo = $codigo;  
        $this->Nome = $nome;  
    }  
  
    // método destrutor  
    function __destruct()  
    {  
        echo "desalocando {$this->Nome}\n";  
    }  
  
    $maria = new Pessoa(27, 'Maria da Silva');  
    $joana = new Pessoa(28, 'Joana Maranhão');  
    var_dump($maria, $joana);  
    unset($maria);  
    unset($joana);  
?>
```

## output

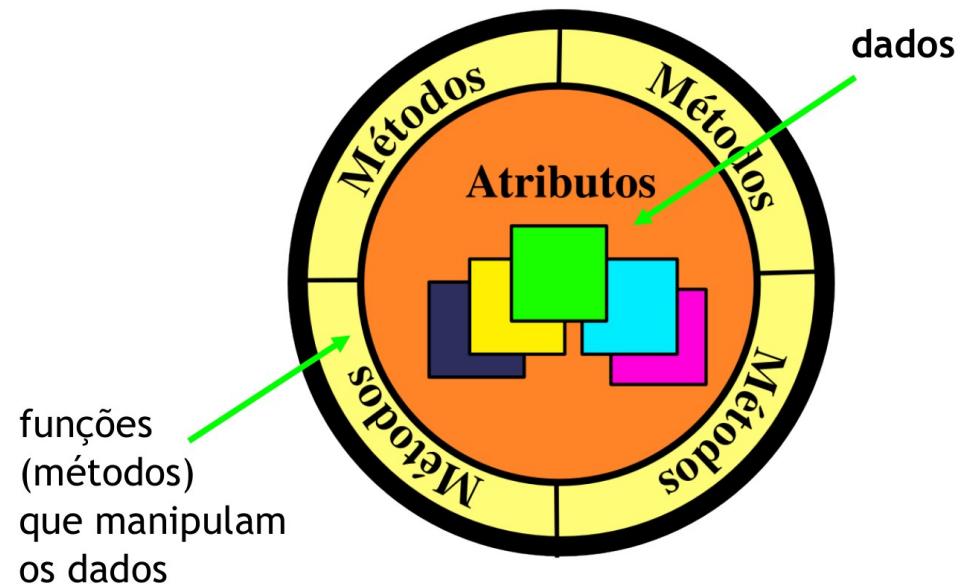
```
object(Pessoa)#1 (2) {  
    ["Codigo"]=> int(27)  
    ["Nome"]=>  
    string(14) "Maria da Silva"  
}  
  
object(Pessoa)#2 (2) {  
    ["Codigo"]=> int(28)  
    ["Nome"]=>  
    string(14) "Joana Maranhão"  
}  
  
desalocando Maria da Silva  
desalocando Joana Maranhão
```

Método construtor

Método destrutor

# Encapsulamento

- A proteção de acesso ao conteúdo de um objeto se dá por meio de mecanismos de encapsulamento
  - O encapsulamento visa separar os aspectos externos de um objeto dos detalhes internos daquele objeto
  - É uma forma de proteger os atributos, evitando que estes sejam manipuladas indevidamente



# Encapsulamento

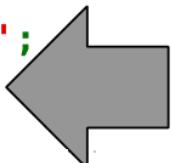
```
<?php  
class Pessoa  
{  
    private $Codigo;  
    public $Nome;  
    private $Altura;  
  
    function __construct($codigo)  
    {  
        $this->Codigo = $codigo;  
    }  
  
    function setAltura($altura)  
    {  
        $this->Altura = $altura;  
    }  
  
}  
  
$maria = new Pessoa(27);  
$maria->Nome = 'Maria da Silva';  
$maria->setAltura(1.7);  
  
$joana = new Pessoa(28);  
$joana->Nome = 'Joana Maranhão';  
$joana->Altura = 1.8;  
?>
```

output

```
Fatal error: Cannot access private  
property Pessoa::$Altura in  
exemplo.php on line 25
```

## Tipos de visibilidade

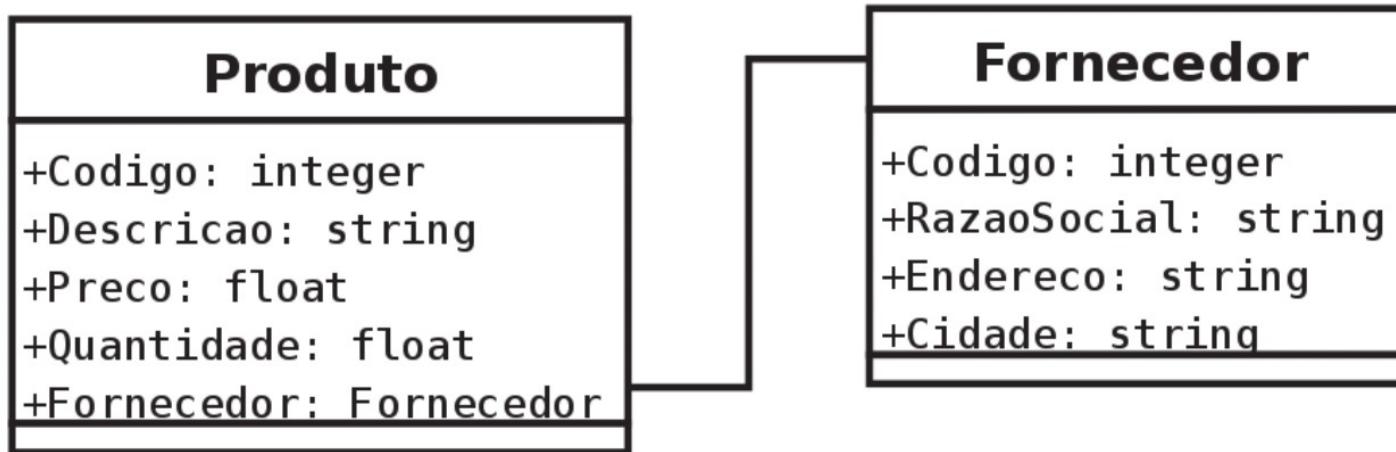
```
PRIVATE :: SOMENTE PRÓPRIA CLASSE  
PROTECTED :: CLASSE E DESCENDENTES  
PUBLIC :: DE QUALQUER PONTO
```



Acesso não permitido

# Relacionamento

- Um **relacionamento** permite que um objeto possua uma referência à outro objeto, podendo visualizar seus atributos ou mesmo acionar uma de suas funcionalidades
- A forma mais comum de implementar um relacionamento é ter um objeto como atributo de outro



# Relacionamento

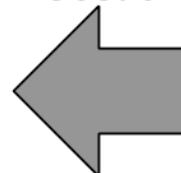
```
<?php
class Produto
{
    public $Codigo;
    public $Descricao;
    public $Preco;
    public $Fornecedor;

    // exibe os dados do produto
    function ExibeDados()
    {
        echo 'Codigo: ' . $this->Codigo . "<br>\n";
        echo 'Descricao: ' . $this->Descricao . "<br>\n";
        echo 'Preço: ' . $this->Preco . "<br>\n";
        echo 'Fornecedor:' . $this->Fornecedor->Nome . "<br>\n";
    }

    // atribui um fornecedor ao produto
    function setFornecedor(Fornecedor $fornecedor)
    {
        $this->Fornecedor = $fornecedor;
    }
}
?>
```

```
<?php
class Fornecedor
{
    public $Nome;
    public $Telefone;
    public $Endereco;
}

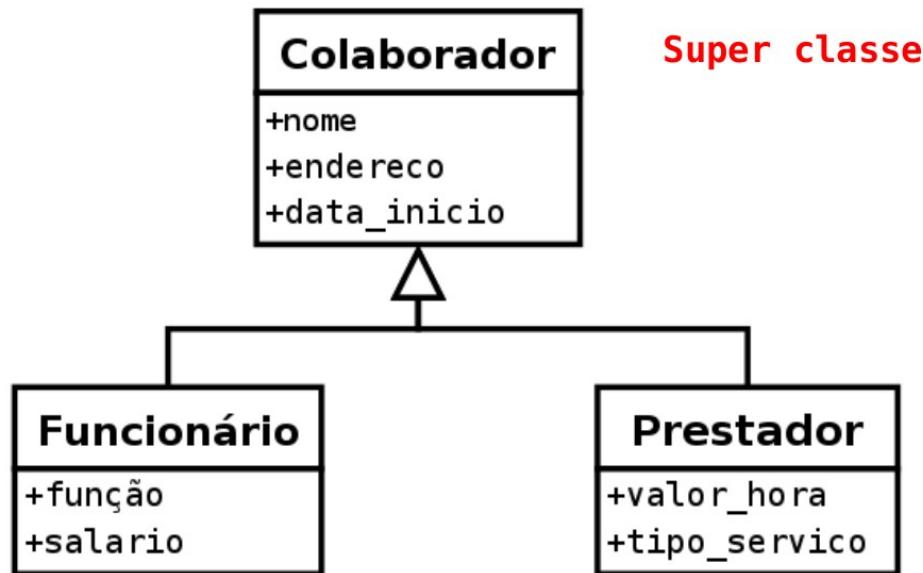
?>
```



Associação interna

# Herança

- Uma **herança** é um tipo de relacionamento que permite **especializar** uma classe, criar versões refinadas dela
- Na herança, as classes são organizadas em **hierarquias**
- A herança é uma forma de reutilizar componentes de software aperfeiçoando-os ou adicionando características específicas



# Herança

- A herança permite o **compartilhamento** de atributos e métodos entre as classes de uma hierarquia
- Cada subclasse **herda** todas as propriedades (atributos e métodos) de suas **ancestrais**
- Uma **subclasse** pode estender ou redefinir a estrutura e/ou o comportamento de sua super classe
  - Leia-se um objeto – é tipo de – outro objeto
- É um poderoso instrumento de **reusabilidade**, pois
  - Permite que atributos e operações comuns à hierarquia sejam especificados apenas uma vez
  - Permite que novas classes sejam criadas contendo apenas a diferença entre ela e a classe-pai

# Herança

```
<?php
abstract class Conta
{
    private $Agencia;
    private $Numero;
    protected $Saldo;

    function __construct($agencia, $numero, $saldo)
    {
        $this->Agencia = $agencia;
        $this->Numero = $numero;
        $this->Saldo = $saldo;
    }

    function Depositar($valor)
    {
        $this->Saldo += $valor;
    }

    function getSaldo()
    {
        return $this->Saldo;
    }
}
?>
```

Uma classe abstrata  
não pode ser instanciada  
diretamente.

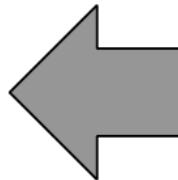
# Herança

```
<?php
final class ContaCorrente extends Conta
{
    public $Limite;

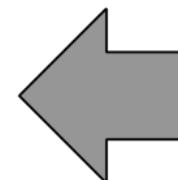
    function Retirar($valor)
    {
        if ($this->Saldo + $this->Limite > $valor)
        {
            $this->Saldo -= $valor;
        }
    }
}

final class ContaPoupanca extends Conta
{
    function Retirar($valor)
    {
        if ($this->Saldo > $valor)
        {
            $this->Saldo -= $valor;
        }
    }
}
?>
```

Uma classe final não pode ser extendida.



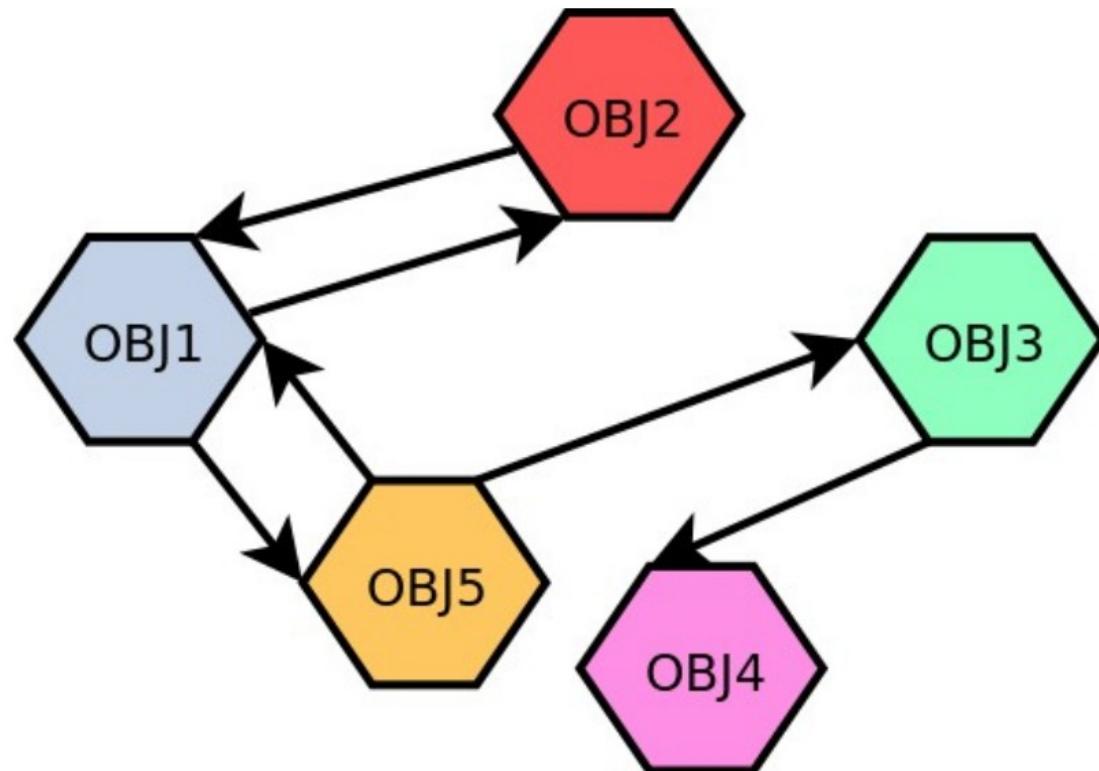
Classe filha



Classe filha

# Sistema OO

- Um sistema OO é modelado, implementado e efetivamente funciona como um conjunto de objetos que interagem entre si.



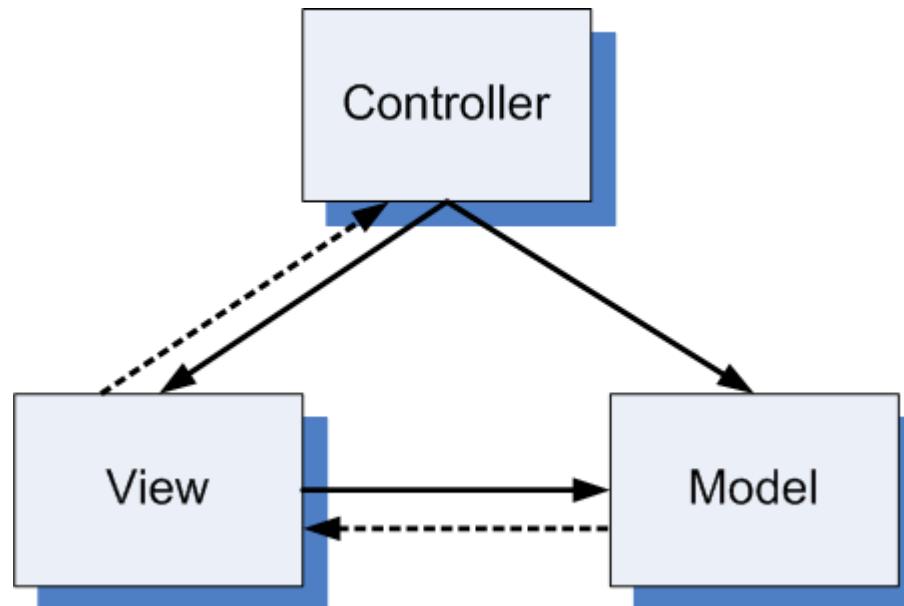
# O que é um Framework

- Um **framework** é um conjunto de bibliotecas que visam facilitar o desenvolvimento de sistemas completos
- Desenvolver uma aplicação do zero é inviável e muito trabalhoso em muitos casos
  - Reusabilidade de código torna tudo mais fácil, rápido, confiável, e possivelmente mais seguro



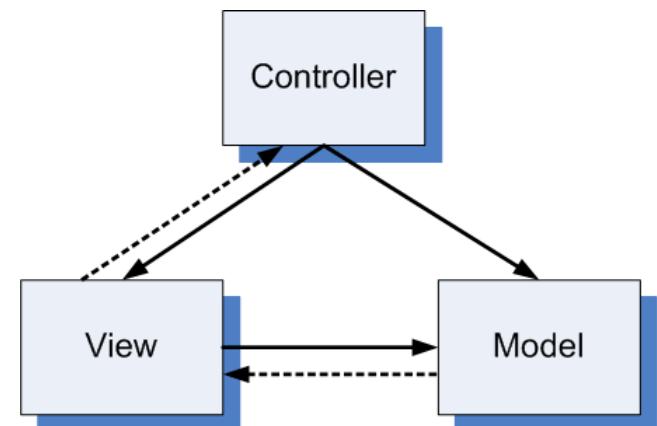
# Conceitos Fundamentais

- Os frameworks são criados utilizando certos padrões de design e desenvolvimento
- Para o Yii, os dois padrões de desenvolvimento mais importantes são **Programação orientada a objetos** (POO), e o **Modelo-Visão-Controlador** (MVC)

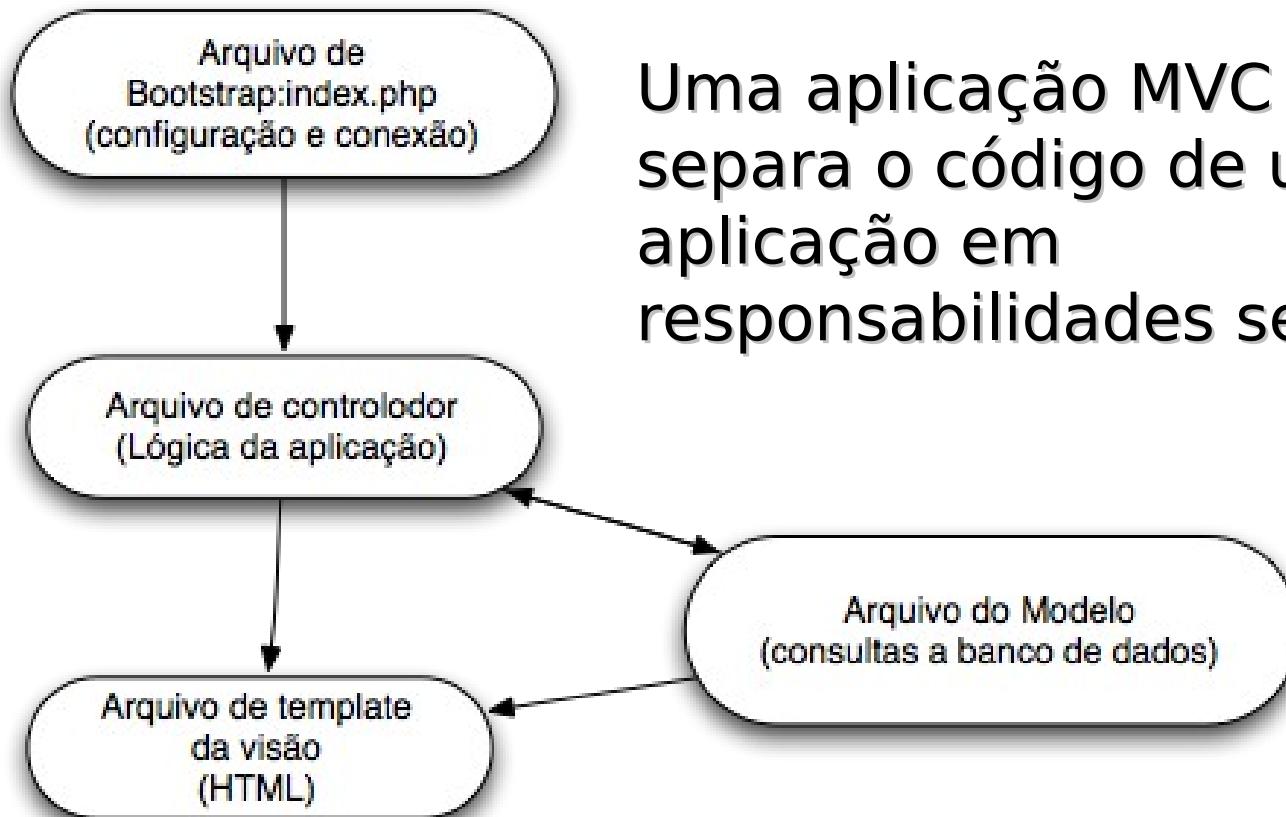


# Conceitos Fundamentais

- O MVC organiza uma aplicação em três partes de código: os **dados**, as **ações** dos usuários, e a **interface**
- Desta forma, o MVC representa uma aplicação em três partes distintas:
  - **Modelo**, que é uma combinação dos dados usados pela aplicação, com o modelo de negócios associado a tais dados
  - **View**, que é a interface com que os usuários interagem com a aplicação
  - **Controlador**, que é o agente que responde as ações dos usuários, fazendo uso dos modelos e apresentando os resultados através das views



# Modelo-Visão-Controlador



# Trabalhando com o Composer

- Muitas bibliotecas e frameworks PHP, incluindo o Yii 2, são instalados através do **Composer**
- O Composer é um gerenciador de dependências para projetos desenvolvidos na linguagem PHP
- Através do Composer, você define quais são os requerimentos de seu projeto PHP
- Após isso, o Composer irá abaixar e instalar todos os pacotes definidos por você



# Trabalhando com o Composer

- Muitas bibliotecas e frameworks PHP, incluindo o Yii 2, são instalados através do **Composer**
- O Composer é uma ferramenta poderosa para gerenciar dependências PHP. Ele é essencial para um bom programador PHP sem ele requeridos pacotes.
- Após isso, o Composer irá abaixar e instalar todos os pacotes definidos por você



# Instalando o Composer

- Para instalar o Composer, basta executar o seguinte comando no diretório da sua aplicação:

```
curl -sS https://getcomposer.org/installer | php
```

```
david@coiote:/home/www/icomp
david@coiote:/home/www/icomp$ curl -sS https://getcomposer.org/installer | php
#!/usr/bin/env php
All settings correct for using Composer
Downloading...

Composer successfully installed to: /home/www/icomp/composer.phar
Use it: php composer.phar
david@coiote:/home/www/icomp$
```



# Instalando o Composer

- Adicionalmente, você vai precisar instalar um plugin do Composer chamado **Asset Manager**

```
php composer.phar global require  
"fxp/composer-asset-plugin:1.0.0"
```

```
david@coiote: /home/www/icomp  
david@coiote:/home/www/icomp$ php composer.phar global require "fxp/composer-asse  
t-plugin:1.0.0"  
Changed current directory to /home/david/.composer  
.composer.json has been updated  
Loading composer repositories with package information  
Updating dependencies (including require-dev)  
- Removing fxp/composer-asset-plugin (v1.0.0-beta3)  
- Installing fxp/composer-asset-plugin (v1.0.0)  
  Downloading: 100%  
  
Writing lock file  
Generating autoload files  
david@coiote:/home/www/icomp$ █
```



# Criando um Projeto com Yii 2

- A versão 2 do framework Yii define dois possíveis templates: **basic** e **advanced**
- A versão advanced possui **back-end** e **front-end** separados, além de definir um modelo **User**
- A versão basic é mais simples, e será adotada ao longo das aulas



# Criando um Projeto com Yii 2

- Para criar um novo projeto com o Yii 2 usando o template basic, basta dar o seguinte comando:

```
php composer.phar create-project  
--prefer-dist yiisoft/yii2-app-basic
```

```
david@coiote:/home/www/icomp$ php composer.phar create-project --prefer-dist yiisoft/yii2-app-basic  
Installing yiisoft/yii2-app-basic (2.0.4)  
- Installing yiisoft/yii2-app-basic (2.0.4)  
  Downloading: 100%  
  
Created project in /home/www/icomp/yii2-app-basic  
Loading composer repositories with package information  
Installing dependencies (including require-dev)  
- Installing yiisoft/yii2-composer (2.0.3)  
  Downloading: 100%  
  
- Installing ezyang/htmlpurifier (v4.6.0)
```



# Criando um Projeto com Yii 2

- Para criar um novo projeto com o Yii 2 usando o template basic, basta dar o seguinte comando:

```
php
```

**Exercício:** Criar um diretório para uma nova aplicação, instalar o composer e o Yii 2 nesse diretório.

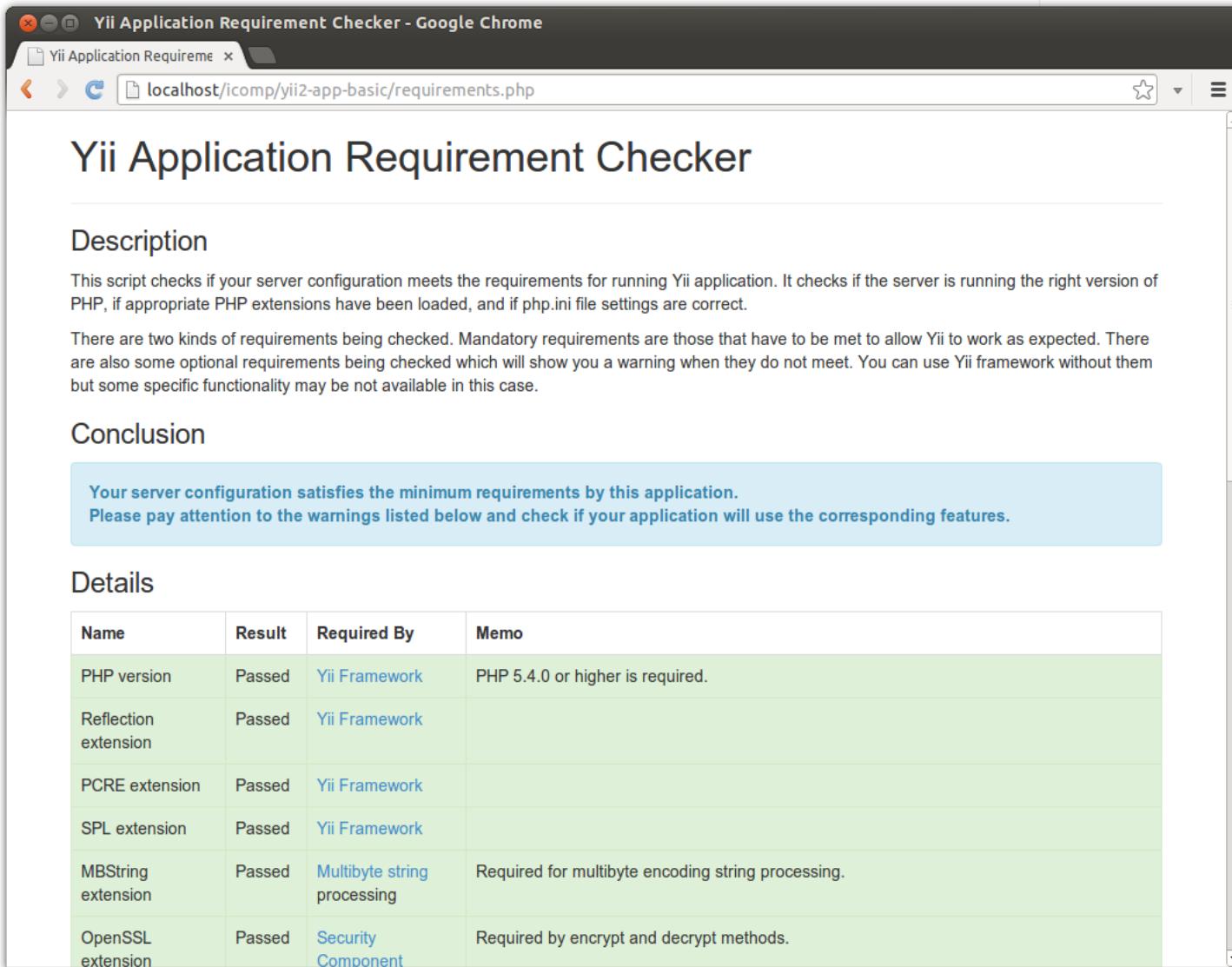
```
--p  
david@soft/yi...  
er-dist yiis
```

```
Installing yiisoft/yii2-app-basic (2.0.4)  
- Installing yiisoft/yii2-app-basic (2.0.4)  
  Downloading: 100%
```

```
Created project in /home/www/icomp/yii2-app-basic  
Loading composer repositories with package information  
Installing dependencies (including require-dev)  
- Installing yiisoft/yii2-composer (2.0.3)  
  Downloading: 100%  
  
- Installing ezyang/htmlpurifier (v4.6.0)
```



# Checando Requerimentos



The screenshot shows a Google Chrome browser window with the title "Yii Application Requirement Checker - Google Chrome". The address bar displays "localhost/icomp/yii2-app-basic/requirements.php". The main content area is titled "Yii Application Requirement Checker". It contains a "Description" section with text about the script's purpose and two types of requirements. Below this is a "Conclusion" section with a message about the server configuration and a "Details" section with a table of requirements.

## Description

This script checks if your server configuration meets the requirements for running Yii application. It checks if the server is running the right version of PHP, if appropriate PHP extensions have been loaded, and if php.ini file settings are correct.

There are two kinds of requirements being checked. Mandatory requirements are those that have to be met to allow Yii to work as expected. There are also some optional requirements being checked which will show you a warning when they do not meet. You can use Yii framework without them but some specific functionality may be not available in this case.

## Conclusion

Your server configuration satisfies the minimum requirements by this application.  
Please pay attention to the warnings listed below and check if your application will use the corresponding features.

## Details

Name	Result	Required By	Memo
PHP version	Passed	Yii Framework	PHP 5.4.0 or higher is required.
Reflection extension	Passed	Yii Framework	
PCRE extension	Passed	Yii Framework	
SPL extension	Passed	Yii Framework	
MBString extension	Passed	Multibyte string processing	Required for multibyte encoding string processing.
OpenSSL extension	Passed	Security Component	Required by encrypt and decrypt methods.

# Acessando seu Projeto

The screenshot shows a Yii application running in Google Chrome. The title bar says "My Yii Application - Google Chrome". The address bar shows "localhost/icompiyii2-app-basic/web/". The page has a dark header with "My Company" on the left and "Home" (which is highlighted), "About", "Contact", and "Login" on the right. The main content area features a large "Congratulations!" heading, followed by the text "You have successfully created your Yii-powered application." Below this is a green button labeled "Get started with Yii". Further down, there are two sections, each with a "Heading" and some placeholder text from the Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. At the bottom, there is a "Yii Documentation »" link and a Yii Debugger footer with various metrics.

My Company

Home About Contact Login

# Congratulations!

You have successfully created your Yii-powered application.

Get started with Yii

## Heading

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

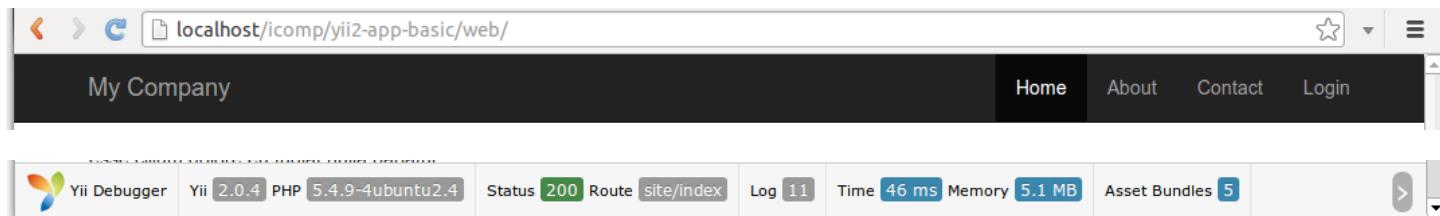
Yii Documentation »

## Heading

yii 2.0.4 PHP 5.4.9-4ubuntu2.4 Status 200 Route site/index Log 11 Time 46 ms Memory 5.1 MB Asset Bundles 5

# Acessando seu Projeto

- Como funcionalidades, a aplicação que você acabou de gerar já inclui:
  - Uma página principal
  - Uma página About
  - Um formulário de contato, com Captcha
  - Um formulário de login
  - Funcionalidades de login/logout (demo/demo)
  - Twitter Bootstrap
  - Um debugger



# Diretório do Projeto

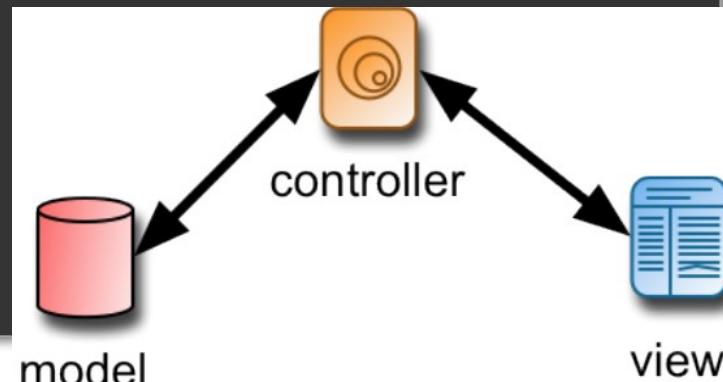
- No diretório **raiz da aplicação**, também conhecido como **diretório base**, encontramos os seguintes arquivos e diretórios:

```
david@coiote: /home/www/icomp/yii2-app-basic
david@coiote:/home/www/icomp/yii2-app-basic$ ls
assets           config        models        tests      yii
commands         controllers   README.md    vendor    yii.bat
composer.json    LICENSE.md   requirements.php views
composer.lock    mail          runtime      web
david@coiote:/home/www/icomp/yii2-app-basic$ █
```

# Diretório do Projeto

- Observe que o diretório raiz possui um diretório para cada um dos componentes de um sistema MVC: **controllers**, **models** e **views**:

```
david@coiote: /home/www/icomp/yii2-app-basic
david@coiote:/home/www/icomp/yii2-app-basic$ ls
assets           config        models      tests      yii
command          controllers  README.md   vendor    yii.bat
composer.json    LICENSE.md   requirements.php views
composer.lock    mail         runtime
david@coiote:/home/www/icomp/yii2-app-basic$
```



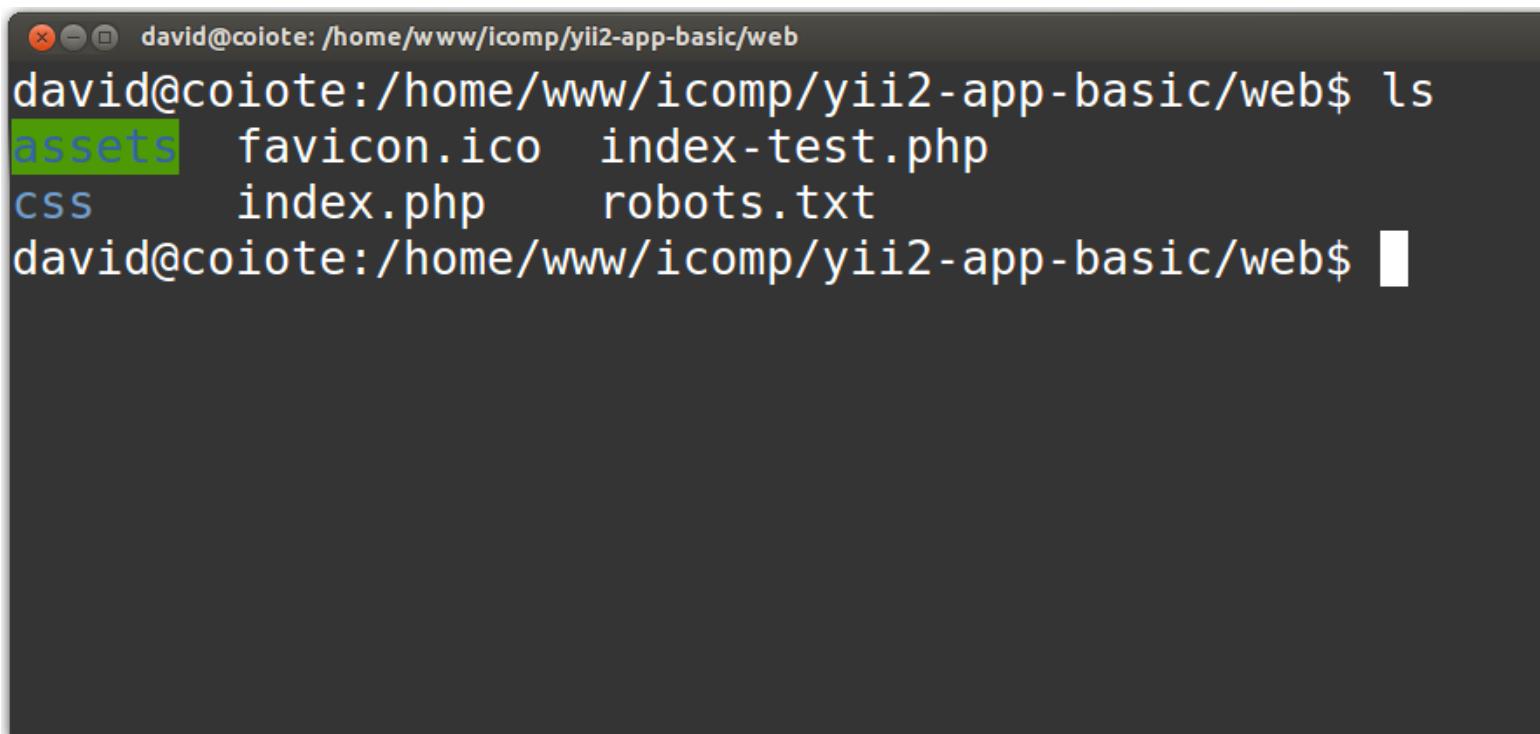
# Diretório do Framework Yii 2

- Diferentemente da versão 1.0 do Yii, o framework fica instalado em um diretório dentro de **vendor**
- O diretório **vendor** contém todas as bibliotecas e frameworks instalados via **composer**

```
david@coiote:/home/www/icomp/yii2-app-basic/vendor/yiisoft$ ls  
extensions.php  yii2-codeception  yii2-faker  
yii2           yii2-composer    yii2-gii  
yii2-bootstrap  yii2-debug     yii2-swiftmailer  
david@coiote:/home/www/icomp/yii2-app-basic/vendor/yiisoft$ █
```

# Diretório do Framework Yii 2

- O diretório **web** é o diretório raiz do website (não da aplicação)



```
david@coiote: /home/www/icomp/yii2-app-basic/web
david@coiote:/home/www/icomp/yii2-app-basic/web$ ls
assets  favicon.ico  index-test.php
css      index.php    robots.txt
david@coiote:/home/www/icomp/yii2-app-basic/web$ █
```

A screenshot of a terminal window titled "david@coiote: /home/www/icomp/yii2-app-basic/web". The window shows the output of the "ls" command, listing files and directories: assets, favicon.ico, index-test.php, css, index.php, and robots.txt. The "assets" directory is highlighted with a green background. The terminal has a dark theme with white text and a black background.

# O Arquivo Bootstrap

- O arquivo `index.php`, incluso no template básico do Yii 2, possui o seguinte conteúdo:

```
david@coiote: /home/www/icomp/yii2-app-basic/web
<?php

// comment out the following two lines when deployed to production
defined('YII_DEBUG') or define('YII_DEBUG', true);
defined('YII_ENV') or define('YII_ENV', 'dev');

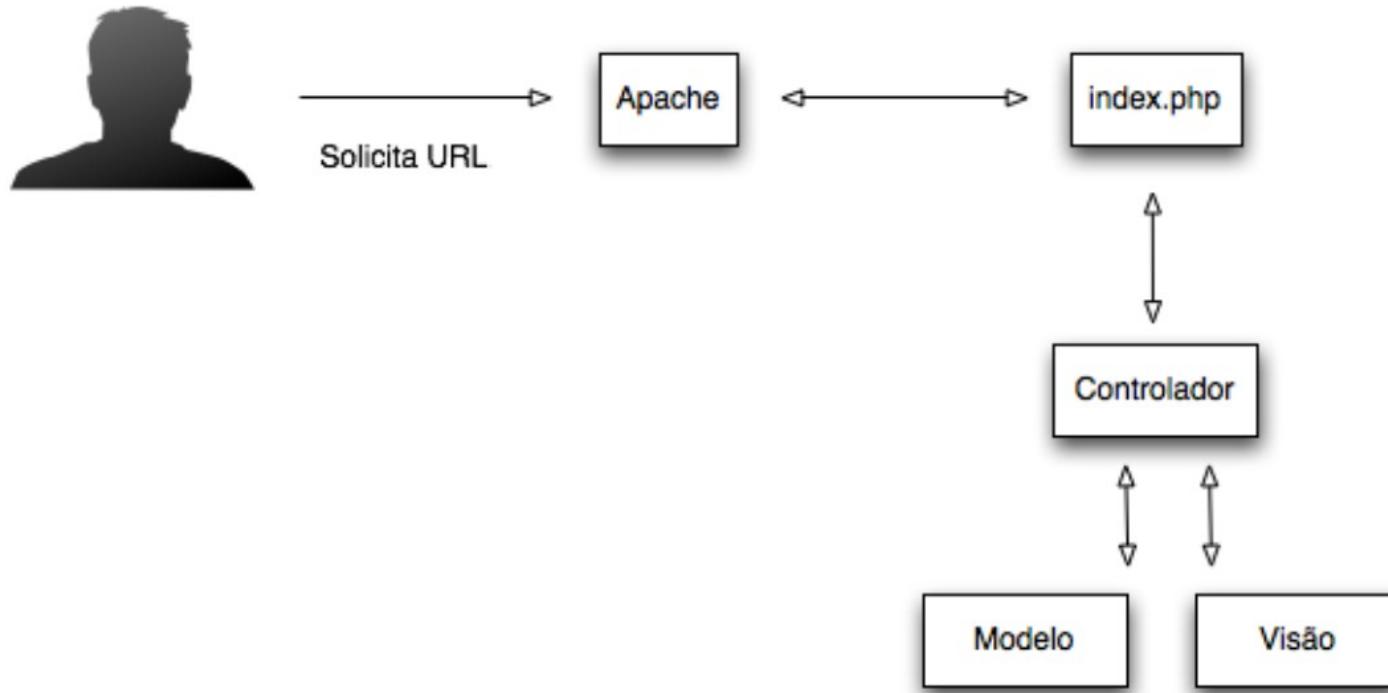
require(__DIR__ . '/../vendor/autoload.php');
require(__DIR__ . '/../vendor/yiisoft/yii2/Yii.php');

$config = require(__DIR__ . '/../config/web.php');

(new yii\web\Application($config))->run();
~
~
```

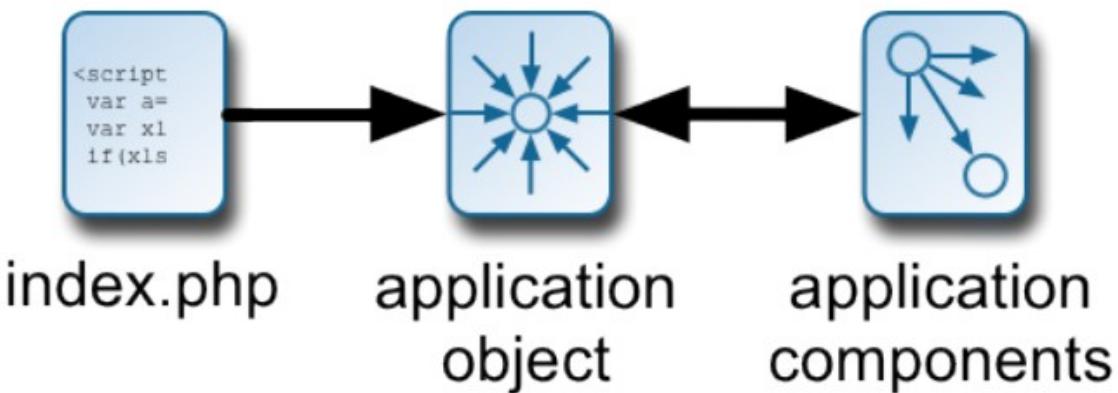
# Instalando o YII Framework

- O arquivo `index.php` é conhecido como **Bootstrap**, que recebe todas as requisições de acesso à aplicação
  - Adotando o padrão de projeto **front controller**
- Uma vez que uma requisição é recebida, o Bootstrap invoca o controlador especificado na URL solicitada



# O Arquivo Bootstrap

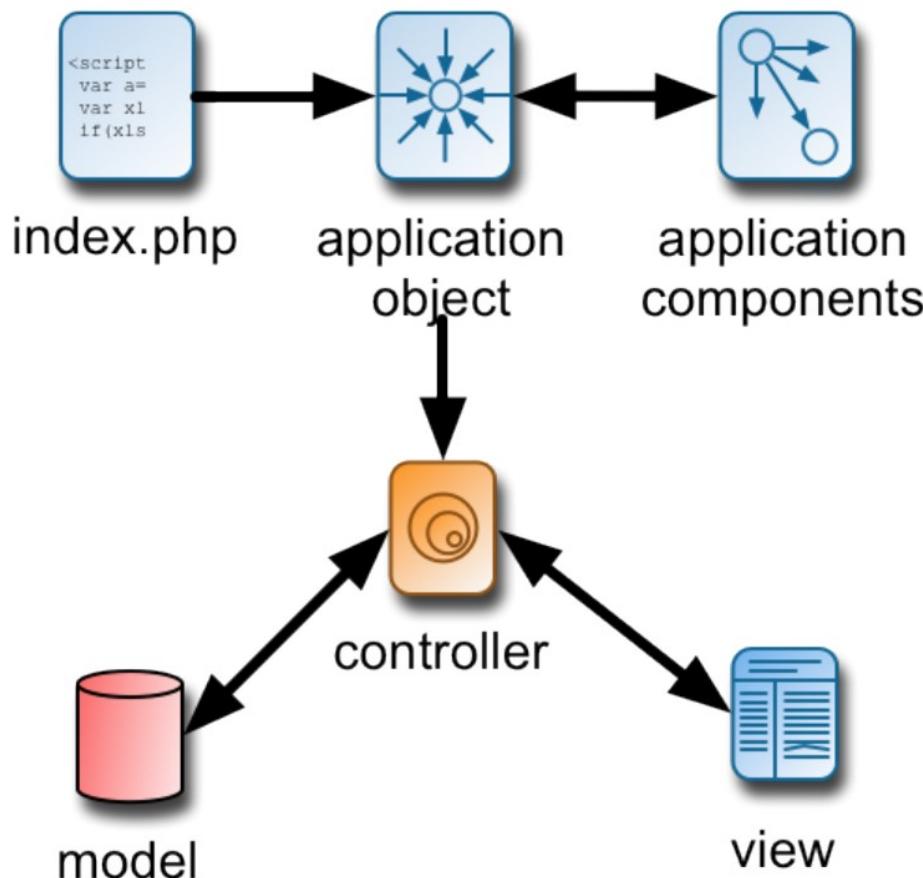
- O arquivo bootstrap cria um objeto da aplicação que carrega todos os componentes necessários



```
$config = require(__DIR__ . '/../config/web.php');  
  
(new yii\web\Application($config))->run();
```

# O Paradigma MVC

- Após o objeto da aplicação ser carregado, o roteamento identifica o controlador a ser executado



# Roteamento

- Em sites construídos sem o uso de frameworks, as requisições envolvem diferentes arquivos PHP
  - Ex: `http://example.com/page.php`
- No caso do Yii, todas as requisições são feitas para o arquivo bootstrap, isto é, o `index.php`
- Para executar uma requisição específica, uma variável de rota `r` é adicionada a cada URL
- Nessa variável, indicamos o **controlador**, e a **action** que processará a requisição:
  - `index.php?r=ControllerID/ActionID`

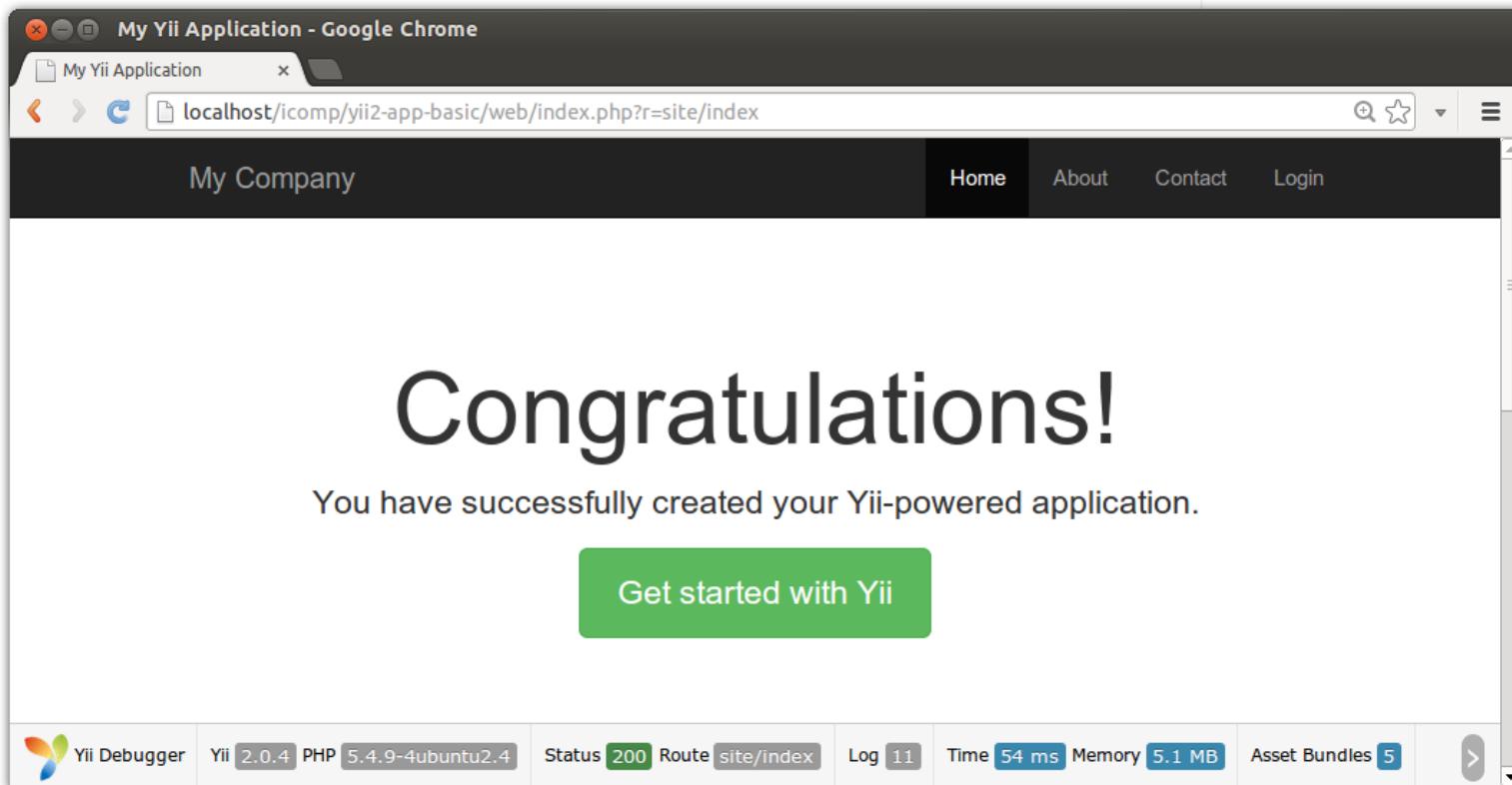
# Roteamento

- Os **controladores** são organizados em **actions**, sendo que cada action é responsável por carregar uma única página do site

```
david@coiote:/home/www/icomp/yii2-app-basic/controllers$ cat Site  
Controller.php | grep -v actions | grep action  
    public function actionIndex()  
    public function actionLogin()  
    public function actionLogout()  
    public function actionContact()  
    public function actionAbout()  
david@coiote:/home/www/icomp/yii2-app-basic/controllers$ █
```

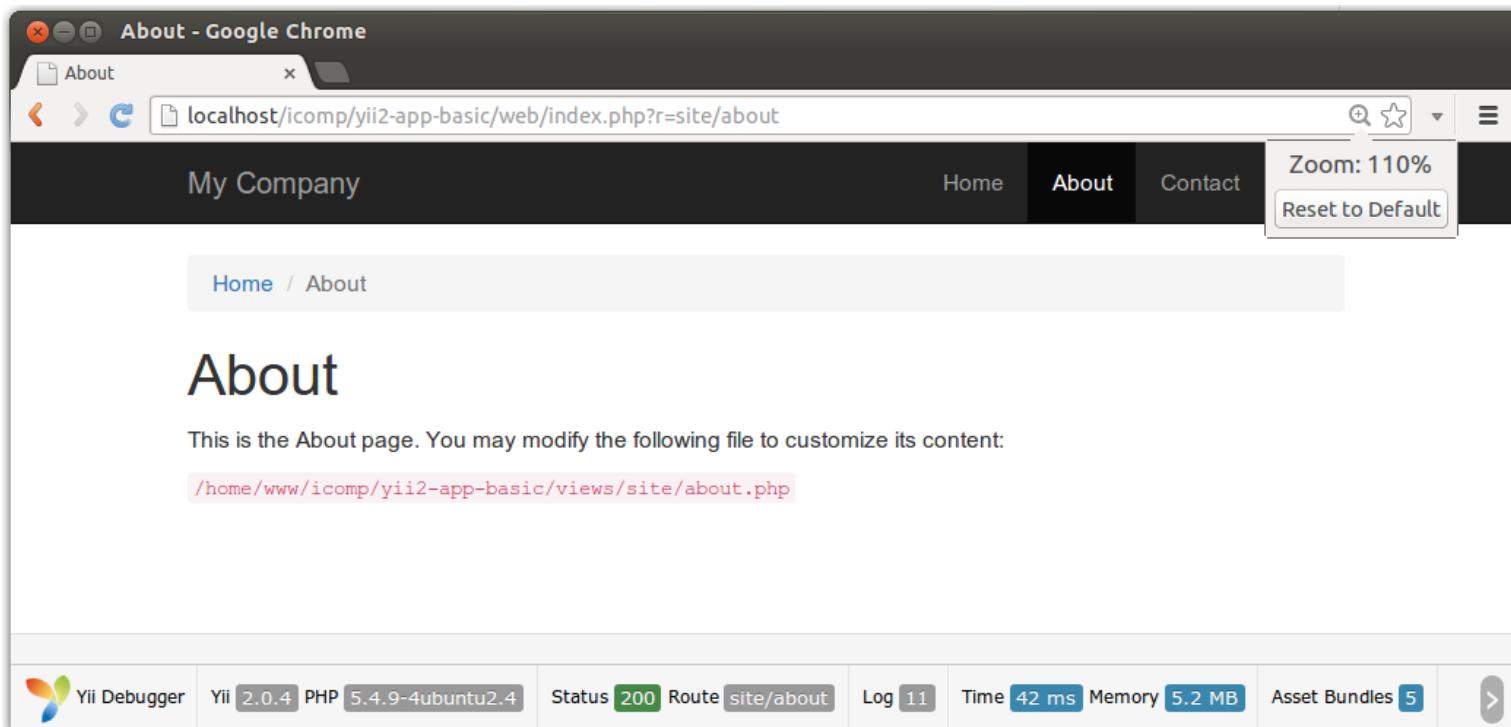
# Roteamento

- Por exemplo, `index.php?r=site/index` carrega a action **index** do controlador **site**



# Roteamento

- Por exemplo, `index.php?r=site/about` carrega a action **about** do controlador **site**

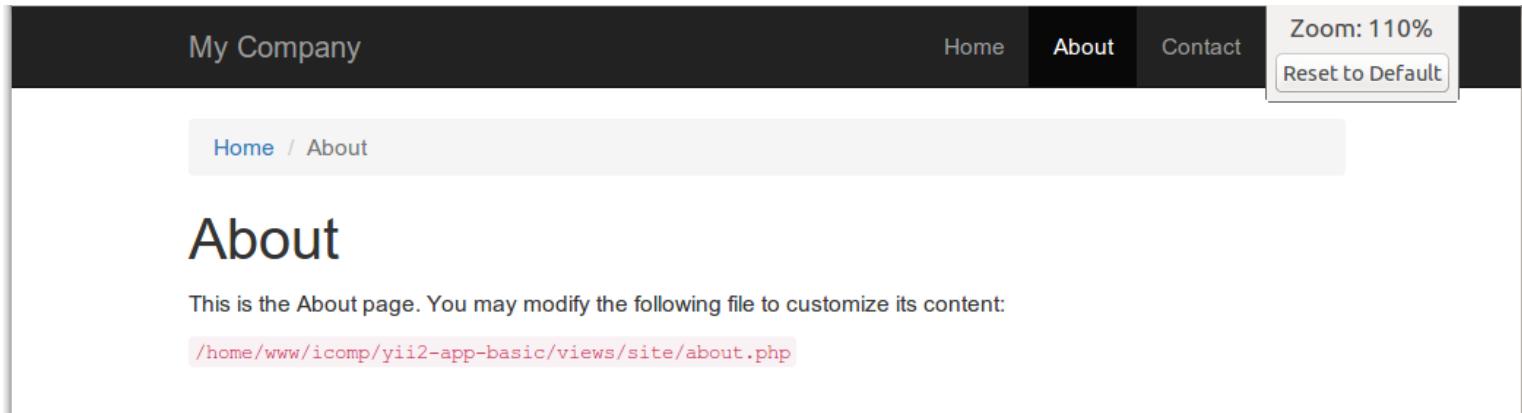


# Roteamento

- Se a action não for informada na URL, então a action padrão do controlador informado será executada
  - Normalmente, a action padrão de um controlador é a **index**
  - Por exemplo, **index.php?r=aluno** irá executar o código da action **index** do controlador **aluno**
- Se o controlador também não for informado, então o controlador padrão do site será executado
  - Por default, o controlador padrão é **site**, que possui as actions da página principal, de login, e de contato

# Controller X View

- A página **about** não usa nenhum dado do banco de dados para ser carregada
- Por causa disso, ela não faz uso de **models** – faz uso apenas de um controller e de uma view
- Vamos tentar entender como o controlador **site** e a action **about** fazem para carregar a página



# Controller X View

- Os controladores ficam armazenados no diretório **controllers**, do diretório base da aplicação
  - Cada controlador é um arquivo desse diretório
- Logo após ser gerada, a aplicação possuirá apenas um controlador, chamado **site**
- No Yii, o arquivo de um controladore deve iniciar com seu nome, seguido da string **Controller**

```
david@coiote: /home/www/icomp/yii2-app-basic/controllers
david@coiote:/home/www/icomp/yii2-app-basic/controllers$ ls
SiteController.php
david@coiote:/home/www/icomp/yii2-app-basic/controllers$ █
```

# Controller X View

- O arquivo **SiteController** define uma classe chamada **SiteController**

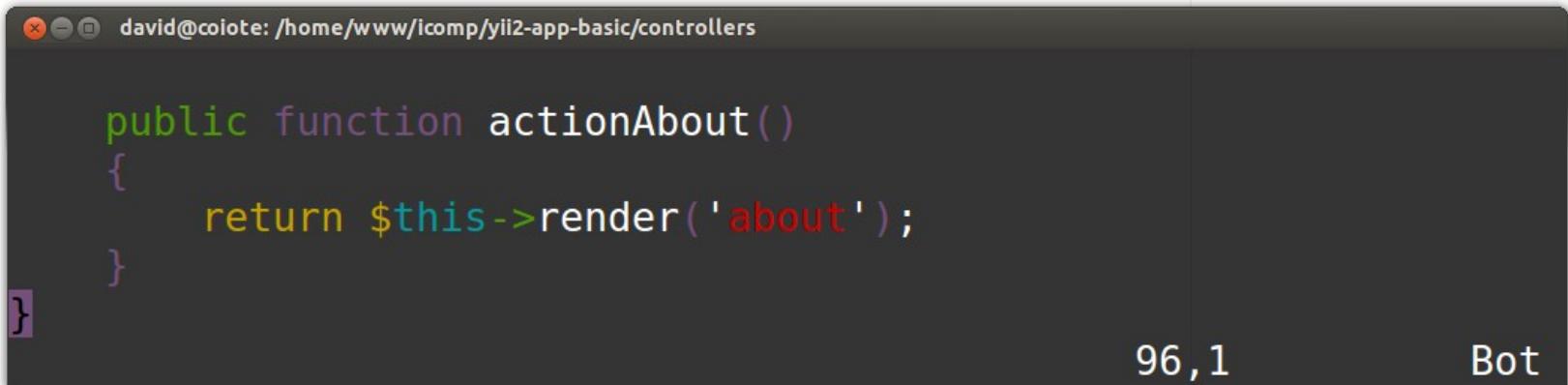
```
david@colote: /home/www/icomp/yii2-app-basic/controllers

class SiteController extends Controller
{
    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'only' => ['logout'],
                'rules' => [
                    [
                        'actions' => ['logout'],
                        'allow' => true,
                        'roles' => ['@'],
                    ],
                ],
            ],
            'verbs' => [

```

# Controller X View

- A classe `SiteController` possui um método chamado `actionAbout()`
- Esse método contém o código que é carregado quando acessamos `index.php?r=site/about`



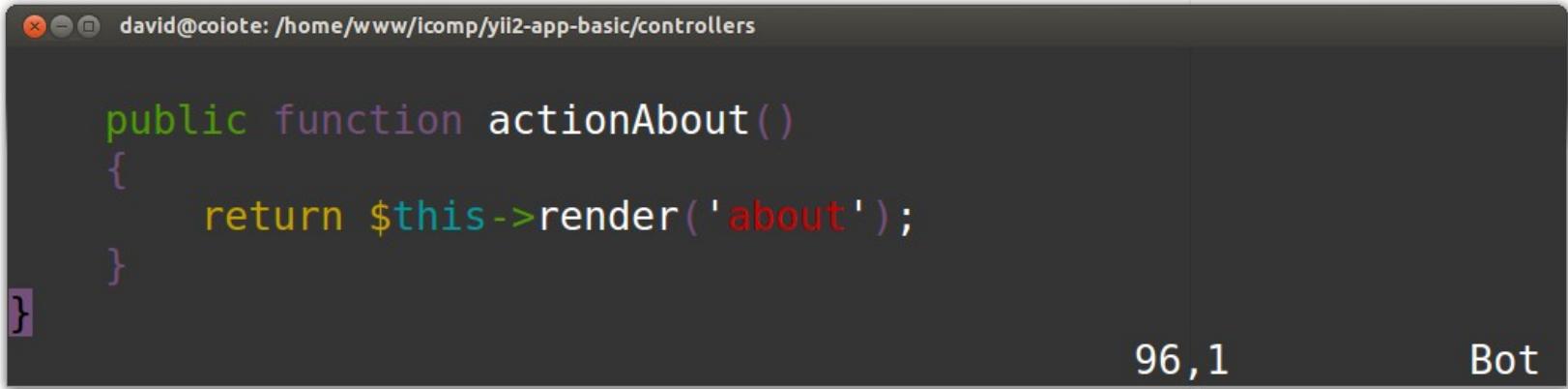
```
david@coiote: /home/www/icomp/yii2-app-basic/controllers

public function actionAbout()
{
    return $this->render('about');
}
```

96,1      Bot

# Controller X View

- Observe que a página **about** não precisa de nenhum processamento, e nem acesso a dados
  - Isso porque ela é uma página de texto simples
- Desta forma, a única tarefa do controlador é indicar a view que contém o texto da página
- Para fazer isso, o controlador chama o método **render**, passando o nome da view como parâmetro



```
david@coiote: /home/www/icomp/yii2-app-basic/controllers

public function actionAbout()
{
    return $this->render('about');
}
```

# Controller X View

- As views são armazenadas no diretório `views`, dentro do diretório base da aplicação
- Após sua geração inicial, a aplicação possui 2 diretórios de views: **layouts**, e **site**
- Mais tarde vamos falar sobre layouts, mas por enquanto vamos nos ater às views do diretório site

```
david@coiote: /home/www/icomp/yii2-app-basic/views
david@coiote:/home/www/icomp/yii2-app-basic/views$ ls
layouts  site
david@coiote:/home/www/icomp/yii2-app-basic/views$ █
```

# Controller X View

- O diretório site contém as views de todas as actions do controlador **site**

```
david@coiote:/home/www/icomp/yii2-app-basic/controllers$ cat SiteController.php | grep -v actions | grep action
    public function actionIndex()
    public function actionLogin()
    public function actionLogout()
    public function actionContact()
    public function actionAbout()
david@coiote:/home/www/icomp/yii2-app-basic/controllers$ █
```

```
david@coiote:/home/www/icomp/yii2-app-basic/views/site$ ls
about.php  contact.php  error.php  index.php  login.php
david@coiote:/home/www/icomp/yii2-app-basic/views/site$ █
```

# Controller X View

- O arquivo `site/about.php` contém a view que é carregada quando `SiteController` executa o comando `$this->render('about')`

```
david@coiote: /home/www/icomp/yii2-app-basic/views/site
<?php
use yii\helpers\Html;

/* @var $this yii\web\View */
$this->title = 'About';
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="site-about">
    <h1><?= Html::encode($this->title) ?></h1>

    <p>
        This is the About page. You may modify the following file
        to customize its content:
    </p>

    <code><?= __FILE__ ?></code>
</div>
```

# Controller X View

- Variáveis criadas no controller podem ser enviadas para a view

```
david@coiote: /home/www/icomp/yii2-app-basic/controllers
```

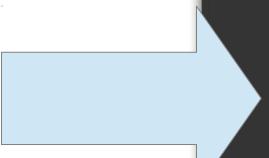
```
public function actionAbout()  
{  
    $teste = "Apenas um teste";  
    return $this->render('about', ['teste' => $teste]);  
}
```

97,1

Bot

# Controller X View

- A variável `$teste`, passada pelo controlador, pode ser acessada diretamente pela view



```
david@coiote: /home/www/icomp/yii2-app-basic/views/site
$this->title = 'About';
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="site-about">
    <h1><?= Html::encode($this->title) ?></h1>

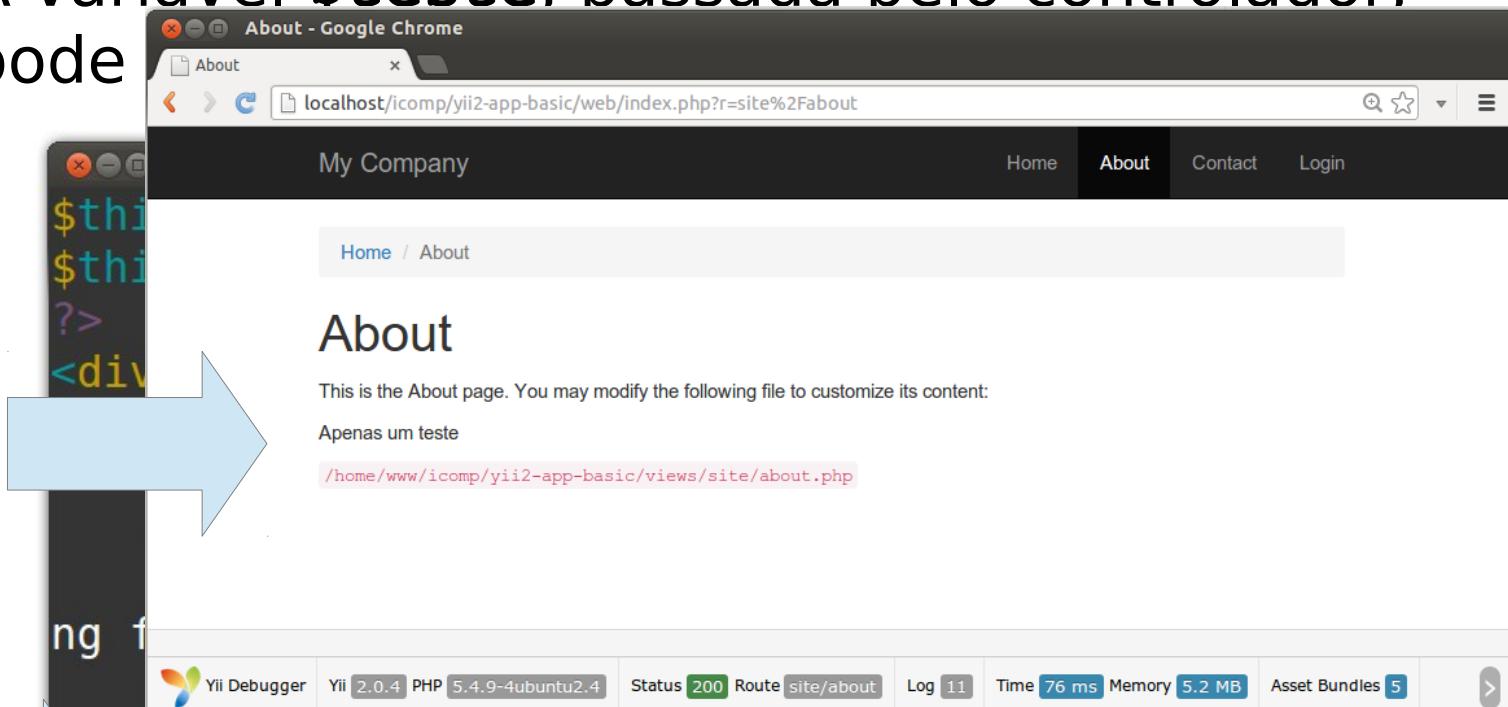
    <p>
        This is the About page. You may modify the following file to customize its content:
    </p>

    <p><?= $teste ?></p>

    <code><?= __FILE__ ?></code>
</div>
```

# Controller X View

- A variável `$teste`, passada pelo controlador, pode

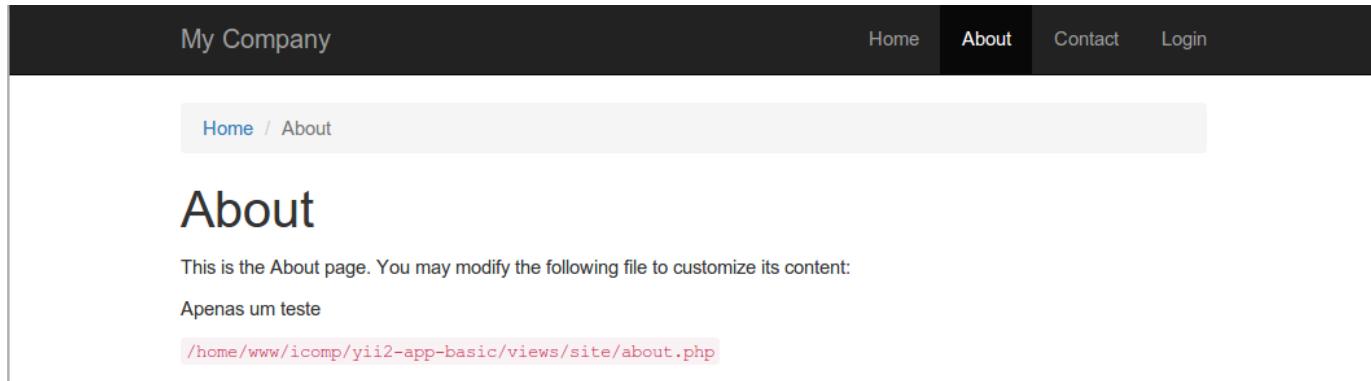


```
<p><?= $teste ?></p>

<code><?= __FILE__ ?></code>
</div>
```

# Exercício

- Mudar a página **About** para que ela exiba uma breve descrição da aplicação a ser desenvolvida
- Usando a função `date()`, crie uma variável no controlador contendo a data atual, e mostre essa data na view da página **About**



# Usando o Debugger

- Uma das principais melhorias do Yii 2 em relação a primeira versão do framework é o **debugger**



- Por padrão, o debugger apresenta:
  - A versão do Yii 2 e do PHP em uso
  - Status da requisição HTTP da página atual
  - Rota da página requisitada
  - Número de mensagens de log geradas pela página
  - Tempo de renderização da página
  - Memória requerida pra gerar a página
  - Número de asset bundles requisitados

# Usando o Debugger

- Podemos clicar em um dos itens do debugger para obter mais detalhes sobre ele

The screenshot shows the Yii Debugger interface running in Google Chrome. The title bar says "Yii Debugger - Google Chrome". The address bar shows the URL "icomp.localhost/index.php?r=debug%2Fdefault%2Fview&tag=5591471913d94&panel=log". The top navigation bar includes the Yii Debugger logo, Yii 2.0.4, PHP 5.4.9-4ubuntu2.4, Status 200, Route site/index, Log 11, Time 44 ms, Memory 5.2 MB, Asset Bundles 5, and a three-dot menu.

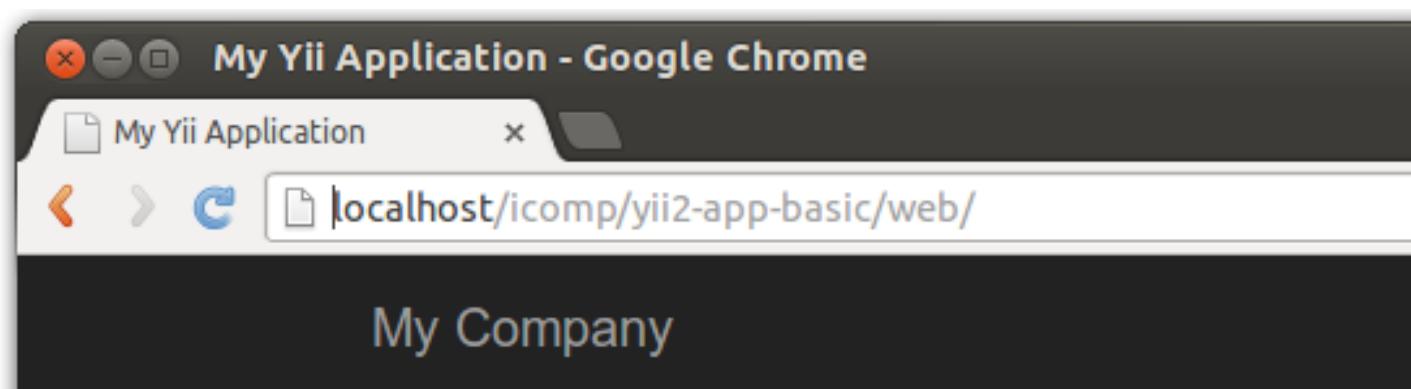
The left sidebar has links for Configuration, Request, Logs (which is selected and highlighted in blue), Profiling, Database, Asset Bundles, and Mail.

The main content area is titled "Log Messages" and displays "Total 11 items." Below this is a table with columns: #, Time, Level, Category, and Message.

#	Time	Level	Category	Message
1	13:24:41.077	trace	yii\base\Application::bootstrap	Bootstrap with yii\log\Dispatcher
2	13:24:41.077	trace	yii\base\Module::getModule	Loading module: debug
3	13:24:41.080	trace	yii\base\Application::bootstrap	Bootstrap with yii\debug\Module::bootstrap()
4	13:24:41.082	trace	yii\base\Module::getModule	Loading module: gii
5	13:24:41.082	trace	yii\base\Application::bootstrap	Bootstrap with yii\gii\Module::bootstrap()
6	13:24:41.083	trace	yii\web\UrlManager::parseRequest	Pretty URL not enabled. Using default URL parsing

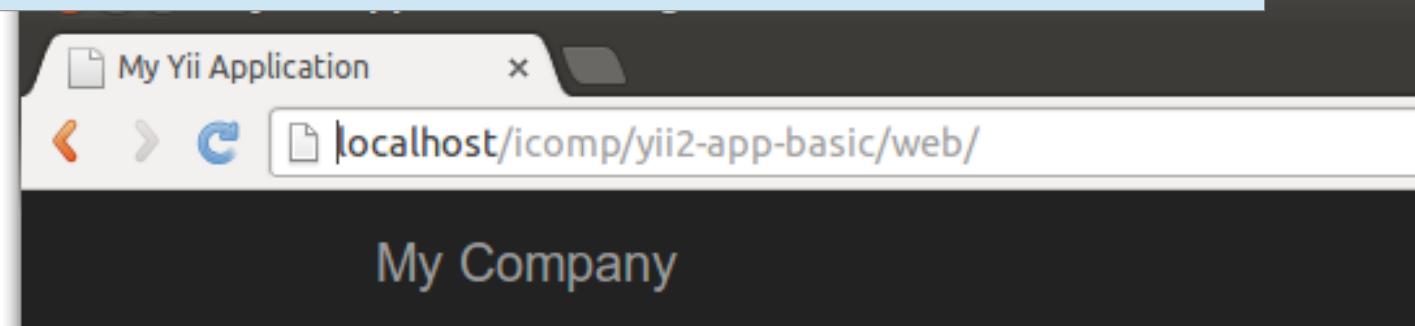
# Configurando o Apache

- No Yii 2, o diretório `web` é criado para ser o diretório root da aplicação a ser desenvolvida
- Desta forma, por padrão, o endereço da aplicação é `http://localhost/icomp/yii2-app-basic/web/`



# Configurando o Apache

- No Yii 2, o diretório **web** é criado para ser o diretório root da aplicação a ser desenvolvida
- Teoricamente, não há problema algum de usar **localhost/icomp/yii2-app-basic/web/** uma URL tão grande, mas o acesso ao site fica mais simples com URLs pequenas



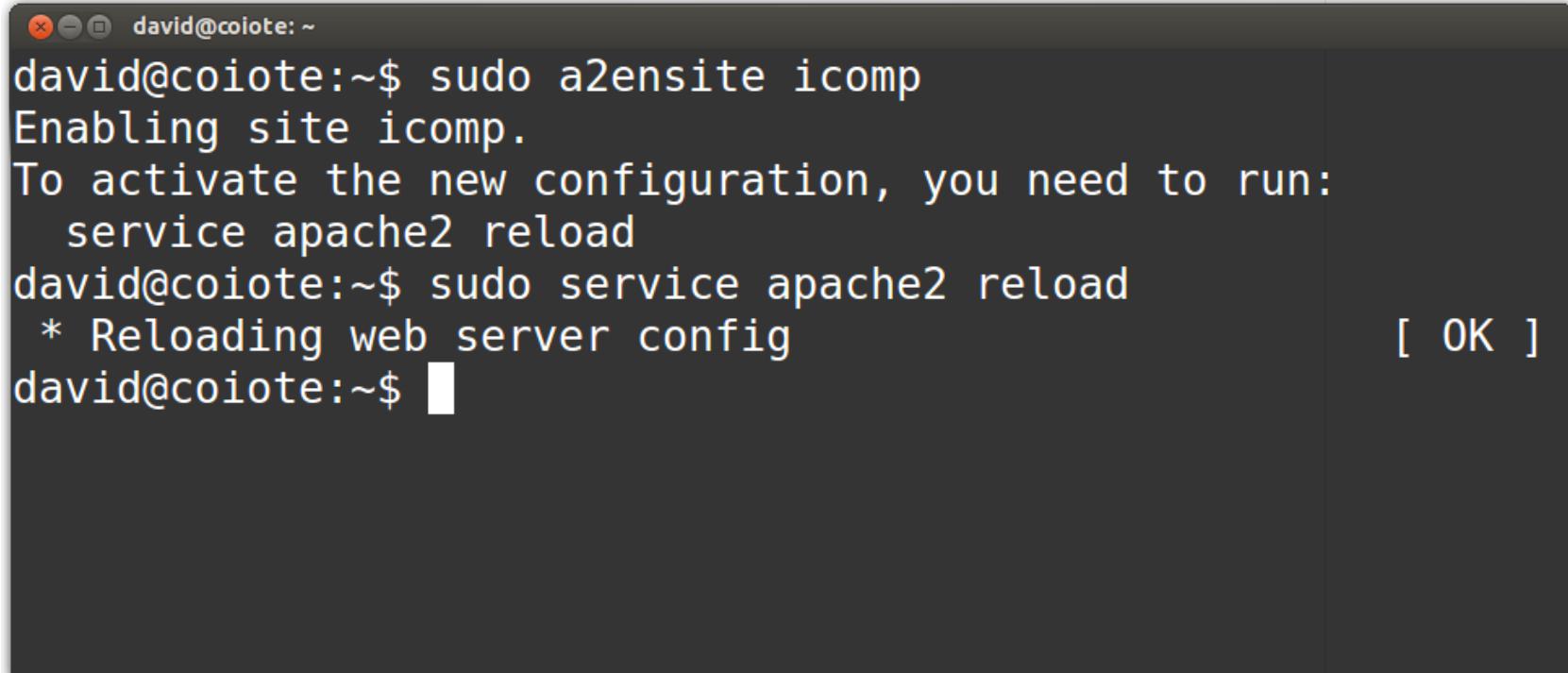
# Configurando o Apache

- Para mudar a URL, basta criar o seguinte arquivo no diretório **sites-available** do apache
  - Diretório **/etc/apache2/sites-available** no Ubuntu

```
david@coiote: /etc/apache2/sites-available
<VirtualHost *:80>
    ServerName icomp.localhost
    DocumentRoot /home/www/icomp/yii2-app-basic/web/
    SetEnv APPLICATION_ENV "development"
    <Directory /home/www/icomp/yii2-app-basic/web/>
        DirectoryIndex index.php
        AllowOverride All
        Order allow,deny
        Allow from all
    </Directory>
</VirtualHost>
```

# Configurando o Apache

- Após a criação do arquivo, use o comando `a2ensite` para ativar a configuração do novo domínio



```
david@coiote:~$ sudo a2ensite icomp
Enabling site icomp.
To activate the new configuration, you need to run:
  service apache2 reload
david@coiote:~$ sudo service apache2 reload
 * Reloading web server config [ OK ]
david@coiote:~$ █
```

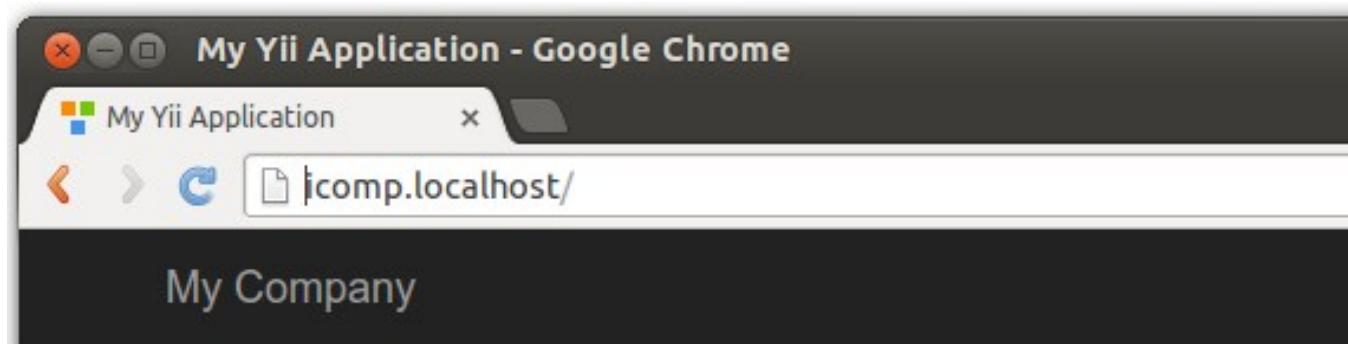
# Configurando o Apache

- Também é importante dizer para o resolvedor de nomes local sobre a existência do novo domínio

```
david@coiote:~$ cat /etc/hosts | grep icomp
127.0.0.1      icomp.localhost
david@coiote:~$ █
```

# Configurando o Apache

- Após a configuração do domínio, podemos acessar a aplicação usando `http://icomp.localhost/`

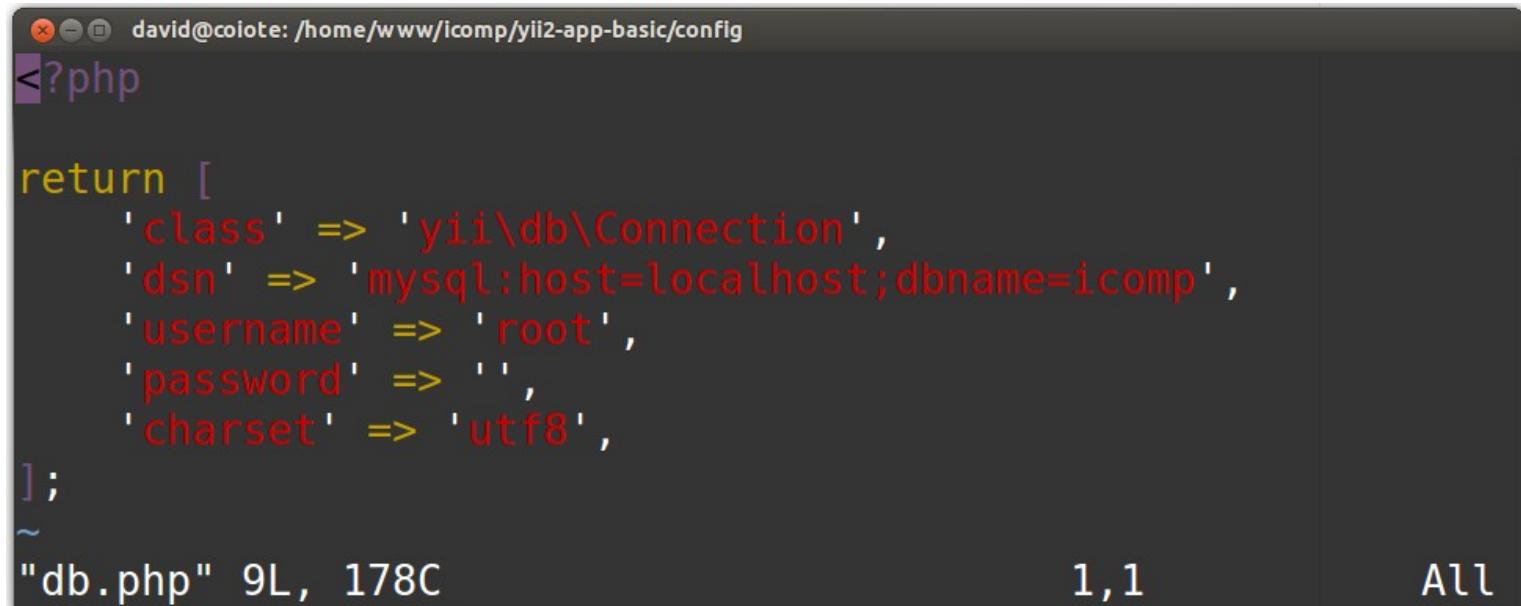


# Configurações Básicas

- No diretório `config`, iremos encontrar os seguintes arquivos de configuração:
  - `console.php`, configura aplicações de console
  - `db.php`, configurações do banco de dados
  - `params.php`, armazena variáveis do site
  - `web.php`, configura aplicações web

# Configuração do Banco

- O acesso ao banco é configurado através do arquivo `web.php`, do diretório `config`
- A string de conexão é a **DSN**, ou **Database Source Name**, que precisa ser mudada de acordo com os dados de acesso ao banco



```
david@coiote: /home/www/icomp/yii2-app-basic/config
<?php

return [
    'class' => 'yii\db\Connection',
    'dsn' => 'mysql:host=localhost;dbname=icomp',
    'username' => 'root',
    'password' => '',
    'charset' => 'utf8',
];
~
"db.php" 9L, 178C
```

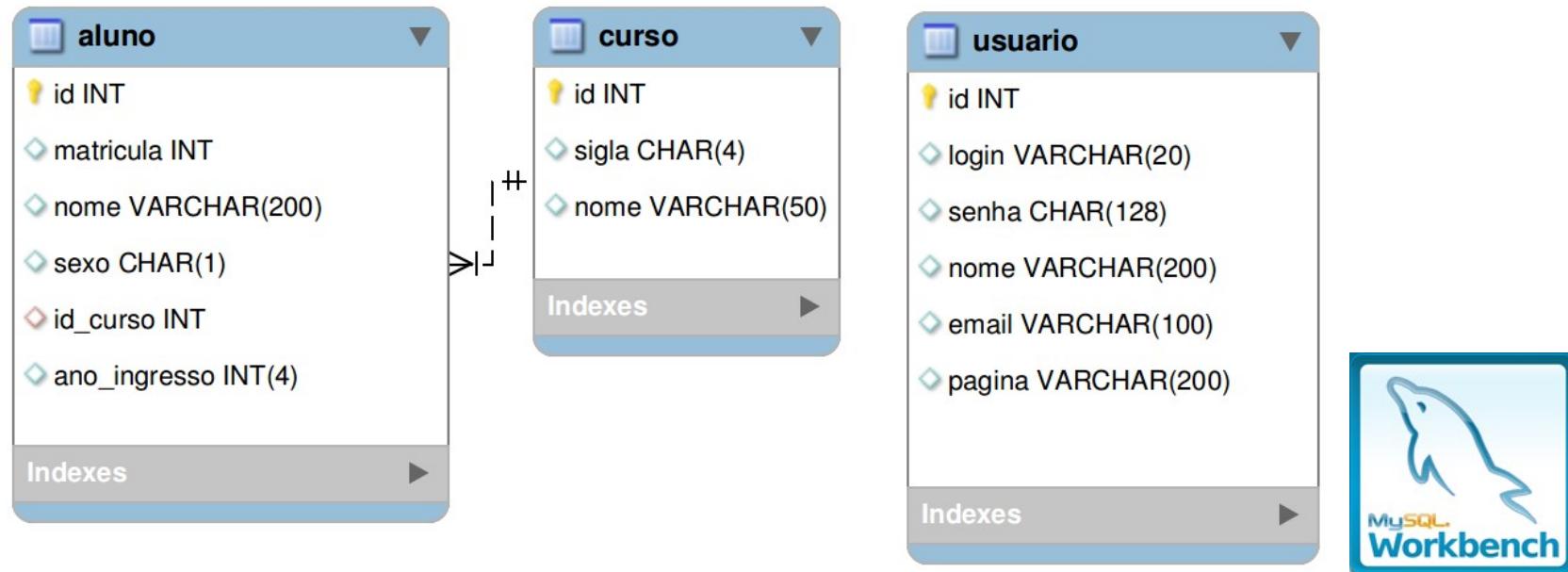
The screenshot shows a terminal window with the command `david@coiote: /home/www/icomp/yii2-app-basic/config`. It displays the contents of the `db.php` file, which defines a database connection configuration. The configuration is an array with the following key-value pairs:

- 'class' => 'yii\db\Connection'
- 'dsn' => 'mysql:host=localhost;dbname=icomp'
- 'username' => 'root'
- 'password' => '' (empty string)
- 'charset' => 'utf8'

The file has 9 lines and a total size of 178 bytes. The bottom right corner of the terminal shows the numbers "1,1" and "All".

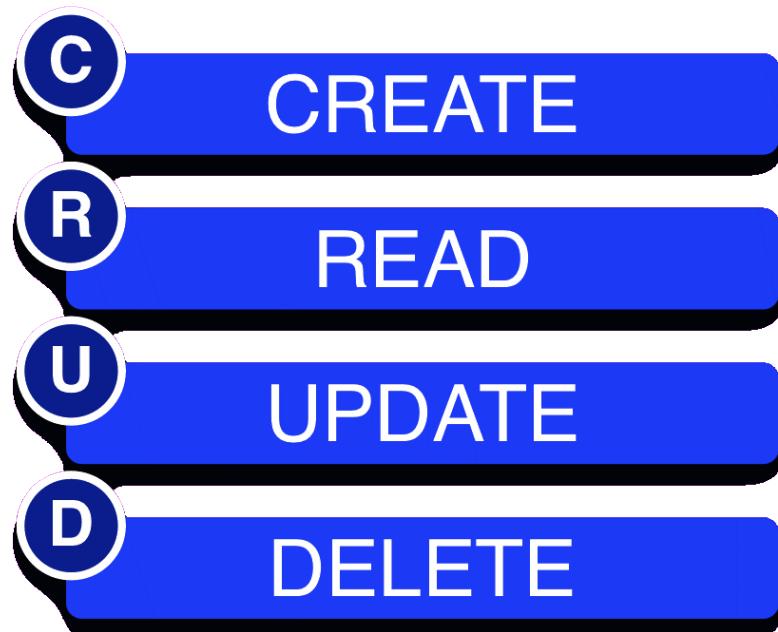
# Exemplo de Aplicação

- Para exemplificar o desenvolvimento de uma aplicação usando o Yii 2, consideremos um sistema que gerencia os dados de todos os alunos do Instituto de Computação
- Através do **MySQL Workbench**, podemos criar o esquema do banco **icomp**, e importá-lo através do PHPMyAdmin



# O CRUD

- Nosso objetivo é criar uma aplicação Web que permita a **adição, edição, apresentação e deleção** de cursos e alunos do Instituto de Computação da UFAM
  - Essas quatro funções são conhecidas como CRUD: Create (i.e., INSERT), Read (i.e., SELECT), Update, and Delete



# Definindo o Banco de Dados

- O banco **icomp** possui três tabelas (curso, aluno e usuario)
- As tabelas **curso** e **aluno** possuem um relacionamento one-to-many entre elas

```
CREATE TABLE IF NOT EXISTS `aluno` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `matricula` int(11) NOT NULL,
    `nome` varchar(200) DEFAULT NULL,
    `sexo` char(1) DEFAULT NULL,
    `id_curso` int(11) DEFAULT NULL,
    `ano_ingresso` int(11) DEFAULT NULL,
    PRIMARY KEY (`id`),
    UNIQUE KEY `matricula` (`matricula`),
    KEY `fk_Alunos_1` (`id_curso`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

# Definindo o Banco de Dados

- O banco **icomp** possui três tabelas (curso, aluno e usuario)
- As tabelas **curso** e **aluno** possuem um relacionamento one-to-many entre elas

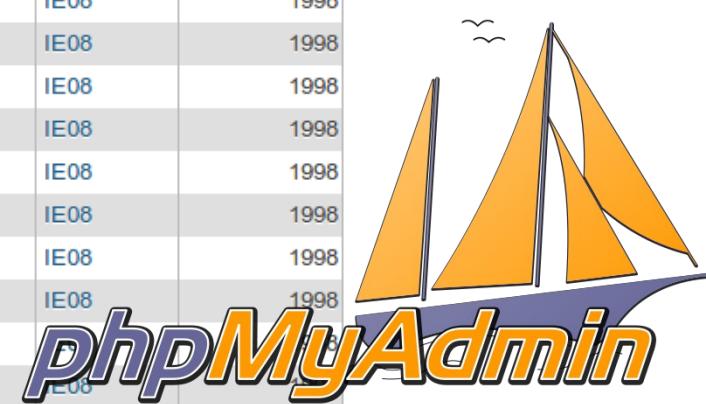
```
CREATE TABLE IF NOT EXISTS `curso` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `sigla` char(4) NOT NULL,
    `nome` varchar(50) DEFAULT NULL,
    PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

# Definindo o Banco de Dados

- O banco **icomp** possui três tabelas (curso, aluno e usuario)
- As tabelas **curso** e **aluno** possuem um relacionamento one-to-many entre elas

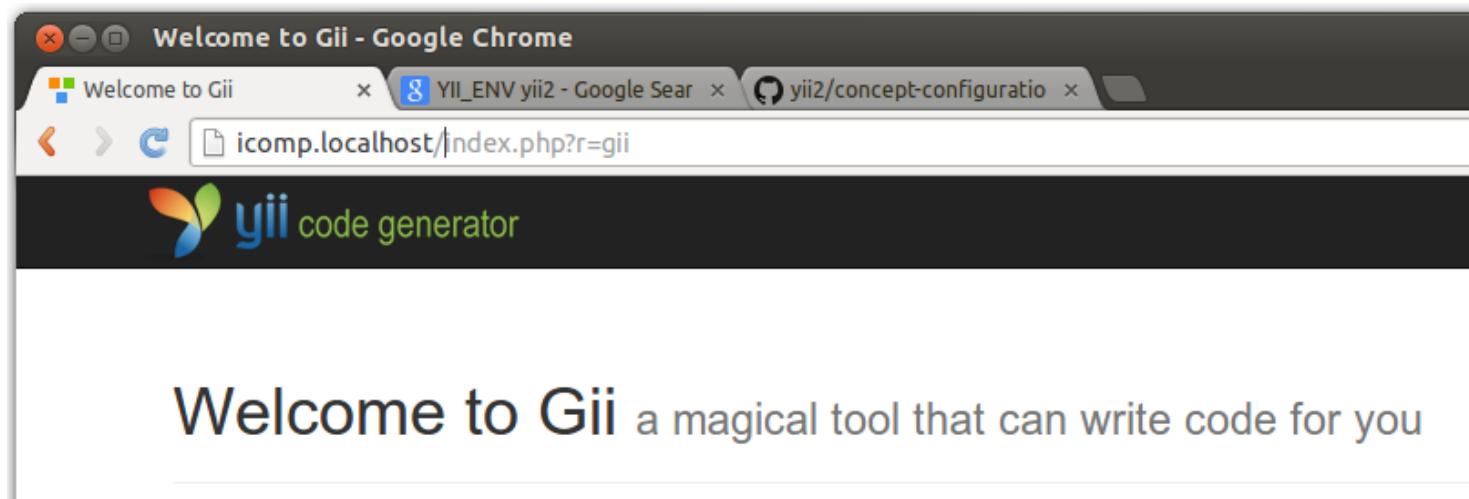
```
CREATE TABLE IF NOT EXISTS `usuario` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `login` varchar(20) NOT NULL,
    `senha` char(128) NOT NULL,
    `nome` varchar(200) NOT NULL,
    `email` varchar(50) NOT NULL,
    `tipo` ENUM('aluno','professor','admin') NOT NULL,
    `data_cadastro` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (`id`),
    UNIQUE INDEX login_UNIQUE (username ASC),
    UNIQUE INDEX email_UNIQUE (email ASC)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

	<input type="checkbox"/>	<input type="checkbox"/>	matricula	nome	sexo	sigla_curso	ano_Ingresso
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	18910303 ALESSANDRO DE MEDEIROS	M	IE08	1989
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19110321 MACSANDRO ROCHA PARENTE	M	IE08	1991
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19110327 PLACIDO VICENTE LAVOR MITOSO	M	IE08	1991
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19210498 RUTH DE ALMEIDA CUNHA	M	IE08	1998
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19310384 EDUARDO RAMOS CORREA	M	IE08	1993
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19510007 EUDESIO COELHO MACIEL	M	IE08	1995
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19510397 LUCIANA DE SOUZA AMPARO MONTREZOR	F	IE08	2001
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19610321 ELIANE BARBOSA FROTA	F	IE08	1996
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19610323 ERBETT HINTON RIBEIRO OLIVEIRA	M	IE08	1996
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19610329 JOSE LUIS CORREIA MARINHO	M	IE08	1996
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19610343 PAULO FERREIRA TEIXEIRA JUNIOR	M	IE08	1996
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19610349 WENDELL DOS SANTOS MOREIRA	M	IE08	1996
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19710290 ABENILSON FERREIRA DA SILVA	M	IE08	1997
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19710295 CHRISTIAN OTERO DA SILVA	M	IE08	1997
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19710312 NELSON BARBOSA DOS REIS JUNIOR	M	IE08	1997
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19710318 SERGIO RICARDO DE OCAVALCANTE	M	IE08	1997
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19710473 DANIEL DE OLIVEIRA CHAGAS	M	IE08	1998
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19710477 LUZILENE ANDRADE APOLINARIO	F	IE08	1999
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19710485 MONICA RENATA DE O DA COSTA M SILVA	F	IE08	2000
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19810306 JOAO PAULO DE ARAUJO ROLAND	M	IE08	1998
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19810310 ROBERTO DA COSTA ROCHA	M	IE08	1998
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19810318 ALEXANDRE AMORIM DE SOUZA CRUZ	M	IE08	1998
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19810320 LEONARDO MONTEIRO MARANHAO RODRIGUES	M	IE08	1998
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19810328 ALEXANDRE FREDERICO COSTA BASTOS	M	IE08	1998
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19810331 JOSIMAR MOTA MICHLIES	M	IE08	1998
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19810336 WESLEY SILVA GAMA	M	IE08	1998
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19810337 SERGIO FREITAS DE MORAES	M	IE08	1998
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19810338 LUCIANO DA COSTA PINTO	M	IE08	1998
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19810383 MARCIO GOMES DE LIMA	M	IE08	1998
	<a href="#"></a>	<a href="#"></a>	<a href="#"></a>	19910275 CARLOS EDUARDO DE SOUZA AGUIAR	M	IE08	1998



# Definindo a Aplicação

- Para gerarmos o CRUD dos alunos e cursos, vamos usar uma ferramenta chamada Gii
- Para usar o Gii, basta acessar a aplicação através da URL `index.php?r=gii`



# Definindo a Aplicação

The screenshot shows the 'Welcome to Gii' page of the Yii framework. The browser title bar reads 'Welcome to Gii - Google Chrome'. The address bar shows 'icomp.localhost/index.php?r=gii'. The page header features the 'yii code generator' logo and navigation links for 'Home', 'Help', and 'Application'. The main content area has a heading 'Welcome to Gii a magical tool that can write code for you'. Below it, there's a section titled 'Start the fun with the following code generators:' followed by six generator descriptions, each with a 'Start »' button.

**Welcome to Gii** a magical tool that can write code for you

Start the fun with the following code generators:

**Model Generator**  
This generator generates an ActiveRecord class for the specified database table.  
[Start »](#)

**CRUD Generator**  
This generator generates a controller and views that implement CRUD (Create, Read, Update, Delete) operations for the specified data model.  
[Start »](#)

**Controller Generator**  
This generator helps you to quickly generate a new controller class with one or several controller actions and their corresponding views.  
[Start »](#)

**Form Generator**  
This generator generates a view script file that displays a form to collect input for the specified model class.  
[Start »](#)

**Module Generator**  
This generator helps you to generate the skeleton code needed by a Yii module.  
[Start »](#)

**Extension Generator**  
This generator helps you to generate the files needed by a Yii extension.  
[Start »](#)

Yii Debugger    Yii 2.0.4 PHP 5.4.9-4ubuntu2.4    Status 200 Route gii/default/index    Log 11    Time 52 ms Memory 5.6 MB    Asset Bundles 6

# Gerando os Modelos

- O propósito do Gii é gerar modelos, views, e controladores
- P
- U

The screenshot shows a web browser window titled "Model Generator - Google Chrome". The address bar displays "localhost / localhost / icor" and the URL "icomp.localhost/index.php?r=gii%2Fdefault%2Fview&id=model". The page itself is titled "Model Generator" and contains the following content:

This generator generates an ActiveRecord class for the specified database table.

**Table Name**:

**Model Class**:

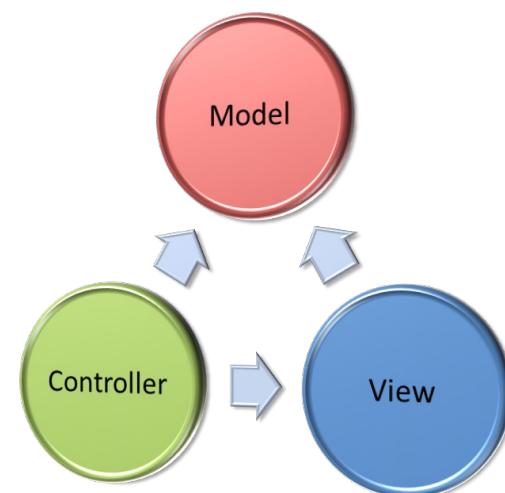
**Namespace**:  (highlighted in yellow)

**Base Class**:  (highlighted in yellow)

The left sidebar lists other generators: Model Generator, CRUD Generator, Controller Generator, Form Generator, Module Generator, and Extension Generator.

# Arquitetura MVC

- O MVC separa uma aplicação em três partes distintas:
  - **Modelo (M)**: que representa os dados, ou o banco de dados, usados pela aplicação
  - **Visões (V)**: que representa a interface com que os usuários interagem com a aplicação
  - **Controlador (C)**: que representa os agentes que respondem às ações dos usuários



# Gerando os Modelos

- O primeiro passo é geramos os modelos das tabelas para as quais queremos gerar o CRUDs
- Para tanto, clique no link **Model Generator** do GII

## Model Generator

This generator generates an ActiveRecord class for the specified database table.

**Table Name**

aluno

**Model Class**

Aluno

**Namespace**

app\models

**Base Class**

yii\db\ActiveRecord

**Database Connection ID**

db

**Use Table Prefix**

**Generate Relations**

**Generate Labels from DB Comments**

# Gerando os Modelos

- O primeiro passo é geramos o modelo da tabela para o qual queremos gerar o CRUD
- Para tanto, clique no link **Model Generator** do GII

## Model Generator

This generator generates an ActiveRecord class for the specified database table.

Table Name

aluno

Model Class

Aluno

Namespace

app\models

Base Class

yii\db\ActiveRecord

Preview

Generate

Click on the above **Generate** button to generate the files selected below:

Create  Unchanged  Overwrite

Code File	Action	
models/Aluno.php	create	<input checked="" type="checkbox"/>

# Gerando os Modelos

- O primeiro passo é geramos o modelo da tabela para o qual queremos gerar o CRUD
- Para tanto, clique no link **Model Generator** do GII

## Model Generator

This generator generates an ActiveRecord class for the specified database table.

**Table Name**

aluno

**Model Class**

Aluno

**Namespace**

app\models

**Base Class**

yii\db\ActiveRecord

The code has been generated successfully.

```
Generating code using template "/home/www/icomp/yii2-app-basic/vendor/yiisoft/yii2-gii
generated models/Aluno.php
done!
```

# Gerando os Modelos

- O primeiro passo é geramos o modelo da tabela

para a tabela que queremos

- Para tanto, no link

**Generat**

## Model Generator

This generator generates an ActiveRecord class for the specified database table.

Table Name

Seguir os mesmos passos para criar os **modelos** de Curso e Usuário

yii\db\ActiveRecord

The code has been generated successfully.

```
Generating code using template "/home/www/icomp/yii2-app-basic/vendor/yiisoft/yii2-gii  
generated models/Aluno.php  
done!
```

# Gerando o CRUD

- Com os modelos criados, o segundo passo é a geração dos CRUDs
- Em nosso exemplo, vamos criar os CRUDs da entidade Aluno
  - O resultado é um conjunto de páginas através das quais se pode **criar, ler, atualizar e deletar** registros de alunos

## CRUD Generator

This generator generates a controller and views that implement CRUD (Create, Read, Update, Delete) operations for the specified data model.

### Model Class

```
app\models\Aluno
```

### Search Model Class

```
app\models\AlunoSearch
```

### Controller Class

```
app\controllers\AlunoController
```

### View Path

```
views\aluno
```

### Base Controller Class

```
yii\web\Controller
```

# Gerando o CRUD

- Com os modelos criados, o segundo passo é a geração dos CRUDs
- Em nosso exemplo, vamos criar os CRUDs da entidade Aluno
  - O resultado é um conjunto de páginas através das quais se pode **criar, ler, atualizar e deletar** registros de alunos

## CRUD Generator

This generator generates a controller and views that implement CRUD (Create, Read, Update, Delete) operations for the specified data model.

### Model Class

app\models\Aluno

[Preview](#) [Generate](#)

Click on the above [Generate](#) button to generate the files selected below:

Create  Unchanged  Overwrite

Code File	Action	
controllers/AlunoController.php	create	<input checked="" type="checkbox"/>
models/AlunoSearch.php	create	<input checked="" type="checkbox"/>
views/aluno/_form.php	create	<input checked="" type="checkbox"/>
views/aluno/_search.php	create	<input checked="" type="checkbox"/>
views/aluno/create.php	create	<input checked="" type="checkbox"/>
views/aluno/index.php	create	<input checked="" type="checkbox"/>
views/aluno/update.php	create	<input checked="" type="checkbox"/>
views/aluno/view.php	create	<input checked="" type="checkbox"/>

# Gerando o CRUD

- Com os modelos criados, o segundo passo é a geração dos CRUDs
- Em nosso exemplo, vamos criar os CRUDs da entidade Aluno
  - O resultado é um conjunto de páginas através das quais se pode **criar, ler, atualizar e deletar** registros de alunos

## CRUD Generator

This generator generates a controller and views that implement CRUD (Create, Read, Update, Delete) operations for the specified data model.

### Model Class

app\models\Aluno

### Search Model Class

app\models\AlunoSearch

Preview

The code has been generated successfully.

```
Generating code using template "/home/www/icomp/yii2-app-basic/vendor/yiisoft/yii2-gii
generated controllers/AlunoController.php
generated models/AlunoSearch.php
generated views/aluno/_form.php
generated views/aluno/_search.php
generated views/aluno/create.php
generated views/aluno/index.php
generated views/aluno/update.php
generated views/aluno/view.php
done!
```

# Gerando o CRUD

- Com os modelos criados, o segundo passo é gerar o CRUD

## CRUD Generator

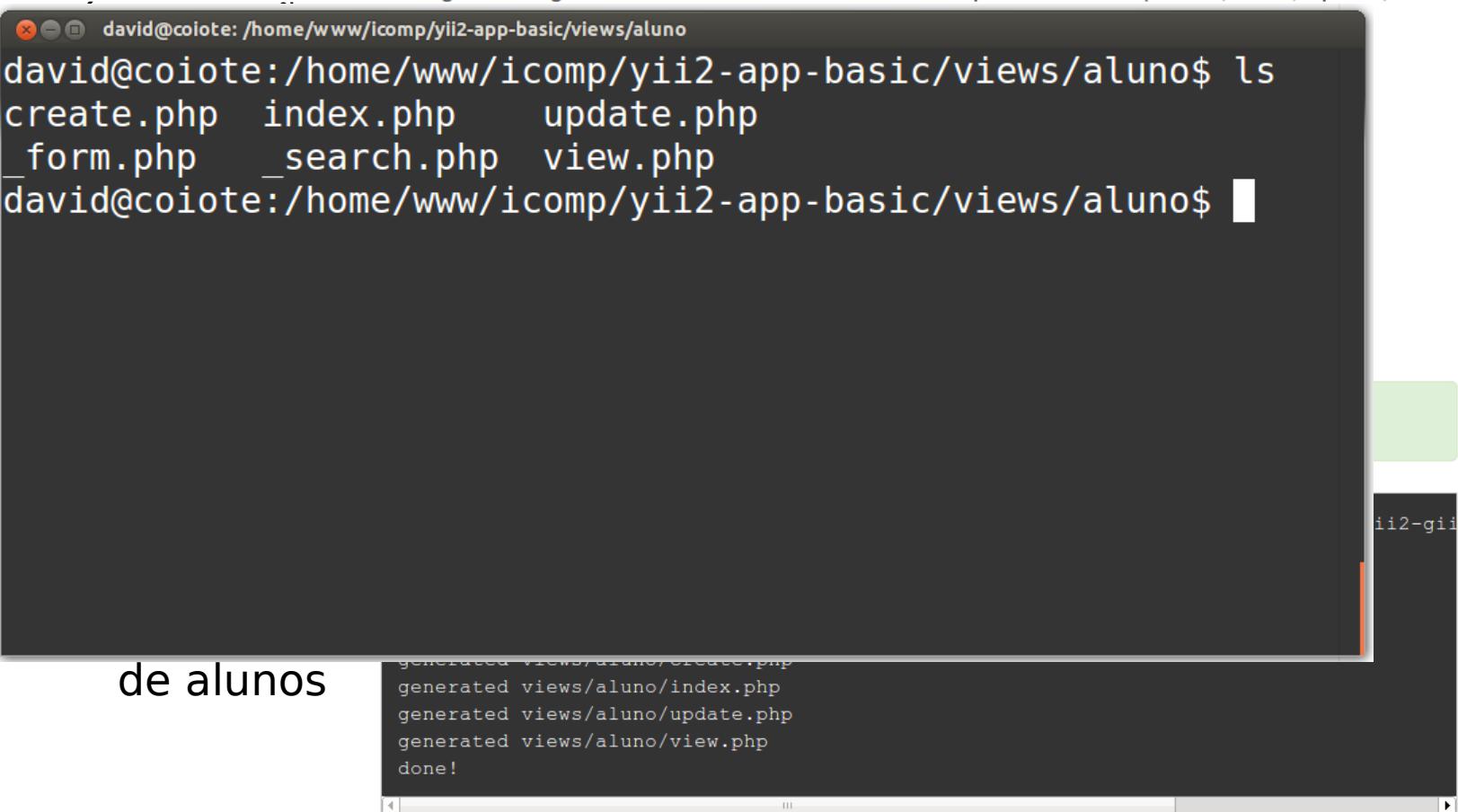
This generator generates a controller and views that implement CRUD (Create, Read, Update, Delete)

- Em seguida, é só executar o comando

CRUD

- O resultado é o conjunto de arquivos que separamos

de gerar o CRUD de alunos



```
david@coiote:/home/www/icomp/yii2-app-basic/views/aluno$ ls
create.php    index.php    update.php
_form.php     _search.php   view.php
david@coiote:/home/www/icomp/yii2-app-basic/views/aluno$ ii2-gi
generated views/aluno/create.php
generated views/aluno/index.php
generated views/aluno/update.php
generated views/aluno/view.php
done!
```

# Alunos

[Create Aluno](#)

Showing 1-20 of 1586 items.

#	ID	Matricula	Nome	Sexo	Id Curso	
1	1	18910303	ALESSANDRO DE MEDEIROS	M	1	 
2	2	19110321	MACSANDRO ROCHA PARENTE	M	1	 
3	3	19110327	PLACIDO VICENTE LAVOR MITOSO	M	1	 
4	4	19210498	RUTH DE ALMEIDA CUNHA	M	1	 
5	5	19310384	EDUARDO RAMOS CORREA	M	1	 
6	6	19510007	EUDESIO COELHO MACIEL	M	1	 
7	7	19510397	LUCIANA DE SOUZA AMPARO MONTREZOR	F	1	 

Alunos

[icomp.localhost/index.php?r=aluno](#)

My Company

Home / Alunos

Alunos

[Create Aluno](#)

Showing 1-20 o

#	ID
1	1
2	2
3	3
4	4
5	5
6	6
7	7

Create Aluno

[icomp.localhost/index.php?r=aluno%2Fcreate](#)

My Company

Home About Contact Login

Home / Alunos / Create Aluno

## Create Aluno

**Matricula****Nome****Sexo****Id Curso****Ano Ingresso**[Create](#)

Yii Debugger

Yii 2.0.4 PHP 5.4.9-4ubuntu2.4

Status 200 Route aluno/create

Log 15

Time 60 ms Memory 6.9 MB

DB 2 2 ms

Asset Bundles 7



Alunos

[icomp.localhost/index.php?r=aluno](#)

My Company

[Home](#) / [Alunos](#)

# Alunos

[Create Aluno](#)

Showing 1-20 o

#	ID
1	1
2	2
3	3
4	4
5	5
6	6
7	7

## Update Aluno: 2 - Google Chrome

Update Aluno: 2

icomp.localhost/index.php?r=aluno%2Fupdate&id=2

### My Company

[Home](#) [About](#) [Contact](#) [Login](#)

[Home](#) / [Alunos](#) / 2 / Update

## Update Aluno: 2

**Matricula**

**Nome**

**Sexo**

**Id Curso**

**Ano Ingresso**

**Update**

Yii Debugger | Yii 2.0.4 PHP 5.4.9-4ubuntu2.4 | Status 200 Route aluno/update | Log 16 | Time 80 ms Memory 7.8 MB | DB 3 2 ms Asset Bundles 7

Alunos

icomp.localhost/index.php?r=aluno

My Company

Home / Alunos

# Alunos

[Create Aluno](#)

Showing 1-20 o

#	ID
1	1
2	2
3	3
4	4
5	5
6	6
7	7

1

[Update](#)[Delete](#)

ID	1
Matricula	18910303
Nome	ALESSANDRO DE MEDEIROS
Sexo	M
Id Curso	1
Ano Ingresso	1989



Yii Debugger | Yii 2.0.4 PHP 5.4.9-4ubuntu2.4 | Status 200 Route aluno/view Log 15 Time 76 ms Memory 7.5 MB DB 3 1 ms Asset Bundles 5

# Gerando o CRUD

- Com os modelos criados, o segundo passo é a geração dos CRUDs.
- Em nosso exemplo, vamos gerar os CRUDs da entidade Aluno.
- O resultado é um conjunto de páginas através das quais se pode **criar, ler, atualizar e deletar** registros de alunos.

## CRUD Generator

This generator generates a controller and views that implement CRUD (Create, Read, Update, Delete) operations for the specified data model.

Seguir os mesmos passos para criar os **CRUDs** de Curso e Usuário

```
Generating code using template "/home/www/icomp/yii2-app-basic/vendor/yiisoft/yii2-gii"
generated controllers/AlunoController.php
generated models/AlunoSearch.php
generated views/aluno/_form.php
generated views/aluno/_search.php
generated views/aluno/create.php
generated views/aluno/index.php
generated views/aluno/update.php
generated views/aluno/view.php
done!
```

# Gerando o CRUD

- Com os modelos criados, o segundo passo é a geração dos CRUDs.
- Em nosso exemplo, vamos gerar os CRUDs da entidade Aluno.
- O resultado é um conjunto de páginas através das quais se pode **criar, ler, atualizar e deletar** registros de alunos.

## CRUD Generator

This generator generates a controller and views that implement CRUD (Create, Read, Update, Delete) operations for the specified data model.

Após a geração dos CRUDs, desabilitar o Gii em config/web.php

```
Generating code using template "/home/www/icomp/yii2-app-basic/vendor/yiisoft/yii2-gii"
generated controllers/AlunoController.php
generated models/AlunoSearch.php
generated views/aluno/_form.php
generated views/aluno/_search.php
generated views/aluno/create.php
generated views/aluno/index.php
generated views/aluno/update.php
generated views/aluno/view.php
done!
```

# Trabalhando com Modelos

- Os **modelos** representam os dados de uma aplicação
- Cada modelo representa uma tabela do banco de dados
  - Por exemplo, em nossa aplicação criamos três modelos distintos, **Aluno**, **Curso** e **Usuario**
  - O modelo **Aluno** conterá códigos para acesso e manipulação de todas as linhas da tabela **aluno**



# Trabalhando com Modelos

- Por padrão, os modelos do Yii2 são armazenados no diretório **models**
- Cada arquivo define a classe de apenas um modelo, e possui nome de arquivo igual ao nome da classe seguido da extensão **.php**
  - Por exemplo, o modelo da tabela **aluno** é definido em **models/Aluno.php**



# Trabalhando com Modelos

- As classes dos modelos estendem `\yii\db\ActiveRecord`, e possuem a seguinte estrutura geral:

```
class ClassName extends \yii\db\ActiveRecord {  
    // Métodos...  
    public function tableName() { }  
    public function rules() { }  
    public function relations() { }  
    public function attributeLabels() { }  
}
```



# Trabalhando com Modelos

- Os modelos possuem métodos que precisam ser editados durante o desenvolvimento de uma aplicação
- Dentre tais métodos, **rules()** define as regras de validação de dados para cada atributo do modelo
- O método **rules()**, tal como vários outros métodos do Yii, retorna um array de dados

```
public function rules() {  
    return [  
        ['matricula', 'id_curso'], 'required',  
        // Outras regras.  
    ];  
}
```



# Trabalhando com Modelos

- Para uma mesma regra de validação para um atributo, basta usar a seguinte estrutura:

```
['attr', 'regra', [outros parâmetros]]
```

Para aplicar uma mesma regra para múltiplos atributos, basta usar um array como primeiro parâmetro

```
[['attr1', 'attr2'], 'regra', [outros parâmetros]]
```



# Editando os Modelos

- A primeira regra, e talvez a mais óvia, é a indicação de preenchimento obrigatório de alguns campos

```
[ ['matricula', 'id_curso', 'ano_ingresso',  
  'nome', 'sexo'], 'required' ]
```

- Também é possível informar se um valor precisa ser numérico, ou mais especificadamente, inteiro

```
[ ['id_curso', 'ano_ingresso'], 'integer' ]
```



# Editando os Modelos

- É possível restringir os tamanhos mínimo e máximo de uma string

```
[ ['nome'], 'string', 'max' => 200],  
[ 'idade', 'integer', 'min'=>13, 'max'=>100],
```

- Ou um tamanho exato

```
[ 'matricula', 'length' => 8],
```



# Editando os Modelos

- O validador **in** garante que o valor pertence a um dado intervalo ou lista de valores:

```
[ 'boi_favorido', 'in', 'range' =>  
[ 'garantido', 'caprichoso' ] ],  
[ 'rating', 'in', 'range' => range(1,10) ],
```

- **range()** é uma função do PHP que retorna um array contendo uma faixa de valores



# Editando os Modelos

- Os validadores `email` e `url` compararam o valor do campo com expressões regulares próprias para essas sintaxes

```
[ 'email', 'email' ],  
[ 'website', 'url' ],
```

- O validador `match` testa o valor do campo contra uma expressão regular personalizada

```
[ 'senha', 'match', 'pattern'=> '/^ [a-zA-Z0-9_-] {6,20} $/' ],
```



# Editando os Modelos

- Quando o usuário infringe uma das regras da função **rules()**, o Yii 2 mostra uma mensagem de erro

Home / Alunos / Create Aluno

## Create Aluno

**Matricula**

Matricula cannot be blank.

**Nome**

Nome cannot be blank.



# Editando os Modelos

- Para mudar essa mensagem, basta setar o campo **message** da regra, tal como mostrado abaixo:

```
[ ['matricula', 'id_curso', 'ano_ingresso',  
  'nome', 'sexo'], 'required', 'message'=>'Este  
  campo é obrigatório'],
```

**Matricula**

Este campo é obrigatório

**Nome**

Este campo é obrigatório



# Editando os Modelos

- Outra regra importante é a comparação, comumente usada para garantir que o campo de confirmação de senha é igual ao campo de senha

```
['pass', 'compare', 'compareAttribute' =>  
 'passCompare']
```



# Editando os Modelos

- Outros tipos de validadores:

<code>boolean</code>	<code>image</code>	<code>unique</code>
<code>captcha</code>	<code>in</code>	<code>url</code>
<code>date</code>	<code>match</code>	
<code>default</code>	<code>number</code>	
<code>double</code>	<code>required</code>	
<code>email</code>	<code>safe</code>	
<code>exist</code>	<code>string</code>	
<code>file</code>	<code>trim</code>	



# Editando os Modelos

- Outros tipos de validadores:

`boolean`

`image`

`unique`

`captcha`

`in`

`url`

`date`

`default`

**Exercício:** editar os métodos `rules()` dos modelos de seu projeto de acordo com as regras mostradas.

`email`

`exist`

`file`

`trim`



# Editando os Modelos

- Um método mais trivial, embora ainda assim importante, é **attributeLabels()**
- Este método retorna um array associativo contendo os labels dos atributos do modelo, para serem usados em formulários

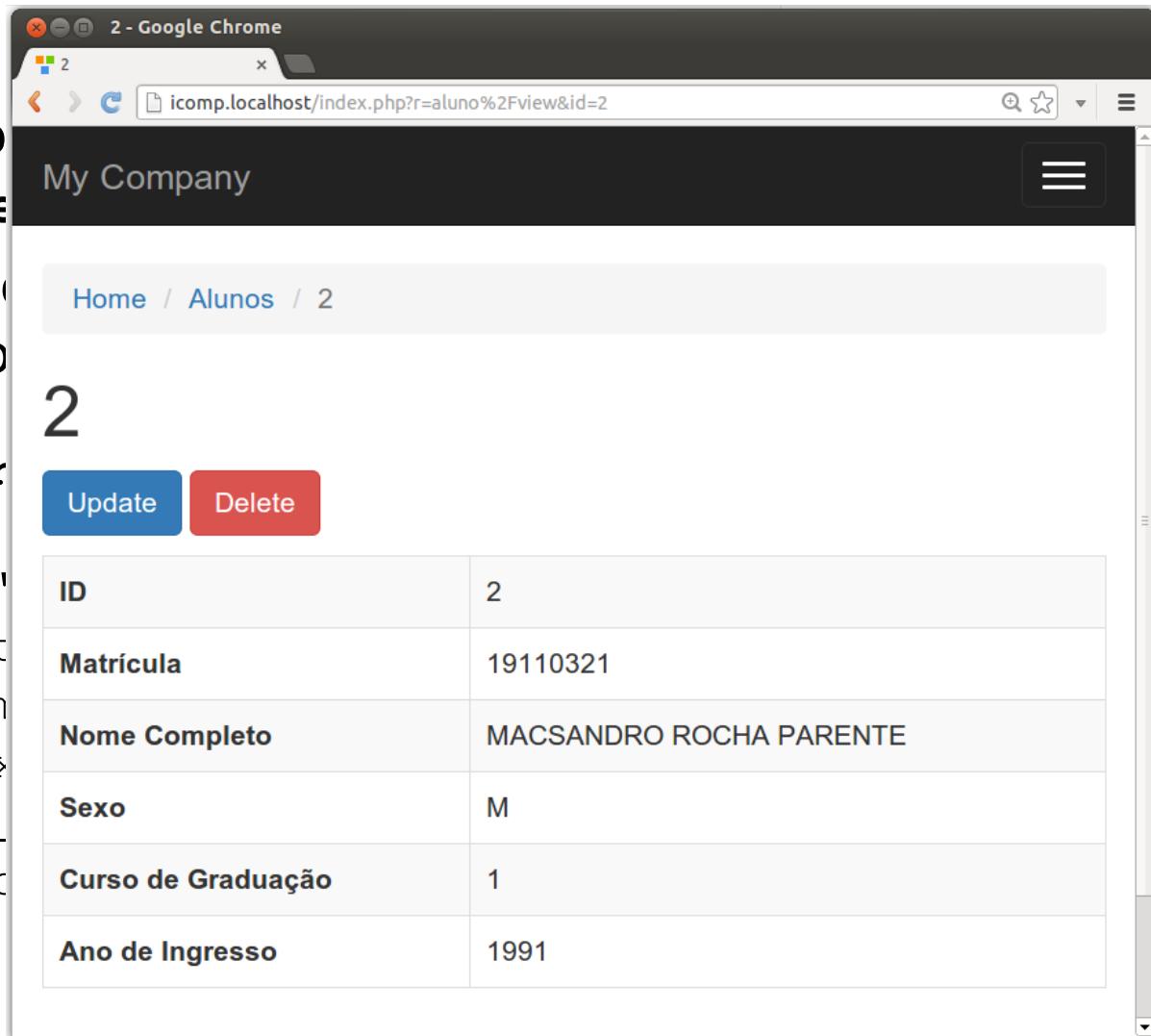
```
public function attributeLabels() {  
    return [  
        'id' => 'ID',  
        'matricula' => 'Matrícula',  
        'nome' => 'Nome Completo',  
        'sexo' => 'Sexo',  
        'id_curso' => 'Curso de Graduação',  
        'ano_ingresso' => 'Ano de Ingresso',  
    ];  
}
```



# Editando os Modelos

- Um método para editar os **atributos**
- Este método deve ser responsável por todos os atributos

```
public function edit($id) {
    return [
        'id' => $id,
        'matricula' => $this->getMatricula(),
        'nome' => $this->getNomeCompleto(),
        'sexo' => $this->getSexo(),
        'id_curso' => $this->getIdCurso(),
        'ano_ingresso' => $this->getAnoIngresso()
    ];
}
```



The screenshot shows a Google Chrome window with the URL `icomp.localhost/index.php?r=aluno%2Fview&id=2`. The page title is "My Company". The breadcrumb navigation shows "Home / Alunos / 2". Below the navigation, there is a large number "2". Two buttons are visible: "Update" (blue) and "Delete" (red). A table displays the following student information:

ID	2
Matrícula	19110321
Nome Completo	MACSANDRO ROCHA PARENTE
Sexo	M
Curso de Graduação	1
Ano de Ingresso	1991



# Editando os Modelos

- Um método para editar os **atributos**
- Este método é responsável por todos os atributos

```
public  
re
```

**Exercício:** editar os labels de seus modelos de acordo com o desejado.

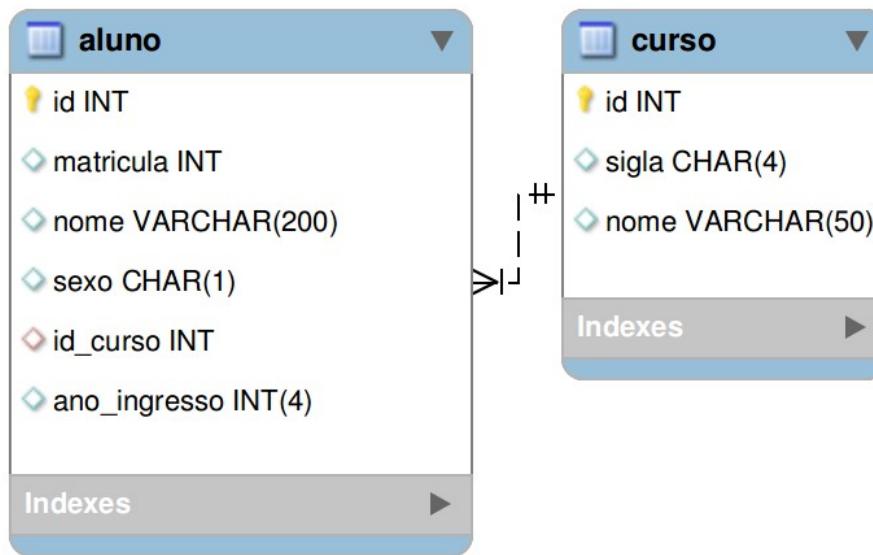
```
'ses  
'id_  
'anc  
];  
}
```

<b>Sexo</b>	M
<b>Curso de Graduação</b>	1
<b>Ano de Ingresso</b>	1991



# Relações entre Modelos

- Outro importante aspecto é a relação entre modelos: como um modelo está relacionado com outro no banco de dados
- Relacionamentos entre modelos são cruciais, pois frequentemente você precisará recuperar dados relacionados
  - Por exemplo, para recuperar o nome do curso (tabela **Curso**) de um aluno (tabela **Aluno**)



# Relações entre Modelos

- No Yii 2, cada relacionamento de um modelo é definido através de um método independente
- Os dois códigos abaixo foram gerados pelo Gii, sendo o primeiro do modelo Aluno, e o segundo do modelo Curso

```
public function getIdCurso() {
    return $this->hasOne(
        Curso::className(), ['id' => 'id_curso']
    );
}

public function getAlunos() {
    return $this->hasMany(
        Aluno::className(), ['id_curso' => 'id']
    );
}
```



# Editando os Modelos

- Outros métodos comumente usados em modelos do Yii:
  - **beforeSave()** , cujo código é executando antes de um objeto da classe do modelo ser salvo
  - **afterSave()** , cujo código é executado depois de um objeto ser salvo
  - **beforeValidate()** , cujo código é executado antes da validação dos dados



# Os Controladores

- Os **Controladores** são responsáveis por atender as requisições dos usuários
- Considerando a entidade Aluno, um usuário pode fazer as seguintes requisições:
  - **Create (C)**: Inserir um novo aluno no banco;
  - **Read (R)**: Visualizar os dados de um ou mais alunos;
  - **Update (U)**: Atualizar os dados de um aluno;
  - **Delete (D)**: Remover um aluno existente;
- Desta forma, o **Controlador** da entidade Aluno deverá conter o código para execução de cada uma destas ações



# Os Controladores

- No padrão de desenvolvimento MVC, as operações de um controlador são chamadas de **Actions**
  - Por exemplo, no controlador da entidade Aluno, existe uma action para cada operação do CRUD de Aluno
- No Yii 2, os controladores ficam armazenados no diretório **controllers**
- Além disso, todos os controladores devem possuir o sufixo **Controller** em seu nome
  - Por exemplo, o controlador da entidade Aluno é **controllers/AlunoController.php**



# Os Controladores

- No Yii 2, um controlador é definido como uma classe que estende a classe **Controller**
- Acessando o controlador AlunoController, percebemos a existência das seguintes funções:
  - `public function behaviors()`
  - `public function actionView($id)`
  - `public function actionCreate()`
  - `public function actionUpdate($id)`
  - `public function actionDelete($id)`
  - `public function actionIndex()`
  - `public function actionAdmin()`
  - `public function findModel($id)`



# As Visões

- As visões contêm o código HTML, CSS e JavaScript das páginas, e refletem aquilo que é visto pelo usuário durante sua interação com a aplicação
- O diretório `views` possui um subdiretório para cada controlador da aplicação, onde serão armazenados as views desse controlador
  - Por exemplo, as views do controlador aluno estão armazenadas no diretório `views/aluno`



# As Visões

- Acessando o diretório `views/aluno`, vemos os seguintes arquivos (entre outros):
  - `admin.php`,
  - `create.php`,
  - `index.php`,
  - `update.php`, e
  - `view.php`
- Isto é, o diretório possui uma view para cada action do controlador Aluno (com exceção de delete)



# As Visões

- Considere a action de Aluno abaixo, que é responsável por mostrar as informações de um dado aluno

```
public function actionView($id) {  
    return $this->render('view', [  
        'model' => $this->findModel($id),  
    ]);  
}
```

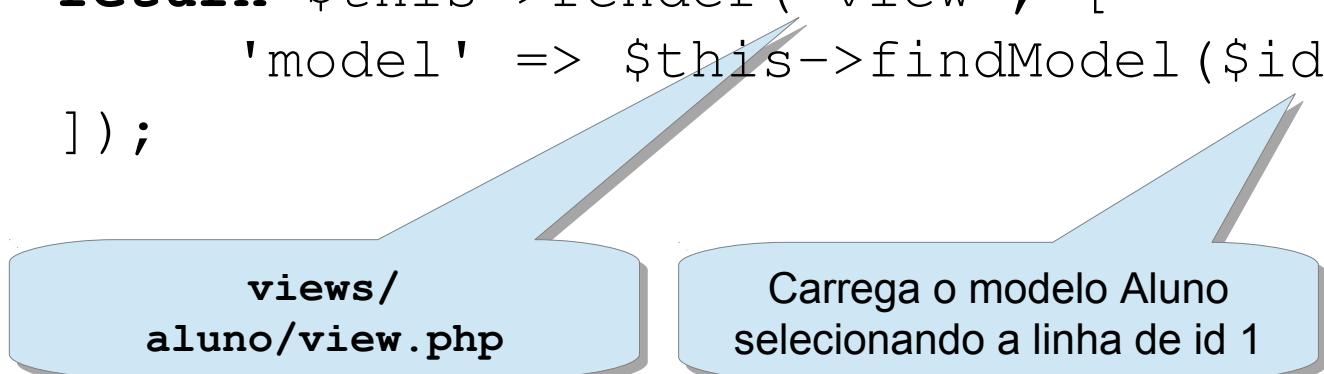
- Conforme já mostramos, dentro da action existe uma chamada à view que contém o código HTML responsável por apresentar o aluno desejado ao usuário



# As Visões

- Digamos que estamos visualizando a página  
`icomp.localhost/index.php?r=aluno/view&id=1`
- Observe que esta página mostra os dados do empregado cujo ID (chave primária) é igual a 1
- Para carregar esta página, o Yii vai executar o método `actionView()` do controlador **AlunoController**

```
public function actionView($id) {  
    return $this->render('view', [  
        'model' => $this->findModel($id),  
    ]);  
}
```



The diagram illustrates the flow of execution. A blue arrow points from the code block above to a light blue rounded rectangle labeled "views/aluno/view.php". Another blue arrow points from this rectangle to a larger light blue rounded rectangle below it, which contains the text "Carrega o modelo Aluno selecionando a linha de id 1".



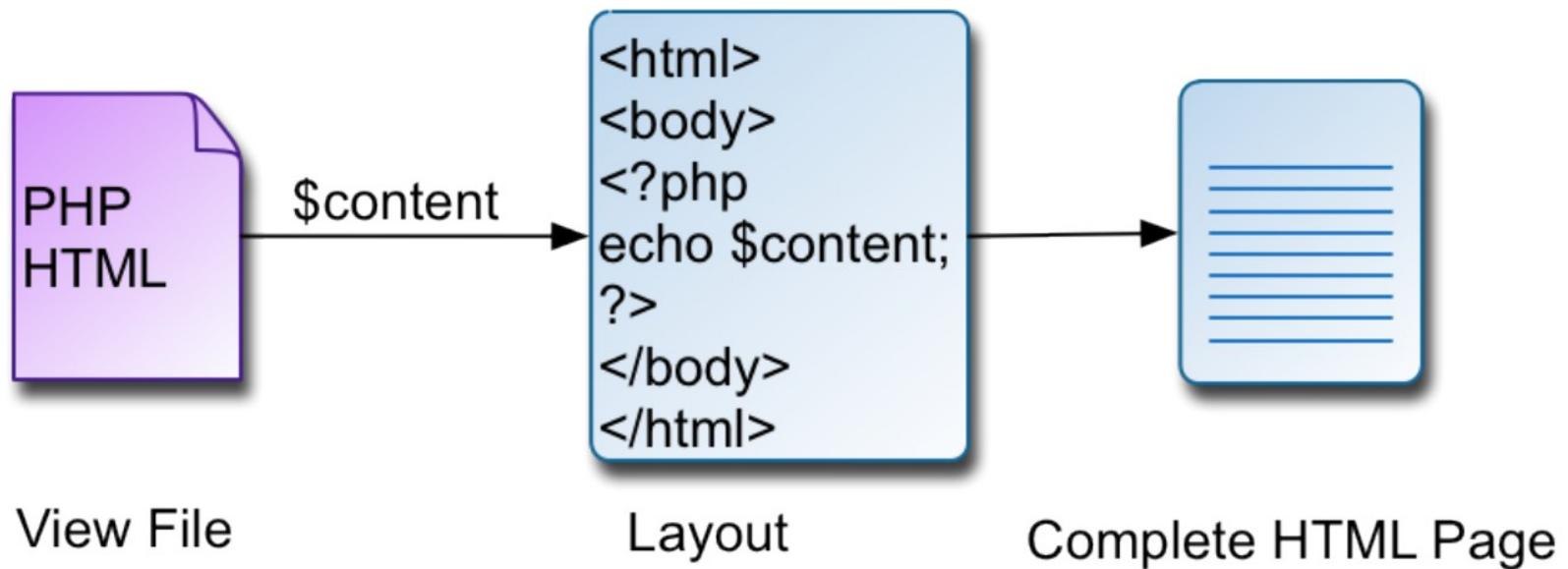
# Layout X Views

- O layout constitui todo o conteúdo HTML que é comum a todas as páginas, isto é, aquilo que não é específico de uma única página (`views`)
- Por padrão, o principal arquivo de layout é `views/layouts/main.php`
- Ao editarmos esse arquivo, verificamos que ele possui a string DOCTYPE e as tags `<html>` e `</html>`
- No meio deste arquivo, vemos a seguinte linha:
  - `<?= $content; ?>`
- A variável `$content` possui o conteúdo específico de cada página, isto é, o conteúdo de uma view



# Layout X Views

- A variável **\$content** possui o conteúdo específico de cada página, isto é, o conteúdo de uma view



# Layout X Views

- Por exemplo, considere que a página `aluno/create` seja requisitada por um dado usuário
  - Para essa requisição, a action `actionCreate()` do controlador `AlunoController` será executada
- A view deste método é `views/aluno/create.php`
- O conteúdo dessa view será carregada dentro do conteúdo do layout, através do comando `<?= $content; ?>`



# Layout X Views

Layout

Create Aluno - Google Chrome

Create Aluno

icomp.localhost/index.php?r=aluno%2Fcreate

My Company

Home About Contact Login

Home / Alunos / Create Aluno

## Create Aluno

Matrícula

Nome Completo

Sexo

Curso de Graduação

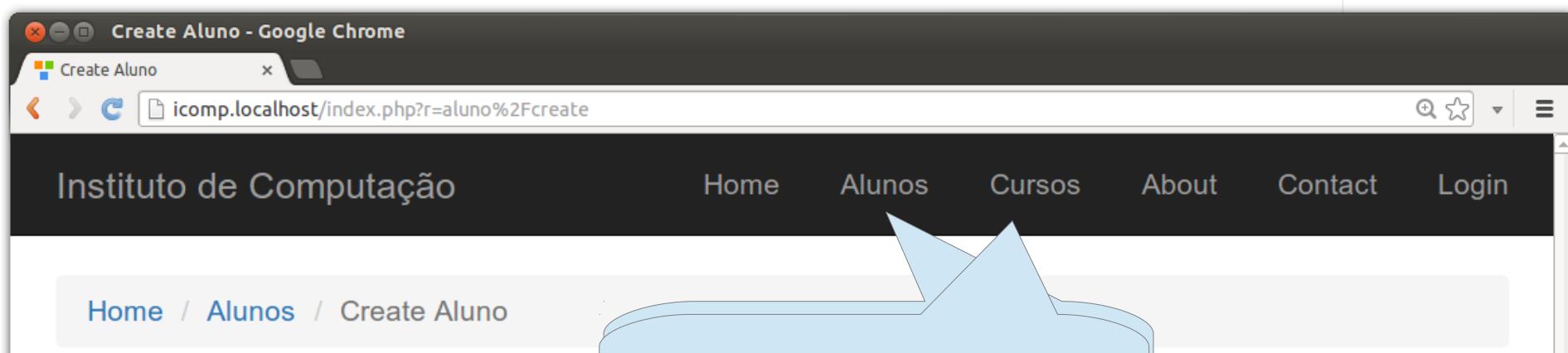
Ano de Ingresso

view aluno/create



# Exercício: Editando o Menu

- Acesse o principal arquivo de layout de sua aplicação, e mude o nome da aplicação e os itens do menu
- Isso é feito editando as configurações do widget **NavBar**



# Editando as Visões

- É possível definir diferentes layouts para diferentes páginas da aplicação, mudando o valor do parâmetro layout dentro da action que se deseja mudar o layout

```
public function actionView($id) {  
    $this->layout = 'main';  
    $this->render('view', array(  
        'model'=>$this->loadModel($id),  
    ));  
}
```

- Também é possível mudar o layout de todas as actions de um controlador de uma única vez:

```
class AlunoController extends Controller {  
    public $layout='//layouts/column2';  
    ...
```



# Editando as Visões

- Para cada controlador criado através do Gii, iremos encontrar os seguintes arquivos de visões:
    - **\_form.php**
    - **\_search.php**
    - **\_view.php**
    - **admin.php**
    - **create.php**
    - **index.php**
    - **update.php**
    - **view.php**
- Contém o formulário incluído pelas visões create e update



# Editando as Visões

- Para passar uma variável do controlador para a view, previsamos usar um segundo argumento na função **render()**

```
public function actionIndex() {  
    $num = 23;  
    $this->render('index',  
        array('num' => $num)  
    );  
}
```

- A partir de agora, a view **index.php** pode acessar a variável **\$num**, que irá conter o valor 23.

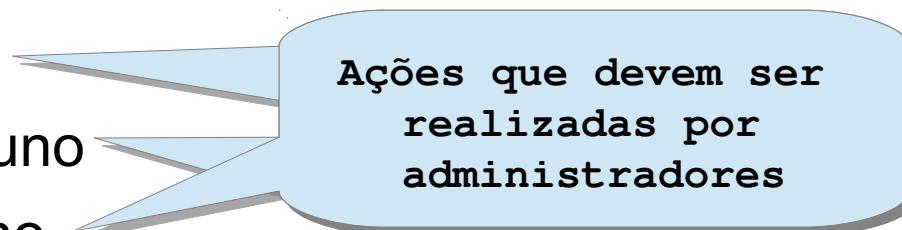


# Editando os Controladores

- Para melhor entender as funções dos controladores, basta pensar nas ações que podem ser requisitadas a um site
- Por exemplo, em se tratando de um aluno, podemos realizar as seguintes operações:
  - Criar um aluno
  - Atualizar um aluno
  - Apagar um aluno
  - Mostrar dados de um aluno
  - Listar alunos



# Editando os Controladores

- Para melhor entender as funções dos controladores, basta pensar nas ações que podem ser requisitadas a um site
  - Por exemplo, em se tratando de um aluno, podemos realizar as seguintes operações:
    - Criar um aluno
    - Atualizar um aluno
    - Apagar um aluno
    - Mostrar dados de um aluno
    - Listar alunos
- 
- Ações que devem ser realizadas por administradores



# Editando os Controladores

- O código do controlador Aluno gerado pelo Gii possui as definições dos seguintes métodos:
  - `actionCreate()`, `actionIndex()`, `actionView()`,  
`actionUpdate()`, `actionDelete()` e `actionAdmin()`
- Além desses, o código gerado irá conter as definições dos seguintes métodos adicionais:
  - `filters()`, `accessRules()`, `loadModel()`,  
`performAjaxValidation()`



# Editando os Controladores

- **actionIndex** é a action padrão dos controladores
  - Desta forma, quando a action não é especificada em uma requisição, o método **actionIndex()** será chamado
- Para mudar esse comportamento, basta assinalar uma action diferente para o atributo **\$defaultAction** da classe do controlador

```
class SomeController extends Controller {  
    public $defaultAction = 'view';
```



# Editando os Controladores

- O controlador padrão de um dado site é **SiteController**
- Para mudar esse comportamento padrão, basta adicionar a seguinte linha de código em **protected/config/main.php**

```
return array(
    /* outras configurações */
    'defaultController' => 'YourControllerId',
    /* ainda outras configurações */
);
```



# Editando os Controladores

- Uma das operações mais comuns em controladores é criar e/ou acessar instâncias de um modelo e passá-las para a view:
  - Em **ActionCreate** é preciso criar uma instância nova e vazia
  - Em **ViewCreate** e **UpdateCreate**, é preciso carregar uma instância já existente
  - Em **indexCreate**, é preciso carregar todas as instâncias de um modelo



# Editando os Controladores

- Para criar uma nova instância de um modelo, basta criar um novo objeto da classe desse modelo:

```
public function actionCreate() {  
    $model = new Aluno;  
    ...  
    $this->render('create',  
        array('model'=>$model)  
    );  
}
```

- Esse código irá criar uma variável chamada `$model` na view evocada



# Editando os Controladores

- Para criar uma nova instância de um modelo, basta criar um novo objeto da classe desse modelo:

```
public function actionCreate() {  
    $model = new Page;  
    ...  
    $th  
}  
);  
}  
• Esse código
```

**Exercício:** Visualizar a view e a action responsáveis por criar uma nova instância de **Aluno**

view



# Editando os Controladores

- Algumas vezes, será necessário imprimir uma view sem incorporar o conteúdo do layout
- Para fazer isso, basta chamar a função `renderPartial()`, ao invés da função `render()`



# Editando os Controladores

- Carregar uma instância de um modelo é necessário em várias funções, por exemplo:
  - **actionView()**, **actionUpdate()** e **actionDelete()**
- Para carregar uma instância, os controladores gerados pelo Gii possuem a seguinte função:

```
public function loadModel ($id) {  
    $model = Page::model ()->findByPk ($id);  
    if ($model === null) {  
        throw new CHttpException (...);  
    }  
    return $model;  
}
```

Chave primária



# Editando os Controladores

- A action **actionView** é um exemplo de função que usa **loadModel**:

```
public function actionView($id) {  
    $this->render('view', array(  
        'model' => $this->loadModel($id),  
    ));  
}
```



# Editando os Controladores

- A action **actionView** é um exemplo de função que usa **loadModel**:

```
public function actionView($id) {  
    $this->render('view', array(  
        'model' => $this->  
        loadModel('Aluno', $id)  
    ));  
}
```

**Exercício:** Visualizar a view e a action responsáveis por visualizar uma dada instância **Aluno**



# Editando os Controladores

- A action `actionView` é um exemplo de função que usa `loadModel`:

Instituto de Computação

Home Alunos About Contact Logout (a)

Home » Alunos » 1

## View Aluno #1

ID	1
Matrícula	18910303
Nome	ALESSANDRO DE MEDEIROS
Sexo	M
Curso	2
Ano de Ingresso	1989

Troque por Visualizar Aluno

Remova

Troque pelo Nome do Curso



# Editando os Controladores

- A action **actionUpdate** é outro exemplo de função que usa **loadModel**:

```
public function actionUpdate($id) {  
    $model=$this->loadModel($id);  
    ...  
    $this->render('update', array(  
        'model'=>$model,  
    ));  
}
```



# Editando os Controladores

- A action **actionUpdate** é outro exemplo de função que usa **loadModel**:

```
public function actionUpdate($id) {  
    $model=$this->loadModel($id);  
    ...  
    $th  
} );
```

**Exercício:** Visualizar a view e a action responsáveis por atualizar uma dada instância de **Aluno**



# Editando os Controladores

- Para carregar todos as instâncias de um modelo, podemos usar a função **findAll()** da classe CActiveRecord

```
$models = Aluno::model()->findAll();
```
- Essa função irá retornar um array com todas as instâncias do modelo



# Editando os Controladores

Instituto

Coloque em português

Home Alunos About Contact Logout /edit/  
Home » Alunos » Create

Troque por  
Novo Aluno

## Create Aluno

Fields with \* are required.

Matrícula \*

Coloque em português

Nome

Troque por um select box

Sexo

Troque por um select box

Curso

Troque por um select box

Ano de Ingresso

# Editando os Controladores

- Os controladores também possuem métodos não ligados a ações, dentre os quais destacamos accessRules()
- Este método é uma peça chave de segurança, determinando quem pode fazer o que na aplicação

```
public function accessRules() {  
    return array(  
        array('allow', 'actions'=>array('index', 'view'),  
              'users'=>array('*'),  
              ),  
        array('allow', 'actions'=>array('create', 'update'),  
              'users'=>array('@'),  
              ),  
        array('allow', 'actions'=>array('admin', 'delete'),  
              'users'=>array('admin'),  
              ),  
        array('deny',  
              'users'=>array('*'),  
              ),  
    );  
}
```



# Editando as Visões

- Para passar uma variável do controlador para a view, previsamos usar um segundo argumento na função **render()**

```
public function actionIndex() {  
    $num = 23;  
    $this->render('index',  
        array('num' => $num)  
    );  
}
```

- A partir de agora, a view **index.php** pode acessar a variável **\$num**, que irá conter o valor 23.

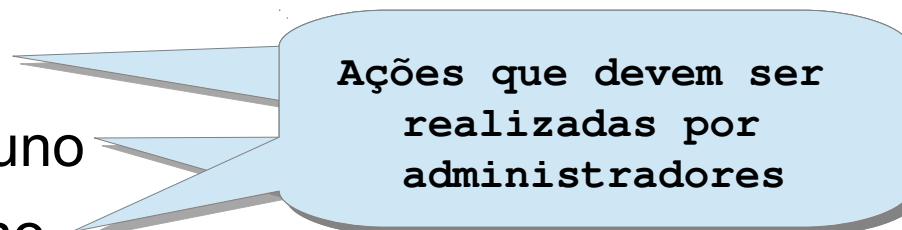


# Editando os Controladores

- Para melhor entender as funções dos controladores, basta pensar nas ações que podem ser requisitadas a um site
- Por exemplo, em se tratando de um aluno, podemos realizar as seguintes operações:
  - Criar um aluno
  - Atualizar um aluno
  - Apagar um aluno
  - Mostrar dados de um aluno
  - Listar alunos



# Editando os Controladores

- Para melhor entender as funções dos controladores, basta pensar nas ações que podem ser requisitadas a um site
  - Por exemplo, em se tratando de um aluno, podemos realizar as seguintes operações:
    - Criar um aluno
    - Atualizar um aluno
    - Apagar um aluno
    - Mostrar dados de um aluno
    - Listar alunos
- 
- Ações que devem ser realizadas por administradores



# Editando os Controladores

- O código do controlador Aluno gerado pelo Gii possui as definições dos seguintes métodos:
  - `actionCreate()`, `actionIndex()`, `actionView()`,  
`actionUpdate()`, `actionDelete()` e `actionAdmin()`
- Além desses, o código gerado irá conter as definições dos seguintes métodos adicionais:
  - `filters()`, `accessRules()`, `loadModel()`,  
`performAjaxValidation()`



# Editando os Controladores

- **actionIndex** é a action padrão dos controladores
  - Desta forma, quando a action não é especificada em uma requisição, o método **actionIndex()** será chamado
- Para mudar esse comportamento, basta assinalar uma action diferente para o atributo **\$defaultAction** da classe do controlador

```
class SomeController extends Controller {  
    public $defaultAction = 'view';
```



# Editando os Controladores

- O controlador padrão de um dado site é **SiteController**
- Para mudar esse comportamento padrão, basta adicionar a seguinte linha de código em **protected/config/main.php**

```
return array(
    /* outras configurações */
    'defaultController' => 'YourControllerId',
    /* ainda outras configurações */
);
```



# Editando os Controladores

- Uma das operações mais comuns em controladores é criar e/ou acessar instâncias de um modelo e passá-las para a view:
  - Em **ActionCreate** é preciso criar uma instância nova e vazia
  - Em **ViewCreate** e **UpdateCreate**, é preciso carregar uma instância já existente
  - Em **indexCreate**, é preciso carregar todas as instâncias de um modelo



# Editando os Controladores

- Para criar uma nova instância de um modelo, basta criar um novo objeto da classe desse modelo:

```
public function actionCreate() {  
    $model = new Aluno;  
    ...  
    $this->render('create',  
        array('model'=>$model)  
    );  
}
```

- Esse código irá criar uma variável chamada `$model` na view evocada



# Editando os Controladores

- Para criar uma nova instância de um modelo, basta criar um novo objeto da classe desse modelo:

```
public function actionCreate() {  
    $model = new Page;  
    ...  
    $th  
}  
);  
}  
• Esse código
```

**Exercício:** Visualizar a view e a action responsáveis por criar uma nova instância de **Aluno**

view



# Editando os Controladores

- Algumas vezes, será necessário imprimir uma view sem incorporar o conteúdo do layout
- Para fazer isso, basta chamar a função `renderPartial()`, ao invés da função `render()`



# Editando os Controladores

- Carregar uma instância de um modelo é necessário em várias funções, por exemplo:
  - **actionView()**, **actionUpdate()** e **actionDelete()**
- Para carregar uma instância, os controladores gerados pelo Gii possuem a seguinte função:

```
public function loadModel ($id) {  
    $model = Page::model ()->findByPk ($id);  
    if ($model === null) {  
        throw new CHttpException (...);  
    }  
    return $model;  
}
```

Chave primária



# Editando os Controladores

- A action **actionView** é um exemplo de função que usa **loadModel**:

```
public function actionView($id) {  
    $this->render('view', array(  
        'model' => $this->loadModel($id),  
    ));  
}
```



# Editando os Controladores

- A action **actionView** é um exemplo de função que usa **loadModel**:

```
public function actionView($id) {  
    $this->render('view', array(  
        'model' => $this->  
        loadModel('Aluno', $id)  
    ));  
}
```

**Exercício:** Visualizar a view e a action responsáveis por visualizar uma dada instância **Aluno**



# Editando os Controladores

- A action `actionView` é um exemplo de função que usa `loadModel`:

Instituto de Computação

Home Alunos About Contact Logout (a)

Home » Alunos » 1

## View Aluno #1

ID	1
Matrícula	18910303
Nome	ALESSANDRO DE MEDEIROS
Sexo	M
Curso	2
Ano de Ingresso	1989

Troque por Visualizar Aluno

Remova

Troque pelo Nome do Curso



# Editando os Controladores

- A action **actionUpdate** é outro exemplo de função que usa **loadModel**:

```
public function actionUpdate($id) {  
    $model=$this->loadModel($id);  
    ...  
    $this->render('update', array(  
        'model'=>$model,  
    ));  
}
```



# Editando os Controladores

- A action **actionUpdate** é outro exemplo de função que usa **loadModel**:

```
public function actionUpdate($id) {  
    $model=$this->loadModel($id);  
    ...  
    $th  
} );
```

**Exercício:** Visualizar a view e a action responsáveis por atualizar uma dada instância de **Aluno**



# Editando os Controladores

- Para carregar todos as instâncias de um modelo, podemos usar a função **findAll()** da classe CActiveRecord

```
$models = Aluno::model()->findAll();
```
- Essa função irá retornar um array com todas as instâncias do modelo



# Editando os Controladores

Instituto

Coloque em português

Home Alunos About Contact Logout /edit/  
Home » Alunos » Create

Troque por  
Novo Aluno

## Create Aluno

Fields with \* are required.

Matrícula \*

Coloque em português

Nome

Sexo

Curso

Ano de Ingresso

Troque por um select box

Troque por um select box

Troque por um select box

# Editando os Controladores

- Os controladores também possuem métodos não ligados a ações, dentre os quais destacamos accessRules()
- Este método é uma peça chave de segurança, determinando quem pode fazer o que na aplicação

```
public function accessRules() {  
    return array(  
        array('allow', 'actions'=>array('index', 'view'),  
              'users'=>array('*'),  
              ),  
        array('allow', 'actions'=>array('create', 'update'),  
              'users'=>array('@'),  
              ),  
        array('allow', 'actions'=>array('admin', 'delete'),  
              'users'=>array('admin'),  
              ),  
        array('deny',  
              'users'=>array('*'),  
              ),  
    );  
}
```

