

CAPÍTULO 1

Conceitos de extensões Joomla!

Entendendo o que é extensão

Extensão pode ser entendida como uma pequena aplicação desenvolvida com regras de construção estabelecidas pelo ambiente Joomla!. É um produto instalável que tem por finalidade adicionar novas características ao Joomla!.

As extensões são classificadas em componentes, módulos, plug-ins, temas (templates) e idioma:

- **Componente:** é um tipo de aplicação responsável por apresentar conteúdo na página principal. Somente um componente pode ser executado ou exibido por página carregada. Entre os demais tipos de extensões é o mais complexo no que se refere ao esforço de desenvolvimento e administração. Cada componente em uma instalação Joomla! tem um nome único. Em geral, seu código é dividido em duas partes, frontend e backend. Cada parte fica situada em pastas (diretórios) distintas.
- **Módulo:** é uma pequena porção de software que normalmente é apresentada como parte de um conteúdo de uma página. Mais de um módulo pode ser apresentado em uma página na forma de uma barra lateral, menu, cabeçalho ou rodapé. A configuração de um módulo consiste basicamente em definição de parâmetros de apresentação, incluindo a sua posição na página.
- **Plug-in:** é uma pequena porção de código contido em uma função ou método que pode ser executado em resposta a eventos que ocorrem no Joomla!. Como exemplo, é possível desenvolver um plug-in que coloca parte do texto de um artigo em negrito durante a sua apresentação na página.

- **Template:** é responsável pela forma com que o site será apresentado ao usuário, incluindo: cores, fontes de caracteres, estilos etc. É ele quem determina a organização dos objetos de uma página no navegador web.
- **Idioma:** permite o desenvolvimento de sites para vários idiomas. Ao instalar esse tipo de extensão e selecioná-lo como padrão, todas as mensagens, menu e textos pertencentes do Joomla! serão apresentados no novo idioma.

A figura 1.1 ilustra uma página disponível aos visitantes destacando as extensões mencionadas anteriormente.

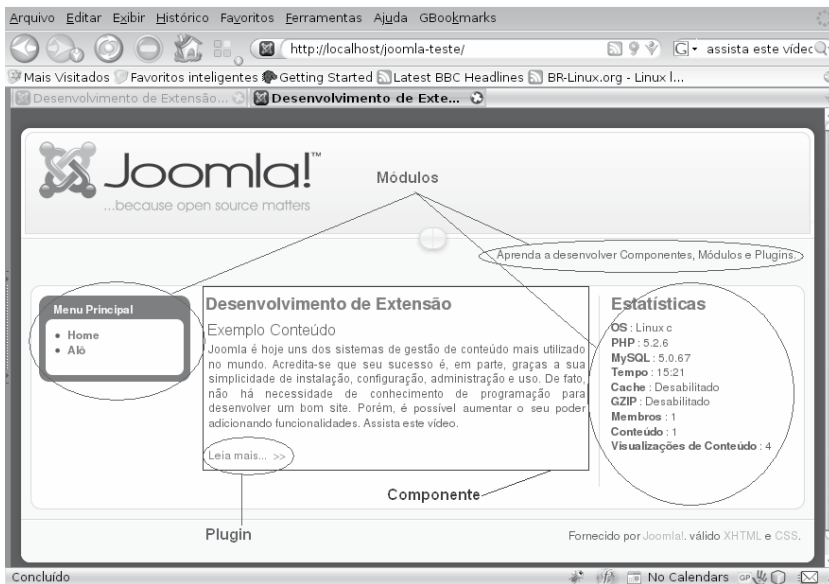


Figura 1.1 – Exemplo de disposição de componente, módulos e plug-in em uma página Joomla! (frontend).

Entendendo componentes do Joomla!

Um componente é dividido em duas partes, frontend e backend. O frontend é a parte responsável pela apresentação do conteúdo para o público usuário em geral. Frontend pode ser entendido também como website. O backend é parte responsável pela manutenção do componente. Geralmente, usamos o backend para administrar componentes no Joomla! e definir quando, onde, de que forma e com qual conteúdo um componente deve ser apresentado no site. As figuras 1.2a e 1.2b ilustram melhor esse conceito.



Figura 1. 2a – Apresentação do frontend do Joomla!.

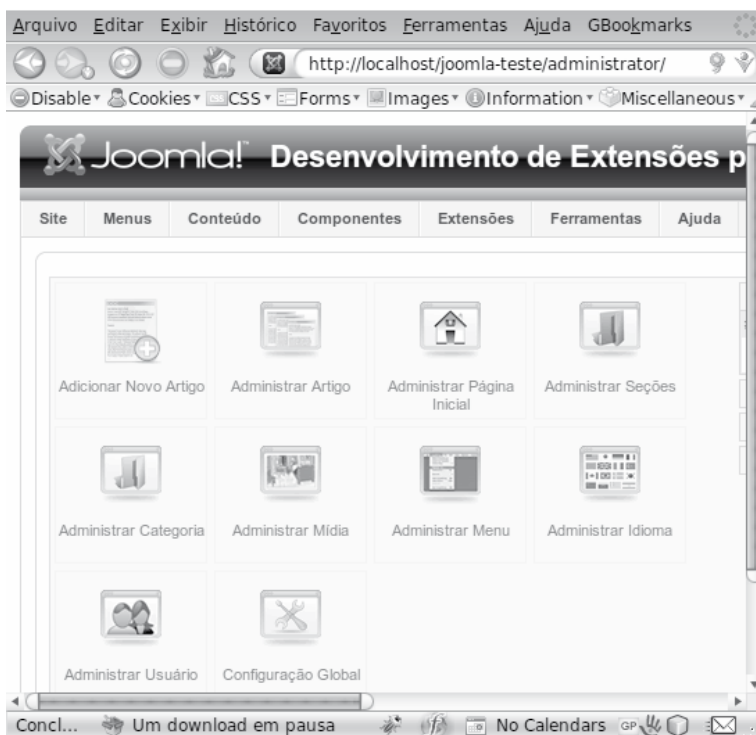


Figura 1.2b – Apresentação do backend do Joomla!.

Estrutura de arquivo de um componente

Um componente em Joomla! consiste em um ou mais arquivos (scripts PHP, HTML, JavaScript, CSS, mídia etc.) divididos em basicamente duas pastas principais: `administrator/components/`, para efetuar operações de administração (backend), e `components/`, para executar operações de apresentação ao usuário final (frontend). A figura 1.3 ilustra a estrutura de diretório do Joomla!.

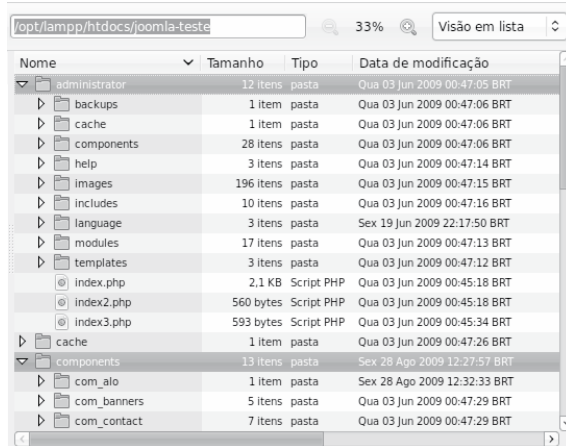


Figura 1.3 – Estrutura de pastas do Joomla! destacando backend e frontend.

Ainda em relação à figura 1.3, observe que o Joomla! foi instalado na pasta `/opt/lampp/htdocs/joomla-teste`. As linhas destacadas representam as pastas utilizadas para os componentes. No Windows, utilizando o XAMPP, a instalação equivalente seria no diretório `c:\xampp\htdocs\joomla-teste`.

Como ponto de partida, vamos iniciar com o desenvolvimento de um componente cuja única função será apresentar a mensagem **Alô** na área de conteúdo da página principal. Para tanto, crie um script PHP conforme mostrado a seguir.

Arquivo `alo.php`:

```
<?php
    echo 'Alô!'
?>
```

Usando como referência a pasta de instalação do Joomla!, entre no diretório `components`, crie a pasta `com_alo` e grave o script `alo.php`. A figura 1.4 ilustra essa operação.

Nome	Tamanho	Tipo
▶ administrador	12 items	pasta
▶ cache	1 item	pasta
▼ components	13 items	pasta
▼ com_alo	1 item	pasta
alo.php	25 bytes	Script PHP
▶ com_banners	5 items	pasta

Figura 1.4 – Apresentação do script alo.php gravado em components/com_alo.

É muito importante que o leitor não confunda as pastas administrator/ components/ e components/. A primeira está relacionada à administração e não é objeto deste capítulo.

Para execução do componente, digite a URL *http://localhost/joomla-teste/index.php?option=com_alo* em seu navegador web. O resultado deverá se parecer com o conteúdo da figura 1.5:



Figura 1.5 – Execução do componente Alô (frontend).

Observe que a execução do componente é delegada ao Joomla! via URL com o parâmetro "*?option=com_alo*". Isso é suficiente para que o script *alo.php* seja executado. Contudo, da forma que o componente foi construído, é possível burlar o sistema e executá-lo sem o controle do Joomla!. Para verificar isso,

execute o script `alo.php` usando a URL `http://localhost/joomla-teste/components/com_alo/alo.php`. O resultado dessa operação é simplesmente uma página limpa contendo a mensagem 'Alô!'. A figura 1.6 ilustra esse procedimento.

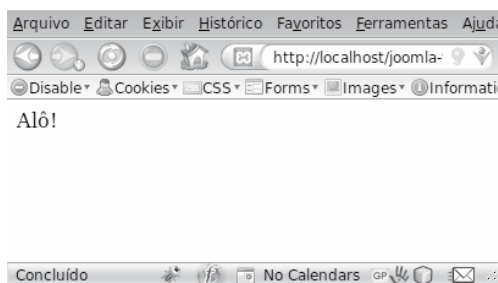


Figura 1.6 – Execução direta do script `alo.php`.

É claro que o comportamento mostrado na figura 1.6 não é desejável. Qualquer pessoa com conhecimento da estrutura de pastas do Joomla! poderia ter acesso direto ao componente. Para evitar isso, é recomendado que todo script PHP escrito para o Joomla! inicie com o código `"defined('_JEXEC') or die('Acesso restrito ao Joomla!');"`, conforme indicado no código a seguir.

Arquivo `alo.php` protegido do acesso direto:

```
<?php
    defined('_JEXEC') or die('Acesso restrito ao Joomla!');
    echo 'Alô!';
?>
```

Faça a alteração indicada no script `alo.php` e tente executar a URL `http://localhost/joomla-teste/components/com_alo/alo.php` novamente. O resultado deverá parecer com a figura 1.7.

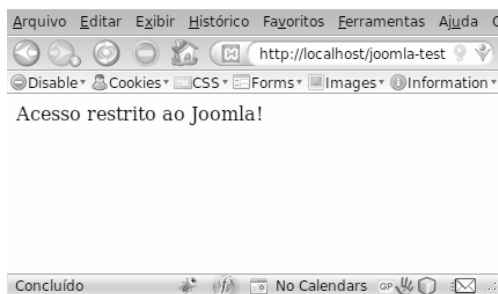


Figura 1.7 – Evitando acesso direto ao componente.

Explicação: a constante `_JEXEC` é definida no script `index.php` existente no diretório raiz da instalação do Joomla!. Se o leitor abrir esse script verá o comando `define('_JEXEC',1)`. É importante que o leitor perceba que `index.php` é o ponto de entrada para execução de qualquer componente. O que determina qual componente será executado é o conteúdo do argumento `option` (por exemplo: `index.php?option=com_a1o`). Dessa forma, é possível verificar no script `alo.php` se a constante `_JEXEC` foi definida, caso contrário a execução será cancelada.

Interação com o componente

Vamos fazer uma pequena alteração no código `alo.php` de forma a deixá-lo mais amigável. Isto é, vamos interagir com o componente enviando parâmetros via URL.

Altere o código `alo.php` conforme mostrado a seguir.

```
<?php
    defined('_JEXEC') or die('Acesso restrito ao Joomla!');
    $task = JRequest::getVar('task',' ');
    $nome = JRequest::getVar('nome',' ');
    switch ($task) {
        case 'show':
            echo 'Alô ' . $nome . '. Seja bem vindo ao Joomla!';
            break;
        default:
            echo 'Alô visitante desconhecido. Seja bem vindo ao Joomla!';
            break;
    }
?>
```

A alteração sugerida apresenta alguns elementos novos. A chamada `JRequest::getVar` obtém de `$_REQUEST`, `$_POST` ou `$_GET` os parâmetros desejados. Dessa forma, eliminamos a possibilidade de utilização de técnicas de invasão como `code injection`.

Se a URL `http://localhost/joomla-teste/index.php?option=com_a1o&task=show&nome=Ricardo` for executada, `$task` receberá o valor 'show' e `$nome` receberá o valor 'Ricardo', respectivamente. O parâmetro `option` informa ao Joomla! a pasta do componente a ser executado. Em nosso caso, a pasta é `com_a1o` e o script é `alo.php`.

Continuando com o fluxo de execução, quando os parâmetros `task` e `nome` não são incluídos na URL, uma string vazia é atribuída tanto a variável `$task` como `$nome`. Dessa forma, o comando `switch` desvia o fluxo de execução para

a cláusula `default` e uma mensagem genérica é apresentada no navegador. Se `$task` e `$nome` contiverem 'show' e 'Ricardo', respectivamente, uma mensagem personalizada será apresentada.

No navegador web digite a URL `http://localhost/joomla-teste/index.php?option=com_alo` e em seguida a URL `http://localhost/joomla-teste/index.php?option=com_alo&task=show&nome=Ricardo`. As figuras 1.8a e 1.8b ilustram os respectivos resultados.



Figura 1.8a – Apresentação de mensagem genérica na área de conteúdo.

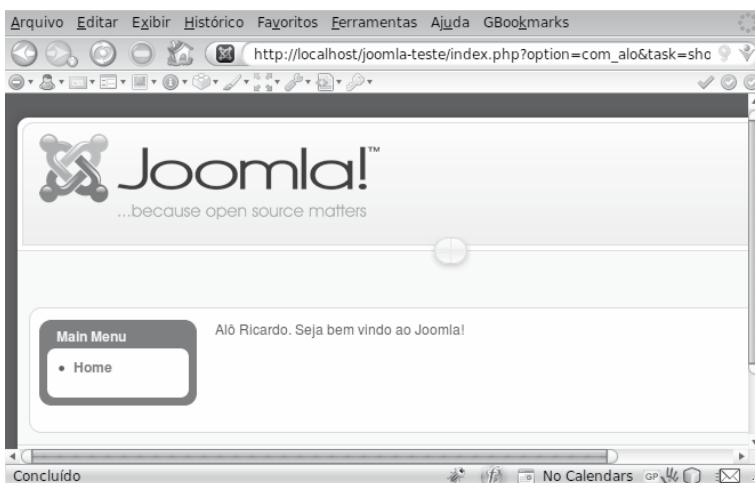


Figura 1.8b – Apresentação de mensagem personalizada na área de conteúdo.

Joomla! é um sistema de código aberto. Das várias vantagens nessa modalidade de distribuição, destaca-se a possibilidade de aprender analisando o próprio código-fonte do sistema. Em particular, o Joomla! vem com um conjunto básico de extensões. Uma maneira interessante de aprender a desenvolver é estudar o código dos componentes, módulos e plug-ins instalados. Comece observando como os arquivos estão organizados, tanto no lado administrador (backend) como no site (frontend). Se necessário, utilize J!Dump, apresentado a seguir, para inspecionar variáveis.

Uso de J!Dump para mostrar dados em uma janela pop-up

J!Dump é uma versão avançada das funções do PHP `var_dump` e `print_r`. Essa ferramenta torna o desenvolvimento de extensões, incluindo templates, mais fácil. J!Dump é dividido e distribuído em duas extensões, um componente e um plug-in. Tanto o componente como o plug-in devem ser instalados, caso contrário ocorrerá erro ao tentar usá-lo. O processo de instalação é o mesmo utilizado para instalar qualquer extensão.

Baixando o componente e o plug-in do J!Dump

Entre no site <http://extensions.joomla.org/> e localize J!Dump ou vá direto a <http://joomla.code.org/gf/project/jdump/frs>. Na elaboração deste livro os arquivos baixados foram `jdump_v1.1.0_component.zip` (componente) e `jdump_v1.1.0_plugin.zip` (plug-in).

Importante: após a instalação, no menu **Extensões** selecione **Administrar plug-in** e altere o status de J!Dump para habilitado.

Inspeção de variáveis com J!Dump

Uma vez instalado e configurado corretamente o J!Dump, inspecionar variáveis se torna uma tarefa tão simples quanto usar as funções `print_r()` e `var_dump()`, bastando, para tanto, fazer a chamada da função `dump()` em qualquer lugar do código. Em tempo de execução, uma janela pop-up aparecerá, mostrando o conteúdo da variável ou variáveis inspecionadas. Para ilustrar essa funcionalidade vamos fazer uma alteração no código de nosso componente conforme mostrado a seguir.

Alteração do script `alo.php` para inspecionar variáveis.

```
<?php
    defined('_JEXEC') or die('Acesso restrito ao Joomla!');
    $task = JRequest::getVar('task', ' ');
    $nome = JRequest::getVar('nome', ' ');
    dump($task, 'Valor de task:');
    dump($nome, 'Valor de Nome:');
    switch ($task) {
        case 'show':
            echo 'Alô ' . $nome . '. Seja bem vindo ao Joomla!';
            break;
        default:
            echo 'Alô visitante desconhecido. Seja bem vindo ao Joomla!';
            break;
    }
?>
```

Insira a URL `http://localhost/joomla-teste/index.php?option=com_alo&task=XYZ&nome=Fulano`.

Uma janela pop-up será aberta conforme ilustra a figura 1.9.

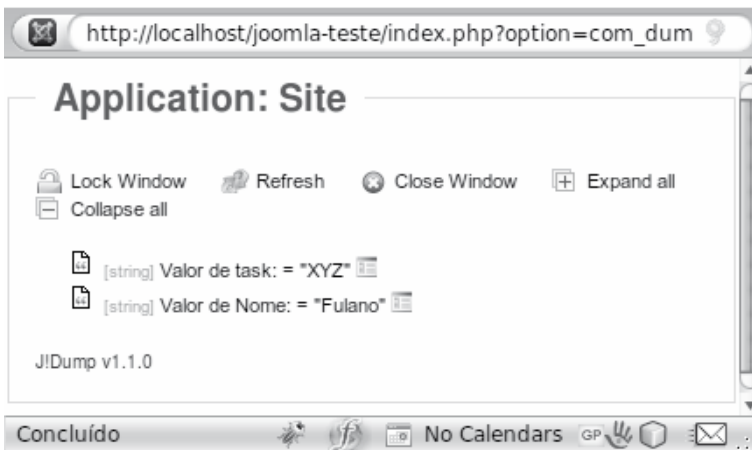


Figura 1.9 – Janela pop-up do J!Dump.

Além da função `dump()`, você poderá usar `dumpMessage()`, `dumpSysinfo()`, `dumpTrace()` e `dumpTemplate()`. A tabela 1.1 detalha essas funções.

Tabela 1.1 – Funções do J!Dump

Função	Descrição
dump()	Conforme visto anteriormente, apresenta o conteúdo de uma variável (simples, array ou objeto).
dumpMessage()	Apresenta uma mensagem em uma janela pop-up (por exemplo, <code>dumpMessage('inclua sua mensagem aqui')</code>).
dumpSysinfo()	Mostra informações importantes sobre o sistema.
dumpTrace()	Apresenta a pilha de execução efetuadas até o momento.
dumpTemplate()	Deve ser usada dentro de um template (arquivo <code>index.php</code>). Apresenta conteúdo de parâmetros.

Separação da lógica do componente de sua apresentação

Em desenvolvimento de aplicações PHP em geral, é comum encontrar trecho de código em PHP misturado com código HTML, ou seja, a lógica e a apresentação estão em um mesmo arquivo. O fato é que essa abordagem de desenvolvimento deixa a aplicação por demais complexa no que tange à visibilidade ou ao entendimento do código. Para fazer uma aplicação mais legível buscamos separar o código responsável pela lógica do código responsável pela apresentação. Para tanto, usamos dois arquivos diferentes conforme podemos ver a seguir.

Efetue as modificações em `alo.php` destacadas no código a seguir.

```
<?php
    defined('_JEXEC') or die('Acesso restrito ao Joomla!');
    jimport('joomla.application.helper');
    require_once(JApplicationHelper::getPath('front_html', 'com_alo'));
    $task = JRequest::getVar('task', ' ');
    $nome = JRequest::getVar('nome', ' ');
    switch ($task) {
        case 'show':
            HTML_alo::mostraNome($nome);
            break;
        default:
            HTML_alo::mostraPadrao();
            break;
    }
?>
```

Crie um novo script `alo.html.php` e grave na mesma pasta do script `alo.php`.

```
<?php
    defined('_JEXEC') or die('Acesso restrito ao Joomla!');
    class HTML_alo {
        function mostraNome($nome) {
            echo 'Alô ' . $nome . '. Seja bem vindo ao Joomla!';
        }
        function mostraPadrao() {
            echo 'Alô visitante desconhecido. Seja bem vindo ao Joomla!';
        }
    }
?>
```

Pela análise do código-fonte anterior, o leitor pode observar que não há alteração no comportamento do componente. Isto é, ele apresenta exatamente a mesma coisa que a versão anterior. O leitor pode achar que essa estratégia deixou a codificação mais complexa e extensa. Contudo, na proporção que a aplicação evolui, essa abordagem se mostra bem mais legível que a anterior.

Uma observação importante a ser feita no código é a inclusão da chamada `require_once(JApplicationHelper::getPath('front_html','com_alo'))`. Ela é usada para determinar o caminho do arquivo `alo.html.php`. `JapplicationHelper` é uma classe que fornece um conjunto de funções que ajudam o desenvolvedor a obter informações sobre o ambiente, tais como cliente, caminhos de pastas, “parse” XML etc. Em nosso caso, a chamada `JapplicationHelper::getPath('front_html','com_alo')` retornará uma string com o conteúdo `lampp\htdocs\joomla\dev\components\com_alo\alo.html.php`. Com isso, `require_once` incluirá o arquivo `alo.html.php`. A alteração mais significativa do script `alo.php` foi a substituição do comando `echo` pelas chamadas `HTML_alo::mostraNome($nome)` e `HTML_alo::mostraPadrao()`. Toda a API oferecida pelo Joomla! fica localizada na pasta `libraries/`. O comando `jimport` carrega o script `helper.php` existente na pasta `libraries/joomla/application`.

Embora em primeira análise essa abordagem seja mais complexa que a anterior, fica clara a separação entre lógica e apresentação. Veremos no decorrer deste livro que essa abordagem deixa, na realidade, a solução mais simples de ser entendida na proporção em que a solução fica mais complexa.

A figura 1.10 ilustra a disposição da estrutura de pastas e arquivos de nosso componente.

Nome	Tamanho	Tipo
▶ administrator	12 items	pasta
▶ cache	1 item	pasta
▼ components	14 items	pasta
▼ com_alo	2 items	pasta
alo.html.php	371 bytes	Script PHP
alo.php	493 bytes	Script PHP
▶ com_banners	5 items	pasta

Figura 1.10 – Estrutura do frontend (versão final do capítulo 1).

Resumo do capítulo

Este capítulo conceituou extensões e iniciou a construção de um pequeno componente. Mostrou como o Joomla! organiza suas pastas separando-as em administração de extensões (backend) e site (frontend). Introduziu técnicas de segurança para evitar injeção de código via URL. Abordou a importância do desenvolvimento do componente separando-o em lógica e apresentação. Por fim, apresentou o J!Dump como ferramenta de inspeção de variáveis.

O que foi feito neste capítulo?

Criamos uma pequena parte de um componente Joomla! (um frontend).

Instalamos o componente com procedimento manual. Isto é, seguindo as regras de formação de nomes de componentes, criamos manualmente a pasta `com_alo` no diretório `components/` e um arquivo `alo.php`.

Vimos que é possível interagir com o componente usando parâmetros na URL.

Como o componente ainda não foi registrado formalmente no Joomla!, o acesso a ele ocorre usando a URL da seguinte forma:

```
index.php?option=nome_componente&param1=valor1&param2=valor2
```

Para não deixar o componente vulnerável a ataques de hackers adicionamos o código: `defined('_JEXEC') OR die('Acesso restrito ao Joomla!')` no início de cada script.

Inspecionamos variáveis usando a ferramenta J!Dump.

Separamos o componente em dois scripts, um para cuidar da lógica e outro responsável pela apresentação.