Edição especial de lançamento

edição #1 - jan/2007

PHP MAGAZINE

A sua revista digital de PHP

Gerenciadores de conteúdo

Como as ferramentas de CMS podem se tornar parceiras do seu sucesso

Sessões

Descubra o que elas podem fazer pelo seu sistema

Veja também:

- Padrões de Projetos
- Integração de sistemas com SOA

Finalmente lançada a primeira revista digital de PHP no Brasil

Entrevista com Cristian Pedroso

www.phpmagazine.com.br

Editores
Ricardo Aragão, ricardoaragao@phpmagazine.com.br
Flávio Z. Fagundes, zach@phpmagazine.com.br

Administração
Norberto Augusto, augusto@phpmagazine.com.br

Comercial
Norberto Augusto, augusto@phpmagazine.com.br
Ralf Braga, ralf@phpmagazine.com.br

Projeto gráfico
Ricardo Aragão da Silva
Flávio Zacharias Fagundes
Camilla Carvalho

Revisão técnica
Ricardo Aragão da Silva
Flávio Zacharias Fagundes

Revisão português
Camilla Carvalho

Correspondentes e colaboradores
Paulino Michelazzo
Rafael Dohms
Jennifer Franchi
Marcos Pont
Alexandre Altair de Melo
José Walter Pereira Dutra
Leandro Schwarz
Felipe Moreno
Rafael Leoni

Edição de lançamento

Amigo leitor,

Após dois anos de projeto, hoje apresentamos a primeira edição da revista digital PHP Magazine.

Projeto longo, demorado, suado, mas muito compensador, pois todos passam a ter acesso a um excelente conteúdo sobre PHP.

A equipe da revista agradece a todos que participaram da idealização em qualquer uma de suas fases. Desde a escolha do nome da revista até a última revisão antes da publicação, todos são lembrados. Sem a participação de cada um deles, você não estaria lendo este editorial.

Não podemos esquecer dos profissionais altamente capacitados que enviaram seus artigos para esta edição. Acreditando no potencial do projeto, abdicaram de suas horas de lazer para compartilhar seus conhecimentos técnicos.

A equipe estará no próximo mês selecionando voluntários para trabalhos fixos dentro do projeto. Em breve anunciaremos as áreas com vagas disponíveis. Você já pode contribuir também com artigos, notícias, links, e eventos.

Aqueles que se interessarem podem entrar em contato pelo email contato@phpmagazine.com.br

Contamos com seu apoio na próxima publicação.

Uma boa leitura e até a segunda edição.

Equipe PHP Magazine

Conteúdo

Artigos

3, Ferramentas de CMS parceiras do seu sucesso
7, Padrões de Projeto (Design Patterns) em PHP da Teoria a Prática
11, Integração de sistemas utilizando webservices baseado na tecnologia SOA
16,Trabalhando com sessões

Entrevista

22, Cristian Pedroso

Ferramentas de CMS

parceiras do seu sucesso

Por Paulino Michelazzo

As ferramentas de CMS estão melhores, mais eficientes e prometem muito em sua nova safra. Se você pretende desenvolver um site próprio ou ainda para clientes, não as perca de vista. O que elas tem a oferecer para o sucesso de seu website é muito mais do que você pode imaginar.

Começa mais um ano e com ele os planejamentos para seu site novo ou ainda para aquele cliente que você enviou a proposta em meados do ano passado e que agora resolve colocar em prática seus anseios para dar uma "cara nova" a sua imagem na web. Neste momento, você já pensa nas ferramentas que vai usar, nas tecnologias e, principalmente, no tempo que irá gastar para implementar tudo aquilo que é necessário no menor espaço de tempo possível. Diante destas variáveis, o desenvolvedor mais atento pode questionar: por que não usar um CMS para o projeto? Será que eles estão maduros para atender minhas necessidades? As respostas para estas perguntas e outras sobre o assunto estão neste artigo.

A história dos CMS's

Nem sempre a Internet foi dinâmica, eficiente e útil como é atualmente. Em seus primórdios, era estática, feia e com pouca utilidade para nós que a empregamos hoje como fonte de lazer, entretenimento e solução para vários dos problemas da vida moderna.

Com o advento das linguagens de programação para a web, e também da migração de outras que já existiam para este ambiente, começaram a serem oferecidos vários serviços que vão desde a simples busca de uma informação sobre seu bairro ou cidade até a compra de produtos e serviços de empresas inexistentes no mundo físico, mas que podem ser entregues na sua casa de uma forma rápida e cômoda.

Esta facilidade trouxe um grande problema tanto para quem procura uma informação como para quem a disponibiliza: como organizar o grande volume de textos, dados, sons, vídeos e imagens de forma que qualquer pessoa possa rapidamente encontrar aquilo que procura e, em contrapartida, permitir a quem disponibiliza as informações uma forma ágil de organizá-las e mantê-las com o intuito de estarem sempre atualizadas e facilmente encontráveis, mesmo em um cenário onde a velocidade é premissa básica para os negócios e "estar presente" é ordem a ser seguida à risca?

Além destes pontos, existiam outros que precisavam ser atendidos para propiciar um ambiente de fácil manuseio dos dados, separando deles a apresentação e formatação, principalmente a partir do momento que uma mesma informação poderia ser visualizada tanto em um monitor LCD como na tela de um telefone celular ou ainda em um assistente pessoal digital (PDA) dentro de um veículo. Então, formatações especiais, fontes, paginações, posicionamentos, cores, alinhamentos e outras "perfumarias" não deveriam fazer parte do conteúdo em si, mas serem agregadas de acordo com o tipo de interface utilizada para sua visualização.

A solução foi então criar ferramentas gerenciadoras de conteúdo que pudessem fornecer uma interface para a rápida organização da informação, mas que também separassem o design e formatação em camadas diferentes que permitissem a qualquer

momento sua troca ou "chaveamento".

Também era salutar que este gerenciador permitisse a utilização de módulos específicos para tarefas específicas e que estes pudessem ser agregados ou removidos do conjunto sem a modificação do conteúdo, permitindo assim a alteração de todo o sistema, passando de um simples organizador de notícias para uma loja virtual em pouco espaço de tempo e com o menor esforço possível. Como em um brinquedo de montar, cujas peças vão sendo encaixadas para criar carros, barcos, casas ou aquilo que a imaginação permite, o sistema de gerenciamento deveria ter instruções básicas em seu núcleo e disponibilizar conectores para que estes módulos adicionais fossem criados futuramente, estendendo as características da ferramenta como um todo.

Este conjunto de idéias e necessidades forneceu a base para a criação de ferramentas hoje categorizadas como CMS's ou Sistemas de Gestão de Conteúdo, as quais tratam a informação totalmente separada do sistema e da interface, permitindo dezenas de variações de acordo com as partes que estão no conjunto, incorporando assim várias identidades, ao mesmo tempo ou não.

Os CMS's hoje

As ferramentas de gestão de conteúdo, ou CMS's são atualmente poderosos frameworks que fornecem um vasto conjunto de API's básicas ao desenvolvedor de aplicações para que consigam criar grandes produtos ou serviços partindo-se de uma mesma base de trabalho. Estes frameworks são feitos nas mais diversas linguagens e suportados pelas mais diferentes plataformas de sistemas operacionais, bancos de dados e servidores web, permitindo, assim, sua integração e uso em qualquer ambiente de trabalho, desde o mais simples servidor até o mais complexo datacenter.

Suas funcionalidades variam de acordo com o uso que se pretende, existindo CMS's específicos para áreas como mídia (áudio e vídeo) e notícias (textos e imagens) ou ainda aqueles genéricos que podem ser utilizados em qualquer área, agregando-se ao sistema básico, componentes e módulos adicionais de acordo com as necessidades apresentadas. Entretanto, basicamente um CMS é composto das seguintes partes:

• Gerenciador de conteúdo que permite a manutenção de conteúdo de vários tipos categorizados dentro da aplicação de alguma forma e que fornece diferentes formas de visualização;

• Gerenciador de usuários que permite a administração tanto dos usuários que acessam o sistema com o intuito de obter informações quanto dos administradores e mantenedores do sistema, divididos em categorias com níveis de permissões diferentes se assim for necessário;

• Gerenciador de mídia que permite a execução de tarefas básicas de inclusão e remoção de arquivos de mídia, tais como imagens, sons e vídeos;

Com estas partes básicas já é possível a publicação de conteúdo informativo na Internet com alguma organização e verificação de acesso. Entretanto, ainda é muito pouco para as necessidades hoje existentes dos usuários que precisam de, por exemplo, álbuns de fotos, blog's, download de arquivos, páginas em diversos idiomas, enquetes e assim por diante. Para atender estas necessidades, a grande maioria das ferramentas de CMS permite sua expansão mediante pequenos componentes ou módulos responsáveis por tarefas específicas, mas totalmente integrados ao framework principal. Assim, é possível ter gerenciadores de banners, enquetes, pesquisas e até sistemas de carrinhos de compras, transação eletrônica, controle de notas e trabalhos escolares e o que mais o desenvolvedor ou seu cliente desejem.

As opções existentes

Desde os mais simples até os mais completos, existem dezenas de opções (livres ou não) de CMS's. Alguns são específicos para determinados segmentos como, por exemplo, o Moodle que é um poderoso gestor de conteúdo para ensino à distância ou ainda o osCommerce voltado para comércio eletrônico. Outros podem ser chamados de "genéricos", atendendo as mais diferentes necessidades, como é o caso do Mambo, um dos mais conceituados e antigos CMS livres.

Para escolher uma dentre as dezenas de opções (só listando as baseadas em PHP e livres), o desenvolvedor deve levar em consideração alguns pontos importantes que podem impactar futuramente em todo o seu trabalho

de manutenção e customização. Antes, uma pergunta deve ser respondida: "qual é a minha necessidade de gerenciamento de conteúdo?"

A ferramenta certa para o trabalho

Este é um dos aspectos que leva a maioria dos desenvolvedores a declinar do uso de um CMS e partir para a criação de sua própria solução. Como a maior parte desconhece a enorme variedade de ferramentas disponíveis, imaginam que não existe uma solução que atenda a maioria dos quesitos daquilo que vai desenvolver. Um engano que pode levá-lo a horas adicionais de programação desnecessárias e, muitas vezes, a uma solução de eficiência duvidosa.

Desta forma, o primeiro passo é responder a questão do ponto anterior que pode ser resumida em uma simples opção. Se a necessidade é criar uma loja na Internet, utiliza-se gestores específicos para o gerenciamento de comércio eletrônico. Se a necessidade é criar um blog, parte-se para ferramentas destinadas a este segmento. Então, temos as seguintes categorias (não somente elas):

• Portais – a opção de "faz tudo" em CMS's. Com eles, é possível criar portais dos mais simples aos mais complexos;

• E-commerce – voltados à gestão de lojas eletrônicas na Internet que, em sua maioria, já possuem controles de estoque e clientes, sistemas de gerenciamento de frete e envio, além de módulos de pagamento como cartões de crédito e PayPal;

• Blog – gestores voltados à criação de blog's com vários módulos adicionais;

• E-learning – voltados ao ensino à distância, disponibilizando uma plataforma de aprendizagem eficiente;

Além destas categorias, existem outras dezenas com várias opções e que, dificilmente, não irão atender a necessidade do desenvolvedor que, ao invés de criar toda a aplicação, será responsável somente pela customização do framework básico de acordo com suas necessidades. Ainda assim, além desta pergunta respondida, alguns outros pontos devem ser observados, tais como:

• Maturidade e equipe do projeto – um CMS que é mantido por somente um desenvolvedor pode ser considerado "pessoal" e não deve inicialmente ser usado como ferramenta de produção. Futuramente, podem ocorrer mudanças no humor deste desenvolvedor e ele, de uma hora para outra, resolver encerrá-lo e o usuário ficará literalmente na mão. Paralelamente, um projeto com anos de vida normalmente conta com vários desenvolvedores e uma história de sucesso por trás;

• Ser expansível – o principal conceito das ferramentas de CMS é serem expansíveis, permitindo o adicionamento de novos módulos com o intuito de fornecer novas funcionalidades. Um CMS deve permitir este tipo de expansão tanto para módulos originais do projeto quanto de terceiros;

• Easy-to-use – o CMS deve ser fácil de usar, tanto pelo administrador quanto pelos usuários. Neste ponto podem (e devem) ser incluídos pontos conto editores de texto visuais, facilidade na alteração dos temas (templates) e também a internacionalização, ou seja, a possibilidade de uso em vários idiomas (inclusive simultaneamente).

Algumas opções

Obviamente, a escolha de uma ferramenta não é somente uma questão de tecnicidade, mas também de gosto e adaptação. Mesmo assim, algumas dicas sempre são bem-vindas, principalmente, para aqueles que ainda não as conhecem e gostariam de experimentar algumas delas. A seguir, estão algumas das opções baseadas em PHP mais conhecidas e mais bem conceituadas do mercado.

Drupal – um dos mais antigos e robustos CMS's da atualidade. Com dezenas de desenvolvedores trabalhando em seu código e centenas de módulos externos, é uma opção muito interessante para aqueles que precisam de uma ferramenta multitarefa;

eZPublish – mantido pela empresa eZ Systems da Noruega e também por uma grande comunidade internacional (inclusive no Brasil), o eZPublish é um E-CMS (Enterprise CMS) com um apelo fortemente corporativo e funcionalidades que permitem atender a grande maioria das necessidades de usuários corporativos ou não.

Joomla! - derivado do Mambo, ele é uma ótima opção de CMS para portais, corporativos ou não, e conta com uma vasta rede de desenvolvedores dentro de sua

comunidade. No ano passado, foi escolhido como "O melhor CMS" pela Packt Publish, editora técnica inglesa.

Mambo – desenvolvido desde 2000 o Mambo possui vários prêmios mundiais que atestam sua confiabilidade, robustez e segurança e conta com mais de 1200 módulos adicionais que permitem sua caracterização em um sem número de aplicações;

Moodle – reconhecida como a melhor ferramenta open source de ensino à distância, o Moodle é utilizado por universidades de dezenas de países (inclusive o Brasil) com a mais alta qualidade;

osCommerce – uma das mais conceituadas soluções open source para comércio eletrônico da atualidade. Contanto com uma grande comunidade ao seu redor, ele permite a criação de lojas virtuais em um curto espaço de tempo e com pouca customização por parte do desenvolvedor.

Estas não são as únicas soluções existentes, mas servem para balizar as primeiras escolhas dos usuários. Entretanto, se precisar de maiores informações sobre estas e outras opções, vale uma passada em dois sites interessantes: CMS Matrix e OpenSourceCMS. O primeiro permite a escolha de vários CMS's diferentes e a comparação entre eles em uma matriz de dados. Algo extremamente útil quando desejamos um comparativo com as várias opções existentes. Já o segundo disponibiliza o teste de várias soluções, sem a necessidade de instalar ou configurar qualquer arquivo, permitindo assim o contato tanto com a interface do cliente quanto com a interface administrativa. As duas em conjunto permitem que as escolhas sejam feitas sob critérios técnicos e visuais sem a intervenção de marketing ou qualquer tipo de opinião de vendedores.

Finalizando

Soluções de CMS são, sem dúvida, uma "grande mão na roda" para qualquer desenvolvedor que necessita criar um site rapidamente, mas com muita qualidade. A maioria delas possuem comunidades ativas ao redor dos projetos das ferramentas, o que permite que quaisquer problemas ou dúvidas possam ser sanados rapidamente em fóruns e listas de discussão.

Ao contrário do que alguns podem argumentar, estas ferramentas não apagam a criatividade do desenvolvedor ou jogam-na para segundo plano. Customizar um CMS é tarefa muitas vezes tão complexa quando o desenvolvimento de uma aplicação completa, pois elas não operam milagres, sendo, portanto, necessária esta intervenção do desenvolvedor para que o produto final seja fiel àquilo que é desejado.

No próximo projeto, verifique as opções de CMS's disponíveis. Com toda a certeza, uma delas irá lhe atender em quase tudo o que deseja.

Sobre o autor:

Paulino Michelazzo é desenvolvedor web desde 1995 em diversas linguagens para Internet e assíduo pesquisador de novas tecnologias. É diretor mundial da Mambo Foundation, entidade responsável pela manutenção do CMS Mambo em todo o mundo e desenvolve websites em vários frameworks diferentes.Atualmente, mora na capital do Timor Leste, Dili, onde ocupa o cargo de Systems Development Specialist dentro do projeto UNV das Nações Unidas, desenvolvendo aplicações em PHP no Ministério da Justiça daquele país.

Referências e links sugeridos

Drupal - http://www.drupal.org
eZPublish - http://www.ez.no
Mambo - http://www.mambo-foundation.org
Joomla! - http://www.joomla.org
Moodle - http://www.moodle.org
osCommerce - http://www.oscommerce.com
CMS Matrix - http://www.cmsmatrix.org

Padrões de Projeto

(Design Patterns) em PHP da Teoria a Prática

Por Alexandre Altair de Melo

O presente artigo tem como objetivo demonstrar o uso de padrões de projeto em php e o quanto essa técnica pode trazer de ganhos ao profissional que a utiliza. Vale ressaltar, no entanto, que esta é somente uma das boas práticas de programação que podem ser usadas num projeto. Existem outras que podem vir a ser assuntos de artigos futuros como, por exemplo, a RUP.

Antes de iniciar o artigo em si, acompanhe a história a seguir. A personagem principal se chama AP (Anti Pattern). Esta história é um extremo do que pode acontecer quando somos apenas eficientes na resolução de um problema, mas não eficazes.

Segue a história de AP:

"Para mim, até bem pouco tempo atrás, o desenvolvimento, seja web ou não, era composto das seguintes etapas: fechar um escopo de projeto (nada de requisitos, somente algumas linhas gerais), um prazo e fazer o mesmo à escovação dos bits[1]. Estava feliz fazendo meu espaguete misturando código, html e tudo mais no mesmo lugar. Sem maiores preocupações. Prazo era um dos pontos que menos me preocupava, pois, ao meu ver, era uma saída, só para não ter o cliente no meu pé falando:

- E aí quando você vai me entregar o projeto? E não algo para se levar a sério. Geralmente só voltava a pensar no prazo quando ele já estava no fim. Tanto que eu nem precisava me preocupar em lembrar, tinha a minha agenda pessoal, o "cara" que paga as contas:

- Então semana que vem vamos começar a implantar? O prazo! Lá me via, então, passando noites em claro a base de café. Teve o lado bom, foi a época em que mais lucrei com ações das empresas produtoras de café. Brincadeiras à parte, o importante é que o projeto era entregue e eu recebia por ele. E o mundo continuava a girar.

Vinha outro cliente e pedia a mesma coisa e lá estava eu, novamente, no velho e bom espaguete. Não sabia, mas a época também usava um pattern só que não da POO[2] mais da POG[3], o nome deste era RCP Pattern que significa: Reuse by Copy-and-Paste (Reuso por copiar e colar). O RCP dita que, na pressa, quando não dá pra fazer a coisa por herança, basta copiar e colar, quantas vezes forem necessárias.

E assim foi até que se fez a luz. Eu acho que baixou o nível de café no meu sangue. Só porque faltaram algumas regras de negócio em um ou outro projeto. Hora, francamente... rodou, então funciona. Para que levantamento de requisitos? O pattern RCP sempre funcionou, é verdade que alguns erros só apareciam no cliente. Mas e daí? Eu entreguei no prazo graças ao RCP e isso, acredito, é qualidade.

O fato era que esta mudança ocorreu enquanto eu navegava na web, por falta de trabalho. Visitei um site (o último do resultado de um buscador) que descrevia técnicas. Até então, só tinha ouvido falar na AOO, POO e Design Patterns. Após estudo das mesmas, abandonei o RCP e a POG. Agora minhas consultas sql viraram uma classe que faz o mesmo várias e várias vezes, e o melhor em vários projetos. Os trabalhos voltaram aparecer. E o prazo? Bem, geralmente, agora sou eu que ligo para meus clientes marcando a implantação."

Reza a lenda que AP trocou o nome para UP (Utilizo Patterns) e que descobriu a vida fora do escritório. Vamos então apresentar os conceitos que mudaram a história de AP.

Introdução um pouco de conceito

Afinal, o que é um padrão de projeto? De forma genérica pode-se dizer que é uma solução aplicada a um problema. Assim, todas vezes em que você se deparar com o problema que tenha as mesmas características, aplica-se essa solução. Aqui está o ponto chave entre ser eficiente ou eficaz. Isso nos leva a uma pergunta: de onde surgiram os padrões de projeto?

A história dos padrões de projeto

Respondendo a pergunta da seção anterior, originalmente, os padrões de projeto não eram aplicados à informática. Foram soluções propostas para problemas de arquitetura por Christopher Alexander. Só posteriormente, a Gang of Four GOF (gangue dos quatro, tradução livre), formada por Erich Gamma, Richard Helm, Ralph Johson, John Vlissides, aplicou os conceitos à informática dando origem aos 23 padrões clássicos. Segundo um dos criadores dos padrões, Erich Gamma, um outro conceito para os padrões seria o seguinte: "Os padrões de projeto são descrições de objetos que se comunicam e classes que são customizadas para resolver um problema genérico de design em um contexto específico". Eles basicamente fizeram o que o nosso amigo AP fez posteriormente. Viram que no desenvolvimento de vários projetos às vezes se tinha sensação de "eu já vi isso" durante a resolução de um problema. Porém, ao invés de tentar replicar código, catalogaram as soluções para posterior aplicação diante do mesmo problema.

Por que, então, utilizar padrões de projeto?

A melhor pergunta seria: por que não usar? Os padrões são maneiras comprovadas de resolver um determinado problema sem ter de repensar em uma solução toda vez. Lembre-se da história de AP.

Abaixo alguns motivos para incentivar seu uso:

· Melhorar suas habilidades em programação, usando os conceitos de orientação a objetos, tendendo, assim, a melhorar a sua clareza de raciocínio.

· Desenvolver software sem duplicação de código, diminuindo, assim, o tempo de manutenção do mesmo.

· Facilitar a documentação e aprendizagem de conceitos sobre orientação a objetos.

Apresentando os padrões

Como não será possível descrever cada um dos 23 padrões detalhadamente, vamos nos ater no padrão singleton e no conceito de simplefactory, que serão descritos na prática. Abaixo, estão listados os nomes dos 23 padrões clássicos, conforme descrito no livro Padrões de Projeto Erich Gamma et al. e como estes padrões clássicos foram divididos, conforme as tarefas para qual se destinam.

1. Criação (Factory Method, Abstract Factory, Builder, Prototype, Singleton);

2. Estrutura (Class Adapter, Object Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy);

3. Comportamento (Interpreter, Template Method, Chain of Responsibility, Command, Iterator, Mediator, Memento, Observer, State, Strategy, Visitor);

O Padrão Singleton

O padrão singleton talvez seja um dos padrões mais simples quanto a sua implementação, porém é muito útil quanto ao seu propósito. Conceitualmente, este padrão se propõe a garantir que um objeto tenha uma única instância e a promover algum ponto global de acesso a ela. Você pode se perguntar: mas quando utilizar? Antes de responder a pergunta, será mostrada a representação UML[4] da classe que implementa o padrão (figura 1) e a implementação do mesmo no PHP 5 (figura 2). O mesmo conceito pode ser aplicado no PHP 4, porém o código deverá ser alterado para respeitar as regras de sintaxe da versão.

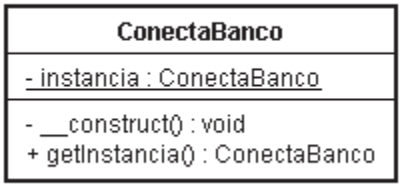

Figura 1 – Modelo UML descrevendo o padrão Singleton

O código mostrado na figura 2 é um exemplo de onde poderia ser empregado o padrão singleton, neste caso em específico, para uma classe de conexão com banco de dados. O acesso à instância da mesma fica encapsulado em único ponto e toda vez que você precisasse fazer algo com o banco chamar-se-ia uma única instância. E o objeto retornado faria toda a comunicação com o banco. Mas, como chamar o objeto em

qualquer lugar? Você poderia guardar o mesmo dentro de uma sessão PHP. E recuperá-lo sempre que fosse necessário utilizá-lo.

```php
<?php

/**
 * Classe para se conectar com banco de dados.
 */
class ConectaBanco{

    /**
     * Propriedade flag utilizada para verificar se já
     * existe uma instância do objeto.
     */
    private static $instancia = null;

    /**
     * Construtor.
     * Aqui está a grande sacada o construtor da classe é privado.
     * Assim só um outro método interno da classe pode ter acesso,
     * a este construtor.
     *
     * @return void
     */
    private function __construct() {}

    /**
     * Método que irá fornecer o ponto global para se
     * ter acesso a instância.
     *
     * @return ConectaBanco
     */
    public function getInstancia() {
        if (is_null(ConectaBanco::$instancia)) {
            ConectaBanco::$instancia = new ConectaBanco();
        }
        return ConectaBanco::$instancia;
    }
}

?>
```

Figura 2 – Classe em PHP 5 implementando o padrão Singleton

O Padrão Factory e o Conceito de SimpleFactory

O padrão factory, como o próprio nome diz, é uma fábrica objetos. Este padrão é utilizado quando há a necessidade de se ter um ponto em comum para criação de objetos. Geralmente, a decisão de criação de um objeto é tomada em tempo de execução, com base em algumas diretivas. Vamos tentar exemplificar da seguinte maneira: imagine que sua aplicação é internacional e precisa lidar com representação monetária. Cada país tem sua própria regra para separação decimal e de milhar. Então, você tem o padrão monetário Brasil, Austrália, ou tantos quanto a sua aplicação trabalhar. Cada objeto monetário sabe como lidar com seus valores monetários e a formatação destes. O problema é qual objeto instanciar quando determinado usuário acessa a sua aplicação. Simples: use o conceito de SimpleFactory. A figura 3 mostra o modelo UML da classe. O código da figura 4 demonstra o conceito de fábrica simples em PHP 5. Ele não é um padrão em si, porém ilustra a necessidade de se instanciar objetos conforme a demanda. Além disso, para que fosse possível trabalhar o conceito dos padrões factory method e abstract factory, seria necessário apresentar algumas teorias de Orientação a Objetos como interfaces e delegação, o que não é objetivo deste artigo. Você agora pode perguntar: e se eu precisasse trabalhar com o padrão monetário russo? Simples altere a classe para retornar um padrão monetário russo. Assim, sua aplicação teria um único ponto para instanciar suas classes de padrões monetários. E não um monte de classes sendo instanciados em vários lugares. Reduzindo o tempo de manutenção e adaptabilidade do seu sistema.

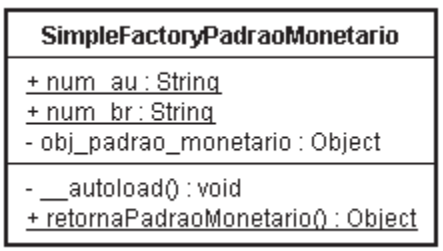

Figura 3 – Modelo UML descrevendo o SimpleFactory

```php
<?php

final class SimpleFactoryPadraoMonetario {

    //Atributos estáticos do objeto
    public static $num_au = "NUM_AU";
    public static $num_br = "NUM_BR";

    //Atributo que irá receber o padrão monetário
    private $obj_padrao_monetario;

    /**
     * Método mágico do php que carrega as classes conforme
     * a necessidade.
     *
     * @return void
     */
    private function __autoload($str_nome_classe) {
        require_once $str_nome_classe . 'php';
    }

    /**
     * Padrão factory simples que retorna a classe conforme
     * o tipo passado.
     *
     * @param $str_tipo_padrao tipo do padrão monetário
     * @return object retorna um objeto do tipo padrão monetário
     */
    public static function retornaPadraoMonetario($str_tipo_padrao) {
        switch ($str_tipo_padrao) {
            case self::$num_au:
                $this->obj_padrao_monetario = new PadraoMonetarioAustralia();
                break;
            case self::$num_br:
                $this->obj_padrao_monetario = new PadraoMonetarioBrasil();
                break;
        }
        return $this->obj_padrao_monetario;
    }
}

?>
```

Figura 4 – Classe em PHP 5 implementando o SimpleFactory

Considerações finais

O uso de padrões de projeto constitui uma ferramenta poderosa no desenvolvimento de sistemas, juntamente com outras técnicas de modelagem, análise, etc. Estas metodologias auxiliam o desenvolvedor a entregar realmente o que deve ser feito, com um menor esforço possível. Um ponto, porém, que sempre fica a quem se inicia numa técnica nova, como os padrões de projetos, é saber quando usar tal ou qual padrão? Não existe uma resposta pronta para isso. Somente com a experiência é que você irá adquirindo confiança para saber quando usar tal ou qual padrão. Que isto não sirva de desmotivação para deixar os padrões de fora de seus projetos, mas de desafio. Falta dizer que o artigo aqui apresentado, bem como os exemplos demonstrados, constitui um guia inicial e não uma solução definitiva de como você pode usar padrões em suas aplicações. Então, sucesso em seus projetos e até a próxima.

[1] Aqui cabe uma observação, para o desenvolvimento de projetos de software, o uso de uma metodologia de desenvolvimento como o RUP, Análise Estruturada, etc. É muito bem-vinda, para não dizer indispensável. Aqui foi descrita uma situação extrema que quase nunca vemos num projeto, ou seja, a falta de planejamento, de levantamento de requisitos, prazo, documentação e por fim a replicação de código. Qualquer semelhança com a realidade é mera coincidência. A idéia desse artigo é levar a uma reflexão das melhores práticas via patterns e aplica-las no seu dia a dia.

[2] POO: Programação Orientada a Objetos. Abordagem usada em programação para abstrair algo do mundo real, para um modelo computacional através de objetos.

[3] POG: Programação Orientada a Gambiarras. Paródia sobre a metodologia POO pode ser conferida em http://desciclo.pedia.ws/wiki/Programa%C3%A7%C3%A3o_Orientada_a_Gambiarras

[4] Unified Modeling Language (Linguagem padrão de Modelagem, tradução livre) modelo para notação de diagramas, proposto por Grady Booch et al.

Sobre o autor:

Alexandre Altair de Melo é bacharel em Sistemas de Informação pela UNIVILLE. Pós-Graduando em Gerenciamento e Planejamento Estratégico PUCPR. Possui 6 anos de experiência em desenvolvimento de sistemas (Delphi, VB, JAVA).
Atua desde 2001 com desenvolvimento WEB. Trabalha com PHP, JAVA, ASP e ASP .NET, Firebird, Interbase, MySQL e Oracle. Possui as certificações: Java – SCJP 1.4 e PHP – ZCE 5.0.
Desenvolve também atividades de treinamento em informática e discussão de idéias no blog http://www.ltsolucoes.com.br/blog

Referências e links sugeridos

[Eduardo Bezerra] – Princípios de Análise e Projetos de Sistemas com UML. Ano: 2002. Editora Campus.
[Eric Freeman et al] – Use a Cabeça! Padrões de Projeto (Design Patterns). Ano: 2005. Editora Alta Books.
[Erich Gamma et al] – Padrões de Projeto. Ano: 2000. Editora Bookman.
[Helder da Rocha] – GoF Design Patterns em Java. http://www.argonavis.com.br/cursos/java/j930/index.html.
[Jason E. Sweat] – php|architect's Guide to PHP Design Patterns - Ano: 2005. Editora NB.

Integração de sistemas

utilizando webservices baseado na tecnologia SOA

Por José Walter Pereira Dutra

Este trabalho descreve o funcionamento de integração de sistemas utilizando Webservices. Tendo como ótica a Arquitetura Orientada a Serviços (SOA), demonstra a sua utilização prática, seus componentes e como esta nova proposta de arquitetura pode ser utilizada na criação de novos negócios e nas soluções de problemas de integração, utilizando recursos de software livre que proporcionam resultados a baixo custo.

Introdução

A integração de sistemas vem sendo um dos principais problemas enfrentados pelas empresas quando se trata do controle das informações e da obtenção de recursos estratégicos que propiciam o crescimento frente a um mercado cada vez mais competitivo. Com o crescimento corporativo desordenado e com a necessidade de resolver problemas pontuais, muitas empresas foram obrigadas a tomar medidas imediatas quanto à organização de suas tecnologias culminando em uma quantidade de dados setoriais descentralizadas, e, com isto, na dificuldade de obtenção de informações estratégicas gerais da empresa que proporcionasse a tomada rápida de decisões.

A integração de sistemas.

A Integração de Aplicações Corporativas (EAI) é uma tecnologia que propicia a troca de informações entre diferentes aplicações, podendo estar desenvolvida em linguagens ou plataformas diferenciadas, estando localmente ou remotamente localizadas.

As soluções de integração – Arquitetura de software baseada em componentes

Muitas soluções foram tentadas com o objetivo de resolver o problema de integração, de forma a impactar pouco nos processos que já estavam em andamento nas empresas. Muitos já vislumbravam esta oportunidade de negócio, tentando com sucesso questionável e quase sempre se deparando com os problemas estruturais e de segurança. Sem falar dos grandes custos desprendidos pelas empresas, frente ao resultado que estas tecnologias proporcionavam.

A principal tecnologia utilizada, conhecida como middleware, consiste basicamente em uma camada de programa sendo executado entre duas aplicações, mediando a conversação de mensagens enviadas através de objetos distribuídos. Tecnologias como CORBA, COM/DCOM e RMI foram amplamente veiculadas e implementadas como a solução do problema, mas surgiram limitações causadas pela falta de interoperabilidade entre diferentes plataformas e a dificuldade em atravessar firewalls/proxys exigidos com a necessidade de segurança causada pela abertura dos sistemas pelas corporações, principalmente com acesso pela Internet.

A tecnologia de webservices

Para entender a integração de sistemas baseada em webservices, é preciso entender como funciona, na prática, esta tecnologia.

Um Webservice (ou serviço Web) consiste na disponibilização de funções ou objetos remotos através de protocolo HTTP a serem acessados por aplicações em ambiente corporativo de intranet ou em ambiente externo à empresa, a Internet. Possuem uma arquitetura baseada na interação de três

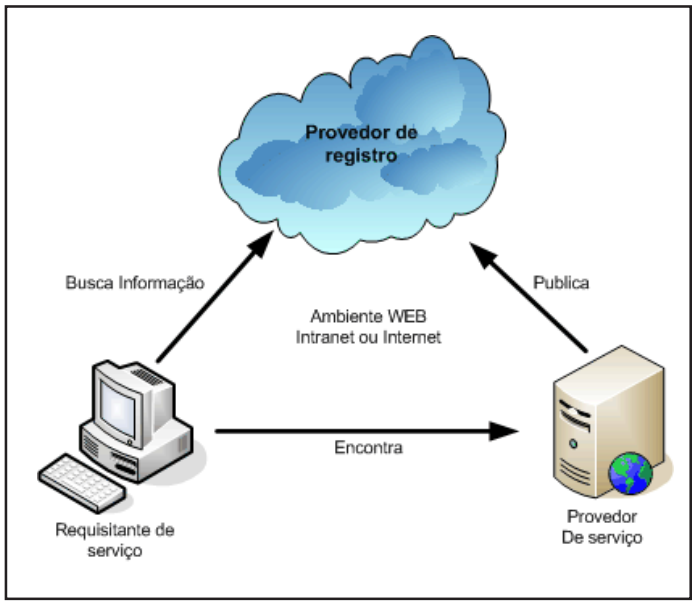

Figura1. Ambiente de webservices

categorias: provedor do serviço (service provider), provedor de registro (registry provider ou registry broker) e o cliente do serviço (service requestor).

Estas categorias envolvem as operações de procura (find), publicação (publish) e acoplamento (bind) do serviço. O provedor publica o serviço com a operação publish. O cliente utiliza a operação busca (find) para obter uma descrição de serviço do provedor de registro e usa esta descrição para, dinamicamente, acessar e interagir (bind) com o provedor do serviço.

Na figura 1, é apresentado um ambiente de webservices com o provedor de registro, que pode ser público ou privado. Público, quando disponibiliza serviços para quem queira acessar e usar, são os casos de notícias públicas, previsões de tempo e outros, e privado no caso de serviços transacionais entre empresas de interesses comuns, são os brokers de corretoras, bolsa de valores e outros.

Estrutura de webservices

Os recursos para a disponibilização de uma estrutura webservice podem ser de alto custo quando feita a opção por tecnologias proprietárias como Microsoft, Borland e outras, mas é possível disponibilizar webservice a baixíssimo custo usando apenas tecnologias open source, fazendo com que seja acessível a quem queira usar, com gastos apenas com hardware e recursos do conhecimento humano.

Algumas tecnologias que suportam o uso de webservices:

· Proprietárias: Microsoft, Borland, IBM.

· Open source: Java, PHP, Ruby e outras

Os webservices são baseados em 3 tecnologias (todas tem como base o XML)

· UDDI [11]– Universal Description, Discovery and Integration)

Protocolo de comunicação para registros. Permite a publicação de serviços em um diretório on-line de modo que estes fiquem acessíveis a partir de qualquer ponto da Internet.

· WSDL [5] – Web Service Description Language

É um documento escrito em XML que especifica e permite descrever ou localizar um Web Service. Contém a localização do serviço e das operações (ou métodos) que o serviço oferece, bem como as regras que devem ser obedecidas para que a troca de mensagens funcione (especificações das requisições a ele enviadas e das respostas que ele envia ao requisitor)

· SOAP [4]– Simple Object Access Protocol

Protocolo de comunicação projetado para invocar aplicações remotas através de RPC ou trocas de mensagens, num ambiente independente da plataforma e linguagem de programação.

Funcionamento de webservice

Componente	Função
Fornecedor de serviços	Disponibiliza o serviço através da rede e publica o serviço em um broker para que seja encontrado
Cliente de serviços	Solicita serviços a um broker e liga-se a um fornecedor de serviços fazendo requisições.
Broker de serviços	Proporciona a localização e disponibilização de serviços de fornecedores a clientes requisitores

Tabela 1. Componentes de um webservice

Basicamente, o funcionamento pode ser explicado conforme as seguintes etapas: o usuário faz a requisição de uma página que é solicitada a um servidor Web contendo uma aplicação. Para montar a página requisitada, a aplicação faz uma requisição a um

servidor de serviços protegido por um firewall/proxy (após ter encontrado o serviço em um broker e ter obtido as regras de comunicação) enviando a solicitação e a requisição de autenticação em uma conexão "encriptada" segura. Uma vez autenticado, o serviço é acionado e, por sua vez, faz uma requisição a um banco de dados que pode estar no mesmo servidor de serviços ou não. Após obter a resposta do banco de dados, o servidor de serviços retorna as informações à aplicação requisitante, que as usa para montar a página solicitada e entregar ao terminal que a requisitou através do browser.

O webservice como solução de integração de sistemas

De acordo com as definições da ANSI/IEEE, uma arquitetura de software trata basicamente de como os componentes fundamentais de um sistema se relacionam intrinsecamente e extrinsecamente. Partindo deste conceito, os webservices propiciam a arquitetura orientada a serviços (*SOA – Service Oriented Architecture*) que tem o conceito de serviços como seu componente fundamental.

Como foi discutido anteriormente nos conceitos de funcionamentos de webservice, é notável a sua versatilidade no provimento deste tipo de arquitetura, que na verdade não há nada de tão novo, pois o ponto fundamental de um webservice é a conversação entre aplicações.

Webservices não foram criados para comunicação com seres humanos, mas sim entre aplicações servidoras e clientes que podem interagir diretamente com seres humanos ou não.

Fazendo um comparativo com arquiteturas orientadas a objetos que tem como característica fundamental a troca de informações entre objetos e componentes, a arquitetura orientada a serviços (SOA) faz este trabalho integrando aplicações por meio de serviços.

Características relevantes em arquitetura orientada a serviços

Reuso "Caixa-preta"

O reuso de software surgiu inicialmente por uma necessidade de economizar recursos de hardware [Clements, 1995]. Com o tempo, esta necessidade foi se estendendo, principalmente com a urgência de criação de aplicações frente a um mercado cada vez mais competitivo e de grande demanda de controle de informação.

Distribuição

Os sistemas informatizados romperam as paredes da organização e agora os processos podem facilmente ser executados mais próximos de onde estão as fontes de informação. Um outro aspecto que deve ser mencionado é a mobilidade proporcionada pelo uso de PDAS que atendem a esta arquitetura.

Heterogeneidade Ambiental

Um provedor de webservices pode ser disponibilizado em várias linguagens e arquiteturas, tendo como clientes aplicações na mesma variedade, desde que obedeça aos padrões de protocolos definidos em sua arquitetura.

Robustez de Protocolos

Arquitetura orientada a serviços é um novo nome para um ambiente que já foi testado e consolidado em seu funcionamento. O uso de Webservices proporcionou de maneira rápida e consistente a sua utilização efetiva em aplicações corporativas.

SOA na prática – uma abordagem do funcionamento

Ambiente de acesso

Figura 3. Ambiente corporativo

No ambiente apresentado acima (figura 2), podemos notar que há uma grande versatilidade na integração de sistemas com uma diversidade de possíveis clientes que podem estar posicionados local ou remotamente. Os terminais podem ficar em plataformas diferenciadas de sistemas operacionais Windows, Linux, Mac ou outra que comporte aplicações, como PDAs, Laptops, estações de Internet ou estações de redes locais. Além disso, não é necessária a rigidez da estrutura do servidor, que pode estar implementado em PHP, Java, NET ou outra linguagem que suporte este recurso.

Ambiente corporativo

A figura 3 mostra um pequeno exemplo de como pode acontecer a integração de sistemas em uma empresa utilizando webservices. Apesar de estar descrito em um ambiente fechado, não impede que servidores estejam interligados remotamente pela Internet.

No caso há dois servidores de serviços, um para processamentos pesados, desonerando recursos do servidor de aplicações e tornando-o mais ágil para operações corriqueiras, e um para serviços compartilhados, possibilitando a reutilização de processos.

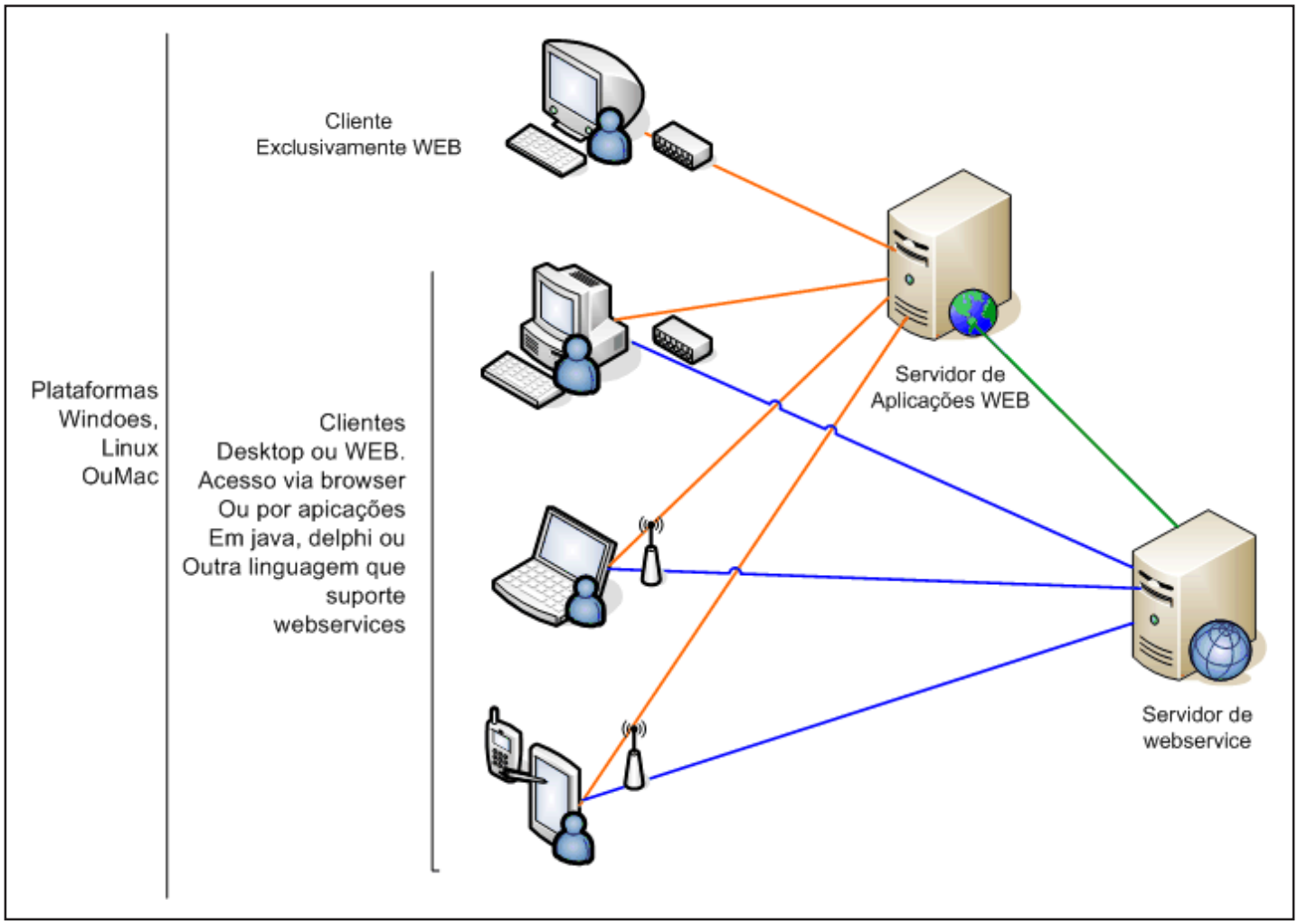

Figura 2. Ambiente de acesso

O setor de RH necessita emitir os contracheques de pagamento de funcionários, e o setor de Contabilidade necessita registrar os pagamentos efetuados aos funcionários, o servidor de serviços compartilhados provê estas informações através de webservice possibilitando estas operações.

Dependendo da quantidade de funcionários o processamento de cálculo da folha pode ser oneroso ao servidor de aplicações ou de serviços compartilhados, neste caso entra em cena um outro webservice que tem como função cuidar dos processamentos pesados, desonerando, assim, os outros servidores.

Conclusão

Na área de tecnologia de sistemas, é comum empresas desenvolvedoras promoverem durante um bom tempo uma queda de braço para definir qual padrão a ser adotado no mercado em soluções específicas, como aconteceu com o padrão tecnológico de integração de sistemas. Empresas como Borland, Microsoft e SUM procuraram durante um bom tempo ganhar o mercado com os seus padrões. Certamente, as tecnologias apresentadas até hoje oferecem ganhos e perdas para as empresas que se propõem a usá-las, mas depois da confusão, alguma coisa tem que acontecer para que a ordem seja restabelecida. A tecnologia do uso de webservices para a integração de sistemas se apresenta como uma solução para este dilema vivido pelo mercado. A sua base é bastante sólida e proporcionada pelo baixo custo de implementação, versatilidade, tecnologia atual baseada em Web, solidez de seus protocolos dentre outros. No entanto, o principal fator que mais beneficia a credibilidade dos webservices é o fato desta não ser uma tecnologia nova e não apresentar a mesma instabilidade inicial, como acontece em inovações. O mercado já conhece e aprovou, agora é só usá-la.

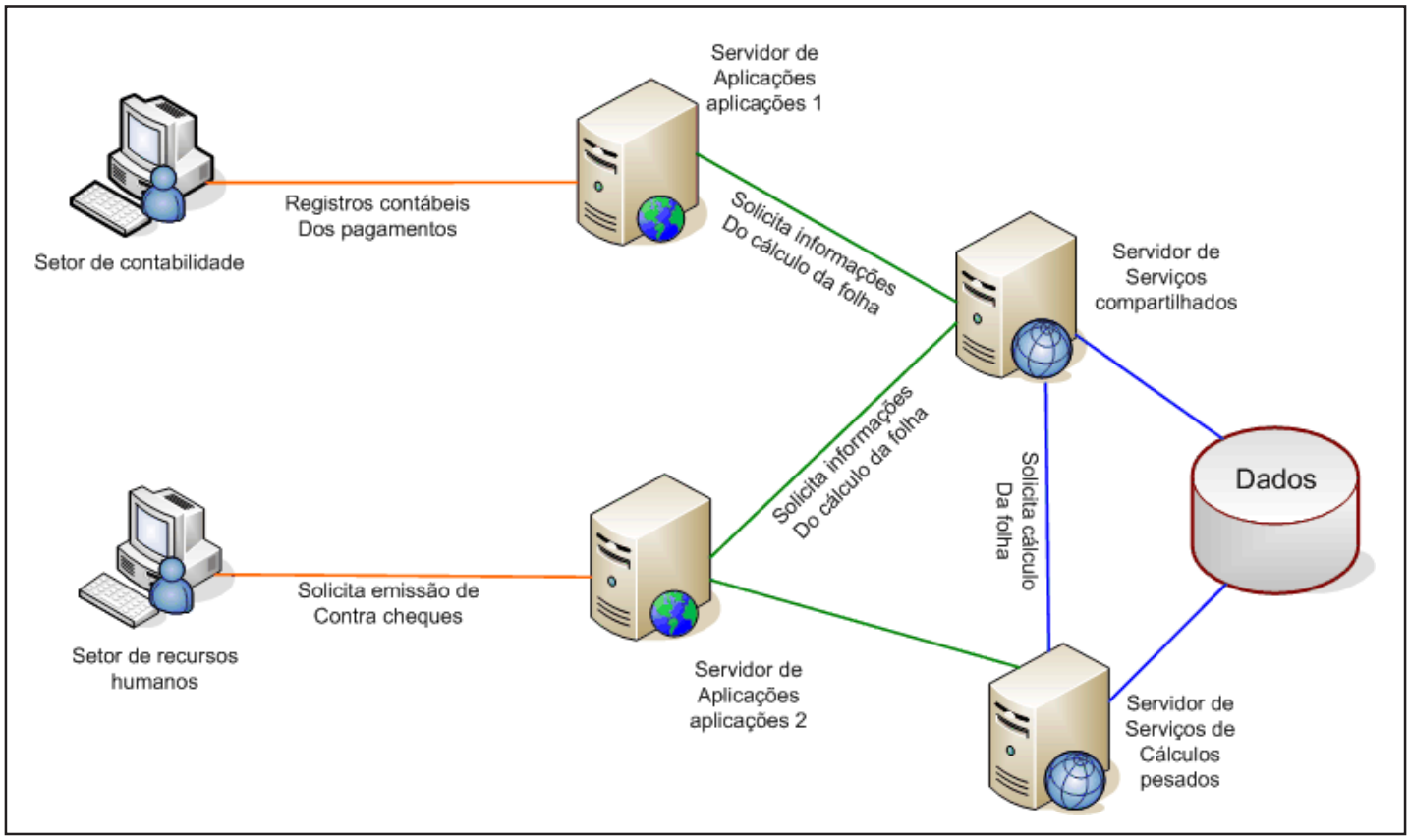

Figura 3. Ambiente corporativo

Sobre o autor:
José Walter Pereira Dutra é analista de sistemas, webdeveloper, Webmaster, graduado em Tecnologia de Desenvolvimento WEB pelo UNIBH. Atuando desde 1990 em trabalhos de hardware e software cliente/servidor e como analista, webmaster e desenvolvedor Web desde 1999. Conhecimento consolidado em ambiente Web com tecnologia Linux, desenvolvimento Web utilizando PHP, MySQL, PorgreSql, Ájax e desenvolvimento desktop cliente/servidor com Delphi.
Analista de sistemas no Grupo Sim, como responsável pelo Portal de conhecimento corporativo, implementações em Delphi e soluções corporativas utilizando tecnologias de software livre.

Referências Bibliográficas
[1] Fuller, J., Fuecks, H., Egervari, K., Waters, B., Solin, D., Stephens, J. and Reynolds, L., Professional PHP Web Services, 2003.
[2] Arquitetura Orientada a Serviço (SOA) http://www.ibm.com/br/products/software/info/topic/openenvironment/soa/
[3] Por dentro do SOA http://www.ibm.com/br/products/software/info/features/futureenterprise/index.phtml 2006, Junho
[4] SOAP version 1.2http://www.w3.org/TR/soap/
[5] Web Services Description Language (WSDL) 1.1 http://www.w3.org/TR/wsdl
[6] Campbell, S.D. (2002) Web Services with NuSOAP http://
www.zend.com/zend/tut/tutorial-campbell.php
[7] Nichol, Scott (2004) Programming with NuSOAP Using WSDL http://www.scottnichol.com/nusoapprogwsdl.htm
[8] Claudimir Zavalik, Guilherme Lacerda, José Palazzo M.de Oliveira, (2004) Implementando Web Services com Software Livre, http://palazzo.pro.br/artigos/04%20Software%20Livre%20-%20Web%20Serv.htm
[9] Greco, Gilnei Borges, Charão , Andrea Schwertner, Webservices uma alternativa para sistemas distribuídos
[10] UDDI - http://www.uddi.org/, 2006, Maio.
[11] Lopes, Carlos J. Feijó, Ramalho, José Carlos, (2004) Web Services: Metodologias de esenvolvimento https://repositorium.sdum.uminho.pt/bitstream/1822/559/1/LR04.pdf

Trabalhando com Sessões

Por Leandro Schwarz

Este trabalho se destina a demonstrar o funcionamento das sessões em PHP, permitindo que o leitor construa *websites* mais seguros e funcionais. Através do uso de sessões, é possível armazenar informações importantes do visitante do *website* e propagá-las por entre as páginas do *website* ou entre acessos em determinado período de tempo.

As sessões em PHP têm como principal função o armazenamento de dados no servidor. Em um *website* de comércio eletrônico, por exemplo, é importante armazenar as informações do usuário, como por exemplo, o *login*, o carrinho de compras, os produtos que o usuário visitou, dentre outros dados. O carrinho de compras e os produtos visitados normalmente são armazenados no banco de dados, no entanto, o *login* constitui uma informação importante que deve estar sempre disponível, pois os acessos ao banco de dados, normalmente, utilizam-no como chave na busca.

Com isso surge o problema de como transmitir o *login* por entre as múltiplas páginas do *website*, após a autenticação do usuário. Pode-se utilizar *cookies*, armazenando o *login* no computador do próprio usuário, entretanto, em computadores públicos ou computadores com alta segurança, o acesso aos *cookies* está normalmente bloqueado, impossibilitando o correto funcionamento do *website*.

Atualmente, a maioria dos portais de comércio eletrônico utilizam sessões para armazenar estes dados, permitindo que o usuário se autentique apenas uma vez. Alguns *websites* fazem uso de sessões e *cookies* em conjunto, armazenando os dados do usuário nas sessões e os dados da sessão no computador do usuário. Com isso é possível permitir que o usuário após ter acessado o *website* e se autenticado, desligue o computador, acesse o *website* um tempo depois e não precise se autenticar novamente. Este recurso está disponível nos provedores de *e-mail* famosos, como o GMail e o Hotmail, através da opção "salvar meu endereço de e-mail e senha".

Definindo sessões

Como já citado anteriormente, as sessões possibilitam o armazenamento de dados entre os acessos sucessivos às páginas de um *website*. Pode-se dizer que as sessões funcionam como *cookies* reversos, guardando os dados no próprio servidor ao invés de salvá-los no computador do usuário.

Quando uma sessão é iniciada, um número único é criado para caracterizar a sessão e um pequeno arquivo ASCII é criado no servidor. O número de identificação da sessão, chamado de ID, é uma seqüência de 16 caracteres alfanuméricos, expresso em sua forma hexadecimal (ocupando, portanto, 32 caracteres de comprimento), como por exemplo "9fff50c722e010d73bdf939ab2a239b1". O arquivo no servidor é criado por *default* no diretório temporário, com o nome do arquivo iniciando por "sess_" e seguido do ID da sessão, por exemplo "sess_9fff50c722e010d73bdf939ab2a239b1".

As variáveis são armazenadas nas sessões de acordo com a precedência, em forma de fila: a primeira variável a ser armazenada ocupa a primeira posição e assim por diante. Apesar de o PHP não trabalhar com variáveis fortemente tipadas, os dados são escritos em uma formatação característica nas sessões, de

acordo com o tipo da variável que se deseja armazenar.

```
$varstring = "meulogin";
$varinteira = 125;
$varreal = 2.5;
$varbool = true;
```

Neste caso, está se armazenando uma variável do tipo seqüência de caracteres, uma variável inteira, uma variável do tipo ponto flutuante e uma variável do tipo booleana. O arquivo no servidor apresentará uma seqüência de caracteres de uma linha, representando todas estas variáveis, como exemplificado a seguir.

```
varstring|s:8:"meulogin";
varinteira|i:125;
varreal|d:2.5;
varbool|b:1;
```

Cada variável foi separada em uma linha diferente para facilitar o entendimento, no entanto, o arquivo do servidor apresenta apenas uma linha com todas as variáveis em seqüência.

A formatação do arquivo segue uma ordem lógica: em primeiro lugar é escrito o nome da variável, no caso anterior "varstring", "varinteira", "varreal" ou "varbool"; depois um caractere barra vertical "|" delimita o fim do nome da variável; a seguir, um caractere indica o tipo de variável que foi armazenada na sessão "s" para seqüências de caracteres, "i" para variáveis inteiras, "d" para variáveis de ponto flutuante e "b" para variáveis do tipo booleana; na seqüência, os dois pontos ":" delimitam o fim do tipo da variável; caso a variável seja do tipo "s", então o número de caracteres da variável é escrito, seguido de outro caractere dois pontos; por fim o valor da variável é escrito, delimitado por um caractere ponto e vírgula ";". Note que se a variável for do tipo "s", seu conteúdo é escrito entre dois caracteres de aspas duplas ("").

Na verdade, este conhecimento todo não é necessário para se trabalhar com sessões, pois o próprio interpretador no servidor cria e gerencia as sessões e suas variáveis. Entretanto, o programador profissional não deve simplesmente resolver problemas, mas sim entender como funcionam as funções e os recursos que utiliza no seu trabalho. Dessa forma pode evitar que problemas ocorram e pode resolvê-los quando for impossível prevê-los.

Nas subdivisões seguintes, será estudado como trabalhar com as sessões através das funções do PHP específicas para o tratamento de sessões e a variável global $_SESSION.

Gerenciamento de sessões

Iniciando uma sessão

O PHP já possui, em sua distribuição padrão, funções para o gerenciamento de sessões através da variável global $_SESSION. Esta variável não existe antes da criação da sessão, sendo iniciada como um vetor em branco no momento que a sessão é criada.

A função **session_start()** é responsável pela criação de uma nova sessão ou pela abertura de uma sessão já iniciada. Uma observação importante é que a função **session_start()** deve vir antes de qualquer saída HTML. Observa-se muito em fóruns e listas de discussões, usuários reclamando que em seus *scripts* aparece uma mensagem de erro semelhante à demonstrada a seguir.

```
Warning: session_start(): Cannot send
session cache limiter - headers
already sent (output started at
/home/leomail/public_html/info.php:2)
in /home/pleomail/public_html/info.php
on line 3
```

Isto acontece, na maioria das vezes, porque o programador, inadvertidamente, incluiu caracteres de espaço, tabulação ou linhas em branco entre o início do arquivo e a tag "<?".

Passagem e recebimento de variáveis

É possível armazenar dados em uma sessão através de duas formas: pelo acesso direto à variável global $_SESSION ou indiretamente, através da função **session_register()**. A seguir, as três formas de se armazenar variáveis em uma sessão por meio dos dois métodos mencionados são demonstradas.

Para o melhor entendimento do *script* a seguir, a figura 1 mostra as variáveis globais e superglobais $_GET, $_POST, $_SESSION, $_FILES e $_COOKIE, além das

```
<?
session_start();

$variavel_01 = "string 01";
$variavel_02 = "string 02";
$variavel_03 = "string 03";

session_register('variavel_01');
$_SESSION['nome_var'] =
$variavel_02;
$_SESSION[] = $variavel_03;
?>
```

variáveis que foram criadas no *script*. Esta saída foi gerada através da função **show_vars()** da biblioteca **debuglib**, constituindo importante recurso, tanto para programadores recém-iniciados, quanto para aqueles com muitos anos de experiência. A biblioteca não faz parte da distribuição padrão do PHP, no entanto, pode ser distribuída e utilizada gratuitamente. O *download* pode ser realizado em http://www.atomar.de/.

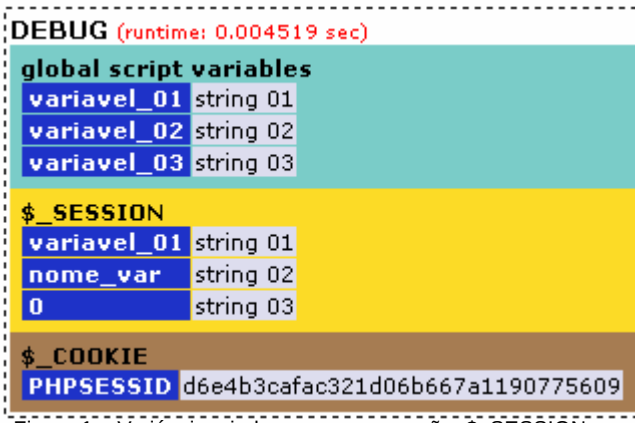

Figura 1 – Variáveis criadas por programação, $_SESSION e $_COOKIES.

Observe neste caso que as variáveis criadas $variavel_01, $variavel_02 e $variavel_03 foram passadas corretamente para a sessão. Há algumas diferenças entre os três métodos. No primeiro caso, a variável armazenada na sessão recebe o mesmo nome que a variável passada. No segundo caso, um novo índice no vetor $_SESSION é criado para armazenar o valor passado. Desta forma, é possível escolher o nome do índice e utilizar um nome diferente do colocado na variável passada. O terceiro caso utiliza a criação automática de índices no vetor $_SESSION, o que pode complicar um pouco a lógica da programação. A terceira forma não é muito indicada, pois requer muito cuidado do programador ao escolher a ordem de passagem e recebimento das variáveis.

Depois que as variáveis são armazenadas em uma sessão, seu valor pode ser retornado pelo acesso direto ao vetor $_SESSION, no entanto, é sempre uma boa idéia verificar se a variável existe na sessão antes de solicitá-la ao servidor. Isto pode ser feito pelo acesso direto ao vetor $_SESSION ou pela variável **session_is_registered()**. O *script* a seguir exemplifica o acesso às variáveis armazenadas na sessão.As variáveis que foram criadas com o script anterior podem ser visualizadas na figura 2.

A variável $variavel_01 foi criada e armazenada na sessão com o nome de *nome_var*. Logo após, esta variável foi recebida pela variável $variavel_02.

```
<?
session_start();

$variavel_01 = "Esta é uma string";
$_SESSION['nome_var'] = $variavel_01;

if(isset($_SESSION['nome_var']))
  $variavel_02 =
$_SESSION['nome_var'];
?>
```

Também é possível excluir uma variável dentro de uma sessão eliminando-se o respectivo índice do vetor $_SESSION ou utilizando-se a função **session_unregister()**.

As variáveis do script anterior estão listadas na figura 3. Observe

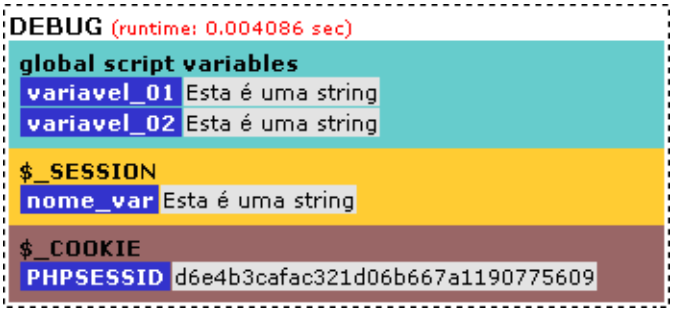

Figura 2 – Variáveis criadas no script anterior.

```
<?
session_start();

$_SESSION['nome_var1'] = "string_01";
$_SESSION['nome_var2'] = "string_02";

if(isset($_SESSION['nome_var1']))
  unset($_SESSION['nome_var1']);
?>
```

Figura 3 – Variáveis criadas no script anterior.

que a variável de sessão de nome *nome_var1* foi excluída da sessão.

Muito cuidado ao tentar limpar todas as variáveis da sessão com o comando **unset()**. Não utilize a instrução **unset($_SESSION)**, pois isto apagará o vetor inteiro e em qualquer tentativa subseqüente de armazenar variáveis na sessão retornará um erro.

Atenção: O uso das funções **session_register()**, **session_is_registered()** e **session_unregister()** não é recomendado, pois estas funções exigem que a variável de configuração **register_globals** esteja habilitada. Utilize o acesso direto ao vetor $_SESSION, para evitar erros deste tipo.

Destruição de sessões

Uma sessão pode ser destruída depois de utilizada. Isto é importante nos casos em que um usuário tentou acessar uma página de conteúdo autenticado, sem que a confirmação da autenticação fosse propagada. Também é possível destruir sessões quando se deseja criar uma nova e descartar todos os dados da sessão anterior. A destruição de sessões é feita com a função **session_destroy()**.

Em *websites* de comércio eletrônico, é comum destruir as sessões antes de iniciar uma nova sessão que conterá a autenticação do usuário, conforme demonstrado no *script* seguinte.

```
<?
session_start();
session_destroy();
session_start();

$variavel_01 = "Esta é uma string";
$_SESSION['nome_var'] = $variavel_01;

$variavel_02 = $_SESSION['nome_var'];
?>
```

A primeira instrução **session_start()** cria uma nova sessão ou reinicia a sessão anterior. A instrução **session_destroy()** fecha esta sessão e a exclui. No momento que a segunda instrução **session_start()** é executada, não há nenhuma sessão anterior para ser reiniciada, portanto, uma nova sessão é criada. Estas três instruções devem ser utilizadas desta forma apenas na autenticação do usuário onde o *login* e a senha são pedidos, pois a cada vez que **session_destroy()** é executada todas as variáveis são perdidas, sendo impossível sua recuperação posterior.

Tempo de vida das sessões

As sessões podem ser configuradas para terem um determinado tempo de vida. Depois de expirado este tempo, a sessão é automaticamente destruída pelo servidor.

A função **session_cache_expire()** é utilizada para retornar o tempo de expiração das sessões ou para definir um novo tempo de expiração em minutos. A cada chamada do *script*,

```
<?
session_cache_expire(25);

session_start();
session_destroy();
session_start();

$tempo = session_cache_expire();
?>
```

o valor do tempo de expiração retorna ao padrão (180 minutos), portanto, a instrução **session_cache_expire()** deve ser chamada novamente a cada página do *website*.

Outra observação importante é que o tempo de vida da sessão deve ser configurado antes da inicialização da mesma, precedendo a instrução **session_start()**. O exemplo anterior configura o tempo de vida da sessão em 25 minutos e depois armazena este valor na variável $tempo.

Limitador de cache

O limitador de *cache* controle os cabeçalhos HTTP que serão enviados e armazenados pelo usuário e pelos *proxies* intermediários. Assim, é possível definir diferentes níveis de segurança, permitindo ou negando o armazenamento destes cabeçalhos no *cache* do cliente e/ou dos *proxies* intermediários.

Os níveis do limitador de *cache* possíveis são *none*, *nocache*, *public*, *private* e *private_no_expire*. Se o limitador estiver definido como *none*, não há limites para o *cache*. O contrário ocorre quando o limitador é marcado como *nocache*, onde nenhum armazenamento é permitido no cliente e nem nos *proxies*. Na configuração *public*, o armazenamento em *cache* está permitido para ambos. Uma opção mais restritiva é a *private*, onde o armazenamento em *cache* é permitido para o cliente, porém negado para os *proxies* intermediários. A opção *private_no_expire* funciona de maneira semelhante à opção *private*, mas o envio de cabeçalhos expirados está desabilitado devido ao fato de alguns *browsers* apresentarem dificuldades ao trabalhar com cabeçalhos expirados.

A função **session_cache_limiter()** apresenta funcionamento semelhante ao da função **session_cache_expire()**, retornando ao valor padrão a cada chamada da página e também deve ser executada antes de qualquer chamada à função **session_start()**. O exemplo anterior foi modificado para configurar o limitador de *cache* para *nocache* e depois armazenar este valor na variável $cache.

```
<?
session_cache_expire(25);
session_cache_limiter('nocache');

session_start();
session_destroy();
session_start();

$cache = session_cache_limiter();
?>
```

Identificadores de sessões

Depois que os dados já foram armazenados, é importante saber como referenciar a uma determinada sessão de forma a receber as variáveis que foram gravadas na sessão em uma outra página. Imagine um *website* de comércio eletrônico onde o usuário se autentica em uma página e realiza as compras em outra. Neste caso, é necessário que as duas páginas abram a mesma sessão.

Como pode ser verificado nos exemplos anteriores, quando iniciamos uma sessão, um *cookie* é criado no cliente, se a configuração dele permitir *cookies*. Este *cookie* contém uma variável chamada **PHPSESSID**, cujo valor é justamente o ID da última sessão aberta. Desta forma, é possível para o PHP abrir a última sessão do cliente sem o envio do ID por parte do programador.

O ID da sessão aberta pode ser obtido por meio da função **session_id()**. Esse valor também pode ser conseguido com a constante **SID**. É possível propagar este valor por entre as páginas de um *website* através dos métodos **POST** e **GET**, ou simplesmente inicializar uma nova sessão e comparar com o ID da sessão anterior. É sempre uma boa sugestão "encriptar" o ID da sessão antes de enviá-lo. Enviar o ID de uma sessão, mesmo não estando "encriptado" é muito mais seguro do que enviar o *login* e/ou senha, mesmo que estejam "encriptados".

É possível também modificar o ID da sessão atual, sem se perder os dados contidos na sessão através da função **session_regenerate_id()**. Na versão 5.1.0 do PHP, a função pode aceitar um parâmetro booleano que determina se a sessão anterior deve ser excluída ou não. Esta função pode ser utilizada para que cada página acessada gere um ID de sessão aleatório, que será propagado pelo *website*, evitando que um ID capturado possa, de alguma forma, ser utilizado.

Uma vantagem muito útil para *framesets* é o emprego de várias sessões em um mesmo *script*. Os dados da sessão são armazenados imediatamente após o término do *script*, no entanto, quando se deseja trabalhar com mais de uma sessão por *script* é necessário solicitar o salvamento da sessão antes de se abrir

uma outra. Isto é realizado pela função **session_write_close()**. Neste caso, a cada vez que se desejar chamar uma nova sessão, deve-se executar primeiro a **session_write_close()**, para salvar a sessão.

O exemplo acima demonstra a utilização de sessões para validação de *login* e senha, além de armazenar o login do usuário e propagá-lo para as outras páginas do *website*. O nome da página é *pagina1.php*.

Este pequeno código abaixo realiza a validação do usuário comparando o *login* e a senha com as informações armazenadas no banco de dados. Na página *pagina2.php*, pode ser utilizado o código a seguir para verificar a validação da sessão.

```
<?
session_cache_expire(25);
session_cache_limiter('nocache');
session_start();
session_destroy();
session_start();
// Aqui a conexão com o banco de
dados deve ser aberta e o login a
senha devem ser adquiridos e
armazenados nas variáveis $login e
$senha
if(isset($_REQUEST['login']))
{
  if(($_REQUEST['login'] ==
$login)&&
     ($_REQUEST['senha'] == $senha))
  {
    $_SESSION['login'] = $login;
    header("Location: pagina2.php?
      session_id=" .
md5(session_id()
      ));
  }
}
?>
 CÓDIGO HTML DO FORMULÁRIO
```

```
<?
session_cache_expire(25);
session_cache_limiter('nocache');
session_start();
if(isset($_REQUEST['session_id']))
{
  if($_REQUEST['session_id'] ==
    md5(session_id()))
  {
    // Validação Aprovada
    $login = $_SESSION['login'];
  }
  else
  {
    // Validação Falhou
    header("Location: pagina1.php");
  }
}
else
{
  // Validação Falhou
  header("Location: pagina1.php");
}
?>
```

Considerações finais

Com o presente artigo, tentou-se demonstrar as principais funcionalidades do uso de sessões, bem como os passos necessários para sua implementação em *websites*.

Pelo exposto, conclui-se que as sessões representam uma forma descomplicada de se armazenar e propagar dados com segurança por um *website*, apresentando maior utilização em *websites* de comércio eletrônico.

Sobre o autor:
Leandro Schwarz é engenheiro eletricista pela UFSC. Atuando desde 2000 com desenvolvimento WEB, possui sólidos conhecimentos em PHP e MySQL.

Atualmente, está finalizando o mestrado em Engenharia Elétrica no Instituto de Engenharia Biomédica da UFSC. Produz *websites* e lojas virtuais como autônomo.

Referências Bibliográficas

[Manual PHP] – http://www.php.net
[iMasters] – http://www.imasters.com.br
[PHP Brasil] – http://www.phpbrasil.com.br

Novas técnicas e boas novidades se aproximam

Por Flávio Fagundes

Apaixonado pelo mundo da computação, Cristian Pedroso, 29 anos, teve seu primeiro contato com a informática aos 10 anos de idade. "Meu pai trouxe para casa um Maxxi (Apple IIe) da extinta Polymax". Como tudo naquela época era programado e interface gráfica não existia, nem nos mais remotos sonhos, Cristian aprendeu a se virar. Comprou umas revistas INPUT e aos 14 anos já trabalhava como programador, com registro em carteira e tudo mais. "Comecei com Clipper Summer 87. Deu pra tirar meus primeiros trocados e vir para Curitiba fazer Eletrotécnica no CEFET-PR, mas nunca consegui me desligar da área de programação.", relembra.

Aos poucos, Cristian iniciou com o PHP, ainda na versão 3.6, enquanto estudava PERL paralelamente. Dali para frente, a produtividade do PHP o conquistou definitivamente. "Foi um desafio integrar SQL Server com PHP 3.06, mas conseguimos com sucesso. Na seqüência, fui desenvolver um projeto audacioso em outra empresa que necessitava criar algumas rotinas COBOL na Web.", conta.

Cristian, que conhece a fundo a rotina de programação de diversas empresas de grande porte, como a Getup e a Midiaweb, é o entrevistado da primeira edição da PHP Magazine.

Como foi a sua trajetória profissional na Getup e na Midiaweb?

Após o sucesso do COBOL integrado com o PHP, a empresa onde eu trabalhava criou um departamento exclusivo para Web e eu fui incumbido de gerenciar esta unidade. Para minha felicidade, esta unidade tomou força e logo foi criada uma empresa exclusiva para ela, a Getup, da qual eu era um dos acionistas e o Diretor de Tecnologia. Nela, desenvolvemos várias aplicações utilizando PHP. Uma das principais é a Assembléia On Line, que era um sistema de transmissão de vídeo de sorteios de consórcio totalmente integrado com um sistema legado em Cobol. Também desenvolvemos nosso próprio CMS, que até hoje vem sofrendo *upgrades* e é uma ferramenta muito poderosa. Em 2005, a Getup fez uma parceria tecnológica com a Midiaweb para desenvolvimento do sistema de VoiP da GVT (Vono). Após alguns meses do encerramento do projeto, a parceria se manteve e recebi um convite da Midiaweb para ser o gerente de projeto e atuar em definitivo na empresa. Aceitei pensando na pluralidade de tecnologias e desafios que eu iria encontrar. Hoje temos projetos de médio e grande porte utilizando as tecnologias PHP, Java e ASP.NET.

De que forma o PHP é/ou foi encarado dentro de cada uma delas?

Certamente, o PHP é uma ferramenta de grande importância na vida profissional e, conseqüentemente, nas empresas pelas quais passei. Hoje, com uma melhor aceitação do mercado com relação à tecnologia, podemos focar ainda mais na utilização da linguagem. Atualmente, grandes empresas, que

antigamente eram escravas de softwares proprietários, renderam-se a performance e praticidade e vem exigindo a utilização do PHP. Sem dúvida, o PHP é ferramenta indispensável tecnicamente e mercadologicamente hoje.

Em quais projetos você já utilizou o PHP?

Já utilizamos PHP em vários projetos dos mais diversos portes. Desde ações dinâmicas de e-mail marketing até grandes portais. Vou citar alguns exemplos de grandes projetos:

assembléia on-line: Sistema de sorteio e transmissão de assembléias de consórcio pela internet. Clientes: Consórcio Nacional New Holland, Consórcio Nacional Mitsubishi Motors, , Consórcio Nacional Embracon, Consórcio Autoplan, Consórcio Nacional Iveco, entre outros.

Web user e Web Atendimento para Consórcios: Desenvolvimento de integração da interface Web com Cobol para criação de serviços on-line (emissão de boletos, oferta de lances, etc.)

Além de sistema de comércio exterior com PostgreSQL, games para endomarketing, sistema de E-Commerce para consórcio, sistema de suporte (chamados e callcenter), Web site 100% dinâmicos (Ex: Clube Atlético Paranaense), sistema de "bolão" da Copa do Mundo para grandes portais.

Até que ponto PHP é ideal e em quem ponto começa deixar a desejar?

Acho o PHP ideal para aplicações Web dos mais diversos portes e penso que ele deixa a desejar quando necessitamos de uma integração mais direta com a máquina do usuário. Acredito que o GTK ainda é muito complexo e deficiente perto de outras soluções com o mesmo fim. O que geralmente usamos é o PHP no backend de outras interfaces para o desktop.

Quais as IDEs utilizadas?

Particularmente, eu prefiro o Zend Encoder, pois oferece muitos recursos para o programador. Também estamos testando o Eclipse com uma extensão para PHP. Esta solução tem sido a preferida por programadores que atuam tanto em Java quanto em PHP.

E quanto à produtividade das equipes?

Certamente, as equipes PHP têm um alto nível de produtividade, uma vez que há grande disponibilidade de recursos na Web. Também é notório que a comunidade do PHP gosta muito de colaborar entre si, o que acaba ajudando na produtividade em relação a outras linguagens. Lógico que tudo isso esta aliado a uma série de outros fatores externos, como a motivação e nível dos profissionais, por exemplo.

Quais os servidores mais utilizados?

Em nossos projetos temos utilizado principalmente o Apache e o Lite. Também temos projetos rodando abaixo de IIS.

E quanto à disponibilidade de profissionais? Você encontrava profissionais capacitados com facilidade ?

Geralmente, a seleção de pessoal é sempre difícil. A maioria das pessoas que faz uma conexão com o banco acha que já sabe programar e isso acaba trazendo prejuízos enormes aos projetos. Temos muitos profissionais de nível intermediário, mas profissionais de nível avançado são raros no mercado e muitas vezes acabam migrando para outras tecnologias.

Como é a seleção de profissionais? Em que critérios é baseada?

A seleção do profissional é sempre baseada em critérios técnicos, testes de conhecimento, análise de código já escrito e fatores psicológicos. Hoje o mercado procura profissionais com a capacidade de resolver problemas por conta própria, ou seja, saber correr atrás de qualquer dificuldade.

Como a empresa encara um profissional que possui certificação Zend em seu currículo ?

Certamente, é um profissional diferenciado, pois este órgão, pelo menos, atesta o conhecimento da linguagem e da lógica básica. Porém, este

profissional ainda é uma jóia rara no mercado.

Como é a integração com outras tecnologias e quais as utilizadas?
Conforme mencionei nos projetos, já realizamos integrações com as mais diversas tecnologias. Temos PHP integrado com Cobol através do Linux onde utilizávamos vários comandos de Shell para acionar o Cobol. Também fizemos várias integrações com aplicações que utilizavam ASP e Java. Em Java, utilizando webservices, tivemos excelentes resultados. Além das tradicionais integrações com bancos de dados (Oracle, Postgre, Db2, Sql Server e MySQL). Também, desde 2002, temos realizado diversos tipos de integrações com Flash. No momento, estamos trabalhando na integração do PHP com Flex para alguns de nossos clientes.

Como encarar o fato de utilizar um *host* de terceiro ?
Hoje em dia é muito comum a utilização de um *host* de terceiro. Eu definiria que somente o cliente de médio porte acaba utilizando seu próprio *host*. O cliente pequeno não tem condição de sequer manter um servidor. O cliente grande gera muito tráfego e acaba terceirizando todo este processo para reduzir custos e principalmente dor de cabeça.

Para qual versão do PHP o desenvolvimento era focado, 4.0 ou 5.0?
Estamos utilizando PHP5 desde a versão beta e acreditamos que as facilidades que a linguagem oferece são de extrema importância para organização e ganho de produtividade.

O que você espera da tecnologia com a vinda do PHP 6.0 ?
Quando se fala de uma nova versão do PHP, sempre tenho boas expectativas. No entanto, me preocupa um pouco a questão da anunciada incompatibilidade de alguns *scripts*. Para empresas como a nossa, que mantém diversos sistemas em diversos servidores diferentes, isto pode se tornar um fator gerador de problemas. No entanto, todas as situações de evolução, como a utilização do Unicode e cada vez mais a linguagem estar embasada em OOP, nos fazem ter a certeza de que boas novidades se aproximam.