Universidade Estadual do Piauí – UESPI
Centro de Tecnologia e Urbanismo – CTU
Coordenação de Ciência da Computação

Guia Para Desenvolver Sistemas Padronizados para UESPI

Outubro / 2010

O que será visto....

- Estrutura de diretórios e arquivos do sistema.
- Configuração de conexão com o banco e de URLs do sistema.
- Criação de módulos do sistema.
- Padronização de nomes de banco de dados, tabelas e campos.

1- Introdução

Este guia foi desenvolvido para tirar algumas dúvidas que você poderá ter para começar o projeto de sistema seguindo o padrão adotado. As explicações dadas aqui serão básicas, mas servirão para você iniciar seu projeto. Lembre-se que as tecnologias utilizadas serão: PHP orientado a objetos, servidor local apache, MySQL, Smarty Framework, CSS, Ajax, Jquery, Prototype e padrões de projeto MVC e Factory.

2- Estrutura de diretórios e arquivos do sistema

Na figura acima, é mostrada toda a estrutura de diretórios necessária para o sistema funcionar. Para começar, instale os seguintes aplicativos no seu computador:

- Netbeans → Para programar em php e ter uma melhor visualização dos arquivos .tpl
- Notepad++ → Para fazer diversas substituições de nomes, necessárias em vários arquivos
- SqlYog ou WorkBench → Para gerenciar o banco de dados, suas tabelas e campos
- WampServer → pacote contendo servidor apache, MySQL e PHP

Após instalar o wampserver, é criada na raiz do seu disco local uma pasta chamada "wamp". Dentro desta pasta existe a pasta raiz do servidor "www" que é onde será colocada a pasta do seu sistema. Veja a figura acima.

3- Configuração de conexão com o banco e de URLs do sistema.

Depois de ter copiado todas as pastas e arquivos necessários para o funcionamento do sistema, como na figura da árvore de diretórios, a primeira coisa a fazer é executar seu script sql no PhpMyAdmin, que vem no pacote do wampserver, ou pelo SQLYog para criar o banco de dados. Feito isso, vamos configurar a conexão com o banco de dados e as URLs do sistema.

Abra o arquivo DatabaseConnectionFactory.class.php, e coloque informações como nome do servidor(host), usuário, senha e nome do banco de dados que deseja usar(dbname). Por padrão o wampserver já define o nome do servidor como localhost ou 127.0.0.1.

```
$params['driver'] = "mysqli";
$params['host'] = "127.0.0.1";
$params['user'] = "root";
$params['password'] = "";
$params['port'] = null;
$params['dbname'] = "dbprojur";
```

Depois acesse a pasta include e abra o arquivo include.init.php. É nele que iremos configurar nossas BASE_URLs. Substitua o "sysprojur" pelo nome da pasta do seu sistema.

```
<?php
define("DIR_BASE", "C:/wamp/www/sysprojur");
define("DIR_TEMP", DIR_BASE . '/temp');
define('BASE_URL', 'http://localhost/sysprojur/');
define('HTTP_URL', 'http://localhost/sysprojur/');

$includes[] = ini_get('include_path');

$includes[] = DIR_BASE;

$includesSet = implode(";",$includes);

ini_set('include_path', $includesSet);

?>
```

4- Criação de módulos do sistema.

Agora iremos construir um módulo simples para você entender. Podemos enxergar os módulos como os menus do sistema(ex.: menu aluno – inserir novo, consultar, alterar e deletar, menu professor e etc) . Toda vez que queremos criar um módulo, devemos verificar primeiro se o módulo que vamos construir possui características parecidas com algum outro módulo para que haja reutilização de código. Claro que mesmo assim, nem tudo é ctrl c + ctrl v. Vamos supor que no meu sistema eu tenha que fazer todo o CRUD(cadastro, alteração...) de alunos. Um módulo bastante parecido com o do aluno que quero construir seria o de funcionário por exemplo, pois os dois possuem nome, idade, sexo e matrícula. Um funcionário ainda teria PIS e outros dados próprios de um funcionário. Mas isso é o suficiente pra construir o módulo de aluno.

Partindo desse ponto, o primeiro passo é criar a classe de entidade ou modelo de aluno e a sua classe DAO, que é a classe responsável por fazer a comunicação com o banco. A classe modelo é gerada a partir de um gerador de classes, localizada na pasta “outros/app/gerador”. Abra o arquivo “gerador.php” e coloque o nome da tabela do banco onde iremos armazenar os alunos como na figura abaixo:

```php
<?php
require_once ("../../../include/initialize.php");

require_once 'classes/modelo/admin/controle/gerador/EntidadeGerador.class.php';
require_once 'classes/base/sistema/Util.class.php';

//$params ['table'] = "tbdiaria";
$params ['table'] = "tbaluno";
$params ['parentTable'] = "";
$params ['pacote'] = "";
$params ['prefix'] = "tb";

$gerador = new EntidadeGerador($params);
echo "ok";

?>
```

No nosso caso a tabela é “tbaluno”. Esse é um padrão de nome de tabela que será explicado mais a frente. Não se esqueça que o parametro “prefix” deve ser o mesmo usado na tabela. Esse prefixo pode ser “stb” ou ”tb”. Tudo sobre nome de tabela, campo e banco será explicado mais a frente. No caso, como nossa tabela é tbaluno, nosso prefix será então “tb”.

Despois disso, execute o arquivo gerador.php no navegador. Não esqueça de ativar o wampserver antes. A classe de entidade gerada estará no mesmo diretório que o “gerador.php”. No caso a classe será “Aluno.class.php”. Crie a pasta “alunos” em “classes/modelo/admin/entidade” e cole a classe gerada lá. Agora vamos criar a classe DAOAluno para fazer consultas e outras operações no banco. Essa classe deve ser feita na mão utilizando funções prontas localizadas na pasta “base”. Como estou fazendo com base em funcionários, eu copio a classe DAOFuncionario.class.php para pasta “alunos”e renomeio para DAOAluno.class.php. Depois eu só preciso substituir onde tem Funcionario para Aluno(por causa do nome da classe) e onde tem funcionario para aluno(pro causa do diretório e variáveis).

Pronto nossa classe entidade tá pronta. Agora vamos fazer a classe de controle. Nessa parte devemos ter cuidado nas modificações, pois qualquer erro fará com que não funcione nada. Então, mais uma vez, criamos a pasta ”alunos” e copiamos o controller de funcionário que será a base do controller do aluno e fazemos as mesmas substituições de nomes além de renomear os arquivos; onde tem Funcionário troca-se por Aluno.

Depois disso devemos alterar os arquivos “GestaoAlunos.class.php”, “ExecCadAlunoAction.class.php”, “ExecDelAlunoAction.class.php” e outros para remover o que tiver de informação sobre funcionário e trocar pela de aluno. Todas essas substituições são feitas rapidamente pelo NotePad++(tecle ctrl + f). Dentro da pasta de controle ainda, devemos registrar esses arquivos que acabamos de alterar no arquivo “ApplicationManagerAdmin.class.php” localizado em “classes/modelo/admin/controle/admin”. A linhas de funcionário lá registradas servem de modelo para aluno.

Por último vamos para as classes de interface. Aqui apenas copiamos a de funcionário e fazemos as mesmas substituições de nomes além de renomear o arquivo. Pronto nossas classes estão prontas. O próximo passo é criar o arquivo javascript de aluno chamado “GestaoAluno.class.js” que também terá a de funcionário como modelo e fazemos apenas as substituições de nomes além de renomear o arquivo.

Logo depois colocamos uma entrada desse arquivo javascript dentro de HTML.tpl localizado em “smarty/templates”. Agora vamos criar os arquivos tpl do aluno. Para isso, crie uma pasta “alunos” dentro de templates. Novamente, tendo os templates de funcionário como modelo, renomeamos os arquivos e fazemos as substituições de nomes. Além disso devemos alterar os templates para a correta mostragem dos formulários e dos dados, como por exemplo, remover o campo PIS de funcionário do template de aluno que é cópia.

Agora para encerrar, devemos criar os menus do sistema fazendo toda essa configuração no banco de dados através das tabelas “stb”(tabelas do sistema). O significado de algumas delas segue abaixo:

- **stbcaso_de_uso**: onde ficarão os nomes dos menus(módulos) e especifica a ordem em que aparecem no sistema.
- **stbfluxo**: onde ficam todas as funções do módulo(cadastrar, alterar, consultar e excluir) e estarão ligados a um caso de uso.
- **stbgrupo**: tipo de perfil do usuário no sistema.
- **stbgrupo_stbcaso-de_uso**: diz quais casos de uso um grupo tem acesso.
- **stbgrupo_stbusuario**: diz a qual grupo pertence um usuário.
- **stbitem_menu**: onde colocamos as opções do menu (Inserir novo e Listar todos – devem ser escrito dessa forma) que ficaram visíveis primeiro e para cada um setamos a função javascript que chama os arquivos necessários para mostragem das telas.
- **stbusuario_fluxo**: onde definimos quais funções do módulo o usuário tem acesso.

5- Nomenclatura dos nomes de banco de dados, tabelas e campos.

- O nome do banco de dados(database) deverá ser: “db” + “nome do banco”.
  - Ex.: dbauditoria, dbprojur, dbead, dbuespi.

- Todo banco de dados terá dois grupos de tabelas: as tabelas comuns a todos os sistemas(stb) e as tabelas do próprio sistema(tb).

  - Tabelas comuns aos sistemas
    - Nessas tabelas nada é alterado, exceto os valores que cada campo deverá ter. Todos os sistemas terão essas tabelas com os mesmos campos. O nome dessas tabelas deverá ser: “stb” + nome_da_tabela.
      - Ex.: stbusuario, stbcaso_de_uso, stbsessao, stbitem_menu.

  - Tabelas do próprio sistema
    - O nome dessas tabelas deverá ser “tb” + nome_da_tabela.
      - Ex.: tbfuncionario, tbapoio_sistema, tbprocesso_impetrado.

  Como vocês podem observar, nomes de tabelas compostas por duas ou mais palavras são separadas por “_”, exceto dos prefixos “stb” e “tb”.

- Os nomes dos campos das tabelas terão a inicial minúscula no primeiro nome e inicial maiúscula nos outros nomes, se tiver.
  - Ex.: idUsuário, nome, idade, tipoProcesso.

- Nomes de chave estrangeira serão definidas como: “fk”+ “nome da tabela referenciada”.
  - Ex.: fkAluno, fkProcesso.

Terminando....

Esse foi um pequeno guia para você seguir a padronização. Outros guias poderão ser feitos para tirar outras dúvidas que surgirem.