Banco de Dados

Banco de Dados

Até então, trabalhamos mais na camada do controlador. Nesta aula, trabalharemos na camada do modelo. É nesta camada que fica o armazenamento, a recuperação e o processamento dos dados específicos da aplicação.

Trabalharemos com o SGBD MySQL. Todos devem ter ele instalado para conseguir fazer os exercícios de aula. No Linux, para instalar o MySQL, utilize o comando:

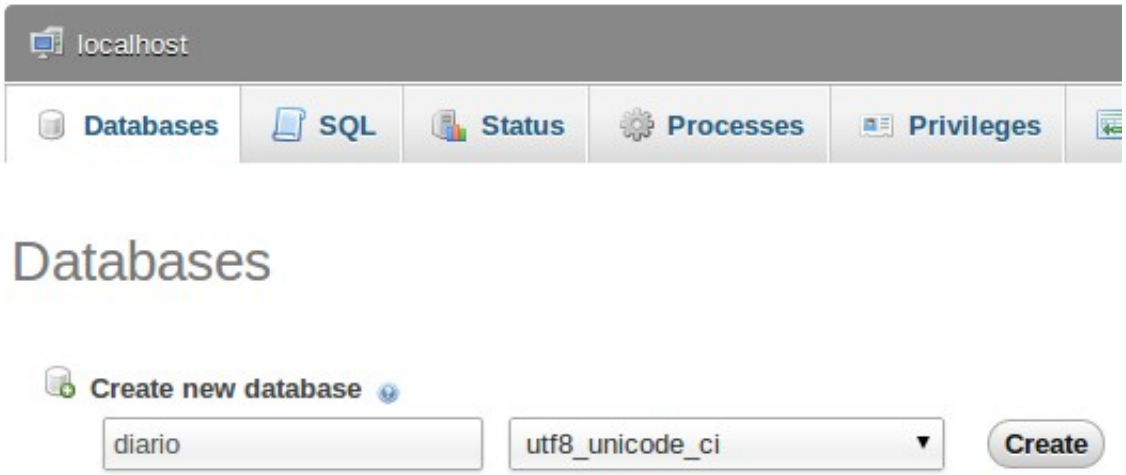
```
sudo apt-get install mysql-server
```

Banco de Dados

Vamos melhorar o nosso framework para oferecer suporte a banco de dados. Nosso framework será bem limitado, pois oferecerá suporte somente ao MySQL. Vocês podem alterar o framework para suporta outros SGBDs.

Banco de Dados

Para a nossa aplicação Diário, será criado um banco chamado **diario**. Utilize a *collation*: **utf8_unicode_ci**.



Banco de Dados

A aplicação terá somente uma tabela: mensagem. Abaixo está a DDL para criar a tabela.

```
CREATE TABLE mensagem (
    id INT NOT NULL AUTO_INCREMENT,
    texto VARCHAR(255) NOT NULL,
    PRIMARY KEY (id)
)
```

Framework

Vamos continuar o desenvolvimento do framework. Iremos criar um novo arquivo chamado **BancoDeDados.class.php** na pasta **framework/classes/**. Ele terá a classe BancoDeDados. Esta classe servirá para criar a conexão com o banco de dados, que no caso será o MySQL.

Como o desenvolvedor não deve alterar as classes internas para desenvolver uma aplicação, as configurações do banco de dados ficarão no arquivo **banco.php** dentro da pasta **framework/**. O desenvolvedor poderá editar este arquivo da mesma forma que o arquivo **config.php**.

Quando a classe BancoDeDados for carregada, ela deve carregar também as configurações do banco de dados.

Framework

As configurações mais comuns para a conexão com o banco de dados são: **servidor, porta, banco, usuário e senha**. No MySQL, utilizaremos as configurações padrão:

- servidor: localhost
- porta: 3306
- banco: diario (específico da nossa aplicação)
- usuário: root
- senha: **** (senha que você escolher para o root)

PHP Data Objects

Para que o PHP se conecte no banco de dados, é necessário utilizar um driver específico do banco de dados para PHP. No nosso caso, precisaremos do driver do MySQL. Para instalar no Linux, utilize o comando:

```
sudo apt-get install php5-mysql
```

PHP Data Objects

Em PHP, existem 3 formas para se conectar no MySQL:

- **mysql_connect(...):** esta é a forma mais antiga de se conectar. Esta função **não deve ser utilizada**, a menos que o sistema já tenha sido desenvolvido e utilize uma versão muito antiga do MySQL e do PHP. Esta forma não suporta OO.
- **mysqli_connect(...):** esta é uma versão melhorada da mysql_connect(). O "i" significa *improved* (melhorada). Esta forma pode ser utilizada tanto em código procedural quanto em código OO.
- **new PDO(...):** esta forma de utilizar o banco de dados é mais interoperável. Ela abstrai as funções específicas de cada banco de dados e fornece ao desenvolvedor somente as funcionalidades mais comuns. A vantagem de se utilizar é que será a mesma interface para qualquer banco de dados que for trabalhar. Ela é totalmente OO.

PHP Data Objects

Para utilizar o PDO, é necessário instanciar a classe **PDO()**. Utilizaremos 3 parâmetros no construtor dela: **fonte de dados, usuário e senha**. O parâmetro "fonte de dados" é uma string no seguinte formato:

```
mysql:host=localhost;port=3306;dbname=diario
```

A primeira palavra da string é o **driver**. O restante são parâmetros passados para o driver. A ordem dos parâmetros não importa. Os parâmetros que não forem passados, podem utilizar valores padrão. Cada banco de dados pode ter parâmetros diferentes.

PHP Data Objects

Na aplicação diário, assumindo o usuário "root" e senha "123mudar", o código ficará assim:

```
$pdo = new
   PDO(  'mysql:host=localhost;port=3306;dbname=diario'
   ,  'root',
'123mudar');
$pdo->exec("set names utf8");
```

A última linha serve para definir que a aplicação PHP enviará os textos na codificação UTF 8.

PHP Data Objects

Podemos deixar o objeto do PDO escondido pelo framework ou deixar o desenvolvedor acessálo diretamente. Caso o PDO fique escondido, é necessário disponibilizar para o desenvolvedor classes e métodos para realizar consultas e atualizações no banco de dados.

Para facilitar, deixaremos a camada de modelo ter acesso ao PDO. Isso não significa que o framework deixará todo o trabalho para o desenvolvedor. Mesmo disponibilizando o PDO, o framework pode ter métodos para auxiliar as tarefas comuns relacionadas ao banco de dados.

É interessante utilizar o padrão de projeto **singleton** para conexões com o banco de dados. Desta forma, só existirá uma conexão aberta na aplicação e ela será reutilizada em toda a camada de modelo.

PHP Data Objects

O objeto PDO tem vários métodos. Nós iremos utilizar somente 2 métodos:

- **query():** este método é utilizado para realizar consultas. Ele possui 1 parâmetro, que é a consulta em SQL. Iremos utilizar esse comando somente quando a consulta não possui dados enviados pelo usuário.
- **prepare():** este método serve para preparar um comando SQL qualquer. Este comando é enviado ao SGBD, "compilado" e retornado para o PHP. Esse método retorna um comando "preparado". Depois de preparado, podemos vincular parâmetros no comando (valores para inserção ou restrição de pesquisa). Depois de vincular os valores, podemos executar o comando com o método **execute()**.

PHP Data Objects

Exemplo de utilização do método query(). Não é possível acessar diretamente cada linha do resultado, pois o resultado da consulta é um objeto e não um vetor. Podemos acessar cada linha utilizando um **foreach**.

```
$linhas = $pdo->query('SELECT * FROM mensagem');
foreach($linhas as $linha) {
    echo $linha['id'] . ', '. $linha['texto'] . '<br>';
}
```

PHP Data Objects

Quando é utilizado um dado enviado pelo usuário, não é seguro concatenar strings, pois o usuário pode tentar burlar a segurança. Neste caso, utilizamos o método **prepare()**. Nos lugares dos dados, podemos colocar **:identificador**. Depois de preparado o comando, associamos os identificadores com seus valores. O comando pode executado várias vezes.

```
$comando = $pdo->prepare(
   'INSERT INTO mensagem(texto) VALUES (:texto)');
$comando-
   >execute(array(  ':texto' =>
   'Olá Mundo.'
));
```

Boas Práticas

Não é uma boa prática deixar strings "espalhadas" pelo código fonte. Em vários casos é aconselhável utilizar constantes, pois assim a string não precisará ser reescrita em vários locais e a manutenção será facilitada.

Como um consulta geralmente é escrita como uma string, é aconselhável definir uma constante para cada consulta. Isto faz com que o seu código fique organizado.

Outro ponto importante é que nem o controlador e nem a visão devem acessar o banco de dados. Lembrese que somente o modelo pode acessar o banco de dados. O modelo também é responsável pela lógica de negócios da aplicação.

Exercício

Altere o framework para salvar as mensagens da aplicação Diário em um banco de dados. Utilize os conceitos aprendidos em aula.

Referências

PHP.NET. **Manual**. Disponível em: <https://php.net/manual/en/>. Acesso em: 23 de maio de 2014.