Introdução

Trady

Esse é um conjunto de classes capaz de mapear um banco de dados de sua escolha
e gerar as classes (em php) para a manipulação dos dados entre o banco e o usuário final.

O Sistema é todo baseado em orientação a objetos, ótimo para quem Já possua um conhecimento elevado na área de OO e bom para quem está iniciando.

Atenção: para quem esta iniciando ira parecer um pouco complicado mas com o passar do tempo (media de uma semana) o programador estara com capacidade de controlar todas as funções das classes geradas.

=> Capitulo 1 - Instalando o Trady

1.1 - Estrai os arquivos no diretorio raiz do seu servidor.

1.2 - Ative o apache e acesse a pasta do sistema pelo navegador, nele ira pedir os dados para a conexão com o servidor MySQL, informe os dados e aperte em instalar.

=> Capitulo 2 - Gerando um conjunto de classes.

2.1 - A pagina inicial escolha um banco de dados a ser mapeado.

2.2 - Na segunda pagina escolha as tabelas que você deseja mapear, e clique em "Gerar Classes".

2.3 - Classes geradas. Agora acesse a pasta "classes_geradas" dentro desta pasta estara a pasta onde estão suas classes

*Nome da pasta foi informado no final da geração das classes.

=> Capitulo 3 - Configurando o arquivo DB.php

3.1 - Abra o arquivo "DB.php".

3.2 - Procure a função "__construct". Nela ira conter o seguinte código:

```
// inicializa as variaveis para a conexão com o banco

public function __construct() {

        self::escreveDados($this->server, "");

        self::escreveDados($this->usuario, "");

        self::escreveDados($this->senha, "");

        self::escreveDados($this->banco, "usuarios");

        self::conexao();

}
```

3.3 - Altera os dados para que fique da seguinte forma:

```
        // inicializa as variaveis para a conexão com o banco

        public function __construct() {

            self::escreveDados($this->server, "localhost"); // endereço do servidor mysql

            self::escreveDados($this->usuario, "root");   // nome do usuario no mysql

            self::escreveDados($this->senha, "123456");// senha do usuario mysql

            self::escreveDados($this->banco, "usuarios");      // nome do banco de dados

            self::conexao();

    }
```

3.4 - Pronto Arquivo configurado.

=> Capitulo 4 - Retornando os dados do banco de dados

4.1 - Primeiro crie um include na sua pagina chamando o arquivo "Principal.php":

Exemplo:

```
<?php

include("classes/Principal.php");

?>
```

*Atenção: Caso você queira renomeie a pasta para um nome de sua escolha

4.2 - Crie uma instancia da classe:

Exemplo:

```
<?php

include("classes/Principal.php");

$principal = new Principal();

?>
```

4.3 - Agora nos vamos chamar os dados que estão no banco de dados:

4.3.1 - Formação do nome da função:

A função e conposta pelas seguintes palavras:

"retorna" = palavra fixa para funções de retorno.

"nome da tabela" = é o nome da tabela de onde serão retornado os dados.

*Atenção: no nome da tabela sempre ira começar com um caractere maiusculo.

nome completo da função.

Exemplo:

retornaTbl_usuarios($extra);

4.3.2 - Parametros passados

repare que o nome da função é formado pela palavra fixa mas o nome da tabela.
além do mais ele pede um parametro para ser passado, nesse caso esse parametro é opcional, o programador so ira passar parametro se desejar uma busca com filtros

4.3.3 - Exemplos de retornos

Exemplo 1: Retornando o nome de todos os usuarios da tabela "tbl_usuarios"

```
<?php

include("classes/Principal.php");
$principal = new Principal();
foreach($principal->retornaTbl_usuarios() as $usuario) {
    echo $usuario->getNome();
}
?>
```

Exemplo 2: Retornando o nome dos ultimos 5 usuarios da tabela "tbl_usuarios"

```
<?php

include("classes/Principal.php");
$principal = new Principal();
$extra = "ORDER BY ID DESC LIMIT 0,5";
foreach($principal->retornaTbl_usuarios($extra) as $usuario) {
      echo $usuario->getNome();
}
?>
```

4.3.4 - Esclarecendo o foreach

Repare que nos chamamos o retorno dos usuario em um foreach. Porque?
O motivo pelo qual nos passamos o retorno para um foreach é pelo motivo que
A função retorna um array de objetos, repare na variavel "$usuario" onde nos passamos para
ela o valor da instancia de retorno. Que a mesma se refer as classes do tipo "Basicas"
que estão contidas na pasta "classesBasicas".
Observer como chamamos o metodo para exibição do nome do usuario

```
$usuario->getNome();
```

No caso desse sistema ele retorna um método que contenha o valor que estava contido no banco
este metodo e composto da seguinte forma:
"get" = palavra fixa
"Nome" = nome do campo referente ao retorno (O primeiro caractere que se refere ao nome do campo deve ser sempre em maiusculo)

=> Capitulo 5 - Inserindo dados no banco

5.1 - Chamando a função de inserção no banco

Para poder inserir dados no banco é necessario chamar duas funções.

Exemplo:

```
<?php

include("classes/Principal.php");
$principal = new Principal();
$principal->tbl_usuarios('', $nome_usuario, $email_usuario, $senha_usuario);
$principal->inserirTbl_usuarios(); // retorna um valor boolean true ou false
?>
```

Explicação:

Repare que nas invocamos outro método nesse caso "tbl_usuarios()";
Esse método é responsavel por alocar os dados temporariamente para que o método de inserção
possa inserir no banco de dados.
Após alocarmos os dados nós chamamos o método de inserção "inserirTbl_usuarios()"

*Atenção: cada tabela tera seu método restrito.

Exemplo: tbl_noticias

```
<?php
include("classes/Principal.php");
$principal = new Principal();
$principal->tbl_noticias('', $titulo, $descricao, $data);
$principal->inserirTbl_noticias(); // retorna um valor boolean true ou false
?>
```

=> Capitulo 6 - Atualização de dados

6.1 - Chamando a função para atualizar os dados

A principio para chamar a função serãm passados os mesmos dados que a função de inserção

mas com a seguinte diferença nã sera chamado o metodo de inserção mas sim o de atualização

Exemplo:

```
<?php
include("classes/Principal.php");
$principal = new Principal();
$principal->tbl_usuarios('', $nome_usuario, $email_usuario, $senha_usuario);
$principal->atualizaTbl_usuarios(); // retorna um valor boolean true ou false
?>
```

=> Capitulo 7 - deletando dados

7.1 - Chmando a função de deletar dados

Essa é a mais facil das funções par deletar os dados você so precisa passar o id do dado a ser apagado

Exemplo 1: Apagando apenas um registro

```
<?php

include("classes/Principal.php");
$principal = new Principal();
$principal->deletaTbl_usuarios($id_usuario); // retorna um valor boolean true ou false
?>
```

Exemplo 1: Apagando todos os registros da tabela "tbl_usuarios"

```
<?php

include("classes/Principal.php");
$principal = new Principal();
foreach($principal->retornaTbl_usuarios() as $usuario) {
    $principal->deletaTbl_usuarios($usuario->getID());
}
?>
```

=> Capitulo 8 - Retornando a quantidade de dados cadastrados

8.1 - Chamando o método que informa a quantidade de dados que estão cadastrados

Este método é parecido com o de retorno exceto pelo motivo que ele retorna um número ou false.

Exemplo 1: Informando o total de dados cadastrados

```
<?php

include("classes/Principal.php");
$principal = new Principal();
$principal->retornaQuantidadeRegistrosTbl_usuarios(); // retorna um valor numerico se verdadeiro ou false se não encontrar dados
?>
```

Exemplo 2: Informando o total de dados cadastrados que tenham estejam ativados

```
<?php

include("classes/Principal.php");
$principal = new Principal();
$extra = "WHERE Status='1'";
$principal->retornaQuantidadeRegistrosTbl_usuarios($extra); // retorna um valor numerico se verdadeiro ou false se não encontrar dados

?>
```