SESIÓN - 33

Listar, obtener, eliminar, actualizar y registrar datos con PDO

Desarrollo:

1. Listar, obtener, eliminar, actualizar y registrar datos con PDO.

Ahora que sabemos cómo conectarnos a la base de datos, vamos a crear funciones para poder insertar, eliminar, modificar, actualizar y listar datos. Para ello crearemos un documento php con el nombre de *local.model.php* que en su interior tendrá una clase que herede de la clase *conectar*.

local.model.php

```
<?php
class LocalModel extends Conectar
{
   public function Listar()
   {
      try
      {
         $result = array();
         $stm = $this->pdo->prepare("SELECT * FROM cetpagoslocal order by id");
         $stm->execute();
         foreach($stm->fetchAll(PDO::FETCH_OBJ) as $r)
         {
            $loc = new Local();
            $loc->__SET('id', $r->id);
            $loc->__SET('denominacion', $r->denominacion);
            $loc->__SET('direccion', $r->direccion);
            $result[] = $loc;
         }
         return $result;
      }
      catch(Exception $e)
      {
         die($e->getMessage());
      }
   }
   public function Obtener($id)
   {
      try
      {
         $stm = $this->pdo
               ->prepare("SELECT * FROM cetpagoslocal WHERE id = ?");
         $stm->execute(array($id));
```

```
        $r = $stm->fetch(PDO::FETCH_OBJ);
        $loc = new Local();
        $loc->__SET('id', $r->id);
        $loc->__SET('denominacion', $r->denominacion);
        $loc->__SET('direccion', $r->direccion);
        return $loc;
    } catch (Exception $e)
    {
        die($e->getMessage());
    }
}
public function Eliminar($id)
{
    try
    {
        $stm = $this->pdo
                ->prepare("DELETE FROM cetpagoslocal WHERE id = ?");
        $stm->execute(array($id));
    } catch (Exception $e)
    {
        die($e->getMessage());
    }
}
public function Actualizar(Local $data)
{
    try
    {
        $sql = "UPDATE cetpagoslocal SET
                    denominacion        = ?,
                    direccion       = ?
                WHERE id = ?";
        $this->pdo->prepare($sql)
            ->execute(
            array(
                $data->__GET('denominacion'),
                $data->__GET('direccion'),
                $data->__GET('id')
                )
            );
    } catch (Exception $e)
    {
        die($e->getMessage());
    }
}
public function Registrar(Local $data)
{
    try
    {
    $sql = "INSERT INTO cetpagoslocal(denominacion, direccion)
            VALUES (?, ?)";
    $this->pdo->prepare($sql)
```

```
        ->execute(
        array(
          $data->__GET('denominacion'),
          $data->__GET('direccion')
          )
        );
     } catch (Exception $e)
     {
        die($e->getMessage());
     }
  }
}
?>
```

Función listar:

```
  public function Listar()
  {
     try
     {
        $result = array();
        $stm = $this->pdo->prepare("SELECT * FROM datosUsuario order by nombre");
        $stm->execute();
        foreach($stm->fetchAll(PDO::FETCH_OBJ) as $r)
        {
           $loc = new Local();
           $loc->__SET('id', $r->id);
           $loc->__SET('nombre', $r->nombre);
           $loc->__SET('direccion', $r->direccion);
           $result[] = $loc;
        }
        return $result;
     }
     catch(Exception $e)
     {
        die($e->getMessage());
     }
  }
```

En esta función como observamos creamos una variable de tipo array, y otra para que guarde la consulta a la base de datos; mediante un foreach obtenemos los datos extraídos de la base de datos, enseguida creamos un objeto para la clase local y este objeto obtendrá mediante __SET los datos correspondientes para luego enviarlos a la variable array creada y retornar el resultado; si hubiera un error mediante el catch podemos poner una excepción.

La instrucción die() tiene la misma función que el exit.

Función obtener:

```
public function Obtener($id)
{
    try
    {
        $stm = $this->pdo
                ->prepare("SELECT * FROM datosUsuario WHERE id = ?");
        $stm->execute(array($id));
        $r = $stm->fetch(PDO::FETCH_OBJ);
        $loc = new Local();
        $loc->__SET('id', $r->id);
        $loc->__SET('nombre', $r->nombre);
        $loc->__SET('direccion', $r->direccion);
        return $loc;
    } catch (Exception $e)
    {
        die($e->getMessage());
    }
}
```

La función obtener nos permite hacer una consulta para extraer de la base de datos todos los registros según el id seleccionado, como observaran en la consulta implantada se coloca "*id = ?" para que según el id en el array se obtengan los datos.*

Función eliminar:

```
public function Eliminar($id)
  {
     try
     {
        $stm = $this->pdo
                ->prepare("DELETE FROM datosUsuario WHERE id = ?");

        $stm->execute(array($id));
     } catch (Exception $e)
     {
        die($e->getMessage());
     }
  }
```

La función eliminar como su propio nombre lo dice nos permite borrar un registro de nuestra base de datos, al observar la consulta podemos ver que la consulta es sencilla y de fácil entendimiento.

Función actualizar:

```
public function Actualizar(Local $data)
 {
    try
    {
       $sql = “UPDATE datosUsuario SET
                nombre        = ?,
                direccion      = ?
             WHERE id = ?”;
       $this->pdo->prepare($sql)
           ->execute(
          array(
             $data->__GET(‘nombre’),
             $data->__GET(‘direccion’),
             $data->__GET(‘id’)
             )
          );
    } catch (Exception $e)
    {
       die($e->getMessage());
    }
 }
```

En la función actualizar usamos la consulta UPDATE para así obtener los datos y modificarlos según sea necesario.

Función registrar:

```
public function Registrar()
   {
      try
         {
               $sql = "INSERT INTO datosUsuario(nombre, direccion)   VALUES (?, ?)";
               $this->pdo->prepare($sql)
                     ->execute(
                      array(
                              $data->__GET('nombre'),
                              $data->__GET('direccion')
                          )
                      );
         }
      catch (Exception $e)
         {
                  die($e->getMessage());
         }
   }
```

En la función registrar nos permitirá guardar los registros en sus respectivos campos generando la consulta INSERT INTO.