**Active Record Pattern**

* The Active Record pattern was originally named such by Martin Fowler in his classic book *Patterns of Enterprise Application Architecture*.
* It's a way to map rows in a database to objects that allows simple access to columnar data.
* Most ORMs are based around this pattern and development frameworks such as Ruby on Rails and Zend Framework has pushed it into common usage.
* Used properly, a good Active Record implementation can save programmers time and sanity.
* In general, Active Record usage improves code by making it easier to read and understand
* Active Record provides a clear separation of the database logic from the code that uses it and isolates and frees the developers from the monotony of repetitive SQL statements.

**Intent**

An object that wraps a row in a database table or view, encapsulates the database access, and adds domain logic on that data.

An object carries both data and behavior. Much of this data is persistent and needs to be stored in a database. Active Record uses the most obvious approach, putting data access logic in the domain object. This way all people know how to read and write their data to and from the database.

**How it works?**

The essence of an *Active Record* is a *Domain Model* in which the classes match very closely the record structure of an underlying database. Each *Active Record* is responsible for saving and loading to the database and also for any domain logic that acts on the data. This may be all the domain logic in the application.

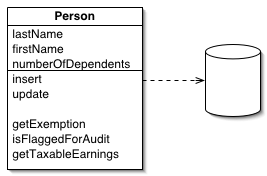
The data structure of the *Active Record* should exactly match that of the database: one field in the class for each column in the table.

The *Active Record* class typically has methods that do the following:

* Construct an instance of the *Active Record* from a SQL result set row
* Construct a new instance for later insertion into the table
* Static finder methods to wrap commonly used SQL queries and return *Active Record* objects
* Update the database and insert into it the data in the *Active Record*
* Get and set the fields
* Implement some pieces of business logic

The getting and setting methods can do some other intelligent things, such as convert from SQL-oriented types to better in-memory types.

**Implementation**



**Applicability**

* *Active Record* is a good choice for domain logic that isn't too complex, such as creates, reads, updates, and deletes.
* Derivations and validations based on a single record work well in this structure.
* *Active Record* has the primary advantage of simplicity. It's easy to build *Active Records,* and they are easy to understand.
* Their primary problem is that they work well only if the *Active Record* objects correspond directly to the database tables: an isomorphic schema. If the business logic is complex, this can lead to the use object's direct relationships, collections, inheritance, and so forth. These don't map easily onto *Active Record,* and adding them piecemeal gets very messy. In such cases it’s better to use use *Data Mapper*.
* Another argument against *Active Record* is the fact that it couples the object design to the database design. This makes it more difficult to refactor either design as a project goes forward.

**Examples**

Active Record Class

<?php

**class** User{

**private** $firstName;

**private** $lastName;

**private** $email;

**private** $table='users';

**public function** \_\_construct(){

}

// set first name

**public function** setFirstName($firstName){

$this->firstName=$firstName;

}

// set last name

**public function** setLastName($lastName){

$this->lastName=$lastName;

}

// set email

**public function** setEmail($email){

$this->email=$email;

}

// fetch row

**public function** fetch($id){

**if**(!$row=mysql\_query("SELECT \* FROM $this->table WHERE id='$id'")){

**throw new** Exception('Error fetching row');

}

**return** $row;

}

// insert row

**public function** insert(){

**if**(!mysql\_query("INSERT INTO $this->table (firstname,lastname,email) VALUES ($this->firstName,$this->lastName,$this->email)")){

**throw new** Exception('Error inserting row');

}

}

// update row

**public function** update($id){

**if**(!mysql\_query("UPDATE $this->table SET firstname='$this->firstName,lastname=$this->lastName,email=$this->email WHERE id='$id'")){

**throw new** Exception('Error updating row');

}

}

// delete row

**public function** delete($id){

**if**(!mysql\_query("DELETE FROM $this->table WHERE id='$id'")){

**throw new** Exception('Error deleting row');

}

}

}

**try**{

$user=**new** User();

// set first name

$user->setFirstName('John ');

// set last name

$user->setlastName('Smith');

// set email

$user->setEmail('smith@domain.com');

// insert row

$user->insert();

// set first name

$user->setFirstName('Johnny');

// set last name

$user->lastName('Smith');

// set email

$user->setEmail('johnny@domain.com');

// update row

$user->update(1);

// delete row

$user->delete(1);

}

**catch**(Exception $e){

**echo** $e->getMessage();

**exit**();

}

**MySql Abstraction Class**

<?php

// define 'MySQL' class

**class** MySQL{

**private** $result;

**private** $select='SELECT \* FROM ';

**private** $where=' WHERE ';

**private** $limit=' LIMIT ';

**private** $like=' LIKE ';

**public function** \_\_construct($host='localhost',$user='user',$password='password',$database='database'){

// connect to MySQL and select database

**if**(!$conId=mysql\_connect($host,$user,$password)){

**throw new** Exception('Error connecting to the server');

}

**if**(!mysql\_select\_db($database,$conId)){

**throw new** Exception('Error selecting database');

}

}

// run SQL query

**public function** query($query){

**if**(!$this->result=mysql\_query($query)){

**throw new** Exception('Error performing query '.$query);

}

}

// fetch one row

**public function** fetchRow(){

**while**($row=mysql\_fetch\_array($this->result)){

**return** $row;

}

**return false**;

}

// fetch all rows

**public function** fetchAll($table='default\_table'){

$this->query('SELECT \* FROM '.$table);

$rows=**array**();

**while**($row=$this->fetchRow()){

$rows[]=$row;

}

**return** $rows;

}

// insert row

**public function** insert($params=**array**(),$table='default\_table'){

$fields = implode(',',array\_keys($params));

$values= (''.implode("','",array\_values($params)).'');

$sql="INSERT INTO $table ($fields) VALUES ('$values')";

$this->query($sql);

}

// update row

**public function** update($params=**array**(),$where,$table='default\_table'){

$args=**array**();

**foreach**($params **as** $field=>$value){

$args[]=$field.'='."'".$value."'";

}

$sql='UPDATE '.$table.' SET '.implode(',',$args).$this->where.$where;

$this->query($sql);

}

// delete one or multiple rows

**public function** delete($where='',$table='default\_table'){

$sql=!$where?'DELETE FROM '.$table:'DELETE FROM '.$table.$this->where.$where;

$this->query($sql);

}

// fetch rows using 'WHERE' clause

**public function** fetchWhere($where,$table='default\_table'){

$this->query($this->select.$table.$this->where.$where);

$rows=**array**();

**while**($row=$this->fetchRow()){

$rows[]=$row;

}

**return** $rows;

}

// fetch rows using 'LIKE' clause

**public function** fetchLike($field,$like,$table='default\_table'){

$this->query($this->select.$table.$this->where.$field.$this->like.$like);

$rows=**array**();

**while**($row=$this->fetchRow()){

$rows[]=$row;

}

**return** $rows;

}

// fetch rows using 'LIMIT' clause

**public function** fetchLimit($offset=1,$numrows=1,

$table='default\_table'){

$this->query($this->select.$table.$this->limit.$offset.','.$numrows);

$rows=**array**();

**while**($row=$this->fetchRow()){

$rows[]=$row;

}

**return** $rows;

}

}

//Testing

**try**{

// connect to MySQL and select a database

$db=**new** MySQL('localhost','root','','patterns');

// insert new row

$db->insert(**array**('firstname'=>'Kate','lastname'=>'Johanson','email'=>

'kate@domain.com'),'users');

// update row

$db->update(**array**('firstname'=>'Kathleen','lastname'=>'Johanson','email'=>

'kate@domain.com'),'id=1','users');

// delete row

$db->delete('id=1','users');

// display all users

$result=$db->fetchAll('users');

**foreach**($result **as** $row){

**echo** $row['firstname'].' '.$row['lastname'].' '.$row['email'].'<br />';

}

// display users where ID > 5

$result=$db->fetchWhere('id>5','users');

**foreach**($result **as** $row){

**echo** $row['firstname'].' '.$row['lastname'].' '.$row['email'].'<br />';

}

// display users where first name contains the 'a' character

$result=$db->fetchLike('firstname',"'%a%'",'users');

**foreach**($result **as** $row){

**echo** $row['firstname'].' '.$row['lastname'].' '.$row['email'].'<br />';

}

// display users with the LIMIT clause

$result=$db->fetchLimit(2,4,'users');

**foreach**($result **as** $row){

**echo** $row['firstname'].' '.$row['lastname'].' '.$row['email'].'<br />';

}

}

**catch**(Exception $e){

**echo** $e->getMessage();

**exit**();

}