mySQL DAO

Generated by Doxygen 1.8.6

Sun Jul 6 2014 12:41:56

Contents

1	Hier	archica	Index	1
	1.1	Class I	Hierarchy	1
2	Clas	s Index		3
	2.1	Class I	ist	3
3	Clas	s Docu	mentation	5
	3.1	Addres	s Class Reference	5
		3.1.1	Detailed Description	6
		3.1.2	Constructor & Destructor Documentation	6
			3.1.2.1construct	6
		3.1.3	Member Function Documentation	7
			3.1.3.1 getCity	7
			3.1.3.2 getld	7
			3.1.3.3 getName	7
			3.1.3.4 getState	7
			3.1.3.5 getStreet1	7
			3.1.3.6 getStreet2	8
			3.1.3.7 getZip	8
			3.1.3.8 setCity	8
			3.1.3.9 setId	8
			3.1.3.10 setName	8
			3.1.3.11 setState	8
			3.1.3.12 setStreet1	9
			3.1.3.13 setStreet2	9
			3.1.3.14 setZip	9
		3.1.4	Member Data Documentation	9
			3.1.4.1 \$dao_typeMap	9
	3.2	MySQI	DAO Class Reference	10
		3.2.1		10
		3.2.2		11
			2.2.2.1 collCtatio	44

iv CONTENTS

	3.2.2.2	dao_generateBindParameters	11
	3.2.2.3	dao_generateFieldList	12
	3.2.2.4	dao_generateParameters	13
	3.2.2.5	dao_generateUpdateParameters	13
	3.2.2.6	delete	13
	3.2.2.7	insert	13
	3.2.2.8	update	14
Index			15

Chapter 1

Hierarchical Index

	1	.1	C	lass	Hier	archy
--	---	----	---	------	------	-------

This inheritance	list	is s	or	ted	ro	ug	hly	ı, b	ut	nc	ot c	100	mp	let	tely	, a	ılpl	hal	bet	tica	ally	y :									
MySQLDAO																															10
Address																								 					 		Ę

2 **Hierarchical Index**

Chapter 2

Class Index

2.1 Class List

Here are the classes,	structs, i	unions a	nd interfaces	with bri	ef descriptions:

Address	
Example class demonstrating the use of MySQLDAO	
MySQLDAO	
Abstract class for the mySQL DAO design pattern	1

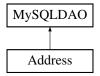
Class Index

Chapter 3

Class Documentation

3.1 Address Class Reference

example class demonstrating the use of MySQLDAO Inheritance diagram for Address:



Public Member Functions

- __construct (\$newId, \$newName, \$newStreet1, \$newStreet2, \$newCity, \$newState, \$newZip)
 constructor for Address
- getld ()

accessor method for id

• setId (\$newId)

mutator method for id

• getName ()

accessor method for name

• setName (\$newName)

mutator method for name

• getStreet1 ()

accessor method for first line of street address

• setStreet1 (\$newStreet1)

mutator method for first line of street address

• getStreet2 ()

accessor method for second line of street address

• setStreet2 (\$newStreet2)

mutator method for second line of street address

• getCity ()

accessor method for city

• setCity (\$newCity)

mutator method for city

• getState ()

accessor method for state

setState (\$newState)

mutator method for state

• getZip ()

accessor method for zip

setZip (\$newZip)

mutator method for zip

Protected Attributes

\$id

id of the address; this is the primary key

• \$name

name of the person or company

• \$street1

first line of the street address

• \$street2

second line of the street address

\$city

city of the street address

• \$state

state of the street address

• \$zip

ZIP code of the street address.

Static Protected Attributes

• static \$dao_primaryKey = "id"

MySQLDAO's name of the primary key.

• static \$dao_tableName = "address"

MySQLDAO's name of the table.

static \$dao_typeMap

MySQLDAO's map of the data types.

Additional Inherited Members

3.1.1 Detailed Description

example class demonstrating the use of MySQLDAO

This is a typical example of a class that represents a row in a mySQL table. Notice the class is simple and clean and consists of only typical accessor and mutator methods. All one needs to do is extend the MySQLDAO class and include the static DAO related variables to deploy the DAO pattern.

Author

Dylan McDonald dylanm@deepdivecoders.com

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Address::__construct (\$newId, \$newName, \$newStreet1, \$newStreet2, \$newCity, \$newState, \$newZip)

constructor for Address

Parameters

integer	new value of id
string	new value of name
string	new value of first line of street address
string	new value of second line of street address
string	new value of city
string	new value of state
string	new value of zip

Exceptions

RuntimeException	if invalid inputs detected

3.1.3 Member Function Documentation

```
3.1.3.1 Address::getCity()
accessor method for city
Returns
      string value of city
3.1.3.2 Address::getId()
accessor method for id
Returns
     integer value of id
3.1.3.3 Address::getName()
accessor method for name
Returns
     string value of name
3.1.3.4 Address::getState ( )
accessor method for state
Returns
      string value of state
3.1.3.5 Address::getStreet1 ( )
accessor method for first line of street address
Returns
```

string value of first line of street address

3.1.3.6 Address::getStreet2 () accessor method for second line of street address Returns string value of second line of street address 3.1.3.7 Address::getZip() accessor method for zip Returns string value of zip 3.1.3.8 Address::setCity (\$newCity) mutator method for city **Parameters** string | new value of city **Exceptions** UnexpectedValueException | if location is empty 3.1.3.9 Address::setId (\$newId) mutator method for id **Parameters** integer | new value of id **Exceptions** RangeException if id is negative UnexpectedValueException if id is invalid 3.1.3.10 Address::setName (\$newName) mutator method for name **Parameters** string new value of name **Exceptions** UnexpectedValueException if name is empty

3.1.3.11 Address::setState (\$newState)

mutator method for state

Parameters

string	new value of state
--------	--------------------

Exceptions

UnexpectedValueException	if state is invalid

3.1.3.12 Address::setStreet1 (\$newStreet1)

mutator method for first line of street address

Parameters

string	new value of first line of street address
--------	---

Exceptions

UnexpectedValueException	if location is empty
--------------------------	----------------------

3.1.3.13 Address::setStreet2 (\$newStreet2)

mutator method for second line of street address

Parameters

string	new value of second line of street address
--------	--

3.1.3.14 Address::setZip (\$newZip)

mutator method for zip

Parameters

strina	new value of zip
July	HOW VAIDE OF ZIP

Exceptions

```
UnexpectedValueException | if zip is invalid
```

3.1.4 Member Data Documentation

```
3.1.4.1 Address::$dao_typeMap [static],[protected]
```

Initial value:

```
= array("id" => "i", "name" => "s", "street1" => "s", "street2" => "s", "city" => "s", "state" => "s", "zip" => "s")
```

MySQLDAO's map of the data types.

The documentation for this class was generated from the following file:

· address.php

3.2 MySQLDAO Class Reference

abstract class for the mySQL DAO design pattern Inheritance diagram for MySQLDAO:



Public Member Functions

delete (&\$mysqli)

deletes this object from mySQL

• insert (&\$mysqli)

inserts this object to mySQL

• update (&\$mysqli)

updates this object in mySQL

Static Public Member Functions

static __callStatic (\$methodName, \$argv)
 dynamically provides getTableByField() static methods

Protected Member Functions

- dao_generateBindParameters (\$fieldList)
 generates an array to pass to mysqli_stmt::bind_param()
- dao_generateParameters (\$numFields)

generates placeholders for an INSERT query template

• dao_generateUpdateParameters (\$fieldList)

generates placeholders for an UPDATE query template

Static Protected Member Functions

• static dao_generateFieldList (\$omitPrimaryKey) generates a comma separated field list

3.2.1 Detailed Description

abstract class for the mySQL DAO design pattern

This class is the abstract class for the mySQL Data Access Object (DAO) pattern. Each subclass of this class will receive the following public methods:

- · delete(&\$mysqli);
- · insert(&\$mysqli);
- update(&\$mysqli);

• Table::getTableByField(&\$mysqli, \$fieldValue)

Note that the static method is dynamically generated based on the field that is queried. In order to use the DAO pattern, subclasses must have the following protected static variables:

- · dao_primaryKey: string containing the primary key's field name
- dao_tableName: string containing the mySQL table name
- dao_typeMap: array containing "field" => "type" pairs, where field is the field name and type is the data type according to mysqli_stmt::bind_param().

Author

Dylan McDonald dylanm@deepdivecoders.com

See Also

```
http://www.php.net/manual/en/mysqli-stmt.bind-param.php
```

3.2.2 Member Function Documentation

```
3.2.2.1 static MySQLDAO::__callStatic( $methodName, $argv ) [static]
```

dynamically provides getTableByField() static methods

The method name must conform to getTableByField() and take exactly two parameters:

- · \$mysqli: pointer to a mySQL connection
- · \$fieldValue: value to search for

Parameters

string	method name being called
array	arguments to the array

Exceptions

BadMethodCallException	if the method name is malformed
mysqli_sql_exception	if a mySQL error occurs
ReflectionException	if the class cannot be reflected
DomainException	if the SQL query returns empty set

3.2.2.2 MySQLDAO::dao_generateBindParameters(*\$fieldList*) [protected]

generates an array to pass to mysqli_stmt::bind_param()

Parameters

string comma separated field list

Returns

array arguments to pass to mysqli_stmt::bind_param()

See Also

```
http://www.php.net/manual/en/mysqli-stmt.bind-param.php
```

3.2.2.3 static MySQLDAO::dao_generateFieldList(*\$omitPrimaryKey*) [static], [protected] generates a comma separated field list

Parameters

boolean whether to omit the primary key

Returns

string comma separated field list

3.2.2.4 MySQLDAO::dao_generateParameters (\$numFields) [protected]

generates placeholders for an INSERT query template

Parameters

integer | number of fields to template

Returns

string templated placeholders

3.2.2.5 MySQLDAO::dao_generateUpdateParameters(*\$fieldList*) [protected]

generates placeholders for an UPDATE query template

Parameters

string comma separated field list

Returns

string templated placeholders

3.2.2.6 MySQLDAO::delete (& \$mysqli)

deletes this object from mySQL

Parameters

mysqli pointer to valid mysqli connection

Exceptions

mysqli_sql_exception | if a mySQL error occurs

3.2.2.7 MySQLDAO::insert (& \$mysqli)

inserts this object to mySQL

Parameters

mysqli pointer to valid mysqli connection

Exceptions

mysqli_sql_exception	if a mySQL error occurs
----------------------	-------------------------

3.2.2.8 MySQLDAO::update (& \$mysqli)

updates this object in mySQL

Parameters

mysqli	pointer to valid mysqli connection
--------	------------------------------------

Exceptions

mysqli_sql_exception	if a mySQL error occurs
----------------------	-------------------------

The documentation for this class was generated from the following file:

• mysqldao.php

Index

\$dao_typeMap Address, 9
callStatic
MySQLDAO, 11
construct
Address, 6
Address, 5
\$dao_typeMap, 9
construct, 6
getCity, 7
getld, 7
getName, 7
getState, 7
getStreet1, 7
getStreet2, 7
getZip, 8
setCity, 8
setId, 8
setName, 8
setState, 8
setStreet1, 9
setStreet2, 9 setZip, 9
Setzip, 9
dao_generateBindParameters
MySQLDAO, 11
MySQLDAO, 11 dao_generateFieldList
MySQLDAO, 11 dao_generateFieldList MySQLDAO, 11
MySQLDAO, 11 dao_generateFieldList MySQLDAO, 11 dao_generateParameters
MySQLDAO, 11 dao_generateFieldList MySQLDAO, 11 dao_generateParameters MySQLDAO, 13
MySQLDAO, 11 dao_generateFieldList MySQLDAO, 11 dao_generateParameters MySQLDAO, 13 dao_generateUpdateParameters
MySQLDAO, 11 dao_generateFieldList MySQLDAO, 11 dao_generateParameters MySQLDAO, 13 dao_generateUpdateParameters MySQLDAO, 13
MySQLDAO, 11 dao_generateFieldList MySQLDAO, 11 dao_generateParameters MySQLDAO, 13 dao_generateUpdateParameters MySQLDAO, 13 delete
MySQLDAO, 11 dao_generateFieldList MySQLDAO, 11 dao_generateParameters MySQLDAO, 13 dao_generateUpdateParameters MySQLDAO, 13
MySQLDAO, 11 dao_generateFieldList MySQLDAO, 11 dao_generateParameters MySQLDAO, 13 dao_generateUpdateParameters MySQLDAO, 13 delete
MySQLDAO, 11 dao_generateFieldList MySQLDAO, 11 dao_generateParameters MySQLDAO, 13 dao_generateUpdateParameters MySQLDAO, 13 delete MySQLDAO, 13 getCity
MySQLDAO, 11 dao_generateFieldList MySQLDAO, 11 dao_generateParameters MySQLDAO, 13 dao_generateUpdateParameters MySQLDAO, 13 delete MySQLDAO, 13
MySQLDAO, 11 dao_generateFieldList MySQLDAO, 11 dao_generateParameters MySQLDAO, 13 dao_generateUpdateParameters MySQLDAO, 13 delete MySQLDAO, 13 getCity Address, 7
MySQLDAO, 11 dao_generateFieldList MySQLDAO, 11 dao_generateParameters MySQLDAO, 13 dao_generateUpdateParameters MySQLDAO, 13 delete MySQLDAO, 13 getCity Address, 7 getId
MySQLDAO, 11 dao_generateFieldList MySQLDAO, 11 dao_generateParameters MySQLDAO, 13 dao_generateUpdateParameters MySQLDAO, 13 delete MySQLDAO, 13 getCity Address, 7 getId Address, 7
MySQLDAO, 11 dao_generateFieldList MySQLDAO, 11 dao_generateParameters MySQLDAO, 13 dao_generateUpdateParameters MySQLDAO, 13 delete MySQLDAO, 13 getCity Address, 7 getId Address, 7 getName Address, 7 getState
MySQLDAO, 11 dao_generateFieldList MySQLDAO, 11 dao_generateParameters MySQLDAO, 13 dao_generateUpdateParameters MySQLDAO, 13 delete MySQLDAO, 13 getCity Address, 7 getId Address, 7 getName Address, 7 getState Address, 7
MySQLDAO, 11 dao_generateFieldList MySQLDAO, 11 dao_generateParameters MySQLDAO, 13 dao_generateUpdateParameters MySQLDAO, 13 delete MySQLDAO, 13 getCity Address, 7 getId Address, 7 getName Address, 7 getState Address, 7 getState Address, 7
MySQLDAO, 11 dao_generateFieldList MySQLDAO, 11 dao_generateParameters MySQLDAO, 13 dao_generateUpdateParameters MySQLDAO, 13 delete MySQLDAO, 13 getCity Address, 7 getId Address, 7 getName Address, 7 getState Address, 7 getStreet1 Address, 7
MySQLDAO, 11 dao_generateFieldList MySQLDAO, 11 dao_generateParameters MySQLDAO, 13 dao_generateUpdateParameters MySQLDAO, 13 delete MySQLDAO, 13 getCity Address, 7 getId Address, 7 getName Address, 7 getState Address, 7 getStreet1 Address, 7 getStreet2
MySQLDAO, 11 dao_generateFieldList MySQLDAO, 11 dao_generateParameters MySQLDAO, 13 dao_generateUpdateParameters MySQLDAO, 13 delete MySQLDAO, 13 getCity Address, 7 getId Address, 7 getName Address, 7 getState Address, 7 getStreet1 Address, 7

Address, 8

```
insert
    MySQLDAO, 13
MySQLDAO, 10
    __callStatic, 11
    dao_generateBindParameters, 11
    dao_generateFieldList, 11
    dao_generateParameters, 13
    dao_generateUpdateParameters, 13
    delete, 13
    insert, 13
    update, 14
setCity
    Address, 8
setId
    Address, 8
setName
    Address, 8
setState
    Address, 8
setStreet1
    Address, 9
setStreet2
    Address, 9
setZip
    Address, 9
update
    MySQLDAO, 14
```