

IS113 - Web Application Development

Week 11

Sections G4/G5

K. J. Shim

Agenda Today

- Data Access Object (DAO)
- PHP Data Objects (PDO)
 - PHP's implementation of Data Access Object (PDO)
 - Connect to MySQL
 - CRUD (Create/Read/Update/Delete)

Go to your **webroot**

- Go to eLearn -> Content -> Session 11 -> In Class → **week11.zip**
- Unzip **week11.zip** such that you have the following directory structure locally:

```
webroot -  
|  
|-- is113  
|   |  
|   |-- week11  
|       |  
|       |-- cat  
|           |-- Cat.php  
|           |-- CatDAO.php  
|           |-- ConnectionManager.php  
|           |-- testCat.php  
|           |-- testCatDAO.php  
|           |-- create.sql  
|           |-- display.php  
|           |-- add.php  
|           |-- delete.php  
|           |-- update.php
```

Exercise 1

cat

`data.php`

- Stores **cats** in a **data structure** (e.g. **multi-dimensional Arrays**)
- Each **cat** is an **Associative Array**
- Makes the **Indexed Array** of **cats** available to whoever that needs to display it

requires

`display.php`

- List all **cats** in an HTML table
- Does NOT contain any data
- It simply **displays** content

Cat.php

- Provides **definition** of what a **cat** should be
- This file alone does NOT give us any actual **cats**

data.php

- Borrows the **person definition** from Cat.php
- Creates **new cats** (“Cat objects”)
- Stores the **new cats** in a **data structure** (e.g. **Indexed Array**)
- Makes the **Indexed Array** available to whoever that needs to display **cats**

display.php

- List all **cats** in an HTML table
- Does NOT contain any data
- It simply **displays** content

requires



requires



Cat.php

- Provides **definition** of what a **cat** should be
- This file alone does NOT give us any actual **cats**

CatDAO.php

- Borrows the **cat definition** from Cat.php
- Creates **new cats** (“Cat objects”)
- Stores the **new cats** in a **data structure** (e.g. **Indexed Array**)
- Makes the **Indexed Array** available to whoever that needs to display **cats**
- Makes available **public methods** for fetching **all cats** as well as **certain cats**

display.php

- List all **cats** in an HTML table
- Does NOT contain any data
- It simply **displays** content

requires

requires

/is113/week11/cat/Cat.php

```
1  <?php
2
3  class Cat {
4
5      private $name;
6      private $age;
7      private $gender;
8      private $status;
9
10     public function __construct($name, $age, $gender, $status) {
11         $this->name = $name;
12         $this->age = $age;
13         $this->gender = $gender;
14         $this->status = $status;
15     }
```

All **cats** share the same set of *properties*:

- name (String)
- age (Integer)
- gender ('M' or 'F') // **M**ale, **F**emale
- status ('A' or 'P') // **A**vailable for adoption, **P**ending adoption

testCat.php
contains sample
test cases

/is113/week11/cat/CatDAO.php

```
1  <?php
2
3  require_once 'Cat.php';
4
5  class CatDAO {
6
7      private $cats;
8
9      // Constructor
10     // Pre-populate static data
11     public function __construct() {
12
13         $this->cats = [
14             new Cat('Dirty', 12, 'M', 'A'),
15             new Cat('Filthy', 7, 'F', 'A'),
16             new Cat('Boring', 3, 'M', 'A'),
17             new Cat('Needy', 3, 'M', 'P'),
18             new Cat('Lazy', 1, 'F', 'P')
19         ];
20
21     }
```

CatDAO doesn't know what **cats** look like. Who knows this info? **Cat.php**

→ So, let's import the **Cat** definition

\$cats (private property) is the **ONLY** property of **CatDAO** object

When a **CatDAO object** is created, **CatDAO Class'** constructor **pre-populates** its only property **\$cats** with **static (hard-coded) data**.

→ All cats can be retrieved via **Public Getter method getCats()**

```
24     // Whoever needs $cats, call this Getter method
25     public function getCats() {
26         return $this->cats;
27     }
```

testCatDAO.php
contains sample test cases

- Complete **display.php**. When completed, **display.php** looks like:

Our Cats

Name	Age	Gender	Status
Dirty	12	M	Available
Filthy	7	F	Available
Boring	3	M	Available
Needy	3	M	Pending Adoption
Lazy	1	F	Pending Adoption

Filter by Status:

Available ▼

Filter

Filter by Status:
Available ▼
Available
Pending Adoption

- Remember how **display.php** no longer stores any **data**
- Hmm, who has the **cat data**?
- CatDAO.php** has it!
- To use **CatDAO**, we create a **CatDAO object** first.

```
1 <?php
2
3 require_once 'CatDAO.php';
4 $dao = new CatDAO();
```

- We can then see which **public methods** of **CatDAO Class**... we can **call** from **display.php**.
- Which **public method** can we call... to retrieve all cats?

- Complete the following **Public Method** in **CatDAO Class**:

```
30      // Returns an Indexed Array of cats with a given 'status'
31      public function getCatsByStatus($status) {
32          $return_array = [];
33
34          // YOUR CODE GOES HERE
35
36          return $return_array;
37      }
```

- Test Cases are available in **testCatDAO.php**:

```
16      // Test 1 (Filter by Status)
17      echo '<hr>';
18      echo "&<h1>Filter by Status: status 'A'</h1>";
19      $cats = $dao->getCatsByStatus('A'); // all cats with status 'A'
20      var_dump($cats);
21
22      echo '<hr>';
23      echo "&<h1>Filter by Status: status 'P'</h1>";
24      $cats = $dao->getCatsByStatus('P'); // all cats with status 'P'
25      var_dump($cats);
```

- Complete **display.php**. When completed, **display.php** looks like:

Our Cats

Name	Age	Gender	Status
Dirty	12	M	Available
Filthy	7	F	Available
Boring	3	M	Available
Needy	3	M	Pending Adoption
Lazy	1	F	Pending Adoption

Filter by Status:

Filter by Status:

Available
Pending Adoption

- After the user selects “**Status**” choice in the dropdown menu and presses **Filter SUBMIT** button...
- display.php** must find all cats matching the user-specified **status**
- Hmm... how do I ask **CatDAO Class** to give me **certain cats**?
- Don't you worry! This is WHY we have **Data Access Object (DAO) Classes!**
- They're designed to fetch and return **data** using **filtering criteria** (such as status, gender, etc.).
- Which **public method** can we call... to retrieve **cats** with a certain **status**?

display.php	display.php (after SUBMIT)																
<p>Filter by Status:</p> <p>Available ▼</p> <p>Select Available in the drop-down and click Filter</p>	<h2>Our Cats</h2> <table border="1"><thead><tr><th>Name</th><th>Age</th><th>Gender</th><th>Status</th></tr></thead><tbody><tr><td>Dirty</td><td>12</td><td>M</td><td>Available</td></tr><tr><td>Filthy</td><td>7</td><td>F</td><td>Available</td></tr><tr><td>Boring</td><td>3</td><td>M</td><td>Available</td></tr></tbody></table> <p>Filter by Status:</p> <p>Available ▼</p> <p>Filter</p> <p>You need to pre-select Status (user's input)</p>	Name	Age	Gender	Status	Dirty	12	M	Available	Filthy	7	F	Available	Boring	3	M	Available
Name	Age	Gender	Status														
Dirty	12	M	Available														
Filthy	7	F	Available														
Boring	3	M	Available														

display.php

Filter by Status:

Pending Adoption ▼

Select **Pending Adoption**
in the drop-down and click
Filter

display.php (after SUBMIT)

Our Cats

Name	Age	Gender	Status
Needy	3	M	Pending Adoption
Lazy	1	F	Pending Adoption

Filter by Status:

Pending Adoption ▼

Filter

You need to pre-select **Status** (user's input)

- Modify **display.php** such that it looks like:

Our Cats

Name	Age	Gender	Status
Dirty	12	M	Available
Filthy	7	F	Available
Boring	3	M	Available
Needy	3	M	Pending Adoption
Lazy	1	F	Pending Adoption

Gender: ☐ Female ☐ Male
Status:

- display.php** must find all cats matching user-specified **gender AND status**

Filter by Status:

Available

Available

Pending Adoption

- Complete the following **Public Method** in **CatDAO Class**:

```
50 // Returns an Indexed Array of cats with:
51 //   a given $gender ('M' or 'F')
52 //   AND
53 //   a given $status ('P' or 'A')
54 public function getCatsByGenderStatus($gender, $status) {
55     $return_array = [];
56
57     // YOUR CODE GOES HERE
58
59     return $return_array;
60 }
```

- Test Cases are available in **testCatDAO.php**:

```
41 // Test 3 (Filter by Gender & Status)
42 echo '<hr>';
43 echo "&<h1>Filter by Gender & Status: gender 'F' AND status 'A'</h1>";
44 $cats = $dao->getCatsByGenderStatus('F', 'A'); // all cats with gender 'F' AND status 'A'
45 var_dump($cats);
46
47 echo '<hr>';
48 echo "&<h1>Filter by Gender & Status: gender 'F' AND status 'P'</h1>";
49 $cats = $dao->getCatsByGenderStatus('F', 'P'); // all cats with gender 'F' AND status 'P'
50 var_dump($cats);
```


display.php	display.php (after SUBMIT)								
<p>Gender: <input checked="" type="radio"/> Female <input type="radio"/> Male</p> <p>Status: <input type="text" value="Available"/></p> <p><i>Make the above selections and click Filter</i></p>	<h2>Our Cats</h2> <table border="1"><thead><tr><th>Name</th><th>Age</th><th>Gender</th><th>Status</th></tr></thead><tbody><tr><td>Filthy</td><td>7</td><td>F</td><td>Available</td></tr></tbody></table> <p>Gender: <input checked="" type="radio"/> Female <input type="radio"/> Male</p> <p>Status: <input type="text" value="Available"/></p> <p><input type="button" value="Filter"/></p> <div><p>You need to pre-check Gender (user's input)</p><p>You need to pre-select Status (user's input)</p></div>	Name	Age	Gender	Status	Filthy	7	F	Available
Name	Age	Gender	Status						
Filthy	7	F	Available						

display.php	display.php (after SUBMIT)								
<p>Gender: <input type="radio"/> Female <input checked="" type="radio"/> Male</p> <p>Status: <input type="text" value="Pending Adoption"/></p> <p><i>Make the above selections and click Filter</i></p>	<h2>Our Cats</h2> <table border="1"><thead><tr><th>Name</th><th>Age</th><th>Gender</th><th>Status</th></tr></thead><tbody><tr><td>Needy</td><td>3</td><td>M</td><td>Pending Adoption</td></tr></tbody></table> <p>Gender: <input type="radio"/> Female <input checked="" type="radio"/> Male</p> <p>Status: <input type="text" value="Pending Adoption"/></p> <p><input type="button" value="Filter"/></p> <div><p>You need to pre-check Gender (user's input)</p><p>You need to pre-select Status (user's input)</p></div>	Name	Age	Gender	Status	Needy	3	M	Pending Adoption
Name	Age	Gender	Status						
Needy	3	M	Pending Adoption						

display.php	display.php (after SUBMIT)												
<p>Gender: <input type="radio"/> Female <input checked="" type="radio"/> Male</p> <p>Status: <input type="text" value="Available"/></p>	<h2>Our Cats</h2> <table border="1"><thead><tr><th>Name</th><th>Age</th><th>Gender</th><th>Status</th></tr></thead><tbody><tr><td>Dirty</td><td>12</td><td>M</td><td>Available</td></tr><tr><td>Boring</td><td>3</td><td>M</td><td>Available</td></tr></tbody></table> <p>Gender: <input type="radio"/> Female <input checked="" type="radio"/> Male</p> <p>Status: <input type="text" value="Available"/></p> <p><input type="button" value="Filter"/></p> <div><p>You need to pre-check Gender (user's input)</p><p>You need to pre-select Status (user's input)</p></div>	Name	Age	Gender	Status	Dirty	12	M	Available	Boring	3	M	Available
Name	Age	Gender	Status										
Dirty	12	M	Available										
Boring	3	M	Available										



Cat.php

- Provides **definition** of what a **cat** should be
- This file alone does NOT give us any actual **cats**

requires

CatDAO.php

- Borrows the **cat definition** from Cat.php
- Creates **new cats** (“Cat objects”)
- Stores the **new cats** in a **data structure** (e.g. **Indexed Array**)
- Makes the **Indexed Array** available to whoever that needs to display **cats**
- Makes available **public methods** for fetching **all cats** as well as **certain cats**

Whoops!
The data are static
(hard-coded)!

requires

display.php

- List all **cats** in an HTML table
- Does NOT contain any data
- It simply **displays** content



Cat.php

- Provides **definition** of what a **cat** should be
- This file alone does NOT give us any actual **cats**

CatDAO.php

- Borrows the **cat definition** from Cat.php
- Creates **new cats** ("Cat objects")
- Stores the **new cats** in a **data structure** (e.g. **Indexed Array**)
- Makes the **Indexed Array** available to whoever that needs to display **cats**
- Makes available **public methods** for fetching **all cats** as well as **certain cats**

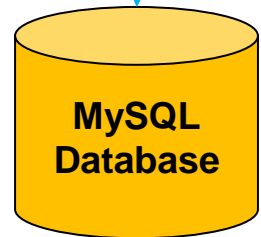
The data are read
from the Database
"dynamically"
(NOT hard-coded)

display.php

- List all **cats** in an HTML table
- Does NOT contain any data
- It simply **displays** content

requires

requires

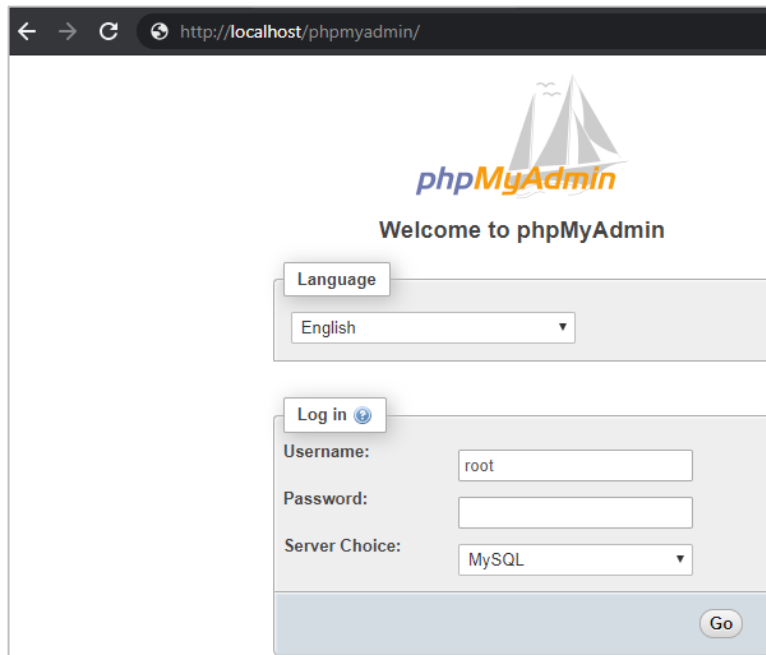


Interact
with

via
SQL

MySQL Database

- Before we modify **CatDAO Class** to make it interact with the **MySQL Database**, we need to first construct the **MySQL Database**.
- The **MySQL Database** is already installed on your local computer (as part the **WAMP Installation**).
- Let's go have a look! In your web browser, go to:
 - <http://localhost/phpmyadmin/>



Default

Username: **root**

Password: **<leave empty>**

http://localhost/phpmyadmin/

The screenshot displays the phpMyAdmin web interface in a browser window. The address bar shows the URL `localhost/phpmyadmin/index.php`. The interface includes a top navigation bar with tabs for Databases, SQL, Status, User accounts, Export, Import, Settings, and More. On the left, there is a sidebar with the phpMyAdmin logo, a 'Current server' dropdown set to 'MySQL', and a list of databases: information_schema, mysql, performance_schema, and sys. The main content area is divided into three panels: General settings, Appearance settings, and Database server. The General settings panel shows options to change the password and set the server connection collation to 'utf8mb4_unicode_ci'. The Appearance settings panel shows the language set to 'English', the theme set to 'pmahomme', and the font size set to '82%'. The Database server panel lists server details: Server: MySQL (127.0.0.1 via TCP/IP), Server type: MySQL, Server connection: SSL is not being used, Server version: 5.7.23 - MySQL Community Server (GPL), Protocol version: 10, User: root@localhost, and Server charset: UTF-8 Unicode (utf8). The Web server panel lists: Apache/2.4.35 (Win64) PHP/7.2.10, Database client version: libmysql - mysqlnd 5.0.12-dev - 20150407 - \$Id: 38fea24f2847fa7519001be390c98ae0acafe3\$, PHP extension: mysqli, curl, mbstring, and PHP version: 7.2.10.

localhost/phpmyadmin/index.php

phpMyAdmin

Current server: MySQL

Recent Favorites

New

- information_schema
- mysql
- performance_schema
- sys

Server: MySQL:3306

Databases SQL Status User accounts Export Import Settings More

General settings

- Change password
- Server connection collation: utf8mb4_unicode_ci

Appearance settings

- Language: English
- Theme: pmahomme
- Font size: 82%
- More settings

Database server

- Server: MySQL (127.0.0.1 via TCP/IP)
- Server type: MySQL
- Server connection: SSL is not being used
- Server version: 5.7.23 - MySQL Community Server (GPL)
- Protocol version: 10
- User: root@localhost
- Server charset: UTF-8 Unicode (utf8)

Web server

- Apache/2.4.35 (Win64) PHP/7.2.10
- Database client version: libmysql - mysqlnd 5.0.12-dev - 20150407 - \$Id: 38fea24f2847fa7519001be390c98ae0acafe3\$
- PHP extension: mysqli, curl, mbstring
- PHP version: 7.2.10

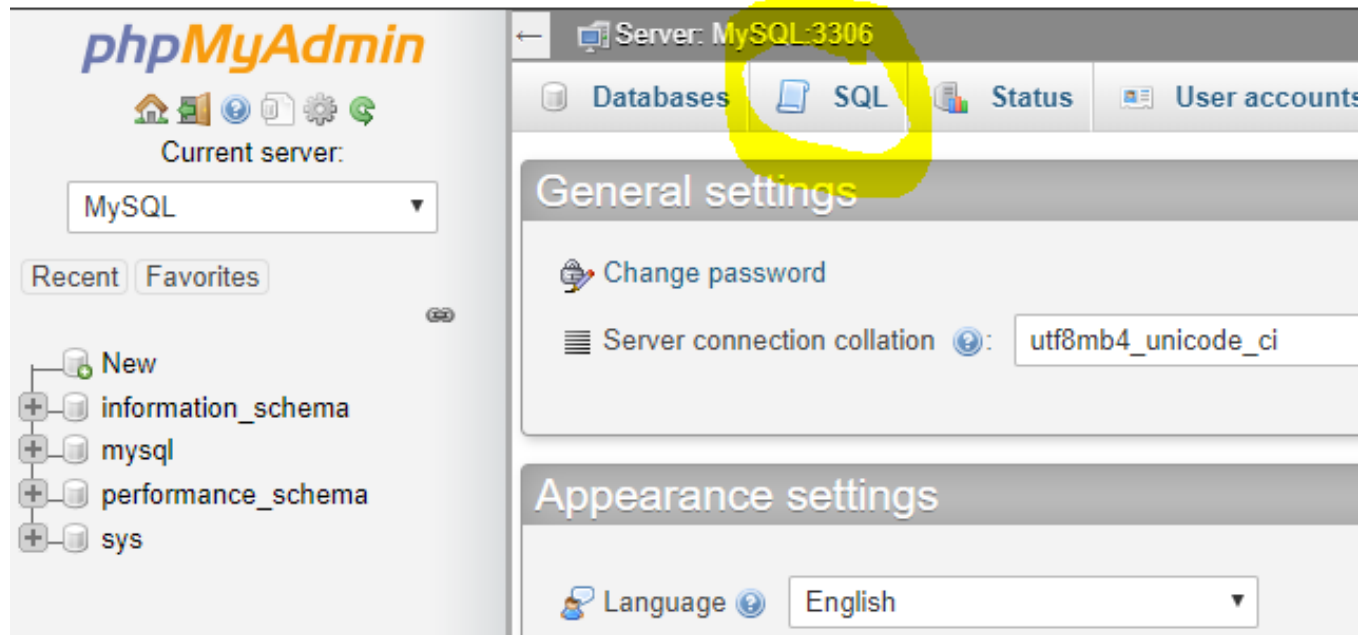
- This file contains a list of **SQL Commands**.
- You need to **load** this file content – to create a new **schema**, new **table** and **pre-populate** the table with some content.
- For demo purposes, we did NOT create any **primary key**. In this exercise, you may **assume** that “**name**” column is **unique**.

```
1  drop database if exists animals;
2  create database animals;
3
4  use animals;
5
6  create table cat (
7      name varchar(50),
8      age integer,
9      gender char(1),
10     status char(1)
11 );
12
13 insert into cat (name, age, gender, status) values ('Dirty', 12, 'M', 'A');
14 insert into cat (name, age, gender, status) values ('Filthy', 7, 'F', 'A');
15 insert into cat (name, age, gender, status) values ('Boring', 3, 'M', 'A');
16 insert into cat (name, age, gender, status) values ('Needy', 3, 'M', 'P');
17 insert into cat (name, age, gender, status) values ('Lazy', 1, 'F', 'P');
18 insert into cat (name, age, gender, status) values ('Stinky', 4, 'F', 'A');
19 insert into cat (name, age, gender, status) values ('Sad', 3, 'F', 'A');
```


PHPMyAdmin

3

- In **PHPMyAdmin**, click **SQL** menu option at the top.



- Copy the content from **create.sql** into the textarea:

Run SQL query/queries on table animals.cat: ⓘ

```
1 drop database if exists animals;
2 create database animals;
3
4 use animals;
5
6 create table cat (
7   name varchar(50),
8   age integer,
9   gender char(1),
10  status char(1)
11 );
12
13 insert into cat (name, age, gender, status) values ('Dirty', 12, 'M', 'A');
14 insert into cat (name, age, gender, status) values ('Filthy', 7, 'F', 'A');
15 insert into cat (name, age, gender, status) values ('Boring', 3, 'M', 'A');
16 insert into cat (name, age, gender, status) values ('Needy', 3, 'M', 'P');
17 insert into cat (name, age, gender, status) values ('Lazy', 1, 'F', 'P');
18 insert into cat (name, age, gender, status) values ('Stinky', 4, 'F', 'A');
19 insert into cat (name, age, gender, status) values ('Sad', 3, 'F', 'A');
```

Columns

- name
- age
- gender
- status

SELECT * SELECT INSERT UPDATE DELETE Clear

Format Get auto-saved query

☐ Bind parameters ⓘ

[Delimiter ;] ☒ Show this query here again ☐ Retain query box ☐ Rollback when finished ☒ Enable foreign key checks

Go

- Press **Go** button

- You should be able to see the following status messages indicating successful execution of SQL queries:

The screenshot displays the PHPMyAdmin interface with a top navigation bar containing links for Browse, Structure, SQL, Search, Insert, Export, Import, Privileges, and a More dropdown menu. Below the navigation bar is a link labeled "Show query box". The main area shows a list of six executed SQL queries, each with a green checkmark icon indicating success. The status messages for each query are: "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0083 seconds.)", "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0023 seconds.)", "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0006 seconds.)", "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0223 seconds.)", "1 row inserted. (Query took 0.0013 seconds.)", and "1 row inserted. (Query took 0.0004 seconds.)". Each query is followed by its SQL code in a light blue box, and each code block has links for "[Edit inline]", "[Edit]", and "[Create PHP code]".

[Show query box](#)

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0083 seconds.)

```
drop database if exists animals
```

[Edit inline] [Edit] [Create PHP code]

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0023 seconds.)

```
create database animals
```

[Edit inline] [Edit] [Create PHP code]

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0006 seconds.)

```
use animals
```

[Edit inline] [Edit] [Create PHP code]

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0223 seconds.)

```
create table cat ( name varchar(50), age integer, gender char(1), status char(1) )
```

[Edit inline] [Edit] [Create PHP code]

✓ 1 row inserted. (Query took 0.0013 seconds.)

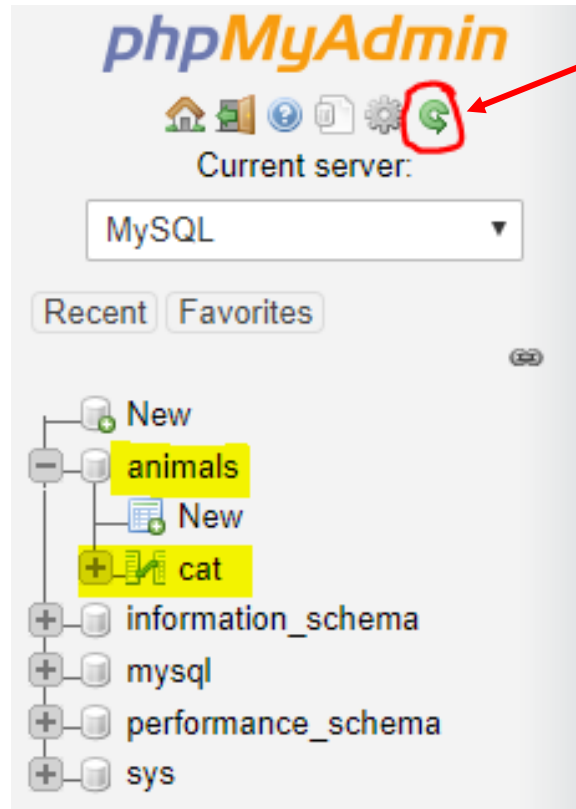
```
insert into cat (name, age, gender, status) values ('Dirty', 12, 'M', 'A')
```

[Edit inline] [Edit] [Create PHP code]

✓ 1 row inserted. (Query took 0.0004 seconds.)

```
insert into cat (name, age, gender, status) values ('Filthy', 7, 'F', 'A')
```

- In the left panel, you should be able to see a new schema **animals**.
- Click on the + sign and it will expand the **animals** schema – and display the newly created **table** called **cat**.
 - If unable to see, press the **Refresh** button at the top.



- Click on the new table name **cat**.
 - In the right panel, you should be able to see the **data**.

The screenshot shows the PHPMyAdmin interface for a MySQL server. The top navigation bar includes links for Browse, Structure, SQL, Search, Insert, Export, Import, Privileges, and More. A warning message states: "Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available." Below this, a green status bar indicates "Showing rows 0 - 6 (7 total, Query took 0.0019 seconds.)". The SQL query editor shows the query: `SELECT * FROM `cat``. Below the query editor, there are checkboxes for "Show all" and "Number of rows: 25", and a "Filter rows: Search this table" input field. The table data is displayed below the query editor.

name	age	gender	status
Dirty	12	M	A
Filthy	7	F	A
Boring	3	M	A
Needy	3	M	P
Lazy	1	F	P
Stinky	4	F	A
Sad	3	F	A

- To run a **Query**, click **SQL** menu option. In the text area, you can write any SQL query. When done writing the query, press **Go** button.

The screenshot shows the PHPMyAdmin interface with the 'SQL' tab selected. The title bar reads 'Run SQL query/queries on table animals.cat:'. The SQL query input area contains the query: `SELECT * FROM `cat` WHERE gender = 'F'`. To the right, the 'Columns' list shows: name, age, gender, status. Below the query area, a green status bar indicates: 'Showing rows 0 - 3 (4 total, Query took 0.0009 seconds.)'. Below this, the query is repeated. At the bottom, there are buttons for 'UPDATE', 'DELETE', and 'Clear', along with a '<<' button. At the very bottom, there are checkboxes for 'Show all', 'Number of rows: 25', 'Filter row', 'here again', 'Retain query box', 'Rollback when finished', and 'Enable foreign key checks' (checked), followed by a 'Go' button.

Showing rows 0 - 3 (4 total, Query took 0.0009 seconds.)

```
SELECT * FROM `cat` WHERE gender = 'F'
```

name	age	gender	status
Filthy	7	F	A
Lazy	1	F	P
Stinky	4	F	A
Sad	3	F	A

Query Execution Results

Keep a copy of CatDAO.php...

- In the next exercise, we will modify **CatDAO class**
- Make a copy of the current **CatDAO.php** and keep it somewhere:
 - e.g. /is113/week11/cat/archive/CatDAO.php

Let's **modify...**

CatDAO.php

so that it pulls data from
MySQL DB

/is113/week11/cat/CatDAO.php


CatDAO will NOT hard-code any static data anymore.

Instead, it will now **query** MySQL database to pull required **data**.

CatDAO doesn't know the details of **Database Connection**.

ConnectionManager Class knows the details!

```
1  <?php
2
3  require_once 'Cat.php';
4  require_once 'ConnectionManager.php';
5
6  class CatDAO {
```



/is113/week11/cat/ConnectionManager.php

```
1  <?php
2
3  class ConnectionManager {
4
5      public function connect() {
6          $servername = 'localhost';
7          $username = 'root';
8          $password = '';
9          $dbname = 'animals';
10         $port = '3306';
11
12         // Create connection
13         $pdoObject = new PDO(
14             "mysql:host=$servername;dbname=$dbname;port=$port",
15             $username,
16             $password);
17
18         $pdoObject->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
19         // if fail, exception will be thrown
20
21         // Return connection object
22         return $pdoObject; // PDO object (containing MySQL connection info)
23     }
24 }
25 }
```

The values can
change depending on
your computer setup

MAMP users...
**Your default MySQL port may
NOT be 3306. Please check!**

In case of issues on
the MySQL
Database-side, our
code will throw
exceptions (**error
messages**) and we
should be able to see
it in **web browser** (for
**debugging
purposes**).

To connect to the MySQL database:

- Create **ConnectionManager Class** object
- Call **connect()** public method
- The method returns a **PHP Data Object (PDO)**

CRUD

READ

```
SELECT ... FROM ...
```

getCats()

```
public function getCats() {  
  
    // STEP 1  
    $connMgr = new ConnectionManager();  
    $pdo = $connMgr->connect(); // PDO object  
  
    // STEP 2  
    $sql = "SELECT name, age, gender, status FROM cat";  
    $stmt = $pdo->prepare($sql); // PDOStatement object  
  
    // STEP 3  
    $stmt->execute(); // RUN SQL  
    $stmt->setFetchMode(PDO::FETCH_ASSOC);  
  
    // STEP 4  
    $cats = []; // Array of Cat objects, empty now  
    while ( $row = $stmt->fetch() ) {  
        $cat = new Cat(  
            $row['name'],  
            $row['age'],  
            $row['gender'],  
            $row['status']  
        ); // new Cat object  
        $cats[] = $cat; // add Cat object to ret array  
    }  
  
    // STEP 5  
    $stmt = null; // clear memory  
    $pdo = null; // clear memory  
  
    // STEP 6  
    return $cats;  
}
```

1) Create a **ConnectionManager** object

- Call public method **connect()**

- **\$pdo** is a **PDO** object

2) Create & prepare an **SQL Query**

- **\$stmt** is an **PDOStatement** object

3) Run the query

- Instruct PDO to return **each row** (of the query results) as an **Associative array**

4) Fetch ONE row at a time

- **\$row** is an **Associative Array**

- As we fetch each row from **cat** table, we create a new **Cat object** out of it.

- We then add this new **Cat object** into an **Indexed Array** – one at a time.

5) Clear **PDO/PDOStatement** objects

6) Return an **Indexed Array** of **Cat** objects

getCats()

- Test Cases are available in **testCatDAO.php**:

```
// Test 0 (Get ALL cats)
echo '<hr>';
echo '<h1>Fetch ALL cat rows - each as an Associative Array</h1>';
$cats = $dao->getCats(); // all cats
var_dump($cats);

echo '<hr>';
echo '<h1>Fetch ALL cat rows - each as an Indexed Array</h1>';
$cats = $dao->getCats2(); // all cats
var_dump($cats);
```

Check **getCats2()** method in **CatDAO.php**
Steps 3-4 are different from **getCats()** method

/is113/week11/cat/display.php

- Open **display.php**. It should display **all cats** as follows:

Our Cats

Name	Age	Gender	Status
Dirty	12	M	Available
Filthy	7	F	Available
Boring	3	M	Available
Needy	3	M	Pending Adoption
Lazy	1	F	Pending Adoption
Stinky	4	F	Available
Sad	3	F	Available

Filter by Status:

Available ▼

Filter

getCatsByStatus(\$status)

```
110 public function getCatsByStatus($status) {
111     // STEP 1
112     $connMgr = new ConnectionManager();
113     $pdo = $connMgr->connect(); // PDO object
114
115     // STEP 2
116     $sql = "SELECT
117         name, age, gender, status
118     FROM
119         cat
120     WHERE
121         status = :status ";
122     $stmt = $pdo->prepare($sql);
123     $stmt->bindParam(':status', $status, PDO::PARAM_STR);
124
125     // STEP 3
126     $stmt->execute();
127     $stmt->setFetchMode(PDO::FETCH_ASSOC);
128
129     // STEP 4
130     $cats = [];
131     while ($row = $stmt->fetch()) {
132         $cat = new Cat(
133             $row['name'],
134             $row['age'],
135             $row['gender'],
136             $row['status']
137         );
138         $cats[] = $cat;
139     }
140
141     // STEP 5
142     $stmt = null;
143     $pdo = null;
144
145     // STEP 6
146     return $cats;
147 }
```

1) Create a **ConnectionManager** object

- Call public method **connect()**. **\$pdo** is a **PDO object**

2) Create & prepare an **SQL Query**

- The query looks a bit weird... what is **:status** ??? It is a **placeholder**.
- We use **bindParam()** to **bind** an **actual value** (in the parameter **\$status**) to **:status**
- Since **\$status** is a **STRING**, we tell **bindParam()** so... via **PDO::PARAM_STR** constant variable.
- For **Integers**, it is **PDO::PARAM_INT**
- For **Decimals/Floats**, it is **PDO::PARAM_STR**

3) Run the query

- Return **each row** as an **Associative array**

4) Fetch ONE row at a time

- As we fetch each row from **cat** table, we create a new **Cat object** out of it. We then add this new **Cat object** into an **Indexed Array** – one at a time.

5) Clear **PDO/PDOStatement** objects

6) Return an **Indexed Array** of **Cat objects**

```
// STEP 2
$sql = "SELECT
        name, age, gender, status
      FROM
        cat
     WHERE
        status = :status ";

$stmt = $pdo->prepare($sql);
$stmt->bindParam(':status', $status, PDO::PARAM_STR);
```

- **:status** is a 'placeholder'
- Use **bindParam** to bind parameter \$status' value to **:status**

- Test Cases are available in **testCatDAO.php**:

```
// Test 1 (Filter by Status)
echo '<hr>';
echo "<h1>Filter by Status: status 'A'</h1>";
$cats = $dao->getCatsByStatus('A'); // all cats with status 'A'
var_dump($cats);

echo '<hr>';
echo "<h1>Filter by Status: status 'P'</h1>";
$cats = $dao->getCatsByStatus('P'); // all cats with status 'P'
var_dump($cats);
```



```
// STEP 2
$sql = "SELECT
        name, age, gender, status
      FROM
        cat
     WHERE
        gender = :gender ";

$stmt = $pdo->prepare($sql);
$stmt->bindParam(':gender', $gender, PDO::PARAM_STR);
```

- **:gender** is a 'placeholder'
- Use **bindParam** to bind parameter \$gender's value to **:gender**

- Test Cases are available in **testCatDAO.php**:

```
33 // Test 2 (Filter by Gender)
34 echo '<hr>';
35 echo "<h1>Filter by Gender: gender 'F'</h1>";
36 $cats = $dao->getCatsByGender('F'); // all cats with gender 'F'
37 var_dump($cats);
38
39 echo '<hr>';
40 echo "<h1>Filter by Gender: gender 'M'</h1>";
41 $cats = $dao->getCatsByGender('M'); // all cats with gender 'M'
42 var_dump($cats);
```

```
// STEP 2
// Prepare SQL statement
$sql = "SELECT *
      FROM
        cat
      WHERE
        gender = :gender
      AND
        status = :status ";

$stmt = $pdo->prepare($sql);
$stmt->bindParam(':gender', $gender, PDO::PARAM_STR);
$stmt->bindParam(':status', $status, PDO::PARAM_STR);
```

- **:gender** and **:status** are 'placeholders'
- Use **bindParam** to bind parameter \$gender's value to **:gender** and parameter \$status' value to **:status**

- Test Cases are available in **testCatDAO.php**:

```
45 // Test 3 (Filter by Gender & Status)
46 echo '<hr>';
47 echo "<h1>Filter by Gender & Status: gender 'F' AND status 'A'</h1>";
48 $cats = $dao->getCatsByGenderStatus('F', 'A'); // all cats with gender 'F' AND status 'A'
49 var_dump($cats);
50
51 echo '<hr>';
52 echo "<h1>Filter by Gender & Status: gender 'F' AND status 'P'</h1>";
53 $cats = $dao->getCatsByGenderStatus('F', 'P'); // all cats with gender 'F' AND status 'P'
54 var_dump($cats);
55
56 echo '<hr>';
57 echo "<h1>Filter by Gender & Status: gender 'M' AND status 'A'</h1>";
58 $cats = $dao->getCatsByGenderStatus('M', 'A'); // all cats with gender 'M' AND status 'A'
59 var_dump($cats);
60
61 echo '<hr>';
62 echo "<h1>Filter by Gender & Status: gender 'M' AND status 'P'</h1>";
63 $cats = $dao->getCatsByGenderStatus('M', 'P'); // all cats with gender 'M' AND status 'P'
64 var_dump($cats);
```

No changes needed in display.php

Test the below test case – see if it works as expected

display.php	display.php (after SUBMIT)												
Gender: <input type="radio"/> Female <input checked="" type="radio"/> Male Status: <input type="text" value="Available"/>	<h2>Our Cats</h2> <table border="1"><thead><tr><th>Name</th><th>Age</th><th>Gender</th><th>Status</th></tr></thead><tbody><tr><td>Dirty</td><td>12</td><td>M</td><td>Available</td></tr><tr><td>Boring</td><td>3</td><td>M</td><td>Available</td></tr></tbody></table> <p>Gender: <input type="radio"/> Female <input checked="" type="radio"/> Male Status: <input type="text" value="Available"/></p> <p><input type="button" value="Filter"/></p>	Name	Age	Gender	Status	Dirty	12	M	Available	Boring	3	M	Available
Name	Age	Gender	Status										
Dirty	12	M	Available										
Boring	3	M	Available										

```
// STEP 2
$sql = "SELECT
    name, age, gender, status
FROM
    cat
WHERE";

$have_status = False;
$have_gender = False;
$have_max_age = False;

if( $status == '-' ) {
    $sql .= " status IN ('A', 'P')";
}
else {
    $sql .= " status = :status";
    $have_status = True;
} // Status

if( $gender == 'M' || $gender == 'F' ) {
    $sql .= " AND gender = :gender";
    $have_gender = True;
} // Gender

if( $max_age != '' ) {
    $sql .= " AND age <= :max_age";
    $have_max_age = True;
} // Max Age
```

```
$stmt = $pdo->prepare($sql);

if( $have_status )
    $stmt->bindParam(':status', $status, PDO::PARAM_STR);
if( $have_gender )
    $stmt->bindParam(':gender', $gender, PDO::PARAM_STR);
if( $have_max_age )
    $stmt->bindParam(':max_age', $max_age, PDO::PARAM_INT);
```

- Test Cases are available in **testCatDAO.php**:

```
// Test 4 (Filter by Status, Gender, Max Age)
echo '<hr>';
echo "<h1>Filter by Status & Gender & Max Age</h1>";
echo "<h2>status 'A' AND gender 'F' and max_age 8</h2>";
$status = 'A';
$gender = 'F';
$max_age = 8;
$cats = $dao->getCatsFilter($status, $gender, $max_age);
var_dump($cats);

echo '<hr>';
echo "<h1>Filter by Status & Gender & Max Age</h1>";
echo "<h2>status 'P' AND gender 'M' and max_age 10</h2>";
$status = 'P';
$gender = 'M';
$max_age = 10;
$cats = $dao->getCatsFilter($status, $gender, $max_age);
var_dump($cats);

echo '<hr>';
echo "<h1>Filter by Status & Gender & Max Age</h1>";
echo "<h2>status 'P' AND gender 'M' and max_age 5</h2>";
$status = 'P';
$gender = 'M';
$max_age = 5;
$cats = $dao->getCatsFilter($status, $gender, $max_age);
var_dump($cats);
```

isCatFound(\$name)

```
// Find a cat by $name
// Return TRUE (if the cat is found) or FALSE (otherwise)
public function isCatFound($name) {
    // STEP 1
    $connMgr = new ConnectionManager();
    $pdo = $connMgr->connect(); // PDO object

    // STEP 2
    $sql = "SELECT
            *
        FROM
            cat
        WHERE name = :name
    ";
    $stmt = $pdo->prepare($sql); // SQLStatement object
    $stmt->bindParam(':name', $name, PDO::PARAM_STR);

    // STEP 3
    $stmt->execute(); // RUN SQL

    // STEP 4
    $isFound = False;
    if( $stmt->rowCount() > 0 ) {
        $isFound = True;
    }

    // STEP 5
    $stmt = null; // clear memory
    $pdo = null; // clear memory

    // STEP 6
    return $isFound;
}
```

1) Create a **ConnectionManager** object
- Call public method **connect()**

2) Create & prepare an **SQL Query**
- bind parameter value(s) to placeholders

3) Run the query
- **execute()** returns **TRUE** (if query ran fine) or **FALSE** (SQL INSERTION was unsuccessful)

4) How do you determine whether the cat is found in **cat** table or not?

5) Clear **PDO/PDOStatement** objects

6) Return **TRUE** if the cat is found. **FALSE** otherwise.

/is113/week11/cat/CatDAO.php

isCatFound(\$name)

9

- Test Cases are available in **testCatDAO.php**:

```
// Test 5 - (Database Select - Find Cat by Name - Is he/she found?)
echo '<hr>';
echo "<h1>Find Cat by Name - Is he/she found?</h1>";
$name = 'Filthy';
if( $dao->isCatFound($name) ) {
    echo "<font color='blue'>
        Cat named <b>$name</b> exists in database!
    </font>";
}
else {
    echo "<font color='red'>
        Oh no! Cat named <b>$name</b> does NOT exist in database!
    </font>";
}

echo '<hr>';
echo "<h1>Find Cat by Name - Is he/she found?</h1>";
$name = 'Smarty';
if( $dao->isCatFound($name) ) {
    echo "<font color='blue'>
        Cat named <b>$name</b> exists in database!
    </font>";
}
else {
    echo "<font color='red'>
        Oh no! Cat named <b>$name</b> does NOT exist in database!
    </font>";
}
```

getCatByName(\$name)

```
public function getCatByName($name) {  
    // STEP 1  
    $connMgr = new ConnectionManager();  
    $pdo = $connMgr->connect(); // PDO object  
  
    // STEP 2  
    $sql = "SELECT  
        *  
    FROM  
        cat  
    WHERE name = :name  
    ";  
    $stmt = $pdo->prepare($sql); // SQLStatement object  
    $stmt->bindParam(':name', $name, PDO::PARAM_STR);  
  
    // STEP 3  
    $stmt->execute(); // RUN SQL  
    $stmt->setFetchMode(PDO::FETCH_ASSOC);  
  
    // STEP 4  
    $cat = NULL;  
    if( $stmt->rowCount() > 0 ) {  
        // Assume cat's name is unique, there should only be 1 row  
        $row = $stmt->fetch(); // Fetch row  
  
        // Make a new Cat object out of $row  
        $cat = new Cat(  
            $row['name'],  
            $row['age'],  
            $row['gender'],  
            $row['status']  
        ); // new Cat object  
    }  
  
    // STEP 5  
    $stmt = null; // clear memory  
    $pdo = null; // clear memory  
  
    // STEP 6  
    return $cat;  
}
```

1) Create a **ConnectionManager** object

- Call public method **connect()**

2) Create & prepare an **SQL Query**

- bind parameter value(s) to placeholders

3) Run the query

- **execute()** returns **TRUE** (if query ran fine) or **FALSE** (insertion did not go through)

4) How do you determine whether the cat is found in **cat** table or not?

5) Clear **PDO/PDOStatement** objects

6) Return a new **Cat object** containing the details of the found cat. Return **NULL** otherwise.

getCatByName(\$name)

- Test Cases are available in **testCatDAO.php**:

```
// Test 6 - (Database Select - Find Cat by Name - Get Cat object)
echo '<hr>';
echo "<h1>Find Cat by Name - Get Cat object</h1>";
$name = 'Filthy';
$cat = $dao->getCatByName($name);
if( $cat ) {
    echo "<font color='blue'>
        Cat named <b>$name</b> exists in database!
    </font>";
    var_dump($cat);
}
else {
    echo "<font color='red'>
        Oh no! Cat named <b>$name</b> does NOT exist in database!
    </font>";
}

echo '<hr>';
echo "<h1>Find Cat by Name - Get Cat object</h1>";
$name = 'Smarty';
$cat = $dao->getCatByName($name);
if( $cat ) {
    echo "<font color='blue'>
        Cat named <b>$name</b> exists in database!
    </font>";
    var_dump($cat);
}
else {
    echo "<font color='red'>
        Oh no! Cat named <b>$name</b> does NOT exist in database!
    </font>";
}
```

CRUD

CREATE

INSERT INTO . . .

/is113/week11/cat/CatDAO.php

add(\$name, \$age, \$gender)

11

```
public function add($name, $age, $gender) {  
    // For new cats, default is 'A' (available)  
    $status = 'A';  
  
    // STEP 1  
    $connMgr = new ConnectionManager();  
    $pdo = $connMgr->connect(); // PDO object  
  
    // STEP 2  
    $sql = "INSERT INTO CAT  
           ( name, age, gender, status )  
           VALUES  
           ( :name, :age, :gender, :status )";  
  
    $stmt = $pdo->prepare($sql);  
    $stmt->bindParam(':name', $name, PDO::PARAM_STR);  
    $stmt->bindParam(':age', $age, PDO::PARAM_INT);  
    $stmt->bindParam(':gender', $gender, PDO::PARAM_STR);  
    $stmt->bindParam(':status', $status, PDO::PARAM_STR);  
  
    // STEP 3  
    $isOk = $stmt->execute();  
  
    // STEP 4  
    $stmt = null;  
    $pdo = null;  
  
    // STEP 5  
    return $isOk;  
}
```

1) Create a **ConnectionManager** object
- Call public method **connect()**

2) Create & prepare an **SQL Query**
- bind parameter values to placeholders

3) Run the query
- **execute()** returns **TRUE** (if query ran fine) or **FALSE** (insertion did not go through)
- Save the return value in **\$isOk** variable

4) Clear **PDO/PDOStatement** objects

5) Return **\$isOk**, the result of query execution

/is113/week11/cat3/CatDAO.php

add(\$name, \$age, \$gender)

11

- Test Cases are available in **testCatDAO.php**:

```
// Test 7 - (Database Add)
echo '<hr>';
echo "&<h1>Database Add</h1>";
$name = 'Ugly';
$age = 8;
$gender = 'M';
if( $dao->add($name, $age, $gender) ) {
    echo "<font color='blue'>
        Your cat $name has been added successfully!<br>
        Let's see all the cats in the database!
    </font>";
    var_dump( $dao->getCats() );
}
else {
    echo "<font color='red'>Oh no! Your cat couldn't be added!</font>";
}
```

- Please note that **cat** table does NOT have any **primary key** at the moment.
- Thus**, technically, you can insert **duplicates** (e.g. cats with the same name and other attributes).
- Assume that **name** is unique.

- How do you know if the **Database INSERTION** worked?
- Go to <http://localhost/phpmyadmin> and check!

phpMyAdmin

Current server: MySQL

Recent Favorites

Server: MySQL:3306 » Database: animals

Browse Structure SQL

⚠ Current selection does not contain a unique index. Copy and Delete features are not available.

✓ Showing rows 0 - 7 (8 total, Query took 0.0001 sec)

`SELECT * FROM `cat``

☐ Profiling [Edit inline] [Edit] [Explain]

☐ Show all | Number of rows: 25

+ Options

name	age	gender	status
Dirty	12	M	P
Filthy	7	F	A
Boring	3	M	A
Needy	3	M	P
Lazy	1	F	P
Stinky	4	F	A
Sad	3	F	A
Ugly	8	M	A

Testing CRUD & Resetting the Database

- For **testing/debugging purposes**, you can always go **re-set** the database.
- Simply copy/paste the content from **create.sql** (or any SQL file provided to you) into **PHPMysqlAdmin** → **SQL** and run the queries (click **Go** button).
- Check the table content. In our case, there should be 7 cats.

create.sql

```
drop database if exists animals;
create database animals;

use animals;

create table cat (
  name varchar(50),
  age integer,
  gender char(1),
  status char(1)
);

insert into cat (name, age, gender, status) values ('Dirty', 12, 'M', 'A');
insert into cat (name, age, gender, status) values ('Filthy', 7, 'F', 'A');
insert into cat (name, age, gender, status) values ('Boring', 3, 'M', 'A');
insert into cat (name, age, gender, status) values ('Needy', 3, 'M', 'P');
insert into cat (name, age, gender, status) values ('Lazy', 1, 'F', 'P');
insert into cat (name, age, gender, status) values ('Stinky', 4, 'F', 'A');
insert into cat (name, age, gender, status) values ('Sad', 3, 'F', 'A');
```

Copy

PHPMysqlAdmin → SQL

The screenshot shows the PHPMysqlAdmin interface with the 'SQL' tab selected. The query area contains the following SQL code:

```
create database animals;
use animals;
create table cat (
  name varchar(50),
  age integer,
  gender char(1),
  status char(1)
);
insert into cat (name, age, gender, status) values ('Dirty', 12, 'M', 'A');
insert into cat (name, age, gender, status) values ('Filthy', 7, 'F', 'A');
insert into cat (name, age, gender, status) values ('Boring', 3, 'M', 'A');
insert into cat (name, age, gender, status) values ('Needy', 3, 'M', 'P');
insert into cat (name, age, gender, status) values ('Lazy', 1, 'F', 'P');
insert into cat (name, age, gender, status) values ('Stinky', 4, 'F', 'A');
insert into cat (name, age, gender, status) values ('Sad', 3, 'F', 'A');
```

The interface includes buttons for 'SELECT *', 'SELECT', 'INSERT', 'UPDATE', and 'DELETE'. Below these are 'Clear', 'Format', and 'Get auto-saved query' buttons. At the bottom, there are checkboxes for 'Bind parameters', 'Show this query here again', 'Retain query box', and 'Rollback when finished', along with a 'Go' button.

All Cats

Name	Age	Gender	Status
Dirty	12	M	Available
Filthy	7	F	Available
Boring	3	M	Available
Needy	3	M	Pending Adoption
Lazy	1	F	Pending Adoption
Stinky	4	F	Available
Sad	3	F	Available

Add a new cat

Name:

Age:

Gender: ☐ Female ☐ Male

Complete **add.php**

At the top, display all cats

- Re-use your code from **display.php**

This is a **form**

- action='add.php', method='POST'

- User must key in & select all 3 fields.

- Upon clicking **Add Cat SUBMIT button**, **add.php** must call **add()** public method in **CatDAO Class**.

add.php

All Cats

Name	Age	Gender	Status
Dirty	12	M	Available
Filthy	7	F	Available
Boring	3	M	Available
Needy	3	M	Pending Adoption
Lazy	1	F	Pending Adoption
Stinky	4	F	Available
Sad	3	F	Available

Add a new cat

Name:

Age:

Gender: ☒ Female ☐ Male

add.php
(after SUBMIT)

All Cats

Name	Age	Gender	Status
Dirty	12	M	Available
Filthy	7	F	Available
Boring	3	M	Available
Needy	3	M	Pending Adoption
Lazy	1	F	Pending Adoption
Stinky	4	F	Available
Sad	3	F	Available
Selena	12	F	Available

A new cat has been added

Add a new cat

Name:

Age:

Gender: ☐ Female ☐ Male

Also, check the **MySQL DB "cat" table**

Cat "Selena" should have been inserted there.

CRUD

DELETE

DELETE FROM . . . WHERE . . .

delete(\$name)

```
public function delete($name) {  
    // STEP 1  
    $connMgr = new ConnectionManager();  
    $pdo = $connMgr->connect(); // PDO object  
  
    // STEP 2  
    $sql = "DELETE  
           FROM cat  
           WHERE  
             name = :name";  
  
    $stmt = $pdo->prepare($sql);  
    $stmt->bindParam(':name', $name, PDO::PARAM_STR);  
  
    // STEP 3  
    $isOk = $stmt->execute();  
  
    // STEP 4  
    $stmt = null;  
    $pdo = null;  
  
    // STEP 5  
    return $isOk;  
}
```

1) Create a **ConnectionManager** object
- Call public method **connect()**

2) Create & prepare an **SQL Query**

3) Run the query
- **execute()** returns **TRUE** (if query ran fine) or **FALSE** (deletion did not go through)
- Save the return value in **\$isOk** variable

4) Clear **PDO/PDOStatement** objects

5) Return **\$isOk**, the result of query execution

- Test Cases are available in **testCatDAO.php**:

```
// Test 8 - (Database Delete)
echo '<hr>';
echo "&<h1>Database Delete</h1>";
$name = 'Ugly';
if( $dao->delete($name) ) {
    echo "<font color='blue'>
        Your cat $name has been deleted successfully!
    </font>";
    var_dump( $dao->getCats() );
}
else {
    echo "<font color='red'>
        Oh no! Your cat $name couldn't be deleted!
    </font>";
}
```

- How do you know if the **Database DELETION** worked?
- Go to <http://localhost/phpmyadmin> and check!

name	age	gender	status
Dirty	12	M	A
Filthy	7	F	A
Boring	3	M	A
Needy	3	M	P
Lazy	1	F	P
Stinky	4	F	A
Sad	3	F	A
Angry	5	F	A

Cat '**Ugly**' is no longer in the **cat** table.

Delete a cat

Name	Age	Gender	Status	Delete Link
Dirty	12	M	Available	Delete
Filthy	7	F	Available	Delete
Boring	3	M	Available	Delete
Needy	3	M	Pending Adoption	Delete
Lazy	1	F	Pending Adoption	Delete
Stinky	4	F	Available	Delete
Sad	3	F	Available	Delete

Complete delete.php

This is a **form**

- action='delete.php', method='GET'
- There is **NO SUBMIT button** in this form.
- However... check out the **URL** of the "Delete" hyperlink.
- Clicking on the **hyperlink** is equivalent to **submitting a form** with an **input field** "name" via **GET** method.

`Delete`

delete.php

Delete a cat

Name	Age	Gender	Status	Delete Link
Dirty	12	M	Available	Delete
Filthy	7	F	Available	Delete
Boring	3	M	Available	Delete
Needy	3	M	Pending Adoption	Delete
Lazy	1	F	Pending Adoption	Delete
Stinky	4	F	Available	Delete
Sad	3	F	Available	Delete
Selena	12	F	Available	Delete

delete.php

(after clicking "Delete" hyperlink)

Cat Stinky has been deleted

Delete a cat

Name	Age	Gender	Status	Delete Link
Dirty	12	M	Available	Delete
Filthy	7	F	Available	Delete
Boring	3	M	Available	Delete
Needy	3	M	Pending Adoption	Delete
Lazy	1	F	Pending Adoption	Delete
Sad	3	F	Available	Delete
Selena	12	F	Available	Delete

CRUD

UPDATE

UPDATE . . . WHERE . .

/is113/week11/cat/CatDAO.php

updateStatus(\$name, \$status)

13

```
public function updateStatus($name, $status) {  
    // STEP 1  
    $connMgr = new ConnectionManager();  
    $pdo = $connMgr->connect(); // PDO object  
  
    // STEP 2  
    $sql = "UPDATE cat  
           SET  
             status = :status  
           WHERE  
             name = :name";  
  
    $stmt = $pdo->prepare($sql);  
    $stmt->bindParam(':name', $name, PDO::PARAM_STR);  
    $stmt->bindParam(':status', $status, PDO::PARAM_STR);  
  
    // STEP 3  
    $isOk = $stmt->execute();  
  
    // STEP 4  
    $stmt = null;  
    $pdo = null;  
  
    // STEP 5  
    return $isOk;  
}
```

1) Create a **ConnectionManager** object
- Call public method **connect()**

2) Create & prepare an **SQL Query**

3) Run the query
- **execute()** returns **TRUE** (if query ran fine)
or **FALSE** (update did not go through)
- Save the return value in **\$isOk** variable

4) Clear **PDO/PDOStatement** objects

5) Return **\$isOk**, the result of query execution

/is113/week11/cat/CatDAO.php

updateStatus(\$name, \$status)

13

- Test Cases are available in **testCatDAO.php**:

```
// Test 9 - (Database Update - Update a specified cat's status)
echo '<hr>';
echo "<h1>Update a specified cat's status</h1>";
$name = 'Dirty';
$new_status = 'P';

echo "<h3>Before Updating $name's status</h3>";
var_dump( $dao->getCatByName($name) );

if( $dao->updateStatus($name, $new_status) ) {
    echo "<font color='blue'>
        Cat $name's status changed to $new_status
    </font>";
    echo "<h3>After Updating $name's status</h3>";
    var_dump( $dao->getCatByName($name) );
}
else {
    echo "<font color='red'>
        Cat $name's status could not be updated to $new_status
    </font>";
}
```

- How do you know if the **Database UPDATE** worked?

- Go to <http://localhost/phpmyadmin> and check!

name	age	gender	status
Dirty	12	M	P
Filthy	7	F	A
Boring	3	M	A
Needy	3	M	P
Lazy	1	F	P
Stinky	4	F	A
Sad	3	F	A
Angry	5	F	A

Dirty's status changed from **A** to **P**

/is113/week11/cat/update.php

Update a cat's status

Name	Age	Gender	Status	Update Link
Dirty	12	M	Pending Adoption	Update
Filthy	7	F	Available	Update
Boring	3	M	Available	Update
Needy	3	M	Pending Adoption	Update
Lazy	1	F	Pending Adoption	Update
Stinky	4	F	Available	Update
Sad	3	F	Available	Update
Angry	5	F	Available	Update

Complete update.php

This is a **form**

- action='update.php', method='GET'
- There is **NO SUBMIT button** in this form.
- However... check out the **URL** of the "Update" hyperlink.
- Clicking on the **hyperlink** is equivalent to **submitting a form** with an **input field** "name" via **GET** method.

`Update`

/is113/week11/cat/update.php

update.php

Update a cat's status

Name	Age	Gender	Status	Update Link
Dirty	12	M	Pending Adoption	Update
Filthy	7	F	Available	Update
Boring	3	M	Available	Update
Needy	3	M	Pending Adoption	Update
Lazy	1	F	Pending Adoption	Update
Stinky	4	F	Available	Update
Sad	3	F	Available	Update
Angry	5	F	Available	Update

update.php?name=Dirty

(after clicking "Update" hyperlink)

Update a cat's status

Name	Dirty
Age	12
Gender	M
Status	P
Submit Update Request	

Plain Text
(un-editable)

Plain Text
(editable Input Text field)

CatDAO's public method **getCatByName(\$name)** returns a **Cat object**.

→ You can call this method and retrieve a given cat's details.

→ Also, check out what it returns... in case it's not able to find a cat's **name** in **cat** table.

update.php?name=Dirty

Update a cat's status

Name	Dirty
Age	12
Gender	M
Status	A

Submit Update Request

I wish to change cat **Dirty's** status from **P** to **A**.
Hence, I updated the **Status** INPUT TEXT field value to **A**.

update.php?status=A&name=Dirty
(after SUBMIT)

Cat Dirty's status has been updated to A

Update a cat's status

Name	Dirty
Age	12
Gender	M
Status	A

Submit Update Request

Click [here](#) to go back to Update's Main Page

Click on this Hyperlink

/is113/week11/cat/update.php

Update a cat's status

Name	Age	Gender	Status	Update Link
Dirty	12	M	Available	Update
Filthy	7	F	Available	Update
Boring	3	M	Available	Update
Needy	3	M	Pending Adoption	Update
Lazy	1	F	Pending Adoption	Update
Stinky	4	F	Available	Update
Sad	3	F	Available	Update
Angry	5	F	Available	Update

- Cat **Dirty**'s status is now **A** (**Available**).

update(\$name, \$age, \$gender, \$status)

```
public function update($name, $age, $gender, $status) {  
    // STEP 1  
    $connMgr = new ConnectionManager();  
    $pdo = $connMgr->connect(); // PDO object  
  
    // STEP 2  
    $sql = "UPDATE cat  
        SET  
            age = :age,  
            gender = :gender,  
            status = :status  
        WHERE  
            name = :name";  
  
    $stmt = $pdo->prepare($sql);  
    $stmt->bindParam(':name', $name, PDO::PARAM_STR);  
    $stmt->bindParam(':age', $age, PDO::PARAM_INT);  
    $stmt->bindParam(':gender', $gender, PDO::PARAM_STR);  
    $stmt->bindParam(':status', $status, PDO::PARAM_STR);  
  
    // STEP 3  
    $isOk = $stmt->execute();  
  
    // STEP 4  
    $stmt = null;  
    $pdo = null;  
  
    // STEP 5  
    return $isOk;  
}
```

1) Create a **ConnectionManager** object
- Call public method **connect()**

2) Create & prepare an **SQL Query**

3) Run the query
- **execute()** returns **TRUE** (if query ran fine)
or **FALSE** (update did not go through)
- Save the return value in **\$isOk** variable

4) Clear **PDO resources**

5) Return **\$isOk**, the result of query execution

- Test Cases are available in **testCatDAO.php**:

```
// Test 10 - (Database Update - Update a specified cat's details)
echo '<hr>';
echo "<h1>Update a specified cat's details</h1>";
$name = 'Dirty';
$age = 13;
$new_status = 'P';

echo "<h3>Before Updating $name's status</h3>";
var_dump( $dao->getCatByName($name) );

if( $dao->update($name, $age, $gender, $status) ) {
    echo "<font color='blue'>
        Cat $name's details have been updated!
    </font>";
    echo "<h3>After Updating $name's status</h3>";
    var_dump( $dao->getCatByName($name) );
}
else {
    echo "<font color='red'>
        Cat $name's details could not be updated!
    </font>";
}
```

- How do you know if the **Database UPDATE** worked?
- Go to <http://localhost/phpmyadmin> and check!

name	age	gender	status
Dirty	12	M	A
Filthy	7	F	A
Boring	3	M	A
Needy	3	M	P
Lazy	1	F	P
Stinky	4	F	A
Sad	3	F	A

Update

name	age	gender	status
Dirty	13	M	P
Filthy	7	F	A
Boring	3	M	A
Needy	3	M	P
Lazy	1	F	P
Stinky	4	F	A
Sad	3	F	A

Dirty's age changed from **12** to **13**.
His status changed from **A** to **P**.

Week 12

- Session Management & Authentication
 - PHP's **\$_SESSION** (superglobal variable)
 - Creating session
 - Create session variables
 - Modify session variable value
 - Removing/unsetting session variables
 - Destroying session
 - Login/Logout
 - Authenticating username/password against database

Exercises

- Try **Week 11 Exercises**:

- <https://smu.sg/2021-spring-is113-wk11-doc> (Google Doc)

- Complete

- **Question 1** (Person Filter)
 - **Question 3** (Kpop Stars)
 - **Question 2** (Warehouse)
 - **Question 4** (Location and Store Filter)
 - **Question 7** (Blog Posts)
 - **Question 6** (Maintain a Restaurant Menu)