



Project Flexxter

19.12.2020

MD ASHIKUR RAHMAN

FH-WEDEL

borgfelder straÙe 16

Hamburg, Postleitzahl 20537, Deutschland

Overview

This is a sample task that needs to be submitted to Flexxter Company.

Goals

1. Task 1: Extend the given SQL database by any useful properties or tables to map the scenario. Present the adjustments to the database as SQL statements.
2. Assuming the classes "Employee" and "Machine" already exist in the PHP backend, with the following properties. Add a function to the Machine class that assigns the machine (resource) to an employee and thus describes the „process of checking out a resource“. Also add a function to the Machine class that represents the feature of returning the machine to the warehouse.
3. Write a function that returns all resources as objects of type Machine that are currently checked out by the employee named 'Sandy'.

Necessary Steps taken to complete the tasks

In order to test the functions names are "checkout(Employee \$employee)" & "back_to_warehouse()" with my custom Database file which i made with my thought, i had to make certain changes to the classes and the function's parameters.

I have made a php class named machineController and consider other two classes Machine & Employee as a Model class. Because i thought it was becoming messy and unethical to put the task2's two functions whom are "checkout(Employee \$employee)" & "back_to_warehouse()" in a Machine Model class and check the functions output with database's query. That is why I added a Machine class as an instance to the function's parameter "checkout(Employer \$employee, Machine \$machine)" to perform. Here, I took both EmployeeID and MachineID to search for employees who need to be assigned with a

machine, using “mysql raw query” and update the database according to task’s requirement.

I made a machineController which has a custom function named “getCheckedOutMachinesByEmployees” which gives an object type machine which was a requirement of Task 3.

N.B.

Please see the outputs of screenshots to have a clear understanding of why I have considered solving my tasks in this approach.

Output or Results with Custom Database

I. Task 1

Structures of database Table of Machine and Employees name “tblmachines” and “tblemployees”. Added employeeID as a foreign key in “tblmachines” & added “borrowed” as TINYINT type field to determine whether employee was assigned with a machine, if so then “borrowed” field will changed into “1” which considered as “TRUE”, Otherwise, set an default value to “0” as which considered as “FALSE”.

The screenshot displays the phpMyAdmin interface for a database named 'flexxter'. The 'Table structure' view for the 'tblemployees' table is shown. The table has four columns: EmployeeID (int(11), primary key), Surname (varchar(50)), Password (varchar(50)), and Email (varchar(50)). The 'EmployeeID' column is highlighted as the primary key.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	EmployeeID	int(11)			No	None			Change Drop More
2	Surname	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop More
3	Password	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop More
4	Email	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop More

Below the table structure, there are options to 'Check all', 'With selected', 'Add to central columns', and 'Remove from central columns'. There are also buttons for 'Print', 'Propose table structure', 'Track table', 'Move columns', and 'Normalize'.

The 'Indexes' section shows a primary key index on the 'EmployeeID' column.

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Drop	PRIMARY	BTREE	Yes	No	EmployeeID	0	A	No	

At the bottom, there is a form to 'Create an index on' 1 column(s) with a 'Go' button.

The screenshot shows the phpMyAdmin interface for a database named 'flexxter'. The 'tblmachines' table structure is displayed. The table has four columns: 'MachineID' (int(11), primary key), 'Title' (varchar(50)), 'EmployeeID' (int(11)), and 'borrowed' (tinyint(1)). The 'Indexes' section shows two indexes: 'PRIMARY' on 'MachineID' and 'EmployeeID' on 'EmployeeID'.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	MachineID	int(11)			No	None			Change Drop More
2	Title	varchar(50)	latin1_swedish_ci		Yes	NULL			Change Drop More
3	EmployeeID	int(11)			Yes	NULL			Change Drop More
4	borrowed	tinyint(1)			Yes	0			Change Drop More

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Drop	PRIMARY	BTREE	Yes	No	MachineID	1	A	No	
Edit Drop	EmployeeID	BTREE	No	No	EmployeeID	1	A	Yes	

The screenshot shows the phpMyAdmin interface for a database named 'flexxter'. The 'tblemployees' table data is displayed. The table has four columns: 'EmployeeID', 'Surname', 'Password', and 'Email'. The data shows one row with EmployeeID 1, Surname 'Sandy', Password 'sandy123', and Email 'sandy125@gmail.com'.

Showing rows 0 - 0 (1 total, Query took 0.0013 seconds.)

```
SELECT * FROM `tblemployees`
```

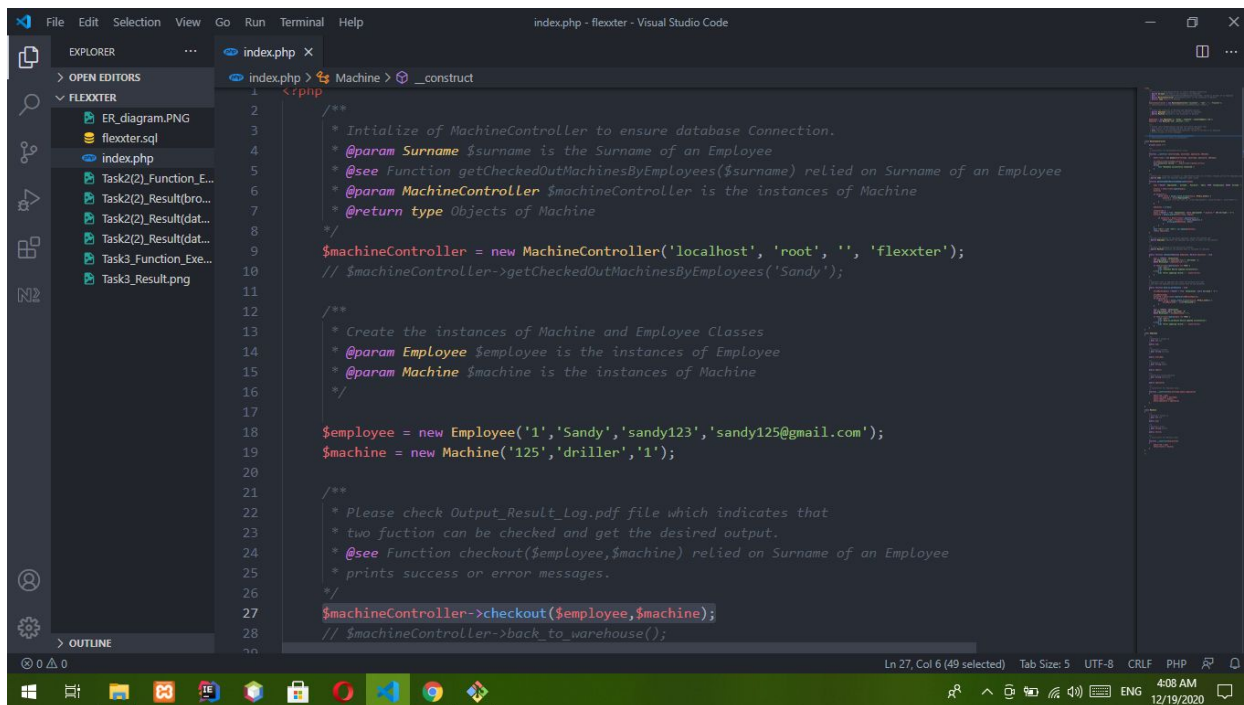
Options: Show all | Number of rows: 25 | Filter rows: Search this table

EmployeeID	Surname	Password	Email
1	Sandy	sandy123	sandy125@gmail.com

Options: Show all | Number of rows: 25 | Filter rows: Search this table

II. Task 2 and Task 3

Below output example of the function “checkout(Employee, Machine)”



```

1  <:php
2  /**
3   * Initialize of MachineController to ensure database Connection.
4   * @param Surname $surname is the Surname of an Employee
5   * @see Function getCheckedOutMachinesByEmployees($surname) relied on Surname of an Employee
6   * @param MachineController $machineController is the instances of Machine
7   * @return type Objects of Machine
8   */
9  $machineController = new MachineController('localhost', 'root', '', 'flexxter');
10 // $machineController->getCheckedOutMachinesByEmployees('Sandy');
11
12 /**
13  * Create the instances of Machine and Employee Classes
14  * @param Employee $employee is the instances of Employee
15  * @param Machine $machine is the instances of Machine
16  */
17
18 $employee = new Employee('1','Sandy','sandy123','sandy125@gmail.com');
19 $machine = new Machine('125','driller','1');
20
21 /**
22  * Please check Output_Result_Log.pdf file which indicates that
23  * two fuction can be checked and get the desired output.
24  * @see Function checkout($employee,$machine) relied on Surname of an Employee
25  * prints success or error messages.
26  */
27 $machineController->checkout($employee,$machine);
28 // $machineController->back_to_warehouse();

```



Database successfully connected.
checkout Record updated successfully



Before Executing the function "checkout(Employee, Machine)"

The screenshot shows the phpMyAdmin interface for the 'flexxter' database. The 'tblmachines' table is selected, and the 'Browse' tab is active. The table structure is displayed with columns: MachineID, Title, EmployeeID, and borrowed. The query results show one row with MachineID 125, Title 'driller', EmployeeID NULL, and borrowed 0.

MachineID	Title	EmployeeID	borrowed
125	driller	NULL	0

After Executing the function "checkout(Employee, Machine)"

The screenshot shows the phpMyAdmin interface for the 'flexxter' database. The 'tblmachines' table is selected, and the 'Browse' tab is active. The table structure is displayed with columns: MachineID, Title, EmployeeID, and borrowed. The query results show one row with MachineID 125, Title 'driller', EmployeeID 1, and borrowed 1.

MachineID	Title	EmployeeID	borrowed
125	driller	1	1

Below output example of the function "checkout(Employee, Machine)"

```

1  <?php
2
3  /**
4   * Initialize of MachineController to ensure database Connection.
5   * @param Surname $surname is the Surname of an Employee
6   * @see Function getCheckedOutMachinesByEmployees($surname) relied on Surname of an Employee
7   * @param MachineController $machineController is the instances of Machine
8   * @return type Objects of Machine
9   */
10 $machineController = new MachineController('localhost', 'root', '', 'flexxter');
11 // $machineController->getCheckedOutMachinesByEmployees('Sandy');
12
13 /**
14  * Create the instances of Machine and Employee Classes
15  * @param Employee $employee is the instances of Employee
16  * @param Machine $machine is the instances of Machine
17  */
18 $employee = new Employee('1','Sandy','sandy123','sandy125@gmail.com');
19 $machine = new Machine('125','driller','1');
20
21 /**
22  * Please check Output_Result_Log.pdf file which indicates that
23  * two fuction can be checked and get the desired output.
24  * @see Function checkout($employee,$machine) relied on Surname of an Employee
25  * prints success or error messages.
26  */
27 // $machineController->checkout($employee,$machine);
28 $machineController->back_to_warehouse();

```



Before Executing the function "back_to_warehouse()"

The screenshot shows the phpMyAdmin interface for the 'flexxter' database. The 'tblmachines' table is selected, and the SQL query 'SELECT * FROM `tblmachines`' is executed. The results show one row with MachineID 125, Title 'driller', EmployeeID 1, and borrowed 1.

MachineID	Title	EmployeeID	borrowed
125	driller	1	1

After Executing the function "back_to_warehouse()"

The screenshot shows the phpMyAdmin interface for the 'flexxter' database. The 'tblmachines' table is selected, and the SQL query 'SELECT * FROM `tblmachines`' is executed. The results show one row with MachineID 125, Title 'driller', EmployeeID NULL, and borrowed 0.

MachineID	Title	EmployeeID	borrowed
125	driller	NULL	0

Before Executing the function “getCheckedOutMachinesByEmployees(\$employeeSurname)”

```

1  <?php
2
3  /**
4   * Initialize of MachineController to ensure database Connection.
5   * @param Surname $surname is the Surname of an Employee
6   * @see Function getCheckedOutMachinesByEmployees($surname) relied on Surname of an Employee
7   * @param MachineController $machineController is the instances of Machine
8   * @return type Objects of Machine
9   */
10 $machineController = new MachineController('localhost', 'root', '', 'flexxter');
11 $machineController->getCheckedOutMachinesByEmployees('Sandy');
12
13 /**
14 * Create the instances of Machine and Employee Classes
15 * @param Employee $employee is the instances of Employee
16 * @param Machine $machine is the instances of Machine
17 */
18
19 $employee = new Employee('1','Sandy','sandy123','sandy125@gmail.com');
20 $machine = new Machine('125','driller','1');
21
22 /**
23 * Please check Output_Result_Log.pdf file which indicates that
24 * two fuction can be checked and get the desired output.
25 * @see Function checkout($employee,$machine) relied on Surname of an Employee
26 * prints success or error messages.
27 */
28 // $machineController->checkout($employee,$machine);
  
```

After Executing the function “getCheckedOutMachinesByEmployees(\$employeeSurname)”



Database successfully connected.

```
array(1) { [0]=> object(stdClass)#4 (4) { ["MachineID"]=> string(3) "125" ["Title"]=> string(7) "driller" ["EmployeeID"]=> string(1) "1" ["borrowed"]=> string(1) "1" }
```

