Web Data Management

152-155

# MVC and PDO

50 points

This project will require you to combine all your HTML, JavaScript, PHP and MySQL skills into one project.

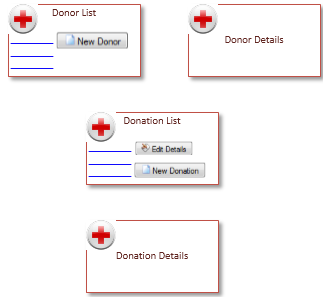
You may accomplish this assignment in one of two ways:

* Come up with your own web-base, database project.
  + You’ll have to design and create your own MySQL database. Remember, if you already have a database (Access maybe) you can export and import records so you don’t have to re-enter them.
  + Your database must include at least two linked tables. If your database has more tables, you will only be required to implement two of them for this assignment.
    - Discuss your database design with me before starting development.
  + Create a web site to manage the data in a MySQL database.
    - If you haven’t already, verify the database structure with me.
    - The database should include two tables.
    - You will be required to provide CRUD processing for both tables.
    - Include at least one list view with a search feature
    - Include at least one combo box that is filled from the database
  + All JavaScript and PHP validation must be implemented.
* Develop MVC applications for the project you’re working on in Systems Implementation.
  + You will be required to implement JavaScript and PHP validation.
  + You will be required to implement two, linked table forms.
  + Discuss your selection with me before starting development.

**Here is an example showing what I am looking for:**

The Red Cross implementation guide appears on the next page. This is a capstone project from a prior semester.

Red Cross Donor/Donations MVC & PDO Project Implementation Guide



* If you haven’t done so already, import *redcrossimport.sql* into your MySQL server (XAMPP) using MySQL Browser.
* Review the Red Cross Data Dictionary included in the starter folder.
* Open the controller file and become familiar with its structure.
* The starter forms should be viewable. View the initial appearance of each form.
* First view the donorList form (change the default action)
* View the other three forms (change the default action to display each).
* Restore the default action to donorList
* In the donors model, add a method to generate a donor lookup (selection) list.
* In the controller, call the method that generates the donor lookup lists from the appropriate case.
* In the donorList view, add a print\_r (probably in <pre> tags) to display the donor list array.
* After verifying that your data has transferred to the view properly, delete the print\_r/pre.
* Create a list of donors using links. As you can see in the diagram above, the donor list links should link the donations list for the selected donor.
* In the controller, add an echo statement that displays the ID number of the selected donor.
* In the donations model, add a method to generate the a list of donations (lookup) for the selected donor. Display the donation date, donation type and units in the list. Sort the list by donation date, descending.
* In the controller, call the method to retrieve the donations list for the selected donor.
* In the donationList view, add a print\_r (probably in <pre> tags) to display the donation list array.
* After verifying that your data has transferred to the view properly, delete the print\_r/pre.
* Display the donation list appropriately formatted.
* Note the donation list view should also display certain **donor** details
* In the donors model, add a method to collect the necessary details for the selected donor.
* In the controller, call the method to get the details of the selected donor.
* In the donation list view, display the donor details. Note the donor name appears in two places.
* In the donation list, create a list of donations using links. As you can see in the diagram above, the donation list links should link the donation details view for the selected donation.
* Test a few of your links by adding an echo statement in the donation details case in the controller.
* The donation list view also displays the total number of units a donor has given. In the donations model, create a method to determine the total number of units donated by a given donor.
* In the controller, call the method to determine the total number of units donated by the selected donor.
* **Alternative:** in the list view total the units as you display them.
* In the donation list view, display the total number of units.
* The donation list view also links to the donor details view using the provided Edit button.
* Include code in the donation list view to share the donor ID with the donor details view.
* Test this code by adding an echo statement to the donor details case displaying the selected donor ID.

**Congratulations!! You have completed phase 1 of this assignment.**

* Remove your test echo from the donor details view.
* In the donor model, add a function (if necessary) to collect one donor’s details.
* In the controller, call the function to get the donor’s details before displaying the view.
* (optional) Consider adding a statement to display the details array in the details view.
* Test by displaying the details for one donor.
* Note the phone number doesn’t display. Correct this error.
* Provide code in the model, controller and details view to include a list of the blood types in the combo box.
* Update the details view to correctly display the Inactive status. Tip: Helfer is inactive.
* Modify the details view to sense new records and hide the Delete and Donation List buttons for new records.
* Add a donorNew action to the controller.
* Add the donorSaveNew and donorUpdate cases (merged) including code to handle all submit buttons.
* View an existing donor’s details. Ensure the Donors List and Donations Lists buttons function properly.
* Add methods to the donor model to update and insert records. Use the validation function to determine which fields are optional. (Update the error messages *employee🡪donor*)
* Add a method to the donor model to delete records. When a donor is deleted be sure to delete that donor’s donations as well.
* Add code where appropriate to avoid duplicate donor records. Use the donor first name, last name and phone number as the logical keys.
* Add a method to the donations model to get donation details.
* Update the controller to get the donation details
* Update the controller to get the name of the donor for this donation.  
  Tip: Add donorID to the link.
* Update the donation details view to display the donor name.
* Add the donationID and donorID as hidden fields in this form.
* Modify the view to hide the Delete button for new records
* Add a donationNew action to the controller.
* Modify the New button in the donation list view to send the donor ID
* Add the donorSaveNew and donorUpdate cases (merged) including code to handle all submit buttons.
* View an existing donation’s details. Ensure the Donations Lists button functions properly.
* Add methods to the donation model to update and insert records. Use the validation function to determine which fields are optional.
* Update the donation details view action to handle both new and update
* Add a method to the donation model to delete records.
* Add code where appropriate to avoid duplicate donation records. Use the donor id and donation date as the logical keys.
* Add a Search box to the donors list view. Modify the getDonorList function to accept a filter. Modify the controller to send the filter when calling getDonorList.
* A give your application a good *once over* looking for other flaws.
* Evaluate your application using the provided evaluation sheet.
* WHEW!!!