Technical Document

PHP & Databases 2

Daniel Cutajar

6/4/2019

CODING ISSUES

When trying to implement a 7-day cycle for my website I came across a wall when trying to add all 7 days in a form. I had no clue how to have the user input 7 dates and then update the same table with seven different inputs and was later informed that it was unnecessarily complicated anyways. With outside help I fixed the issue by reducing inputs to 1 date, the first day of the 7-day cycle. This input would input into the first row of the cycle table and a loop would go through each row adding to the value of the day by 1.

The initial layout of the website had the user access without log in and only be prompted to log in when trying to direct to the seat page or booking page. This became very complicated due to the not so easy to understand navigation paths used by CodeIgniter. The work around for this was simply to force the user to log into the site before even being able to see anything the site has to offer. This is a poor solution but due to time constraints was unable to be touched upon further.

Trying to replicate the Eden Cinemas website, I wanted to implement a Now Showing page displaying all the movies with a showing time. This would have each movie appear with its details on the side and below a list of days and the times they were available that day. This was hard to implement, and I needed outside help to even understand the logic I needed to go about it. The result was a 3 stage for loop. The first loop cycled through all of the movies getting only those that appeared in the showing table. This would prevent movies not actually showing to appear. The movie date was actually an array containing all of the dates the movie appeared and would cycle through those listing beneath it. Lastly for the times, a third loop was created. The movie times had two values, the time and corresponding date. To make this work visibly for each date being shown in a movie it cycled through all the times that movie had. If the current date matched the date corresponding to the time it would display if not it would move on to the next. This way only the correctly dated times appeared on the page.

For the home page I wanted to display each date of the cycle in a menu and clicking the date would load all movies showing that day and the time. The solution was quite similar to the now showing except for the use of the URL. When the date was pressed the page would reload with the date in the URL. All movies with a showing on that URL date would appear and the same function of showing times with dates matching that date was used. The only issue was that the page would initially load not selecting a date, so no movies would appear. I have yet to find a way to overcome this considering it is the base homepage of the website.

Coming soon was a problem due to not knowing how to show any movies that were not out yet. I was exposed to the date function but found it confusing due to the 7-day system that was present in my database. Eventually I realized I can compare the released date to the last day present in the database. If the release date is after that date it is not in the cinema as of yet and will pass through to display in the coming soon page.

I had an issue retrieving the ticket the user had just purchased. Since the seat was a form and could not pass values through the submit button I was unable to pass through any values that would narrow down the ticket selection process to the one just initiated. This was fixed by implementing a do ticket create method called by the seat method. When the button is pressed it would complete this function and insert the ticket info along with a 32-string code that is unique to each ticket purchase. This code is passed through the URL and used to call the information to display.

After assumed completion I realized that the delete function for movies that had showing slots would cause a website breaking error. Despite my efforts to making multiple functions and ordering them to delete the foreign keys first the issue just wouldn’t resolve itself. After researching online, I found out about using cascading in PHPMyAdmin where this setting deletes all instances of a field in other tables automatically. Implementing this to the foreign keys effected fixed the issue and the movie deletion function works properly.

CONTROLLERS

Bookings.php

__construct()

Constructs the pages linking the models and helpers to the controller.

Parameter: None

Result: None

index()

The base function for the booking controller. Links to the index of the booking views loading a list of all the user bookings.

Parameters: None

Return: Builds the index page with the user bookings found in the database.

Date.php

__construct()

Constructs the pages linking the models and helpers to the controller.

Parameter: None

Result: None

index()

The base function for the date controller. Links to the index of the date views loading a list of the date cycles ranging from the first cycle day to the last.

Parameters: None

Return: Builds the index page with the cycle dates found in the database.

edit()

Goes to the edit view for the date controller. Since there must be a date entered there is no create page and this is the only screen to be edited by the user.

Parameters:

> $submit - If the submit isn't false it will complete the _do_edit function which functionally edits the date.

Return: Builds the edit page with the one cycle input.

_do_edit()

Completes the edit function which updates the cycle dates with the new values. The first date is selected by the user and a for loop updates the next 6.

Parameters: None

Return: Updates the dates using the new input.

Movie.php

__construct()

Constructs the pages linking the models and helpers to the controller.

Parameter: None

Result: None

index()

The base function for the movie controller. Links to the index of the movie views loading a list of the movies that are currently in the database.

Parameters: None

Return: Builds the index page with the cycle dates found in the database.

create()

Goes to the create page which allows the user to input all of the required information to add a movie to the database.

Parameters:

> $submit - If the submit isn't false it will complete the _do_create function which functionally adds the movie.

Return: Builds the create a page with all of the movie inputs.

delete()

Deletes the desired movie from the database, removing it from the page.

Parameters:

> $slug – The slug is used to narrow the movie deletion down to one desired movie.

Return: The erasure of a singular movie row.

edit()

Goes to the edit view for the movie controller. This has the same inputs as the create page with the inputs filled up with the current values.

Parameters:

> $slug – The slug narrows down the edit to the specific movie the user wants to change.
>
> $submit - If the submit isn't false it will complete the _do_edit function which functionally edits the date.

Return: Builds the edit page with the pre-entered input.

_do_create()

Completes the create function which creates the movie with the entered values.

Parameters: None

Return: Creates the movie using the added inputs.

_do_edit()

Completes the edit function which updates the movie with the new values. If none are changed it will remain the same and take the user back to the index page.

Parameters:

$movie – The movie row to be edited.

Return: Updates the movie using the changed inputs. If none are changed it remains the same.

_build_dir()

Checks that the folder exists, creates it if not.

Paramaters:

$dir – The directory being checked.

Return: Creates the directory if none is present.

_upload_poster()

Uploads the poster to the directory to be retrieved and used in the views.

Parameters:

$name – The name of the poster file to be created and stored

Return: Uploads the image to its folder.

_get_poster_path()

Generates the file path to store the poster.

Parameters:

$id - The id of the movie to have a poster saved.

$to_array – If there are multiple files matching the id it will store them and load them as an array.

Return: Creates the file path.

Pages.php

__construct()

Constructs the pages linking the models and helpers to the controller.

Parameter: None

Result: None

index()

The base function for the Pages controller. Links to the index of the page views loading the home page showing the 7 days and the movies showing on each day.

Parameters:

$date – Uses the date to load the 7 specific dates on different instances on the home page.

Return: Builds the index page with the appropriate date information being shown.

now_showing()

Goes to the now showing page which shows the user every movie that has a showing slot and separates them by date and orders them by release date.

Parameters:

> $date – The date is used as a reference to the dates that need to be shown on the page in separation.

Return: Builds the now showing page with all of the movies connected to a showing slot.

coming_soon()

Goes to the coming soon page which shows the user all the movies that have release dates that are after the final date in the date cycle meaning they won't be shown that week or have shown in the past.

Parameters: None

Return: Builds the coming soon page with all of the movies that have yet to be released.

seats()

Goes to the seats page loading the details of the movie slot selected with time and movie details along with a UI to select your seat. Any seats already booked will not be able to be purchased and the grid adjusts based on the cinema being shown in.

Parameters:

> $date – Uses this as a reference to the date the movie is being booked.
>
> $slot – Uses this as a reference to the time the movie is being booked.
>
> $id - Gathers all of the information of the movie slot that is being booked
>
> $submit - If the submit isn't false it will complete the _do_ticket_create function which functionally edits the date.

Return: Builds the seat page with the selected movie slot and the seat UI.

ticket()

The ticket page shows the ticket receipt to be printed and shown at the cinema on entrance.

Parameters:

> $code – The ticket create generates a code that leads to the ticket page. This ticket is linked to a purchase and loads the ticket information related to that purchase.

Return: Builds the ticket page with the ticket information for printing.

_do_ticket_create()

Completes the create function which creates the ticket with the retrieved values also generating a code to call back to the data being inputted to the database at this point.

Parameters:

> $date – Uses this as a reference to the date the movie is being booked.
>
> $slot – Uses this as a reference to the time the movie is being booked.

$id - Gathers all of the information of the movie slot that is being booked

Return: Uploads the ticket to the database to be stored and shown.

_get_poster_path()

Generates the file path to store the poster.

Parameters:

$id - The id of the movie to have a poster saved.

$to_array – If there are multiple files matching the id it will store them and load them as an array.

Return: Creates the file path.

Slot.php

__construct()

Constructs the pages linking the models and helpers to the controller.

Parameter: None

Result: None

index()

The base function for the movie controller. Links to the index of the slot views loading a list of the slots that are currently in the database.

Parameters: None

Return: Builds the index page with the slots found in the database.

create()

Goes to the create page which allows the user to input all of the required information to add a slot to the database.

Parameters:

$submit - If the submit isn't false it will complete the _do_create function which functionally creates the slot.

Return: Builds the create a page with all of the movie inputs.

delete()

Deletes the desired slot from the database, removing it from the page.

Parameters:

$id – The id is used to narrow the slot deletion down to one desired slot.

Return: The erasure of a singular slot row.

edit()

Goes to the edit view for the slot controller. This has the same inputs as the slot page with the inputs filled up with the current values.

Parameters:

$id – The slug narrows down the edit to the specific movie the user wants to change.

$submit - If the submit isn't false it will complete the _do_edit function which functionally edits the date.

Return: Builds the edit page with the pre-entered input.

_do_create()

Completes the create function which creates the slot with the entered values.

Parameters: None

Return: Creates the movie using the added inputs.

_do_edit()

Completes the edit function which updates the slot with the new values. If none are changed it will remain the same and take the user back to the index page.

Parameters:

$slot – The slot row to be edited.

Return: Updates the slot using the changed inputs. If none are changed it remains the same.

_build_dir()

Checks that the folder exists, creates it if not.

Paramaters:

$dir – The directory being checked.

Return: Creates the directory if none is present.

MODELS

Booking_model.php

create_ticket()

The create ticket takes the information provided to input a user's ticket into the database. If the user selected multiple seats, then they will have a ticket entered for each seat they selected.

Parameters:

$user_id – The user_id is required to link a ticket purchased to a selected user.

$showing_id – This contains a reference to the movie details like the title, cinema and date

$slot – Since the showing_id has multiple time slots the specific time must be passed.

$seats – Contains all of the seats in an array and inputs them into separate fields.

$code – This code is used to separate sessions. If a user buys a ticket to a movie and then buys another to the same movie later on. They need a code to split up the group by.

Return: No return type. Adds the field into the database.

get_bookings_by_user()

To be used for the booking page. All bookings by the logged in user are retrieved and grouped by their code to show in a list form. From this the movie title, date, time, room and seats are shown.

Parameters:

$user_id – The user_id is used to get only the bookings made by the logged in user.

Return: A result array containing the relevant information from the ticket, showing, room and movies tables.

get_bookings_by_code()

Takes the code generated when making a booking and finds the fields matching that code. With the method this function is implemented each code should have 1 user and showing id.

Parameters:

$ code – The code generated is passed through and the bookings matched are selected.

Return: The results are grouped together to return one row with the ticket information.

get_seats_taken()

Inputs the respective showing_id and time to find all seats booked. If a seat with those values is already ordered in the seat screen it will appear greyed out and unable to be selected.

Parameters:

$showing_id – This narrows down the movie and date without a specific time.

$time – This join with the showing_id to have one precise movie session.

Return: Returns an array of taken seats in that movie session.

Date_model.php

get_date()

This function gets the first value in the cycle table. This is due to the fact that the entry of weekly cycles is done by entering the date of the first week.

Parameters: None

Return: Returns the row containing the date found with id 0

get_dates()

This gets all the dates in the cycle table.

Parameters: None

Return: Returns a result array with each result containing an id and date. There should be 7 results for a weekly cycle.

update_dates()

After inputting the value for the start of the cycle. It inputs every day into the database counting 7 days from that date.

Parameters:

$id – The input method is done via a for loop for 7 times with the id incrementing to fill up all positions.

$date – The date contains the date entered for the first one then it incremented by 1 in the loop.

Return: Inputs the dates into the database.

Movie_model.php

create_movie()

Creates a movie and inputs it into the database using the appropriate information.

Parameters:

$title – The title of the movie as taken from the input field.

$release_date – The date the movie was released as taken from the input field.

$runtime – The length of the movie as taken from the input field.

$rating – The rating for the movie selected from the dropdown list.

$genre – The single or multiple genre of the movie.

Return: Inputs a row into the movie database.

delete_movie()

Selects a movie field from the database and deletes it as well as all fields containing it in tickets and movie_genres. Using cascading in PHPMyAdmin it deletes all instances of the movie in any other table.

Parameters:

$id – The id is required to find the single movie row that you requested to delete.

Return: The erasure of the row_array.

get_movie()

Gets one specific movie based on its slug.

Parameters:

$slug – The slug is unique to each movie and thus is used to select one row of data.

Return: A row_array with all the movie information via the slug.

get_movies()

Gets all the movies present in the movie table.

Parameters: None

Return: A result_array with each movie and the information pertaining to it.

get_movie_genre()

Gets all of the respective genres linked to one specific movie.

Parameters:

$movie_id – The movie id is used to identify the one movie in which all genre ids are linked to.

Return: A result_array with all of the genres that are linked to the desired movie.

get_ratings()

Retrieves all of the ratings into an array.

Parameters: None

Return: Returns a result_array with all the movie ratings.

get_ratings_array()

Retrieves all of the ratings into an array.

Parameters: None

Return: Returns a variable with all the movie ratings in an array.

get_genres_array()

Retrieves all of the genres into an array.

Parameters: None

Return: Returns a variable with all the movie ratings in an array.

get_titles_array()

Retrieves all of the genres into an array.

Parameters: None

Return: Returns a variable with all the titles ratings in an array.

replace_genre()

Deletes the present id genres and replaces them with the new id and genre taken from an array.

Parameters:

$id – the movie_id that needs to be linked to a genre

$genre – the genre array with all of the genres linked to that movie

Return: Inputs the new genres into the movie_genres database.

update_movie ()

Updates a movie row in the database. Initially had a check function but worked without its implementation. The old values are all erased even if the movie isn't updated and the new movie values take its place.

Parameters:

$id – The id of the movie being updated.

$title - The value of the title passed from the input. Could be changed or the same.

$release_date - The value of the release date passed from the input. Could be changed or the same.

$runtime - The value of the runtime passed from the input. Could be changed or the same.

$rating - The value of the rating passed from the input. Could be changed or the same.

Return: Inputs the movie values into the database.

Slot_model.php

create_slot()

Creates a slot and inputs it into the database using the appropriate information.

Parameters:

$movie – The movie id to link to all of the movie information.

$room – The room the movie is being shown at that day.

$date – The date of the slot made.

$time – The various times the movie will be shown that day.

Return: Adds the slots to the slot database.

<u>delete_slot()</u>

Deletes a singular slot based on the id supplied.

Parameters:

$id – The id that narrows down the deletion to one specific slot

Return: The erasure of a slot from the slot database

<u>get_slot()</u>

Retrieves a singular slot based on the id supplied

Parameters:

$id – The id that narrows down the retrieval to one specific slot

Return: A row_array with the slot details based on the id inputted.

<u>get_date_by_movie()</u>

Uses the movie_id supplied to retrieve all the dates a specific movie is showing.

Parameters:

$movie_id – The movie_id retrieves all dates based on one specific movie.

Return: The result will be a result_array with all of the dates that a requested movie is being shown on.

<u>get_time_by_movie()</u>

Uses the movie_id supplied to retrieve all of the dates and times that a movie is being shown. This will be compared to the date in the previous function to sort the time to the specific date they are being shown in the now showing page.

Parameters:

$movie_id – The movie_id retrieves all the dates and times that a requested movie is being shown at.

Return: A result_array containing the dates and times of a movie being shown.

<u>get_slots()</u>

Retrieves all of the slots present in the slot database.

Parameters: None

Return: A result array with each row containing the data of one unique movie slot.

<u>get_times()</u>

Retrieves all of the times present in the time database.

Parameters: None

Return: A result array with each row containing the data of one unique time.

<u>**get_rooms()**</u>

Retrieves all of the times present in the rooms database.

Parameters: None

Return: A result array with each row containing the data of one unique room.

<u>**get_rooms_array()**</u>

Retrieves all of the rooms into an array.

Parameters: None

Return: Returns a variable with all the cinema rooms in an array.

<u>**get_showing_room()**</u>

Retrieves the number of columns and rows in a cinema based on the cinema room being called.

Parameters:

$id – Identifies which cinema the dimensions are asking for.

Return: A row_array retrieving the column value and row value.

<u>**get_times_array()**</u>

Retrieves all of the showing times into an array.

Parameters: None

Return: Returns a variable with all the times in an array.

<u>**update_showing()**</u>

Updates a showing slot row in the database. Initially had a check function but worked without its implementation. The old values are all erased even if the showing slot isn't updated and the new showing slot values take its place.

Parameters:

$id **-** The id of the showing slot being updated.

$movie **-** The value of the rating passed from the input. Could be changed or the same.

$room **-** The value of the cinema passed from the input. Could be changed or the same.

$date – The value of the date passed from the input. Could be changed or the same.

Return: Inputs the new showing slot values into the database.