Kent State University

CS 33901 Software Engineering Spring 2018

Project 2

Posted: 2/22/18
Due: 3/6/18, 11:59pm
Questions? email me at aalali1@kent.edu

A. Art Store

Vision

Demonstrate not only your ability to generate dynamic pages from multiple data-base tables, but also the ability to design a solution that minimizes code duplication, and the ability to design a maintainable and portable web systems by applying the concepts of modularity, encapsulation, separation of concerns, and reusability. This project also makes use of a CSS Framework called `SemanticUl`. The results when finished will look similar to that shown in Figure below.

Functional Requirements

1. You have been provided with two `HTML` files (`list.html` and `detail.html`) that includes all the necessary markup, download the Project Start Code. You have also been provided with an SQL import script (`art-small.sql`) as well as all the images needed for these two pages. You should create a database named art and import the data, create a user. You are provided with `art-config.inc.php` file, it has the connection string with the database connection string and user credentials.
2. Create `PHP` versions of the two supplied `HTML` files named `browse-paintings.php` and `single-painting.php`. Extract the common header into a separate include file.
3. You will need to retrieve information from the `Paintings` table. You will be accessing the `Artists`, `Shapes`, `Galleries`, `Genres`, `Subjects`, and Reviews tables, along with the

intermediate tables: **PaintingGenres** and **PaintingSubjects**. Since both pages will need to access these tables, you should generalize your database retrieval code into separate database classes (a data base access layer).

4. The **browse-paintings.php** page will initially potentially display all the paintings in the **Painting** table. However, because there are hundreds of paintings, only show the top 30 (use the **sql LIMIT** keyword). Each of the images shown must be links with the appropriate query string to the **single-painting.php** page. The user should be able to filter the list by specifying the artist, museum, and/or shape in the three drop-down lists, populated from the **artists** (sorted by last name), **museums** (sorted by gallery name), and **shapes** (sorted by shape name) tables. As with the unfiltered list, only display the top 30 matches for the filter. Be sure to use the **PHP utf8_encode()** function to properly display some of the foreign characters in the data.

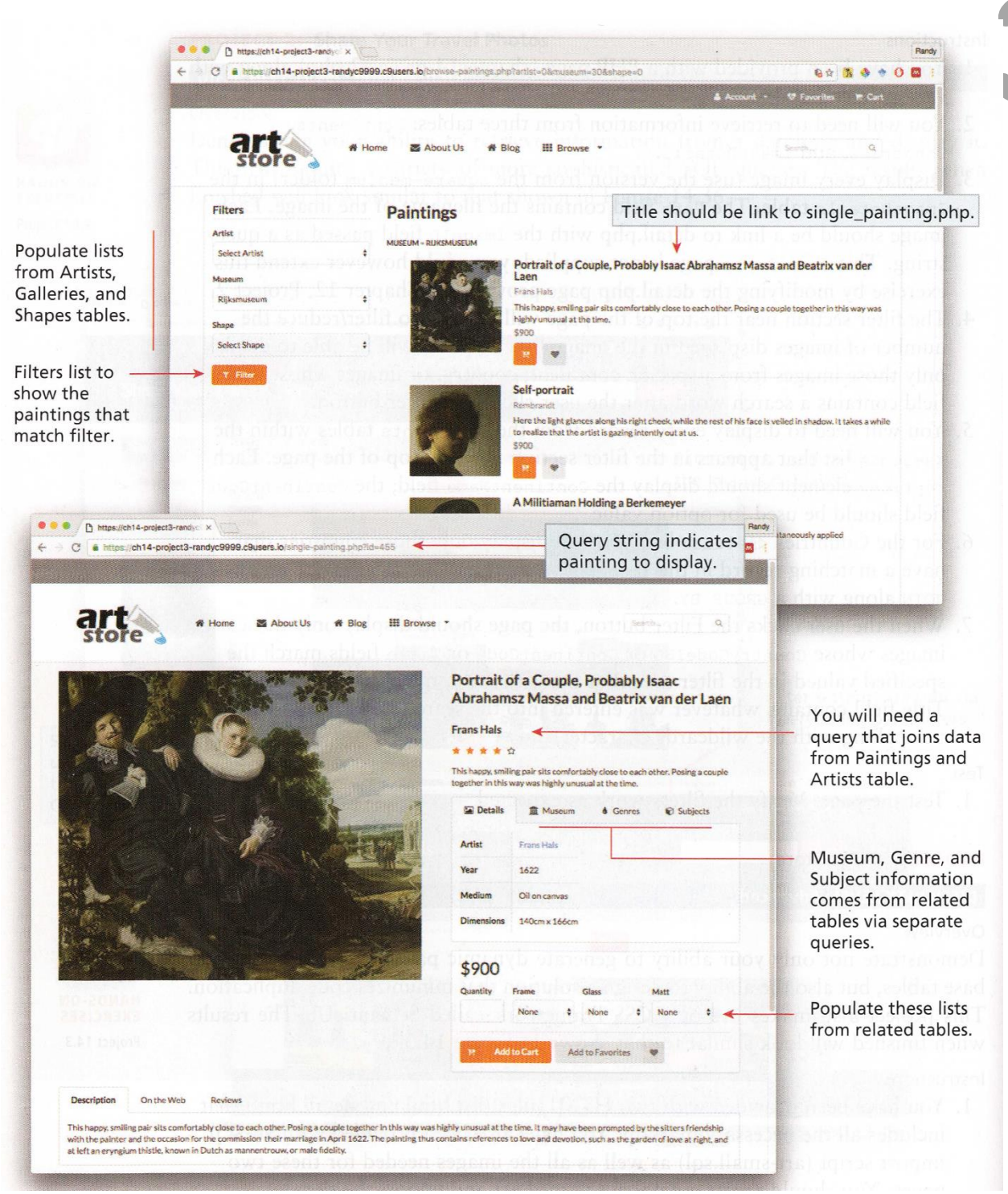
5. It must display the information about a single painting (specified via the id passed in via a query string parameter). This page has a lot of information packed into it, and it uses Tab components (available as part of the **SemanticUl CSS** framework) to make it manageable. This page needs to display data from some other tables (**Galleries**, **Genres**, **Subjects**, and **Reviews**). The Frame, Glass, and Matt select lists should be populated from the appropriate tables (**TypesFrame**, **TypesGlass**, **TypesMatt**). This page should handle a missing or a noninteger query string parameter by displaying a default painting.

Non-Functional Requirements

6. That one should separate that which varies from that which stays the same; Use the least possible **PHP** code mixing with **HTML** markups, utilizes external **PHP** files as classes and utility files.

7. The project needs to be layered that utilizes the 3-tier architecture (UI, domain layer, and data access), as in, **In-Class-Project-5**.

8. The project uses singleton pattern for the database **PDO API** connection as in, **In-Class-Project-5**.


Populate lists from Artists, Galleries, and Shapes tables.
Filters list to show the paintings that match filter.
Title should be link to single_painting.php.
Query string indicates painting to display.
You will need a query that joins data from Paintings and Artists table.
Museum, Genre, and Subject information comes from related tables via separate queries.
Populate these lists from related tables.
art store
Home
About Us
Blog
Browse
Filters
Artist
Select Artist
Museum
Rijksmuseum
Shape
Select Shape
Filter
Paintings
MUSEUM - RIJKSMUSEUM
Portrait of a Couple, Probably Isaac Abrahamsz Massa and Beatrix van der Laen
Frans Hals
This happy, smiling pair sits comfortably close to each other. Posing a couple together in this way was highly unusual at the time.
$900
Self-portrait
Rembrandt
Here the light glances along his right cheek, while the rest of his face is veiled in shadow. It takes a while to realize that the artist is gazing intently out at us.
$900
A Militiaman Holding a Berkemeyer
https://ch14-project3-randyc9999.c9users.io/single-painting.php?id=455
Portrait of a Couple, Probably Isaac Abrahamsz Massa and Beatrix van der Laen
Frans Hals
This happy, smiling pair sits comfortably close to each other. Posing a couple together in this way was highly unusual at the time.
Details
Museum
Genres
Subjects
Artist
Frans Hals
Year
1622
Medium
Oil on canvas
Dimensions
140cm x 166cm
$900
Quantity
Frame
Glass
Matt
None
None
None
Add to Cart
Add to Favorites
Description
On the Web
Reviews
This happy, smiling pair sits comfortably close to each other. Posing a couple together in this way was highly unusual at the time. It may have been prompted by the sitters friendship with the painter and the occasion for the commission: their marriage in April 1622. The painting thus contains references to love and devotion, such as the garden of love at right, and at left an eryngium thistle, known in Dutch as mannentrouw, or male fidelity.

9. Database Access Layer (DB classes) are used by the domain layer and the UI interacts directly with domain layer classes. UI doesn't need to interface directly with neither the database using the **`PDO API`** nor the database access layer.

10. **`ArtStore`** is your main class, your engine or controller, the one that needs to be instantiated first.

11. Minimize the use of the **`PHP`** script in the UI. Use a utility file for helper functions, if needed, you may call it **`art-store-util.php`**.

12. You must submit valid and semantically appropriate **`HTML5`**. Validate the produced **`HTML5`** markup. Submit zero warning and zero error pages.

13. Your submitted repo directory tree structure should be as follows (exact names and structure):

```
.
|-- inclassprojects
|-- projects
|   |-- project1
|   |-- project2
|   |   |-- art-store
|   |   |   |-- browse-paintings.php
|   |   |   |-- single-painting.php
|   |   |   |-- css
|   |   |   |   |-- ...
|   |   |   |-- images
|   |   |   |   |-- ...
|   |   |   |-- includes
|   |   |   |   |-- art-config.inc.php
|   |   |   |   |-- ...
|   |   |   |-- js
|   |   |   |   |-- misc.js
|   |   |   |-- lib
|   |   |   |   |-- ...
|-- YOURNAME.txt
```

Grading

The grade will be broken down as follows:

Points	Mark
1, **2**,	2/100
3,	10/100
4,	30/100
5,	25/100
6, **7**, **8**, **9**, **10**,	15/100
11,	14/100
12, and **13**	2/100