Author : Najam Haque

Date : Sep 14 2014

Document Name : Memory Game

Version : 0.0

Reviewed By : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Reviewd On : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Introduction:

This document describes the accompanied Memory Game. This is the first version of the game, though it is not a feature rich build, however the game can still be used as a proof of concept in order to buy more time and resources from the management.

# The Objective of the Game :

Quoting from the original requirement,

***​"Memory game is a card game in which all of the cards are laid face down on a surface and two cards are flipped face up over each turn. The object of the game is to turn over pairs of matching cards."***

# Demo :

A live demo of the game is available at the following url.

[***http://memory.ca/***](http://memory.ca/)

Users will need to use the following host file entry to make the above work

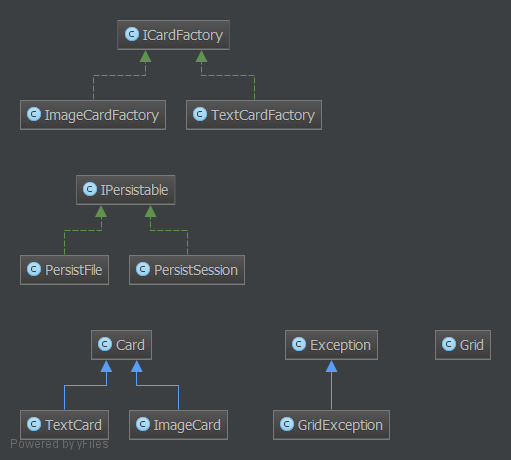
107.21.210.47 memory.ca

107.21.210.47 cdn.memory.ca

# Architecture:

The game is fully using MVC pattern. There is some amalgamation of html in models that needs to be taken out. That aspect deals with rendering of a card and grid , which is completely model implementation dependent. It’s not difficult in future to supply a rendering mechanism to separate those out to a different class.

The game is built using a few core classes. Please refer to the accompanied class diagram memory.uml .



The class Grid is the primary class instantiated from controller. The Grid creation requires a CardFactory as well , so that is also needs to be instantiated within controllers.

The game is extensible to a great extent.

1. Different Card Types

The supplied build only supports two different types of Cards.

* 1. Text Cards
  2. Image Cards

Both of these types conform to an interface , and extends from an abstract class. Future developers can support many other types of cards by supplying requisite functionality as dictated by the ABC Card.

1. Persistence

The Game comes with two types of persistence layers.

* 1. Session Based Persistence
  2. File Based Persistence

As expected, the developers may supply their own implementation of persistence of Grid object . For example, saving the Grid object to RDBMS , MemeCache , or perhaps Redis.

# Code Organization :

All the classes are in /src folder.

Card Abstract Base Class for all types of Cards

CardFactory An interface that Facories Must support. Card factories are used by Grid in order to create different types of Cards. A CardFactory type object is passed to the Grid at the time of Grid Creation.

Grid Represents the Grid.At the time of creation , Grid Must be supplied a CardFactory so that it knows what type of Cards to create. Optionally , a Peristable may be supplied. That allows the Grid to save relevant data to peristable medium. The Persistance mechanism defaults to SessionPersist . However as an example and for usit Testing purposes, PersistFile is supplied which uses a file based mechanism to save the user’s grid state.

GridException A special class extended from Exception . The Grid only throws GridException with different messages. No error code is supplied for now. Any object loading , initialing or saving a grid should catch this exception and handle accordingly.

ImageCard A card that is of type image. This is supplied as an example . The actual implementation still uses textual representation of cards. A future build might use img tag to render the card to fit the requirement.

ImageCardFactory A factory that produces stacks of ImageCard .

TextCard A textual card. Good enough as a proof of concept.

TextCardFactory A factory that produces stacks of TextCard .

Persistable An interface that certain object must support to declare that they persist the gride data . This is not the best name. Before we finalize this version , we must improve upon this name.

PersistFile A class that allows Grdi Data to persist to file.

PersistSession A class that allows Grdi Data to persist to session.

The entry points to the game are at the root level. These are

Index.php :

This Creates a new Grid and renders it. The file acts as both views and controllers. It is recommended that future developers should separate the two and use templates.

Click.php

Called through ajax , whenever the user clicks on any grid position . This supplies the functionality to handle the flip the cards etc.

Assets :

All assets should be loaded from a cdn, just in case we have a widely popular game.

Currently assets are placed in the assets directory.

style.css self explanatory

memory.js self explanatory .. contains the only js function used.

URL Rewriting:

Mode rewrite is used to provide clean urls in the browser ,and still hiding code organization from the browser urls :

/ => maps to src/controllers/index.php

/new => maps to src/controllers/new.php

/click => maps to src/controllers/click.php

# Features:

1. The project is using composer to manage dependency .. even though prod version will not depend on anything other than cdn version library of bootstrap and jquery. However Unit Testing framework is required for dev build to make sure things execute as smooth as possible.
2. By using psr-4 autoloading , we have defined a proper structure of placing files in different areas of application. That frees developers from loading all class files , just like in any modern php framework.
3. All classes placed in Memory NameSpace . This will avoid NameSpace pollution. Future developers are highly encouraged to use Name Spaces . They may create more name spaces as they go. It is not necessary to put all files in Memory . Only the ones that makes sense should go here.
4. The Application uses Structured Exception Handling . In many cases it is easier, and cleaner to throw an exception and handle that exceptionally, instead of placing a bunch of conditions and returns … which will further require checks by the caller, throwing exceptions is preferable.
5. By Design, the game picks up the state from where ever it was left, if the user closes game in a new window etc ... provided that the user's session is not timed out. It uses different cookies by default for each GridType.

# Testing :

Due to time constraints, limited unit testing was done and is supplied. Much stronger unit testing and behavioral testing is recommended. It however demonstrates the ability to create unit tests.

Items To DO :

Following Items are not originally required , however it will be nice to develop these for future

1) Bullet Proof Testing .. As mentioned above phpUnit Test is not exhaustive by any standard and is only supplied as a demo. A much exhaustive Testing is required.

2) Use flickr Images in Image Card … Current Cards don’t use images .. Its not too difficult for developers to support that however.

3) REST API... Just in case there are other developers out there taking advantage of the game .. perhaps creating flash version of the game .

4) Start Client Side Timer / High Score . The game does not recognizes currently that it is completetd. A timer should calculate the total time spent sovling a game, and congratulate after the agem is finished.

5) New Game should alert the user if a game is already in progress.