



Assignment 2 - Composer


You will be creating a **PHP** application that accesses the `deckofcardsapi.com` via `guzzlehttp`.

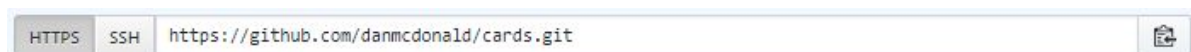


The following steps will help you create the application:

1. Create a new repository on GitHub named `cards`. To avoid errors, do NOT initialize the new repository with `README`, `license`, or `gitignore` files. You can add these files after your project has been pushed to GitHub.
2. Create a new folder called `cards` under `c:\xampp\htdocs`

3. Open your Git bash shell, navigate to

```
$ cd /c/xampp/htdocs/cards
$ echo "# cards" >> README.md
$ git init
$ git add .
#Adds the files in the local repository and stages them for commit.
To unstage a file, use 'git reset HEAD YOUR-FILE'
git commit -m "First Commit"
#Commits the tracked changes and prepares them to be pushed to a
remote repository.
#You can get the URL for your repository at the top of your GitHub
repository's Quick Setup page. Click  at the end of the remote
repository URL:
```

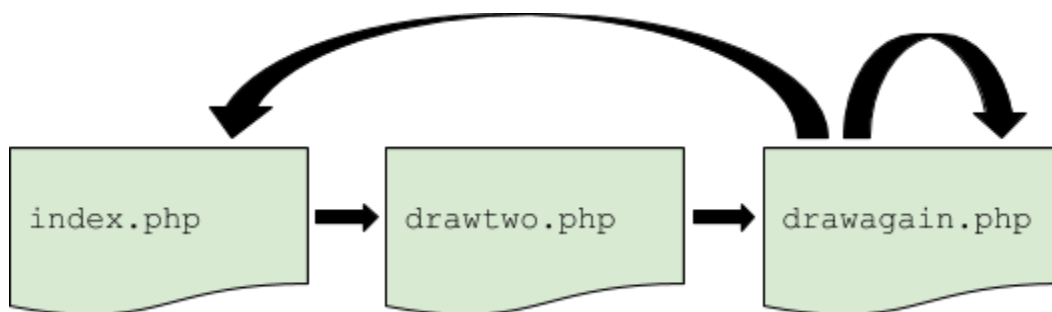


```
git remote add origin https://github.com/danmcdonald/cards.git
#sets the new remote
git remote -v
#verifies the new remote
git push -u origin master
#Pushes the changes in your local repository up to the remote
repository you specified as the origin
```

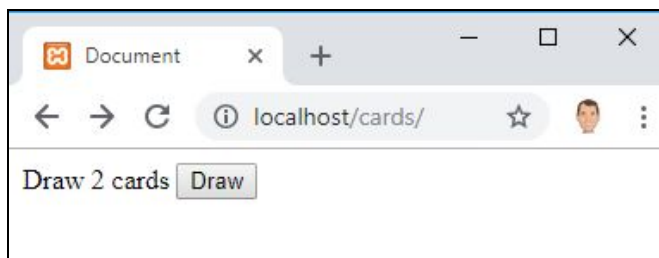
4. If not already done, go to page <https://getcomposer.org/download> and download and install `Composer-Setup.exe`. Use just the default installation, no developer mode, no proxy server. The current version is `v1.9.2`

5. The `php.exe` runtime should be added to your system `PATH` environment variable. You can check this by running `php -version` on the command line.
6. Open the directory `c:\xampp\htdocs\cards` inside of VS Code. Now, open the terminal window inside of **VS Code**. Run the following command:

```
composer require guzzlehttp/guzzle
```
7. We are going to create the card game 21 using the online card deck api of <http://deckofcardsapi.com>. Start by creating three pages in the cards directory, `index.php`, `drawtwo.php`, and `drawagain.php`. The diagram below shows the common flow between the pages.



8. Code the `index.php` page. The `index.php` page is very simple and can look like the following image below:



The Draw button is inside a form. Pressing the Draw button calls `drawtwo.php`.

```
<form action="drawtwo.php" method="get">
  <input type="submit" value="Draw 2 Cards">
</form>
```

At the top of the `index.php` page, make sure to connect to the session and then destroy the session variables, using the following code:

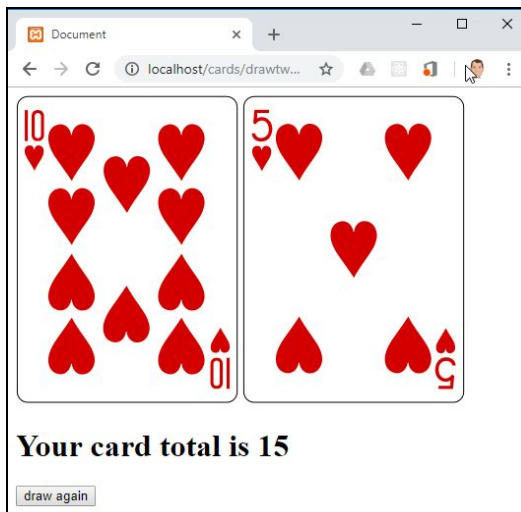
```
session_start();  
session_destroy();
```

We will be storing cards we draw in the session variables, so we want to start with an empty session storage.

9. When clicking on the Draw button, you should load the following page (drawtwo.php):

Notice two cards have been displayed, the numeric total of the two cards has been calculated and if the total is below 21 a draw again button is displayed. If the total of the card values is 21 or higher, a "play again" button is displayed.

Following the page screenshot below are instructions on how to code out the rest of the page.



The above page is where we start using `guzzlehttp` to connect to `deckofcardsapi.com`. First start a **PHP** session and then require the `autoload.php` file as shown below:

```
session_start();  
require 'vendor/autoload.php';
```

Next, we will create an instance of the `GuzzleHttp Client` and make a request of the `deckofcardsapi` site to shuffle a single deck of cards for us. We use a built-in **PHP JSON** library function called `json_decode` to convert the **JSON** to a **PHP** associative array. So, `$response_data` below is an associative array.

```
$client = new \GuzzleHttp\Client();
$response = $client->request('GET', 'https://deckofcardsapi.com/api/deck/new/shuffle/?deck_count=1');
$response_data = json_decode($response->getBody(), TRUE);
```

The data sent back from the shuffle request (from `deckofcardsapi.com`) is shown below:

```
{
  "success": true,
  "deck_id": "3p40paa87x90",
  "shuffled": true,
  "remaining": 52
}
```

Access the `deck_id` from `$response_data` and make another request of the `deckofcardsapi.com` to get two random cards as shown below. We again convert the **JSON** into a **PHP** associative array and we save the associative array as the variable `$response_data2`.

```
$response2 = $client->request('GET',
  'https://deckofcardsapi.com/api/deck/' . $response_data['deck_id'] . '/draw/?count=2');
$response_data2 = json_decode($response2->getBody(), TRUE);
```

The associate array `$response_data2` has been created based on the **JSON** in the format shown below:

```
{
  "success": true,
  "cards": [
    {
      "image": "https://deckofcardsapi.com/static/img/KH.png",
      "value": "KING",
      "suit": "HEARTS",
      "code": "KH"
    },
    {
      "image": "https://deckofcardsapi.com/static/img/8C.png",
      "value": "8",
      "suit": "CLUBS",
      "code": "8C"
    }
  ],
  "deck_id": "3p40paa87x90",
  "remaining": 50
}
```

There is an array called `cards` filled with objects of cards in it. We want to assign the `cards` array to a **PHP** variable called `$card_array`. We then want to calculate the total value of the cards in the array and assign the value to a variable named `$card_total`. The function to calculate the total value of the

cards is called `calc_card_total($card_array)` and is shown after the current code block. Finally, we want to assign the `$card_array` to a session variable called 'card_array' and assign the `$response_data['deck_id']` to a session variable called 'deck_id'.

The code to accomplish the above steps is shown below:

```
$card_array = $response_data2['cards'];
$card_total = calc_card_total($card_array);
$_SESSION['card_array'] = $card_array;
$_SESSION['deck_id'] = $response_data['deck_id'];
```

The function to calculate the total of the cards array is shown below:

```
function calc_card_total($card_array1){
    $card_value1=["KING">10, "QUEEN">10, "JACK">10,"ACE">1, "2">2, "3">3,
    "4">4, "5">5, "6">6, "7">7, "8">8, "9">9, "10">10 ];
    $card_value2=["KING">10, "QUEEN">10, "JACK">10,"ACE">11, "2">2, "3">3,
    "4">4, "5">5, "6">6, "7">7, "8">8, "9">9, "10">10 ];
    $card_total1 = 0;
    $card_total2 = 0;
    $card_face="";
    foreach($card_array1 as $card){
        $card_face = $card['value'];
        $card_total1 = $card_total1 + $card_value1[$card_face];
        $card_total2 = $card_total2 + $card_value2[$card_face];
    }
    if($card_total2 <= 21){
        return $card_total2;
    } else {
        return $card_total1;
    }
}
```

There are two associative arrays declared near the top of the function. The first array uses 1 for the value of an "ACE". The second array uses 11 for the value of an "ACE". The `calc_card_total` function uses the best outcome of the total card value to determine which value for an "ACE" to use.

Now, we transition to the **HTML** portion of the `drawtwo.php` page. You can use the Emmet shortcut `!<tab>` to produce a basic **HTML 5** template page. Under the body tag, use a **PHP** foreach loop to iterate over the `$cards_array` and display an image of the card returned from the deck. The code below shows the **PHP** code for the loop.

```
<?php foreach($card_array as $card) : ?>
    
<?php endforeach; ?>
```

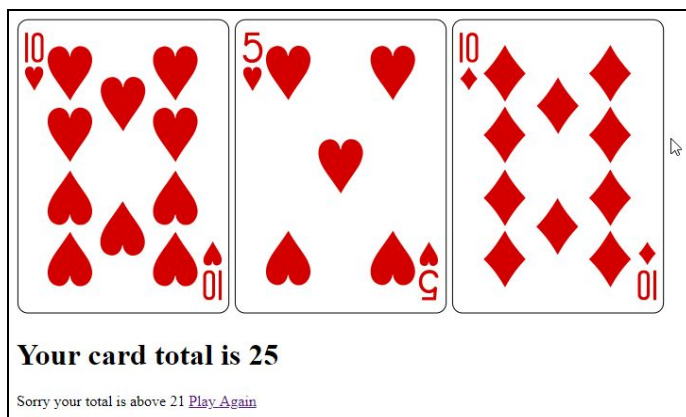
Next, we want to display the total value of the cards drawn. The following code could be used:

```
<h1><?php echo "Your card total is $card_total"; ?></h1>
```

Finally for the `drawtwo.php` page, we want to display the valid options for the user. For example, if the `$card_total` is 21, the user wins and can play again. If the `$card_total` is over 21, the user loses and can play again. Finally, if the `$card_total` is under 21, the user can choose to draw again. The following code can be used to display the correct message:

```
<?php if($card_total > 21): ?>
    Sorry your total is above 21
    <a href="index.php">Play Again</a>
<?php elseif($card_total == 21): ?>
    You win, take a trip to Vegas
    <a href="index.php">Play Again</a>
<?php else: ?>
    <a href="drawagain.php">Draw again</a>
<?php endif; ?>
```

9. When you click on the draw again button, you should get a screen similar to the one below:



In order to achieve this functionality, you need to join the session because we need to access the `$card_array` and `$deck_id`. Like on the page

`drawtwo.php`, we will need to create a GuzzleHttp Client. We can create the required client and access the session variables using the following code:

```
session_start();
require 'vendor/autoload.php';
$client = new \GuzzleHttp\Client();
$card_array = $_SESSION['card_array'];
$deck_id = $_SESSION['deck_id'];
```

Now, that we have our `$deck_id`, we can request another card from the `deckofcardsapi.com`. We want to make the request, then decode the **JSON** as a **PHP** associative array. Once the **JSON** has been decoded, we want to add the newly returned card to our `$card_array`. Once the `$card_array` includes it's newest card, we want to assign the `$card_array` back to the `$_SESSION['card_array']` variable. That way, if another card is drawn, it would be added to the latest deck. The `$deck_id` is already saved in the session, so we don't have to re-assign that value to the session. Finally, we do want to calculate the `$card_total` of the cards in the `$card_array`. We do that by passing the `$card_array` to the function shown above called `calc_card_total`. The following code completes the steps described:

```
$response = $client->request('GET',
'https://deckofcardsapi.com/api/deck/'. $deck_id .'/draw/?count=1');
$response_data = json_decode($response->getBody(), TRUE);
$card_array[] = $response_data['cards'][0];

$_SESSION['card_array'] = $card_array;
$_SESSION['deck_id'] = $response_data['deck_id'];
$card_total = calc_card_total($card_array);
```

Now, we can transition to the **HTML** portion of the `drawagain.php` page. We want to conduct the same type of decision process as we did in the HTML of the `drawtwo.php` page. The code below that should be put in the body of the HTML shows the code for the logic. We first loop through the `$card_array` showing an image of the card face. Next, we print the `$card_total`. Finally, depending on the `$card_total`, we offer the user choices to "Play Again" or "Draw Again".

```

<?php foreach($card_array as $card) : ?>
    
<?php endforeach; ?>

<h1><?php echo "Your card total is $card_total"; ?></h1>

<?php if($card_total > 21): ?>
    Sorry your total is above 21
    <a href="index.php">Play Again</a>
<?php elseif($card_total == 21): ?>
    You win, take a trip to Vegas
    <a href="index.php">Play Again</a>
<?php else: ?>
    <a href="drawagain.php">Draw again</a>
<?php endif; ?>

```

10. Now, the code from your project should be pushed to GitHub. Rerun the following git commands:

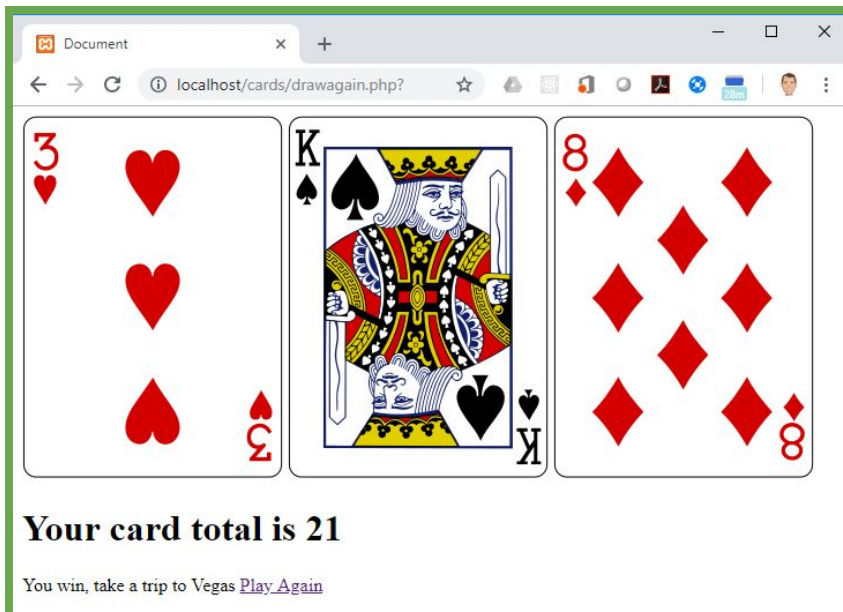
```

$ git add .
git commit -m "PHP Complete"
git push -u origin master

```

11. For Submission, please submit the following two screenshots:

Screenshot #1 - The `drawagain.php` page (cards/card totals do not matter)



Screenshot #2 - The GitHub repository cards

