COMP 3015

Server Side Web Application Development

Week 2

- Working with files
  - In-class exercise: building a book management application
- Call Stack
- Object Oriented Programming (OOP) basics, JSON serialization
  - similar to OOP content covered in the JavaScript course: Classes, Objects, constructors, etc.
- Lab: files and OOP

Differences between single and double quoted strings:

```php
$name = 'Christian';
echo "Hello $name\n";
echo 'Hello $name\n';
```

← escape sequences and variable interpolation is possible
← output is almost exactly “as is”, with few exceptions

```
~/Work/BCIT/COMP3015/Week2 $ php examples.php
Hello Christian
Hello $name\n%
~/Work/BCIT/COMP3015/Week2 $
```

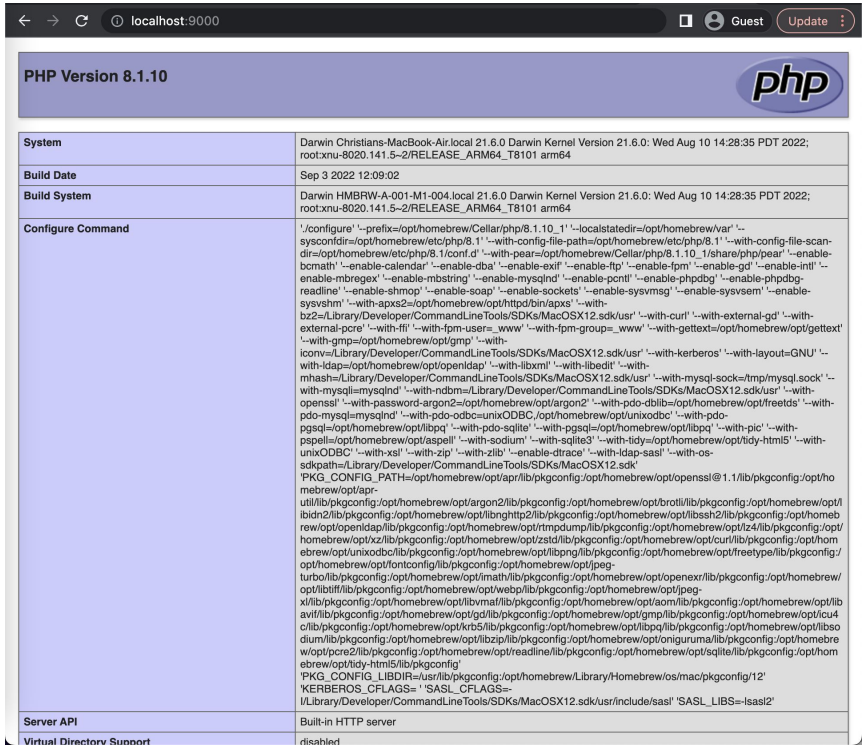
Reading PHP configuration information

```php
<?php
phpinfo();
```

You can run the above code using the built in PHP server on your terminal:

php -S localhost:9000 index.php

phpinfo() will output configuration information. Modifying particular values in your php.ini file will change this output.



System	Darwin Christians-MacBook-Air.local 21.6.0 Darwin Kernel Version 21.6.0: Wed Aug 10 14:28:35 PDT 2022; root:xnu-8020.141.5~2/RELEASE_ARM64_T8101 arm64
Build Date	Sep 3 2022 12:09:02
Build System	Darwin HMBRW-A-001-M1-004.local 21.6.0 Darwin Kernel Version 21.6.0: Wed Aug 10 14:28:35 PDT 2022; root:xnu-8020.141.5~2/RELEASE_ARM64_T8101 arm64
Configure Command	'./configure' '--prefix=/opt/homebrew/Cellar/php/8.1.10_1' '--localstatedir=/opt/homebrew/var' '--sysconfdir=/opt/homebrew/etc/php/8.1' '--with-config-file-path=/opt/homebrew/etc/php/8.1' '--with-config-file-scan-dir=/opt/homebrew/etc/php/8.1/conf.d' '--with-pear=/opt/homebrew/Cellar/php/8.1.10_1/share/php/pear' '--enable-bcmath' '--enable-calendar' '--enable-dba' '--enable-exif' '--enable-ftp' '--enable-fpm' '--enable-gd' '--enable-intl' '--enable-mbregex' '--enable-mbstring' '--enable-mysqlnd' '--enable-pcntl' '--enable-phpdbg' '--enable-phpdbg-readline' '--enable-shmop' '--enable-soap' '--enable-sockets' '--enable-sysvmsg' '--enable-sysvsem' '--enable-sysvshm' '--with-apxs2=/opt/homebrew/opt/httpd/bin/apxs' '--with-bz2=/Library/Developer/CommandLineTools/SDKs/MacOSX12.sdk/usr' '--with-curl' '--with-external-gd' '--with-external-pcre' '--with-ffi' '--with-fpm-user=_www' '--with-fpm-group=_www' '--with-gettext=/opt/homebrew/opt/gettext' '--with-gmp=/opt/homebrew/opt/gmp' '--with-iconv=/Library/Developer/CommandLineTools/SDKs/MacOSX12.sdk/usr' '--with-kerberos' '--with-layout=GNU' '--with-ldap=/opt/homebrew/opt/openldap' '--with-libxml' '--with-libedit' '--with-mhash=/Library/Developer/CommandLineTools/SDKs/MacOSX12.sdk/usr' '--with-mysql-sock=/tmp/mysql.sock' '--with-mysqli=mysqlnd' '--with-ndbm=/Library/Developer/CommandLineTools/SDKs/MacOSX12.sdk/usr' '--with-openssl' '--with-password-argon2=/opt/homebrew/opt/argon2' '--with-pdo-dblib=/opt/homebrew/opt/freetds' '--with-pdo-mysql=mysqlnd' '--with-pdo-odbc=unixODBC,/opt/homebrew/opt/unixodbc' '--with-pdo-pgsql=/opt/homebrew/opt/libpq' '--with-pdo-sqlite' '--with-pgsql=/opt/homebrew/opt/libpq' '--with-pic' '--with-pspell=/opt/homebrew/opt/aspell' '--with-sodium' '--with-sqlite3' '--with-tidy=/opt/homebrew/opt/tidy-html5' '--with-unixODBC' '--with-xsl' '--with-zip' '--with-zlib' '--enable-dtrace' '--with-ldap-sasl' '--with-os-sdkpath=/Library/Developer/CommandLineTools/SDKs/MacOSX12.sdk' 'PKG_CONFIG_PATH=/opt/homebrew/opt/apr/lib/pkgconfig:/opt/homebrew/opt/openssl@1.1/lib/pkgconfig:/opt/homebrew/opt/apr-util/lib/pkgconfig:/opt/homebrew/opt/argon2/lib/pkgconfig:/opt/homebrew/opt/brotli/lib/pkgconfig:/opt/homebrew/opt/libidn2/lib/pkgconfig:/opt/homebrew/opt/libnghttp2/lib/pkgconfig:/opt/homebrew/opt/libssh2/lib/pkgconfig:/opt/homebrew/opt/openldap/lib/pkgconfig:/opt/homebrew/opt/rtmpdump/lib/pkgconfig:/opt/homebrew/opt/lz4/lib/pkgconfig:/opt/homebrew/opt/xz/lib/pkgconfig:/opt/homebrew/opt/zstd/lib/pkgconfig:/opt/homebrew/opt/curl/lib/pkgconfig:/opt/homebrew/opt/unixodbc/lib/pkgconfig:/opt/homebrew/opt/libpng/lib/pkgconfig:/opt/homebrew/opt/freetype/lib/pkgconfig:/opt/homebrew/opt/fontconfig/lib/pkgconfig:/opt/homebrew/opt/jpeg-turbo/lib/pkgconfig:/opt/homebrew/opt/imath/lib/pkgconfig:/opt/homebrew/opt/openexr/lib/pkgconfig:/opt/homebrew/opt/libtiff/lib/pkgconfig:/opt/homebrew/opt/webp/lib/pkgconfig:/opt/homebrew/opt/jpeg-xl/lib/pkgconfig:/opt/homebrew/opt/libvmaf/lib/pkgconfig:/opt/homebrew/opt/aom/lib/pkgconfig:/opt/homebrew/opt/libavif/lib/pkgconfig:/opt/homebrew/opt/gd/lib/pkgconfig:/opt/homebrew/opt/gmp/lib/pkgconfig:/opt/homebrew/opt/icu4c/lib/pkgconfig:/opt/homebrew/opt/krb5/lib/pkgconfig:/opt/homebrew/opt/libpq/lib/pkgconfig:/opt/homebrew/opt/libsodium/lib/pkgconfig:/opt/homebrew/opt/libzip/lib/pkgconfig:/opt/homebrew/opt/oniguruma/lib/pkgconfig:/opt/homebrew/opt/pcre2/lib/pkgconfig:/opt/homebrew/opt/readline/lib/pkgconfig:/opt/homebrew/opt/sqlite/lib/pkgconfig:/opt/homebrew/opt/tidy-html5/lib/pkgconfig' 'PKG_CONFIG_LIBDIR=/usr/lib/pkgconfig:/opt/homebrew/Library/Homebrew/os/mac/pkgconfig/12' 'KERBEROS_CFLAGS= ' 'SASL_CFLAGS=-I/Library/Developer/CommandLineTools/SDKs/MacOSX12.sdk/usr/include/sasl' 'SASL_LIBS=-lsasl2'
Server API	Built-in HTTP server
Virtual Directory Support	disabled

Terminating Programs

A call to exit can be made to explicitly terminate programs.

For using a status code, keep the value >=0 and <= 254.

- Status code 255 is reserved for PHP
- Status code 0 means success (normal exit, non-error state)

```php
<?php
exit;
exit(0);
exit('Exit messages will be printed before the program terminates');
```

File I/O

Working with files: outcomes

- Read files stored on disk
- Write content to files (overwriting and appending)
- Updating content within a file
- Deleting content within a file
- Parsing file contents (using implode, explode functions)

Be able to do Create, Read, Update, Delete (CRUD) operations on file contents.

Writing to files

We can write to a file using file_put_contents, or fopen, fwrite and fclose:

```
file_put_contents('comp3015.txt', 'apple,mango,blueberry,blackberry')
```

Note the **file_put_contents** docs: “This function is identical to calling fopen(), fwrite() and fclose() successively to write data to a file.”

```
$filePointer = fopen('comp3015.txt', 'w');
fwrite($filePointer, 'apple,mango,blueberry,blackberry');
fclose($filePointer);
```

Reading from files

Reading files can be achieved using file_get_contents:

```
$fileContents = file_get_contents('comp3015.txt');
```

This will read the entire file into a string. Similarly to writing files using the fopen approach, we can use fread to read them:

```
$filename = 'comp3015.txt';
$filePointer = fopen('comp3015.txt', 'r');
$fileContents = fread($filePointer, filesize($filename));
fclose($filePointer);
echo $fileContents;
```

Note that the second parameter for fread is the number of bytes to read. We can use the filesize function to get the size of the file we want to read.

implode, explode

implode: join array elements with a string, return a string
explode: split a string by a string, return an array ← in Java, Python this is called “split”

```
$carBrandsAsString = implode( separator: PHP_EOL, ["Honda", "Nissan", "Ford"]);
$listOfCarBrands = explode( separator: PHP_EOL, $carBrandsAsString);
```

$carBrandsAsString:

```
string(17):
"Honda
Nissan
Ford"
```

$listOfCarBrands:

```
array(3) {
  [0]=> string(5) "Honda"
  [1]=> string(6) "Nissan"
  [2]=> string(4) "Ford"
}
```

See **implode-explode/main.php**

See: book-manager example application

Function Call Stack

Function Call Stack

During the execution of an application the order in which functions are invoked must be kept track of.

As a developer, understanding the function call stack is a critical part of being able to effectively debug applications.

Function Call Stack: What's a stack?

- A stack is a Last-In-First-Out (LIFO) data structure
  - Think of a stack of plates

- Two key operations:
  - Push: add a new element to the stack
  - Pop: remove the most recently added element

See: **call_stack.php**

Object Oriented Programming in PHP

Intro to Object Oriented Programming in PHP

- Classes can be thought of as templates for creating objects
  - Classes describe the attributes and behaviour that the objects will have
- Interfaces allow us to create a contract: any class that implements XYZ interface will have to implement methods A, B, C
- Attribute access modifiers
  - **private**: only accessible within the class
  - **protected**: accessible within the current class and subclasses
  - **public**: accessible outside the class

Questions:

- When does a constructor get called?
- Within a method in a class, what's **$this**? ← make sure to be able to answer these
  - What was **this** in JavaScript?

See: **oop/main.php**

Intro to OOP in PHP: classes, creating objects

Book.php × BookRepository.php × main.php ×

```php
<?php

class Book {

    private string $name;
    private string $authorName;
    private string $isbn;

    public function __construct(string $theName = '', string $theAuthor = '', string $theISBN = '') {
        $this->name = $theName;
        $this->authorName = $theAuthor;
        $this->isbn = $theISBN;
    }
```

Default parameters

```php
$bookRepository = new BookRepository( theFilename: 'book_repo.txt');
$lordOfTheRings = new Book( theName: "Lord of the Rings", theAuthor: "J.R.R. Tolkien", isbn: "9780358653035");

$bookRepository->saveBook($lordOfTheRings);
```

Interfaces Example: JSON serialization

The JsonSerializable interface can be implemented by any class that wants to customize how it is represented when passed to json_encode(). The interface has one method which any class implementing the interface, must provide, called jsonSerialize.

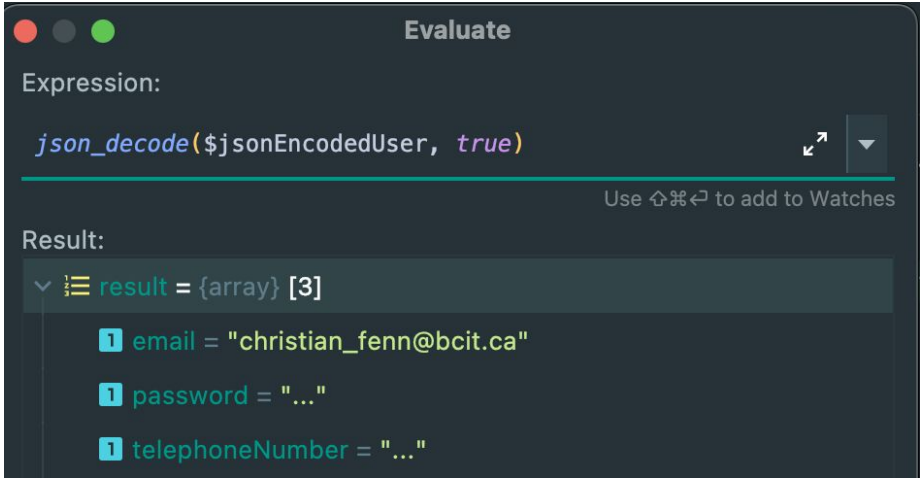
```php
interface JsonSerializable {

    /* Methods */
    public jsonSerialize(): mixed
}
```

```php
<?php

class User implements JsonSerializable {

    private string $email;
    private string $password;
    private string $telephoneNumber;

    // ... typical getters/setters omitted

    public function jsonSerialize(): array {
        return [
            'email' => $this->email,
            'password' => $this->password,
            'telephoneNumber' => $this->telephoneNumber,
        ];
    }
```

Interfaces, JSON serialization

```
require_once 'User.php';
$user = new User( email: 'christian_fenn@bcit.ca',  password: '...',  telephoneNumber: '...');
$jsonEncodedUser = json_encode($user);
echo $jsonEncodedUser;
```

The following value will be output: {"email":"christian_fenn@bcit.ca","password":"...","telephoneNumber":"..."}



json_decode can be used to turn JSON into an associative array, or an stdClass (which is an empty class). Note that the second parameter for json_decode is used to ask for an associative array to be returned.

[json_encode]
[json_decode]

Inheritance

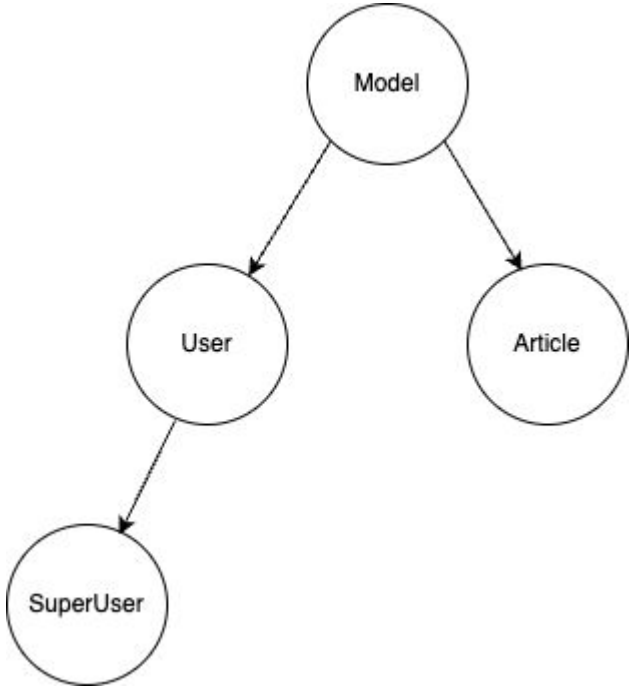
In many cases we have systems with “is-a” relationships.

A **SuperUser** is a **User**.

An **Article** is a **Model**.

In these cases, select functionality from the super class will be shared with the sub class.

See: **inheritance/main.php**



Lab for week 2 will be on D2L

Work for this week

- Ensure you understand how the call stack works
- Complete the lab on D2L
- Review the slides
  - CRUD operations on file contents
  - Basics of Object Oriented Programming (OOP) in PHP
    - Ensure sure you understand:
      - What a class is, what an object is, and how to call methods on an object
      - The **$this** keyword
      - When a constructor is called
      - What an interface is (in the context of OOP): needed for the lab
      - Basics of inheritance