Домашние задания по PHP/ООП/БД

Перечень

1. Создать класс с методами в отдельном файле
2. Продемонстрировать работу исключений
3. Создать БД с заполненными данными и таблицами и получить из неё данные
4. В Index.php указать ссылками задачи, для возможности перехода на них
5. Настроить NGINX (веб-сервер) в docker-compose, для работы вашего кода через localhost в браузере (порт 80)
6. Поместить это все в Doker и выложить в GIT (в том числе dump БД)

Задание 1. Работа с классами и объектами - Интернет-магазин

Описание классов

- `Order` — заказ, содержит два свойства: корзина и цена доставки. Эти свойства должны быть проинициализированы в конструкторе.
- `Basket` — корзина, содержит одно свойство: массив позиций товаров. При создании объекта это свойство содержит пустой массив.
- `BasketPosition` — позиция одного товара в заказе, содержит два свойства: товар и количество. Эти свойства должны быть проинициализированы в конструкторе.
- `Product` — товар, содержит два свойства: название и цена. Эти свойства должны быть проинициализированы в конструкторе.

Класс Order

Этот класс должен содержать следующие методы:

- `public function getBasket()` — возвращает объект-корзину, которая хранится в заказе.
- `public function getPrice()` — возвращает общую стоимость заказа. Сумма стоимости корзины и стоимости доставки заказа.

Класс Basket

Этот класс должен содержать следующие методы:

- public function addProduct(Product $product, $quantity) — создаёт новую позицию и добавляет её в корзину, объединение количества товаров можно не реализовывать.
- public function getPrice() — возвращает стоимость всех позиций в корзине.
- public function describe() — выводит или возвращает информацию о корзине в виде строки:

```
"<Наименование товара> - <Цена одной позиции> - <Количество>"
```

Класс BasketPosition

Этот класс должен содержать следующие методы:

- public function getProduct() — возвращает наименование товара в этой позиции.
- public function getQuantity() — возвращает количество товаров в этой позиции.
- public function getPrice() — возвращает стоимость позиции.

Класс Product

Этот класс должен содержать следующие методы:

- public function getName() — возвращает наименование товара.
- public function getPrice() — возвращает стоимость товара.

Опишите все требуемые классы и реализуйте их методы.

Клиентский код

Создайте корзину, заполните её товарами. Создайте заказ на основе этой корзины таким образом:

```
$order = new Order($basket);
```

Выведите информацию о корзине этого заказа и выведите общую стоимость заказа.

```
"Заказ, на сумму: <сумма заказа> Состав: <информация о корзине>"
```

Задание 2. Валидация с помощью исключений

Описание классов

- User — модель пользователя, класс лишь имитирует взаимодействие с базой данных.
- UserFormValidator — класс, реализующий валидацию данных формы.

Класс User

Этот класс должен содержать следующие методы:

- public function load(int $id) — метод должен формировать исключение, если переданный *$id* не найден в базе данных (придумайте любое условие на значение параметра *$id* для имитации этой ошибки);
- public function save(array $data): bool — метод имитирует сохранение пользователя в базе данных и возвращает *true*, если сохранение прошло успешно, или *false*, если произошла ошибка. Для имитации работы метод должен возвращать случайное значение *true* или *false*.

Класс UserFormValidator

Этот класс должен содержать следующий метод:

- public function validate(array $data) — метод осуществляет валидацию массива с данными. Если валидация не прошла, то метод должен выбросить исключения с текстом ошибки. Для каждого правила валидации текст ошибки должен быть свой.

Правила валидации

- Имя должно быть не пустым.
- Возраст должен быть не менее 18 лет.
- Email должен быть заполнен и соответствовать формату email.

Клиентский код

Создайте простую HTML-страницу с веб-формой. В форме должны быть текстовые поля:

- id — id пользователя;
- name — имя пользователя;
- age — возраст пользователя;
- email — email пользователя.

При отправке формы сделайте следующее:

1. Создайте объект класса User.
2. Попробуйте загрузить пользователя с помощью метода load() класса User.
3. Если пользователя удалось загрузить, то выполните валидацию полей формы с помощью класса UserFormValidator.
4. Если валидация прошла успешно, попробуйте сохранить поля формы с помощью метода save() класса User.
5. Если всё прошло успешно, то выведите сообщение об успешном сохранении формы и снова выведите форму пустой.

6. Если возникла ошибка, то выведите текст сообщения об ошибке и снова выведите форму, подставив в её поля введенные ранее значения.

Задание 3. Создание таблицы SQL

Что нужно сделать

1. Создайте базу данных с таблицами
2. Наполните тестовыми данными
3. Выведите информацию из БД в index.php
4. Сформируйте дамп этой базы данных

Таблица стран — countries:

Поле	**Тип данных**	**Дополнительные параметры**	**Описание поля (можно указать в комментарии к полю)**
id	целое число	обязательное, автоинкремент, первичный ключ	ID - уникальный идентификатор записи
name	строка, без ограничений длины	обязательное	название страны
code	строка, три символа	обязательное	символьный код страны

Таблица городов — cities:

Поле	**Тип данных**	**Дополнительные параметры**	**Описание поля (можно указать в комментарии к полю)**
id	целое число	обязательное, автоинкремент, первичный ключ	ID - уникальный идентификатор записи

name	строка 255	обязательное	название города
founded_at	дата, время	необязательное, может быть NULL	дата основания города
country_id	целое число	обязательное, беззнаковое	ID страны, в которой находится этот город

Таблица классов животных — animal_classes:

Поле	**Тип данных**	**Дополнительные параметры**	**Описание поля (можно указать в комментарии к полю)**
id	целое число	обязательное, автоинкремент, первичный ключ	ID - уникальный идентификатор записи
name	строка, без ограничений длины	обязательное	название класса животных: млекопитающее, рыба, птица и тому подобное
can_flying	TINYINT(1)	по умолчанию false	признак — бывают ли среди них те, кто может летать

Таблица животных — animals:

Поле	**Тип данных**	**Дополнительные параметры**	**Описание поля (можно указать в комментарии к полю)**
id	целое число	обязательное, автоинкремент, первичный ключ	ID - уникальный идентификатор записи

name	строка, без ограничений длины	обязательное	название животного: хрюшка, кот, черепаха и тому подобные
can_flying	TINYINT(1)	по умолчанию false	признак — умеют ли летать
legs_count	целое число	необязательное, может быть NULL	количество лап
class_id	целое число	обязательное, беззнаковое	ID класса животного

Советы и рекомендации

- Обратите внимание: название таблицы — это название хранимого внутри неё элемента во множественном числе. При этом именование происходит в нотации snake_case. Например, наша таблица с животными: каждая запись — это животное (animal), значит, в целом таблица должна называться «Животные» (animals).
- Обратите внимание: название полей также указывается в нотации *snake_case*.
- В названии полей нет лишних уточнений. Например, в таблице городов нет поля city_name, вместо него — поле name. И это логично, так как в таблице городов каждая запись — это город, и поле «Название» — это и есть название города.
- Для создания обязательного поля никакие модификаторы к полю не нужны.
- Посмотрите, какие SQL-запросы формируются для создания таблицы. Не всегда у вас будет возможность использовать графические инструменты, знание синтаксиса может оказаться очень полезным.