**Билет 13**

Суть структуры данных является класс с открытыми переменными и без функций. Иногда такие структуры называются **объектами передачи данных**, или DTO (Data Transfer Object) (в отличие от business object не должен содержать какого-либо поведения). Структуры DTO чрезвычайно полезны, особенно при работе с базами данных, разборе сообщений из сокетов и т. д. С них часто начинается серия фаз преобразования низкоуровневых данных, полученных из базы, в объекты кода приложения.

**Объект передачи данных** (DTO) — это объект, используемый для передачи данных между различными уровнями в вашем приложении. Он не содержит бизнес-данных, а содержит только минимально необходимые данные для передачи между уровнями или приложениями.

Он неизменен и не имеет идентичности. Хотя DTO может существовать как изменяемый объект, если вы этого хотите.

В действительности Объект Передачи Данных действительно содержит данные Объекта. Но это не требование быть DTO. Мы можем преобразовать или собрать DTO из и в другой объект, который существует в нашем приложении или слое. Это означает, что объект передачи данных может передавать части или целые объекты между слоями или приложениями.

Возьмем, к примеру, Запрос из внешнего приложения. Этот запрос может быть создание совершенно новой сущности. Таким образом, запрос является массивом свойств желаемой сущности. Но они могут быть разными по названию или конструкции. Внешний мир не знает точную природу вашего домена, но он хочет каким-то образом изменить вашу модель.

Сам запрос неизменен. Вы получили эту просьбу от внешнего мира, вы не можете изменить этот факт. Таким образом, вместо работы с массивами мы можем создать DTO для работы с ним как с объектом. Затем этот объект можно передать на уровень вашего приложения и собрать в модель вашего домена.

Тогда каково большое преимущество между массивом и объектом? Ну, вы знаете структуру ваших объектов. Вы создаете строгий интерфейс того, какие свойства являются обязательными и разрешенными. При использовании массива вы должны быть осторожны, чтобы ваши данные не изменялись между слоями. И что какое-то внешнее приложение отправляет некоторые свойства, которые не должны изменяться внешним миром.

Работа с объектами передачи данных дает вам больше контроля и позволяет программисту и его пользователям делать меньше ошибок.

**Пример** объекта передачи данных:

Сlass ArticleDTO{

private $title;

private $content;

public function \_\_construct($title,$content) {

$this->title = $title;

$this->content = $content;

}

public function getTitle() {

return $this->title;

}

public function getContent() {

return $this->content;

}

}