

SOLID-ПРИНЦИПЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОЕКТИРОВАНИЯ И ИХ РЕАЛИЗАЦИЯ НА ЯЗЫКЕ PHP

*УО «Белорусский государственный университет информатики и радиоэлектроники»
г. Минск, Республика Беларусь*

Нестер А.А.

Куликов С.С. – к.т.н., доцент

В работе рассмотрены принципы SOLID, используемые при построении объектно-ориентированных систем. Так как данный метод построения систем сегодня используются при построении многих приложений и систем, то данная тема является весьма актуальной.

SOLID-принципы- это аббревиатура пяти основных принципов дизайна классов в объектно-ориентированном проектировании — Singleresponsibility, Open-closed, Liskovsubstitution, Interfacesegregation и Dependencyinversion.

Принцип единственной ответственности гласит — «На каждый объект должна быть возложена одна единственная обязанность». Т.е. другими словами — конкретный класс должен решать конкретную задачу — ни больше, ни меньше. Чтобы проверить, соблюдается ли данный принцип, можно воспользоваться следующим правилом - проверяем, сколько у нас есть причин для изменения класса — если больше одной, то следует разбить данный класс.

Данный принцип гласит — "программные сущности должны быть открыты для расширения, но закрыты для модификации". На более простых словах это можно описать так — все классы, функции и т.д. должны проектироваться так, чтобы для изменения их поведения, нам не нужно было изменять их исходный код. Правило для проверки данного принципа - представляем наш класс как «чёрный ящик» и смотрим, возможно ли в таком случае изменить его поведение.

Принцип подстановки Барбары Лисков (Liskovsubstitution) гласит — «Объекты в программе могут быть заменены их наследниками без изменения свойств программы». Данный принцип широко используется в контрактном программировании. Для проверки соблюдения условия, необходимо проверить, не усилили ли мы предусловия выполнения участка кода и не ослабили ли постусловия. Если это произошло — то принцип не соблюдается

Принцип разделения интерфейса (Interfacesegregation) гласит, что «Много специализированных интерфейсов лучше, чем один универсальный». Соблюдение этого принципа необходимо для того, чтобы классы-клиенты использующий/реализующий интерфейс знали только о тех методах, которые они используют, что ведёт к уменьшению количества неиспользуемого кода. Проверка соблюдения принципа происходит следующим образом - Проверяем, насколько много интерфейс содержит методов и насколько разные функции накладываются на эти методы, и если необходимо — разбиваем интерфейсы.

Принцип инверсии зависимостей (DependencyInvertion) гласит — «Зависимости внутри системы строятся на основе абстракций. Модули верхнего уровня не зависят от модулей нижнего уровня. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций». Данное определение можно сократить — «зависимости должны строится относительно абстракций, а не деталей». Для проверки выполнения принципа используем следующее правило - Проверяем, зависят ли классы от каких-то других классов(непосредственно инстанцируют объекты других классов и т.д) и если эта зависимость имеет место, заменяем на зависимость от абстракции.

Список использованных источников:

1. Роберт С. Мартин, Джеймс В. Ньюкирк, Роберт С. Косс Быстрая разработка программ. Принципы, примеры, практика, стр. 254-316
2. Мартин Р., Мартин М. Принципы, паттерны и методики гибкой разработки на языке C# , стр.357-404

ПРИМЕНЕНИЕ ДОКУМЕНТООРИЕНТИРОВАННЫХ NOSQL СУБД В СОВРЕМЕННЫХ ВЕБ-ПРИЛОЖЕНИЯХ

*УО «Белорусский государственный университет информатики и радиоэлектроники»
г. Минск, Республика Беларусь*

Н.А. Хмурович, О.А. Мацкевич

Куликов С.С. – к.т.н., доцент

В последнее время происходит стремительный рост популярности веб-технологий. Он обусловлен прежде всего тем, что веб остается единственной платформой, приложения которой одинаково хорошо работают на любом аппаратном обеспечении. С увеличением количества устройств, предоставляющих выход в интернет, возрастает популярность веб-приложений. Важной задачей становится хранение и