Публикация на тему

Шаблоны проектирования практических решений

В публикации рассмотрены шаблоны проектирования практических решений.

Автор
Михалькевич Александр Викторович

Публикация

Наименование Шаблоны проектирования практических решений

Автор А.В.Михалькевич

Специальность В публикации рассмотрены шаблоны проектирования практических решений.,

Анотация

Anotation in English

Ключевые слова

Количество символов 8871

Содержание

Введение

Что бы программист ни разрабатывал, на каком бы языке он ни писал, если он стремится к хорошему коду, он будет использовать шаблоны проектирования. Он стремится повторно воспользоваться решениями, которые оказались удачливыми ранее. Все существующие шаблоны проектирования условно можно разделить на две большие группы: архитектурные шаблоны и шаблоны практических решений. Если шаблоны практических решений – это

шаблоны решений конкретных задач, с которыми программист сталкиваемся в процессе решения задач, то архитектурные шаблоны проектирования – это способы структурирования файлов проекта.

1 Типы шаблонов проектирования практических задач

Практические решения, в свою очередь, подразделяются на порождающие шаблоны, структурные и шаблоны поведения.

1. Структурные шаблоны (*structural patterns*) – показывают, как объекты и классы объединяются для образования сложных структур.

2. Порождающие шаблоны (*creational patterns*) – контролируют процесс создания и жизненный цикл объектов.

3. Шаблоны поведения (*behavioral patterns*) – используются для организации, управления и объединения различных вариантов поведения объектов.

Каждый шаблон проектирования описывает задачи, с которыми программисту часто приходится сталкиваться. И затем описывает основу решения этой задачи таким образом, что позволяет воплотить это решение при разработке других программ, ни разу не повторившись.

2 Пораждающие шаблоны практических решений

Порождающие шаблоны или шаблоны создания объектов: абстрактная фабрика, одиночка, прототип, строитель, фабричный метод. Данная подгруппа шаблонов в той или иной степени работают с механизмами создания объектов.

Singleton - обеспечиваем существование в системе ровно одного экземпляра некоторого класса;

Factory Method - делегируем процесс создания объектов классам-наследникам;

Prototype - клонируем объекты на основании некоторого базового объекта;

Builder - отделяем процесс создания комплексного объекта от его представления;

Abstract Factory - описываем сущность для создания целых семейств взаимосвязанных объектов.

3 Структурные шаблоны практических решений

Структурные шаблоны: адаптер, декоратор, заместитель, компоновщик, мост, приспособленец, фасад. Они описывают создание более сложных объектов, либо упрощают работу с другими объектами сисетмы.

Adapter - на основании некоторого класса создаем необходимый клиенту интерфейс;

Facade - описываем унифицированный интерфейс для облегчения работы с набором подсистем;

Composite - работаем с базовыми исоставными объектами единым образом;

Decorator - динамически добавляем новую функциональность некоторому объекту, сохраняя его интерфейс;

Proxy - создаем объект, который перехватывает вызовы к другому объекту;

Bridge - разделяем абстракцию от интерфейса, позволяя им меняться независимо;

Flyweight - эффективно работаем с огромным количеством схожих объектов.

4 Поведенческие шаблоны практических решений

Поведенческие шаблоны: интерпретатор, итератор, команда, наблюдатель, посетитель, посредник, состояние, стратегия, хранитель, цепочка обязанностей, шаблонный метод. Они определяют эффективные способы взаимодействия различных объектов в системе.

Strategy - описывает набор взаимозаменяемых алгоритмов с единым интерфейсом;

Iterator - обеспечивает доступ к коллекциям объектов без раскрытия внутреннего устройства этих коллекций;

Observer - создает объект для отслеживания изменений в подсистеме и нотификации других подсистем;

Memento - сохраняет внутреннее состояние объекта для последующего использования без нарушения инкапсуляции;

Command - описывает объект, представляющий собой некоторое действие, которое можно выполнить в необходимый момент;

Interpreter - определяет способ вычисления выражений некоторого языка;

Mediator - создает объект, которые регулирует взаимодействие между набором подсистем;

State - позволяет объекту менять свое поведение при изменении его внутреннего состояния;

Template method - описывает алгоритм, возлагая реализацию некоторых частей алгоритма на подклассы;

Visitor - отделяет алгоритм от структуры, с которыми алгоритм работает;

Chain of responsibility - пропускает некоторый запрос через набор обработчиков событий, до тех пор пока запрос не будет обработан.

Заключение

Шаблоны группы практических решений применяем только тогда, когда имеется четкое понимание необходимости их использования и дополнительная гибкость действительно необходима. Если за гибкость приходится платить усложнением дизайна либо ухудшением производительности, либо подгоном решения под выбранный паттерн, тогда применение шаблонов практических решений необоснованно. Необоснованное применение сложных паттернов при решении простых задач усложняет задачу. Поэтому дальнейшее проектирование системы зависит от ответа на этот важный вопрос: использовать или не использовать при решении конкретной задачи готовое практическое решение.

Список использованных источников

Приложения