Описание системы контроля управления

Наша система состоит из нескольких блоков, общающихся посредством сообщений по принципу запрос-ответ. Служба доставки - это классическая система, управляемая событиями, основанная на избирательном вещании: события и обработчики сообщений обрабатывают события, происходящие в пользовательском интерфейсе и вне системы, и отправляют их получателю.

Чтобы взаимодействовать с другими участниками сети, устройство отправляет сообщение по их адресу в сети.

1. Событийный подход к управлению

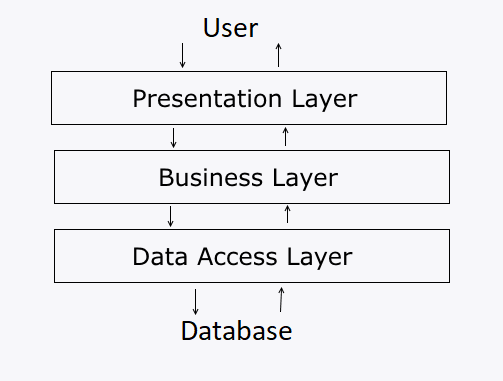
Преимущество этого широковещательного подхода состоит в том, что его устройство относительно простое. Подсистема для обработки определенных классов событий может быть интегрирована путем регистрации ее событий с помощью обработчика событий. Любая подсистема может активировать любую другую подсистему, не зная ее названия или местоположения. Подсистемы могут быть реализованы на распределенных машинах. Это распределение прозрачно для других подсистем.

Недостатком этой модели является то, что подсистемы не знают, будут ли обрабатываться события. Когда подсистема генерирует событие, она не знает, какие другие подсистемы зарегистрировали интерес к этому событию. Для разных подсистем вполне возможно зарегистрироваться на одни и те же события. Это может вызвать конфликты, когда результаты обработки события становятся доступными.

1. N-слойная модель

Многоуровневая архитектура - это клиент-серверная архитектура, в которой функции представления, обработки приложений и управления данными физически разделены.

Слой представляет логический уровень. То есть мы можем иметь уровень доступа к данным, уровень бизнес-логики, уровень представления, уровень обслуживания и так далее. Классическая трехуровневая система состоит из следующих уровней:



* Уровень презентации - это уровень, с которым пользователь напрямую взаимодействует. Этот уровень включает компоненты пользовательского интерфейса, механизмы ввода, контроллеры, объекты контекста запроса.
* Бизнес-уровень содержит набор компонентов, которые отвечают за обработку данных, полученных с уровня презентации, реализуют всю необходимую логику приложения, все вычисления, взаимодействуют с базой данных и передают результат обработки на уровень презентации.
* Уровень доступа к данным хранит модели, которые описывают используемые объекты, а также конкретные классы, работающие с различными технологиями доступа к данным, например, класс данных Entity Framework. Здесь также хранятся репозитории, через которые бизнес-уровень взаимодействует с базой данных.

Использование n-уровневой архитектуры имеет несколько преимуществ:

* Безопасность: вы можете защитить каждый из трех уровней отдельно, используя разные методы.
* Простота управления: вы можете управлять каждым уровнем отдельно, добавляя или изменяя каждый уровень, не затрагивая другие уровни.
* Масштабируемость: если вам нужно добавить больше ресурсов, вы можете сделать это для каждого уровня, не затрагивая другие уровни.
* Гибкость: кроме изолированной масштабируемости, вы также можете расширять каждый уровень любым способом, который диктуют ваши требования.
* Более эффективное развитие. N-уровневая архитектура очень удобна для разработки, так как на каждом уровне могут работать разные команды. Таким образом, вы можете быть уверены, что профессионалы дизайна и презентации работают на уровне представления, а эксперты базы данных работают на уровне данных.
* Легко добавлять новые функции. Если вы хотите ввести новую функцию, вы можете добавить ее к соответствующему уровню, не затрагивая другие уровни. Легко использовать повторно. Поскольку приложение разделено на независимые уровни, вы можете легко использовать каждый уровень для других программных проектов. Например, если вы хотите использовать ту же программу, но для другого набора данных, вы можете просто скопировать уровни логики и представления, а затем создать новый уровень данных.

1. Шаблоны репозитория и единицы работы

**Репозиторий** - это не что иное, как класс, определенный для объекта, со всеми возможными операциями над этим конкретным объектом.

Шаблон репозитория может быть реализован следующими способами:

* Один репозиторий на объект (не универсальный): этот тип реализации предполагает использование одного класса репозитория для каждого объекта. Например, если у вас есть две сущности Order и Customer, каждая сущность будет иметь свой собственный репозиторий.
* Универсальный репозиторий: Универсальный репозиторий - это тот, который может использоваться для всех объектов.

В приложениях ASP.NET шаблон репозитория часто используется для инкапсуляции рабочей логики и источников данных. Кроме того, очень часто мы оперируем несколькими сущностями и моделями, управление которыми требует нескольких классов репозитория. Шаблон Unit of Work позволяет упростить работу с несколькими репозиториями и дает разработчику гарантию того, что все репозитории будут использовать один и тот же контекст данных.

Поскольку мы будем использовать несколько репозиториев для каждой сущности, для упрощения использования соединения с базой данных мы будем использовать шаблон «Единица работы».

В шаблоне Unit of Work мы создаем класс Unit of Work, который позволяет нам предоставлять доступ к репозиториям через отдельные свойства и определяет общий контекст для общих репозиториев.