**Шаблоны [Patterns] проектирования**

|  |
| --- |
| **Название:** Единица работы [ Unit of Work ]   **Источник:**  Фаулер Мартин. Архитектура корпоративных программных приложений - Москва: издательский дом "Вильямс", 2004 г.  **Аннотация:**  Содержит список объектов, охватываемых бизнес-транзакцией, координирует запись изменений в базу данных и разрешает проблемы параллелизма.  **Эскиз:**  http://ooad.asf.ru/image/patterns/UnitOfWork.jpg  **Проблемы:**  Извлекая данные из базы данных или записывая в нее обновления, необходимо отслеживать, что именно было изменено: в противном случае сделанные изменения не будут сохранены в базе данных. Точно так же созданные объекты необходимо вставлять, а удаленные – уничтожать.  Разумеется, изменения в базу данных можно вносить при каждом изменении содержимого объектной модели, однако это неизбежно выльется в гигантское количество мелких обращений к базе данных, что значительно снизит производительность. Более того, транзакция должна оставаться открытой на протяжении всего сеанса взаимодействия с базой данных, что никак не применимо к реальной бизнес-транзакции, охватывающей множество запросов. Наконец, вам придется отслеживать считываемые объекты для того, чтобы не допустить несогласованности данных.  Типовое решение **единица работы** позволяет контролировать все действия, выполняемые в рамках бизнес-транзакции, которые так или иначе связаны с базой данных. По завершении всех действий оно определяет окончательные результаты работы, которые и будут внесены в базу данных.  **Принцип действия:**  Базы данных нужны для того, чтобы вносить в них изменения: добавлять новые объекты или же удалять или обновлять уже существующие. **Единица работы** – это объект, который отслеживает все подобные действия. Как только вы начинаете делать что-нибудь, что может затронуть содержимое базы данных, вы создаёте единицу работы, которая должна контролировать все выполняемые изменения. Каждый раз, создавая, изменяя или удаляя объект, вы сообщаете об этом **единице работы**. Кроме того, следует сообщать, какие объекты были считаны из базы данных, чтобы не допустить их несогласованности (для чего **единица работы** проверяет, не были ли запрошенные объекты изменены во время считывания).  Когда вы решаете зафиксировать сделанные изменения, **единица работы** определяет, что ей нужно сделать. Она сама открывает транзакцию, выполняет всю необходимую проверку на наличие параллельных операций и записывает изменения в базу данных. Разработчики приложений никогда явно не вызывают методы, выполняющие обновления базы данных. Таким образом, им не приходится отслеживать, что было изменено, или беспокоиться о том, в каком порядке необходимо выполнить нужные действия, чтобы не нарушить целостность на уровне ссылок, - **единица работы** сделает это сама.  Разумеется, чтобы **единица работы** действительно вела себя подобным образом, ей должно быть известно, за какими объектами необходимо следить. Об этом ей может сообщить оператор, выполняющий изменение объекта, или же сам объект.  **Назначение:**  Основным назначением **единицы работы** является отслеживании действий, выполняемых над объектами домена, для дальнейшей синхронизации данных, хранящихся в оперативной памяти, с содержимым базы данных. Если вся работа выполняется в рамках системной транзакции, следует беспокоиться только о тех объектах, которые вы изменяете. Конечно же, для этого лучше воспользоваться единицей работы, однако существуют и другие решения.  Пожалуй, наиболее простая альтернатива – явно сохранять объект после каждого изменения. Недостатком этого подхода является необходимость большого количества обращений к базе данных: например, если на протяжении выполнения одного метода объект был изменен трижды, нам придется выполнять три обращения к базе данных, вместо того чтобы ограничиться одним обращением по окончании всех изменений.  Во избежание многократных обращений к базе данных запись всех изменений можно отложить до окончания работы. В этом случае вам придется отслеживать все изменения, которые были внесены в содержимое объектов. Для реализации подобной стратегии в код можно ввести дополнительные переменные, однако учтите: если переменных слишком много, они становятся неуправляемыми. Переменные хорошо использовать со *сценарием транзакции (Transaction Script)*, но они совсем не подходят для *модели предметной области (Domain Model)*.  Вместо того чтобы хранить измененные объекты а переменных, для каждого объекта можно создать флаг, который будет указывать на состояние объекта – изменен или не изменен. В этом случае по окончании транзакции необходимо отобрать все измененные объекты и записать их содержимое в базу данных. Удобство этого метода зависит от того, насколько просто находить измененные объекты. Если все объекты находятся в одной иерархии, вы можете последовательно просмотреть всю иерархию и записать в базу данных все обнаруженные изменения. В свою очередь, более общие структуры объектов, в частности *модель предметной области*, просматривать гораздо труднее.  Огромным преимуществом **единицы работы** является то, что она хранит все данные об изменениях в одном месте. Поэтому вам не придется запоминать море информации, чтобы не упустить из виду все изменения объектов. Кроме того,**единица работы** может послужить прекрасной основой для разрешения более сложных ситуаций, таких, как обработка бизнес транзакций, охватывающих несколько системных транзакций. |