**10 лаба**

1. Что такое Свойства зависимости? Для чего они нужны?

Свойства зависимостей предназначены для предоставления способа вычисления значения свойства по значениям других входных данных.

Эти входные данные могут включать в себя системные свойства, такие как темы и пользовательские параметры, JIT-механизмы определения свойств, такие как привязка данных и анимации (раскадровки), шаблоны многократного использования, например ресурсы и стили, а также значения, известные благодаря отношениям между родительскими и дочерними элементами с другими элементами в дереве.

1. Как создать DependencyProperty?

Если мы хотим применять свойства зависимостей, то нам надо унаследовать свой класс от абстрактного класса DependencyObject. В нашем классе мы определяем два свойства зависимостей: TitleProperty и PriceProperty. Обратите внимание, что они объявляются с модификаторами public static readonly.

Затем свойства регистрируются в статическом конструкторе нашего класса с помощью метода Register. И в конце для них создаются обычные свойства-обертки, в которых мы получаем доступ к значению свойств с помощью методов GetValue и SetValue.

1. Для чего и как используют делегат ValidateValueCallback?

Делегат, который возвращает true, если значение проходит валидацию, и false - если не проходит.

1. Для чего и как используют делегат CoerceValueCallback?

Делегат, который может подкорректировать уже существующее значение свойства, если оно вдруг не попадает в диапазон допустимых значений.

1. Какие типы маршрутизируемых событий есть в WPF (поясните каждый)?

* Прямые (direct events) - они возникают и отрабытывают на одном элементе и никуда дальше не передаются. Действуют как обычные события.
* Поднимающиеся (bubbling events) - возникают на одном элементе, а потом передаются дальше к родителю - элементу-контейнеру и далее, пока не достигнет наивысшего родителя в дереве элементов.
* Опускающиеся, туннельные (tunneling events) - начинает отрабатывать в корневом элементе окна приложения и идет далее по вложенным элементам, пока не достигнет элемента, вызвавшего это событие.

1. Как создать RoutedEvent?

Для определения маршрутизированных событий в классе создвалось статическое поле по типу RoutedEvent.

Это поле, как правило, имеет суффикс Event. Затем это событие регистрируется в статическом конструкторе.

И также класс, в котором создается событие, как правило определяет объект-обертку над обычным событием. В этой обертке с помощью метода AddHandler происходит добавление обработчика для данного события, а с помощью метода RemoveHandler - удаление обработчика.

1. Поясните концепцию Command в WPF? В чем ее преимущества?

Команды представляют механизм выполнения какой-нибудь задачи. Использование команд помогает нам сократить объем кода и использовать одну и ту же команду для нескольких элементов управления в различных местах программы. Таким образом, команды позволяют абстрагировать набор действий от конкретных событий конкретных элементов.

1. Как используются команды?

См. вопрос 7.

1. Как создать RoutedUICommand?

WPF уже обладает большим набором встроенных команд. Все они представляют объекты класса RoutedUICommand, который является производным от RoutedCommand.

**11 лаба**

1. Что такое Binding в WPF? Какие атрибуты мы можем указывать и для чего?

Привязка подразумевает взаимодействие двух объектов: источника и приемника. Объект-приемник создает привязку к определенному свойству объекта-источника.

Некоторые свойства класса **Binding**:

* **ElementName**: имя элемента, к которому создается привязка
* **IsAsync**: если установлено в True, то использует асинхронный режим получения данных из объекта. По умолчанию равно False
* **Mode**: режим привязки
* **Path**: ссылка на свойство объекта, к которому идет привязка
* **TargetNullValue**: устанавливает значение по умолчанию, если привязанное свойство источника привязки имеет значение null
* **RelativeSource**: создает привязку относительно текущего объекта
* **Source**: указывает на объект-источник, если он не является элементом управления.
* **XPath**: используется вместо свойства path для указания пути к xml-данным

1. Что такое ADO.NET? Какие существуют режимы работы?

ADO.NET предоставляет собой технологию работы с данными, которая основана на платформе .NET Framework. Эта технология представляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций.

Функционально классы ADO.NET можно разбить на два уровня: подключенный и отключенный.

Подключенный

►Работает в режиме удержания подключения к базе.

►Обеспечивает максимальную гибкость и эффективность.

►Обеспечивает минимальный расход оперативной памяти.

Отключенный

►Обеспечивает работу с данными в отсутствии подключения к БД.

►Удобна для переноса данных по сети.

►Расходует достаточно много памяти

1. Объясните назначение класса SqlConnection.

Обеспечивают подключение к БД.

1. Объясните назначение класса SqlDataAdapter

Заполняет отсоединенный объект DataSet или обновляют данные на сервере из DataSet.

1. Объясните назначение класса SqlException. Чем данный класс отключается от класса-родителя Exception?

Исключение, которое возникает, когда SQL Server возвращает предупреждение или ошибку. Этот класс не может быть унаследован.

1. Объясните назначение класса SqlCommand.

Управляет источником данных с помощью SQL.

1. В чем разница между методами ExecuteNonQuery, ExecuteReader и ExecuteScalar?

* ExecuteScalar обычно используется, когда ваш запрос возвращает одно значение. Если он возвращает больше, то результатом будет первый столбец первой строки. Примером может быть SELECT @@IDENTITY AS 'Identity'.
* ExecuteReader используется для любого набора результатов с несколькими строками/столбцами (например, SELECT col1, col2 from sometable).
* ExecuteNonQuery обычно используется для операторов SQL без результатов (например, UPDATE, INSERT и т.д.).

1. Объясните назначение класса SqlDataReader.

Обеспечивает последовательный доступ к результату выполнения команды Select. Для работы требуют монопольный доступ к соединению.

1. Объясните назначение класса SqlParameter.

Представляет параметр для [SqlCommand](https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqlcommand?view=dotnet-plat-ext-3.1) и, при необходимости, его сопоставление со столбцами [DataSet](https://docs.microsoft.com/en-us/dotnet/api/system.data.dataset?view=dotnet-plat-ext-3.1) .

1. Объясните назначение класса SqlTransaction.

Представляет транзакцию Transact-SQL, которая должна быть сделана в базе данных SQL Server.

1. Объясните назначение классов DataSet, DataTable, DataColumn, DataRow, DataRelation.

Объекты DataTable используются для представления одной из таблиц базы данных в DataSet.

В свою очередь, DataTable составляется из объектов DataColumn. DataColumn - это блок для создания схемы DataTable.

Объекты DataRelation

Объект DataSet имеет также свойство Relations, возвращающее коллекцию DataRelationCollection, которая в свою очередь состоит из объектов DataRelation. Каждый объект DataRelation выражает отношение между двумя таблицами (сами таблицы связаны по какому-либо полю (столбцу)). Следовательно, эта связь осуществляется через объект DataColumn.

Строки (объект DataRow)

Коллекция Rows объекта DataTable возвращает набор строк (записей) заданной таблицы.

**13 лаба**

* 1. Для чего используют многоуровневые архитектуры?

При этом надо отметить, что крайние уровни не могут взаимодействовать между собой, то есть уровень представления (применительно к ASP.NET MVC, контроллеры) не могут напрямую обращаться к базе данных и даже к уровню доступа к данным, а только через уровень бизнес-логики.

* 1. Опишите назначение слоев многоуровневой архитектуре. Business layer (уровень бизнес-логики), Data Access layer (уровень доступа к данным).

**Business layer** (уровень бизнес-логики): содержит набор компонентов, которые отвечают за обработку полученных от уровня представлений данных, реализует всю необходимую логику приложения, все вычисления, взаимодействует с базой данных и передает уровню представления результат обработки.

**Data Access layer** (уровень доступа к данным): хранит модели, описывающие используемые сущности, также здесь размещаются специфичные классы для работы с разными технологиями доступа к данным, например, класс контекста данных Entity Framework. Здесь также хранятся репозитории, через которые уровень бизнес-логики взаимодействует с базой данных.

* 1. Поясните назначение и принцип использования паттерна Repository?

 Репозиторий позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой.

* 1. Поясните назначение и принцип использования паттерна Unit of Work?

Паттерн Unit of Work позволяет упростить работу с различными репозиториями и дает уверенность, что все репозитории будут использовать один и тот же контекст данных.

* 1. Опишите основные особенности библиотеки Entity Framework?

Entity Framework представляет ORM-технологию, которая позволяет абстрагироваться от структуры базы данных и может выполнять автоматически сопоставление таблиц и их данных с моделями классов, определенных разработчиком.

* 1. Какие преимущества обеспечивает использование слабосвязанного кода?

 Слабо связанный программный код – это код, в котором объекты довольно независимы друг от друга. Если объект сильно связан с другими, то во-первых его приходится часто модифицировать, так как малейшие изменения окружения требуют этого, а во-вторых затрудняется его повторное использование, поскольку при повторном использовании надо восстановить все его связи. В сильно связанном программном коде затруднительно рассмотрение каждого объекта в отдельности.

Как правило, репозиторий привязан к одной конкретной сущности или модели, данными которой он управляет. Хотя это необязательно - в репозитории мы можем предусмотреть механизм для загрузки связанных данных из других таблиц, которые связаны с основной моделью, и ряд аналогичных операций. Но тем не менее, часто для управлению одной сущность создается свой репозиторий.

Если репозитории используют одно и то же подключение, то нередко для организации доступа к одному подключению для всех репозиториев приложения используется другой паттерн - **Unit Of Work**. Класс, который реализует данный паттерн, как правило, содержит набор репозиториев и ряд некоторых общих для них функций.

Юзер.сс

Но если мы обратимся непосредственно к Entity Framework, то мы можем увидеть, то он уже реализует паттерны Unit Of Work и репозиторий. К примеру, контекст данных EF для выше обозначенных моделей мог бы выглядеть следующим образом.

Майдбсонтекст

Фактически класс майдбContext представляет реализацию UnitOfWork - он содержит ряд репозиториев. Каждый репозиторий представлен объектом DbSet, с помощью функциональности которого мы можем получать, добавлять, удалять данные.

К примеру реализуем паттерн репозиторий для работы через EF 6.

контроллеры

**15 лаба**

1. Что такое UML?

**UML** – унифицированный язык моделирования (Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования.

1. Перечислите типы диаграмм UML.

* **Диаграмма** вариантов использования (use case **diagram**)
* **Диаграмма** классов (class **diagram**)
* **Диаграмма** состояний (statechart **diagram**)
* **Диаграмма** последовательности (sequence **diagram**)
* **Диаграмма** кооперации (collaboration **diagram**)
* **Диаграмма** компонентов (component **diagram**)
* **Диаграмма** развертывания (deployment **diagram**)

1. Какие отношения между классами могут быть на диаграмме классов?

Наследование, агрегация, композиция, ассоциация, реализация.

Наследование является базовым принципом ООП и позволяет одному классу (наследнику) унаследовать функционал другого класса (родительского).

в незакрашенной стрелочке

Реализация предполагает определение интерфейса и его реализация в классах.

в незакрашенной стрелочке от класса к интерфейсу, только линия теперь пунктирная

1. Как обозначаются абстрактные классы на диаграмме классов?

 Если **класс абстрактный** — то его имя пишется полужирным курсивом.

5. Как обозначаются интерфейсы на диаграмме классов?

См записку

1. Как отображается доступность членов класса на диаграмме классов?

Так, например, на рисунке ниже атрибут “Атрибут1” является открытым (+) членом класса, а атрибут “Атрибут2” – закрытым (-).

7. Что такое агрегация? Как обозначается?

Композиция определяет отношение **HAS A**, то есть отношение "имеет". Например, в класс автомобиля содержит объект класса электрического двигателя.

При агрегации реализуется слабая связь, то есть в данном случае объекты Car и Engine будут равноправны.

 ромбик будет незакрашенным

8. Что такое ассоциация?

Ассоциация - это отношение, при котором объекты одного типа неким образом связаны с объектами другого типа.

На схемах UML ассоциация обозначается в виде обычно стрелки.

1. Какие обозначения используют на диаграмме последовательности?

Основными элементами диаграммы последовательности являются обозначения [объектов](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) (прямоугольники с названиями объектов), вертикальные «линии жизни» ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *lifeline*), отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), и стрелки, показывающие обмен сигналами или сообщениями между объектами.

1. Для чего нужна диаграмма последовательности?

**Диаграмма последовательности** действий (sequence diagram) отображает взаимодействие объектов, упорядоченное по времени.

1. Каково назначение диаграммы использования, пакетов и активности?

*диаграммы пакетов* (package diagrams) служат для разбиения объемных моделей на составные части, а также (традиционно) для группировки классов моделируемого ПО, когда их слишком много.

*диаграммы активностей* (*activity diagrams*) используются для спецификации бизнес-процессов, которые должно автоматизировать разрабатываемое ПО, а также для задания сложных алгоритмов;

Диаграмма вариантов использования в UML — диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.