1. Опишете накратко платформата Microsoft .NET.

Microsoft .NET Framework е [платформа](http://bg.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0_%28%D0%BA%D0%BE%D0%BC%D0%BF%D1%8E%D1%82%D1%80%D0%B8%29), създадена от [Microsoft](http://bg.wikipedia.org/wiki/Microsoft), която предоставя [програмен модел](http://bg.wikipedia.org/w/index.php?title=%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B5%D0%BD_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB&action=edit&redlink=1), библиотека от класове ([FCL](http://bg.wikipedia.org/w/index.php?title=FCL&action=edit&redlink=1), *Framework Class Library*) и среда за изпълнение на написан специално за нея програмен код ([CLR](http://bg.wikipedia.org/w/index.php?title=CLR&action=edit&redlink=1), *Common Language Runtime*). Тя е ключов елемент от стратегията за развитие на [Microsoft](http://bg.wikipedia.org/wiki/Microsoft), чиято цел е повечето нови приложения за [Windows](http://bg.wikipedia.org/wiki/Windows) да бъдат базирани на .NET Framework.

2. Кои са основните принципи, които залягат в нея (Microsoft .NET)?

* Платформена независимост
* Съвместимост с вече съществуващите технологии
* Common Runtime Engine
* Езикова независимост
* Улеснена инсталация
* Сигурност

3. Избройте четирите компонента, от които тя се състои (Microsoft .NET).

* .Net Framework Visual Studio .Net
* .Net Building Block Services
* .Net Smart Clients
* .Net Enterprise Services

4. Какво представляват .NET Enterprise сървърите?

.NET Enterprise сървърите предоставят

Сървърна инфраструктура за .NET платформата

Среда за изпълнение, управление и интеграция на XML Web услуги

Ключови характеристики

Оркестрация на бизнес процесите в приложенията и услугите (business process orchestration)

Силна поддръжка на XML

Сигурни

Надеждни

Скалируеми

Управляеми

5. Избройте някои от .NET Enterprise сървърите.

* Microsoft Windows Servers Family –фамилия операционни системи
* Microsoft SQL Server – управление на релационни, многомерни данни и XML
* Microsoft BizTalk Server – интеграция на бизнес процеси, услуги и системи
* Microsoft Host Integration Server –интеграция на стари системи
* Microsoft Exchange – координация на съвместната работа в организации
* Microsoft SharePoint Portal Server –сътрудничество и споделяне на информация в реално време

6. Какво е Visual Studio .NET?

Microsoft Visual Studio е мощна [интегрирана среда за разработка](http://bg.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%B0%D0%BD%D0%B0_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D0%B7%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0) на софтуерни приложения за [Windows](http://bg.wikipedia.org/wiki/Windows) и за платформата [.NET Framework](http://bg.wikipedia.org/wiki/.NET_Framework). Използва се за разработка на конзолни и [графични потребителски интерфейс](http://bg.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D0%BD_%D0%BF%D0%BE%D1%82%D1%80%D0%B5%D0%B1%D0%B8%D1%82%D0%B5%D0%BB%D1%81%D0%BA%D0%B8_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) приложения, както и Windows Forms или WPF приложения, [уеб сайтове](http://bg.wikipedia.org/wiki/%D0%A3%D0%B5%D0%B1%D1%81%D0%B0%D0%B9%D1%82), [уеб приложения](http://bg.wikipedia.org/wiki/%D0%A3%D0%B5%D0%B1_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) и [уеб услуги](http://bg.wikipedia.org/wiki/%D0%A3%D0%B5%D0%B1_%D1%83%D1%81%D0%BB%D1%83%D0%B3%D0%B0) на всички поддържани платформи от Microsoft Windows, [Windows Mobile](http://bg.wikipedia.org/wiki/Windows_Mobile), Windows CE, .NET Framework, .NET Compact Framework и Microsoft Silverlight.

7. За какво служат .NET Building Block услугите?

Автентикация – .NET Passport (single sign-on)

Доставка на съобщения

Съхранение на лични потребителски данни – документи, контакти, електронна поща, календар, любими сайтове и т.н.

Съхранение на настройките на приложенията

8. Какво са .NET Smart клиентите?

.NET smart клиентите

Представляват smart клиенти, базирани на .NET Framework

Работят върху различни smart устройства (компютри, мобилни телефони, handhelds, вградени устройства, ...)

Осигуряват достъп до информацията на потребителя – навсякъде и по всяко време

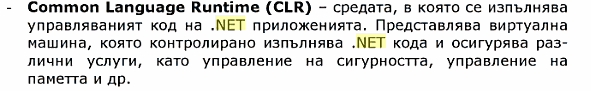
.NET Framework, .NET Compact Framework и VS.NET позволяват разработката на smart клиенти за най-разнообразни smart устройства

9. Какво е характерно за тях (.NET Smart клиентите)?

* .Net предоставя специализирана инфраструктура, която подпомага и улеснява реализацията на Smart Client приложения.
* Работят както на настолни компютри така и на преносими устройства: мобилни устройства, hand held, вградени системи и т.н.
* Основната им задача е да предоставят достъп до информацията, нужна на потребителя, по всяко време и във формат удобен за потребителя.

10. Какво представлява средата за контролирано изпълнение на програмен код Common Language Runtime (CLR)?

* Управление на паметта (garbage collection)
* Интеграция на различни езици
* Поддържа множество версии за компонентите (т. нар. assemblies) – няма вече конфликти
* Интегрирана сигурност
* CLR управлява процеса на изпълнение на кода
* Управлява паметта, конкурентността, ...
* Code-based security
* Role-based security



11. Какво е Framework Class Library (FCL)? Каква функционалност предлага тя?

* Предоставя основната функционалност за разработка:

**ASP.NET, Web Services, ADO.NET, Windows Forms, IO, XML, ...**

****

12. Какво е управляван код?

Управляваният код (Managed code)

* Кодът, който CLR изпълнява
* Win32 кодът не е управляван
* Представлява програмен код на езика от ниско ниво IL (Intermediate Language)
* Описва се от метаданни:

описание на класове, интерфейси, свойства, полета, методи, параметри, типове данни, ...

описание на библиотеките от класове

описание на сигурността

* Програмите, написани на всеки .NET език, се компилират до управляван код

13. Има ли причина да бъде използван управляван код вместо традиционния native код?

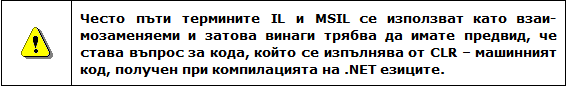
* Обектно-ориентиран
* Защитен от неправилна работа с типове (type-safety)
* Сигурен
* Надежден
* Позволява интеграция между различни езици за програмиране
* Позволява преносимост между различни платформи

Неуправляваният код (Win32 кодът)

* няма защита на паметта и типовете
* създава проблеми с надеждността
* не съдържа метаданни, които го описват

14. Какво е характерно за езика MSIL?

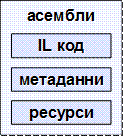
Имплементацията на IL в .NET Framework се нарича **MSIL (Microsoft Intermediate Language)**. IL може да има и други имплементации в други платформи и среди за изпълнение на .NET код.Езикът IL е стандартизиран от организацията ECMA и в съответния стан­дарт се нарича **CIL (Common Intermediate Language)**.



Common Intermediate Language (CIL, произнесена или "SIL" или "Кил") (наричана по-рано Microsoft Intermediate Language или MSIL) е четим за човека език за програмиране на най-ниско ниво, определено от Common Language инфраструктура (CLI) Спецификация и се използва от .NET Framework и Mono. Езици, които са насочени към Runtime Environment CLI-съвместим компилират до CIL, който се сглобява в един код обект, който има формат байткод стил. CIL е обектно-ориентиран език за монтаж и е изцяло стека основа. Нейната байткод е преведена на родния код или - най-често - изпълнен от виртуална машина. (Какво е характерно е подчертано)  
Първоначално CIL е известен като Microsoft Intermediate Language (MSIL) по време на бета версии на езиците .NET. Благодарение на стандартизацията на C # и Common Language инфраструктура, на байткод е вече официално известен като CIL.

15. Какво представляват .NET асемблитата (assemblies)? Каква информация съдържат са метаданните в асемблитата?

Асемблитата са най-малката самостоятелна градивна единица в .NET Framework. Те представляват наследници на познатите ни .exe и .dll файлове и съдържат **IL изпълним код**, **метаданни** и **ресурси**:



За разлика от неуправляваните изпълними файлове, асемблитата са самоописващи се и носят в себе си информация за всички класове, типове и ресурси, които съдържат, както и информация за сигурността, за зависимост от външни компоненти и др. Тази информация се нарича метаданни.

Асемблитата имат собствена версия и дефинират изисквания към правата, свързани със сигурността, на потребителя или процеса, който ги изпъл­нява. Те могат да имат и цифров подпис, положен от създателя им, чрез който се осигурява повишена сигурност.

16. Какво са .NET приложенията?

.NET приложенията се състоят от едно или повече асемблита, в които се съдържат техният код и ресурси. Те пред­став­ля­ват изпълними единици, които могат да бъдат конфигурирани.

В зави­симост от вида си .NET приложенията могат да бъдат самостоятелни или обвързани с други услуги или приложения. Например уеб приложе­нията не са самостоятелни и се изпълняват в средата на ASP.NET, докато конзолните приложения могат да се изпълняват самостоятелно.

За разлика от повечето Win32 приложения, .NET приложенията могат да бъдат инсталирани с просто копиране (XCOPY deployment), без да се налага регистриране на отделните им компоненти (регистрирането се налага само, ако искаме да позволим неуправлявани компоненти да могат да достъпват нашите асемблита). При .NET приложенията не се използва Windows Registry за регистрация на компонентите.

17. Какво е application domain?

Application domain (домейн на приложението) е ново понятие, което се въвежда с .NET Framework. То представлява допълнително ниво на изо­лация между отделни .NET приложения, изпълнявани в един и същ процес на опера­ци­онната система.

За да се ограничат възможните проблеми, свързани с манипулиране на паметта, в операционната система всеки процес разполага със собствена памет, с която работи, и няма право да чете или пише в паметта на друг процес. Ако се налага такова взаимодействие, то се извършва индиректно например чрез прокси обекти.

.NET процеси в CLR

Няколко application domain-а могат да работят в един процес на ОС

изолирани един от друг

без да се нарушава сигурността

Не се създава отделен процес в ОС за всеки application domain (.NET процес)

Подобрява се производителността

Намалява се натоварването и разхода на системни ресурси (памет, процесорно време, ...)

18. Какво е Common Language Specification (CLS)? Защо е необходима тя?

CLS дефинира общите и задължителни характеристики, които един прог­рамен език трябва да притежава, за да бъде съвместим с останалите .NET езици. Тази спецификация има за цел да минимизира разликите между .NET езиците.

CLS, например, налага ограничението да се прави разлика межди малки и главни букви в имената на типовете и техните публични членове, методи, свойства и събития. Ако нарушим това правило, нашият код ще се компи­лира, но ще загуби съвместимостта си с CLS и другите .NET езици.

Друго ограничение, което CLS налага, e езиците да бъдат обектно-ориен­тирани. Това означава, че за да бъде направен даден език съвместим с CLS и .NET Framework, той трябва да бъде разширен да поддържа класове, интерфейси, свойства, изключения и всички останали елементи на обектно-ориентираното програмиране с .NET.

Повечето .NET езици поддържат много повече възможности от тези, които изисква CLS. Поради това трябва да сме внимателни при създаването на класове и други типове и да подхождаме с ясната идея дали искаме те да са CLS съвместими или не.

19. Какво представлява общата система от типове в .NET Framework (Common Type System)?

Общата система от типове в .NET Framework представлява формална спецификация на типовете данни, използвани в различните .NET езици за програмиране. CTS описва различните .NET типове (примитивни типове данни, класове, структури, интерфейси, делегати, атрибути и др.). В CTS се описват съдържанието и начина на дефиниране на типовете, модифи­каторите за достъп, начините за наследяване, времето на живот на обектите и много други технически характеристики.

CTS дефинира двата типа обекти – **референтни** и **стойностни**, според това как се пазят в паметта и как се манипулират. CTS налага задълже­нието всички типове да наследяват системния тип System.Object, дори и примитивните. Благодарение на това извикването "5.ToString()" е на­пълно валидно в езика C#.

20. Избройте няколко от .NET езиците. Какво е общото между тях?

Microsoft предлагат компилатори и поддръжка във Visual Studio .NET (общото между тях) за следните езици:

-     C# - препоръчителният език за програмиране под .NET Framework.

-     Visual Basic .NET – обновена версия на езика Microsoft Visual Basic

-     C++ (managed/unmanaged) – езикът C++ по идея е език от доста по-ниско ниво в сравнение със C# и VB.NET.

-     J# – езикът J# е създаден за да позволи по-лесното прехвърляне на Java приложения към C#.

-     JScript.NET – езикът JScript.NET е представител на слабо типизи­раните скрип­тови езици от фамилията ECMAScript (като JavaScript, VBScript и JScript), но е адаптиран към .NET Framework.

21.Избройте основните библиотеки от Framework Class Library (FCL).

Основните библиотеки, от които се състои FCL, са:

-     Base Class Library – библиотека съдържаща основните средства, нужни за разработване на приложения

-     ADO.NET и XML – осигуряват достъп до бази данни и средства за обработка на XML.

-     ASP.NET – предоставя ни рамкова среда (framework) за разработка на уеб приложения с богата функционалност, както и средства за създаване и консумиране на уеб услуги.

-     Windows Forms – служи за основа при разработването на Windows приложения с прозоречно-базиран графичен потребителски интер­фейс.

*От Лекция 3*

1. Формулирайте основните принципи на обектно-ориентираното програмиране.

Обектно-ориентираното програмиране (ООП) моделира обектите от реалния свят със средствата на програмния език

* Обектите в ООП се характеризират с

атрибути (свойства)

операции (възможни действия)

Основни принципи на ООП

* Капсулация на данните
* Наследяване
* Полиморфизъм
* Използване на изключения

Обектно-ориентираното програмиране позволява преизползване на програмния код (code reuse)

2. Дефинирайте понятията клас, обект, атрибут, метод, капсулация на данните, абстракция на данните и действията, наследяване, полиморфизъм.

* Клас – категория обекти с общи свойства и операции, които могат да се извършват върху тях (например всички сметки в дадена банка)
* Обект – единичен обект от даден клас, инстанция на клас (например банковата сметка на Светлин Наков)
* Интерфейс – спецификация на съвкупност от действия, които даден обект предлага
* Свойство – видима за външния свят характеристика (атрибут) на даден обект
* Метод – операция, която всички обекти от даден клас могат да извършват
* Капсулация на данните – събиране на данните за даден обект и операциите над тях в едно цяло (клас**),** като се ограничава директния достъп до тези данни и операции
* Абстракция на данните – възможността да работим с данни без да се интересуваме от тяхното вътрешно представяне, а само от операциите над тях
* Абстракция на действията – възможността да изпълняваме действия, за които не знаем точно как са реализирани
* Наследяване – възможността един клас (наследник) да придобие свойства и действия от друг клас (родител)
* Полиморфизъм – възможността да разглеждаме обекти от клас-наследник като обекти от базов клас, като класът наследник може да е предефинирал някои от действията на базовия клас.

3. Какви са разликите между клас и структура?

Клас:

В .NET класовете са класическа реализация на понятието клас от обектно-ориентираното програмиране

Много приличат на класовете в C++ и Java

Класовете имат членове (class members)

* полета, константи, методи, свойства, индексатори, събития, оператори, конструктори, деструктори
* вложени типове (вложени класове, структури, интерфейси, делегати, ...)

Членовете имат видимост

* public, private, protected, internal

Членовете могат да бъдат

* статични (общи) или за дадена инстанция

Структура:

Структурите представляват съвкупност от полета с данни

Приличат много на класовете, но са типове по стойност

* Най-често се разполагат в стека
* Предават се по стойност
* Унищожават се при излизане от обхват

За разлика от структурите класовете са типове по референция и се разполагат в динамичната памет

* Създаването и унищожаването им е по-бавно

При правилна употреба заместването на класове със структури може значително да увеличи производителността

Не се препоръчва в структурите да има методи с логика – трябва да съдържат само данни

*Лекция 4*

1. Избройте основните разлики между стойностните и референтните типове.

Типове по стойност (value types)

* Директно съдържат стойността си
* Не могат да приемат стойност null
* Съхраняват стойността си в стека за изпълнение на програмата
* Унищожават се при излизане на съответната променлива от обхват
* При извикване на метод се предават по стойност, записана в стека
* Наследяват типа System.ValueType

Стойностни типове са:

* Примитивните типове (int, char, float, …)
* Структурите, дефинирани от програмиста
* Изброените типове (enumerations)

Типовете по референция (reference types)

* Представляват типово-обезопасени указатели към някакъв обект
* Съхраняват се в динамичната памет, в т. нар. managed heap
* При извикване на метод се предават по референция (по адрес)
* Когато не са необходими се унищожават автоматично от garbage collector-а на CLR
* Могат да приемат стойност null
* Възможно е няколко променливи да сочат към един и същ обект от референтен тип

Референтни типове са:

* System.Object, System.String, указателите, класовете, интерфейсите, масивите, делегатите

Кои от следните типове са стойностни и кои референтни: int, char, string, float, изброени типове, класове, структури, интерфейси, делегати, масиви, указатели, опаковани стойностни типов?

Първите 5 са стойностни, а останалите референтни.

**По-съществени разлики между стойностните и референтните типове:**

* При присвояване на стойностни типове се копира самата им стойност, а при референтни типове – само референцията (адреса)
* При създаване на променлива от стойностен тип тя се заделя в стека, а при референтните типове – в динамичната памет
* Стойностните типове не могат да приемат стойност null, защото не са адреси
* Стойностните типове наследяват типа System.ValueType, а референтните наследяват директно System.Object
* Променливи от стойностен тип могат да се съхраняват в променливи от референтен тип чрез т.нар. опаковане (boxing)

*Лекция 5*

1. Обяснете какво представляват делегатите в .NET Framework.

* Делегатите представляват .NET типове, които описват сигнатурата на даден метод (броя, типа и последователността на параметрите му) и връщания от него тип
* Делегатите приличат на указателите към функции в C и C++ – съдържат силно-типизиран указател (референция) към метод
* Те са структури от данни, които приемат като стойност методи, отговарящи на описаната от делегата сигнатура
* Чрез тях се осъществяват "обратни извиквания" (callbacks)
* Могат да сочат както към статични методи, така и към методи на инстанция

2. Обяснете какво представляват събитията (events) в .NET Framework.

* В компонентно-ориентираното програмиране компонентите изпращат събития (events) към своя притежател за да го уведомят при настъпване на интересна за него ситуация
* Обектът, който предизвиква дадено събитие, се нарича изпращач на събития (event sender)
* Обектът, който получава дадено събитие, се нарича получател на събитието (event receiver)
* За да получават дадено събитие получателите му предварително се абонират за него (subscribe for event)
* В компонентния модел на .NET Framework абонирането, изпращането и получаването на събитията се поддържа чрез делегати и събития
* Събитията в C# са специални инстанции на делегати, декларирани с ключовата дума event
* За променливите от тип събитие C# компилаторът автоматично дефинира операторите += и -= съответно за абониране за събитието и за премахване на абонамент
* Събитията могат да предефинират кода за абониране и премахваме на абонамент

3. Какво се препоръчва от утвърдената конвенция за събитията в .NET Framework?

В .NET Framework се използва утвърдена конвенция за събитията:

* Делегатите, които се използват за събития:

имат имена образувани от глагол + EventHandler (SomeVerb*EventHandler)*

* връщат void и приемат два параметъра – обект-изпращач от тип System.Object и обект, съдържащ данни за събитието от тип, наследник на System.EventArgs

пример:

public delegate ItemChangedEventHandler(

object aSender, ItemChangedEventArgs aEventArgs);

Събитията се обявяват като public, започват с главна буква и завършват с глагол, например:

public event ItemChangedEventHandler ItemChanged;

За предизвикване на събитие се създава protected void метод с име в стил OnVerb, например:

protected void OnItemChanged() { … }

Методът-получател (обработчик) на събитието има име Обект\_Събитие:

private void OrderList\_ItemChanged() { … }

*Лекция 6*

1. Обяснете какво представляват атрибутите в .NET Framework.

* Атрибутите представляват специални декларативни тагове, които служат за прикрепяне на описателна информация (анотации) към декларациите в кода
* При компилация се записват в метаданните на асемблито
* Могат да се извличат от метаданните и да се обработват от различни инструменти
* Представляват инстанции на класове, наследници на System.Attribute
* Програмистите могат да използват стандартните атрибути от .NET Framework и допълнително да дефинират собствени

2. Как се прилагат атрибути?

За да се приложи атрибут, името му се огражда в квадратни скоби и се поставя преди декларацията, за която се отнася:

*// прилагаме атрибута System.FlagsAttribute*

[Flags]

public enum FileAccess

{

Read = 1,

Write = 2,

ReadWrite = Read | Write

}

Атрибутът [Flags] указва, че изброеният тип може да се третира като битово поле, т.е. като множество от битови флагове

3. Как се прилагат атрибути с параметри?

В примера атрибутът [DllImport] се инстанцира от компилатора така:

Създава се обект от класа System.Runtime.

* InteropServices.DllImportAttribute

В конструктора му се подава като параметър стойност "user32.dll"

В публичното му поле EntryPoint се записва стойност "MessageBox"

[DllImport("user32.dll", EntryPoint="MessageBox")]

public static extern int ShowMessageBox(int hWnd,

string text, string caption, int type);

...

ShowMessageBox(0, "Some text", "Some caption", 0);

4. Как се задава цел при прилагане на атрибут?

При прилагането на атрибути може да се задава целта, към която да се приложат:

*// следващите атрибути се прилагат върху цел "assembly"*

[assembly: AssemblyTitle("Attributes Demo")]

[assembly: AssemblyCompany("DemoSoft")]

[assembly: AssemblyProduct("Entreprise Demo Suite")]

[assembly: AssemblyCopyright("(c) 1963-1964 DemoSoft")]

[assembly: AssemblyVersion("2.0.1.37")]

[Serializable] *// подразбира се [type: Serializable]*

class TestClass

{

[NonSerialized] *// подразбира се [field: NonSerialized]*

private int mStatus;

...

}

Къде се използват атрибути?

В .NET Framework вътрешно атрибутите се използват за най-разнообразни цели:

* За декларативно управление на сигурността
* При използване на автоматизирана сериализация на обекти
* В компонентния модел на .NET за взаимодействие с дизайнера
* При създаване на Web-услуги в ASP.NET
* При създаване на потребителски контроли в ASP.NET
* За взаимодействие с неуправляван (Win32) код
* При използване на Remoting технологията

*Лекция 8*

1. Опишете накратко стандарта Unicode. Каква е основната му идея? Има ли връзка между Unicode и типа System.Char в .NET Framework?

Стандартът Unicode съпоставя уникален номер на всеки знак (символ) от азбуките и йероглифните системи на езиците по цял свят

Например:

* знакът "A" (A главно, латинско) има номер 65 (0x41)
* знакът "⅔" (математически символ "две трети") има номер 8532 (0x2154)
* знакът "♥" (сърце) има номер 9829 (0x2665)
* знакът " " (листо на традиционен китайски) има номер 33865 (0x8449)
* знакът " " (музикален символ "нота шестнайсетинка") има номер 119137 (0x1D161)

Номерирането на знаците е глобално и не зависи от платформата, операционната система и езика за програмиране

Unicode 4.0 стандартът дефинира 96 248 знака, но кодиращите схеми, които се използват, позволяват да се поддържат над 1 милион различни знака

Една графема (графичен символ) може да се състои от:

* единичен Unicode знак
* последователност от базов Unicode знак и комбиниращи Unicode знаци

Например:

* графемата "A" се представя с единичен Unicode знак 0x0041
* графемата "А̣" (А с точка отдолу) се представя с последователността 0x0041, 0x0323

За съхранение и пренасяне на Unicode текстове се използват кодиращи схеми, които представят всеки знак като поредица от 8-битови, 16-битови или 32-битови стойности

В .NET Framework се използва кодиращата схема UTF-16, която представя един знак като една или две 16-битови стойности

* По-често използваните Unicode знаци се представят с една 16-битова стойност, а останалите – с две (с кодова двойка)

Стандартът Unicode постоянно се развива под контрола на Unicode консорциумът – [www.unicode.org](http://www.unicode.org/)

2. Опишете какво представляват символните низове в .NET Framework. Какво е характерно за класа System.String?

Типът System.Char е стойностен тип и съхранява 16-битова стойност

Една инстанция на System.Char не винаги съответства на един Unicode знак или графема

Някои Unicode графеми се съхраняват в две или повече Unicode стойности

Основни методи на System.Char:

* За класификация на символите: IsLetter(…), IsDigit(…), IsSeparator(…), IsSymbol(…), IsWhiteSpace(…), GetUnicodeCategory(…), ...
* За смяна на регистъра (малки/главни букви) – ToLower(), ToUpper() – може да се задава култура (ще разгледаме това по-нататък)

Символните низове (strings, стрингове) представляват последователности от Unicode знаци в кодиране UTF-16

В .NET Framework за работа с низове се използва класът System.String

Класът System.String (string в C#)

* Съдържа неизменима последователност от символи (System.Char)
* Използва Unicode за да поддържа много езици едновременно
* Съхранява низовете в динамичната памет (защото System.String е референтен тип)

3. Избройте по-важните методи и свойства на класа System.String и обяснете за какво служат.

По важни методи и свойства на класа System.String:

* Length – връща дължината на низа
* Equals(string) – сравнява низа с друг низ
* Compare(string, string) – сравнява два низа лексикографски един с друг (прави се разлика между малки и главни букви)
* оператори == и != – също сравняват низове
* оператори + и += – слепват низове
* this[int] – индексатор, който връща символа на зададената позиция (броенето започва от 0)
* StartsWith(string) – проверява дали низът започва с посочения низ
* EndsWith(string) – проверява дали низът завършва с посочения низ
* Substring(int startIndex, int Length) – извлича подниз по дадено начало и дължина
* IndexOf(string) – връща позицията на първото срещане на посочения низ или -1.
* LastIndexOf(string str) – връща позицията на последното срещане на посочения низ
* Join(string, string[]) – слепва поредица от низове посредством зададен разделител
* Format(string, params object[]) – връща низ, получен чрез замяна на форматиращите идентификатори с посочените за тях стойности (като Console.WriteLine(…))
* PadLeft(int, char) – подравнява отляво до посочената дължина със зададения символ
* PadRight(int, char) – подравнява отдясно до посочената дължина със зададения символ
* ToLower() – заменя всички главни букви със съответните им малки, обратно на ToUpper()
* Insert(int index, string) – връща нов низ, получен чрез вмъкване на посочения низ преди зададената позиция
* Remove(int index, int count) – връща нов низ, получен чрез премахване на зададена поредица от символи
* Replace(string oldVal, string newVal) – връща низ, получен чрез замяна на всички срещания на даден низ с даден друг низ

*Лекция 9*

1. Опишете накратко какво представляват регулярните изрази. Кои са основните елементи на езика на регулярните изрази? Какви метасимволи познавате?

Регулярните изрази са мощно, удобно и ефективно средство за обработка на текст

С регулярни изрази можем:

* да търсим и извличаме информация от даден текст по даден шаблон
* да валидираме текстова информация
* да заменяме и изтриваме поднизове в даден текст чрез шаблони

Регулярните изрази използват като математическа основа теорията на крайните автомати и регулярните езици

В .NET Framework регулярните изрази имат синтаксиса на Perl 5

Регулярен израз е символен низ, който описва някаква съвкупност от символни низове (регулярен език)

Регулярните изрази използват специална граматика за описание на низовете

Например регулярният израз [0-1]+

обозначава всички непразни низове, които се състоят само от цифрите 0 и 1, а низът 088[0-9]{7}

обозначава всички телефонни номера, които имат вида 088XXXXXXX (X е цифра)

Метасимволите в регулярните изрази са няколко категории:

* Escaping последователности, например \\*
* Класове от символи, например [a-zA-Z]
* Метасимволи за задаване на количеството (quantifiers), например \* и +
* Метасимволи за задаване на местоположението в текста, например \b
* Метасимволи за алтернативен избор, например | (логическо "или")
* Групиращи метасимволи, например ([0-9]+)
* Други метасимволи, например # (за задаване на коментари), заместващи (например $1), …
* \t – табулация
* \r – символ за връщане на каретката CR (0x0D)
* \n – символ за нов ред LF (0x0А)
* \xXX – символ с ASCII код XX (шестнайсетично)
* \uXXXX – Unicode символ с номер XXXX (шестнайсетично)
* \\ – символ \
* \\* – символ \*
* \+ – символ +

. – обозначава произволен символ без \n

(.|\s) – обозначава произволен символ

[символи] – обозначава произволен символ от изброените

* Пример: [01] обозначава цифрата 0 или 1

[^символи] – обозначава произволен символ, който не е сред изброените

* Пример: [^<>\\] обозначава всеки символ без <, > и \

[charX-charY] – обозначава символ в зададения интервал

* Пример: [0-9A-F] обозначава всеки символ, който е цифра или латинска буква между A и F
* **\w – обозначава буквите, цифрите и символа \_ (за всички езици от Unicode)**
* **\W – обозначава всички символи с изключение на буквите, цифрите и \_**
* **\s – обозначава символите за празно пространство (интервал, табулация, нов ред, ...)**
* **\S – обозначава символите, които не са празно пространство**
* **\d – обозначава десетичните цифри [0-9]**
* **\D – обозначава всички символи, които не са десетични цифри**

2. Опишете накратко средствата на .NET Framework за работа с регулярни изрази - основните класове и по-важните им методи.

В .NET Framework средствата за обработка на текст чрез регулярни изрази са разположени в пространството от имена System.Text.RegularExpressions

Класът Regex съдържа неизменим регулярен израз и методи за търсене, заместване и разделяне на низове чрез този израз

* Regex предоставя и статични методи за всички основни операции

Класът Match съдържа описание на едно съвпадение (стойност, начална позиция и дължина), получено в резултат от търсене с регулярен израз

* Дава възможност за намиране на следващото съвпадение от търсенето, ако такова има

Класовете от пространството с имена System.Text.RegularExpressions

* Класът MatchCollection съдържа списък от съвпадения (получени в резултат от търсене)
* Класът Group представлява група от символи, съдържаща се в дадено съвпадение (Match). В едно съвпадение може да има няколко групи
* Класът GroupCollection съдържа списък от групи, съдържащи се в дадено съвпадение
* Делегатът MatchEvaluator се използва при заместване с регулярен израз за обработка на всяко едно съвпадение
* Изброеният тип RegexOptions се използва за задаване на опции за търсенето с рег. изрази

В .NET Framework класът Regex е най-важният клас за работа с регулярни изрази

С Regex може да се работи по два начина:

* Инстанцира се класът и в конструктора му се подава регулярен израз, след което му се извикват методите за обработка на текст (IsMatch, Match, Matches, Replace, Split)
* Използват се статичните методи на класа (IsMatch, Match, Matches, Replace, Split), на които се подава текста за обработка и регулярен израз, който да се използва

Подходът с инстанцирането на Regex е по-ефективен ако с един и същ израз ще се обработват последователно няколко текста

По-важни методи и свойства на Regex:

* IsMatch(text, pattern) – проверява дали в даден текст се среща поне един подниз, който съответства на даден регулярен израз
* Match(text, pattern) – търси зададения регулярен израз в зададения текст и връща първото съвпадение като Match обект
* Matches(text, pattern) – търси зададения регулярен израз в зададения текст и връща MatchCollection от всички съвпадения
* Replace(text, pattern, replacement) – замества всички срещания за даден регулярен израз в даден текст със заместващ текст, който може да съдържа части от намерените съвпадения (групи в рег. израз)

*Лекция 10*

1. Какво знаете за автоматичното управление на паметта и ресурсите в .NET Framework?

* Инстанциите на референтните типове в .NET се съхраняват в динамичната памет (в т. нар. managed heap)
* При "препълване" на динамичната памет се включва системата за почистване на паметта (т. нар. garbage collector)
* Системата за почистване на паметта анализира динамичната памет и освобождава тези обекти от нея, които не се използват от програмата
* При освобождаване на ненужните обекти им се извикват т. нар. finalizers, които освобождават използваните ресурси

2. Какви са предимствата и недостатъците на автоматичното управление на паметта?

Предимства на автоматичното управление на паметта с помощта на garbage collector:

* не трябва ръчно да се освобождава паметта като в C++
* не се получава "изтичане на памет" (memory leaks) – много неприятен проблем
* не е възможно четене и писане по освободена памет или повторно освобождаване
* ресурсите винаги се освобождават
* паметта се заделя много бързо

Недостатъци:

* някои ресурси трябва да се управляват ръчно
* няма гаранция кога се изпълнява garbage collector-ът и колко време отнема

3. Как работи т. нар. garbage collector?

Почистване на паметта (garbage collection) в .NET Framework:

* Активира се при създаване на нов обект, когато има недостиг на памет
* Изчакват се всички нишки да достигнат безопасно състояние и се приспиват
* Намират се използваните от всички нишки обекти и всички обекти, достижими от тях
* Получава се граф на използваните обекти
* Идентифицират се ненужните обекти (тези, за които е установено, че не се използват)
* Ненужните обекти, които изискват финализация (т.е. имат да освобождават ресурси) се преместват специална опашка
* Всички останали ненужни обекти се освобождават
* Динамичната памет се пренарежда, така че лявата част да е заета памет, а дясната част – свободна (премахват се получените "дупки")
* Референциите към всички преместени обекти се пренасочват към новото им местоположение в динамичната памет
* Възобновява се работата на всички нишки

4. Какво знаете за финализацията и интерфейса IDisposable в .NET Framework? Кога се използват? Как се реализират?

Финализацията:

* Някои обекти освен памет използват и други ресурси (например връзки към БД, файлови манипулатори, сокети и др.)
* Финализацията осигурява правилното освобождаване на ресурсите, използвани от обектите
* Всеки наследник на System.Object може да дефинира метод за финализация Finalize() (или деструктор в C#)
* Методът Finalize() се извиква преди даден обект да бъде унищожен от системата за почистване на паметта
* Освен динамичната памет CLR поддържа още две структури:
  + Finalization Queue – опашка с обекти, които имат метод за финализация, но все още се използват от програмата
  + Freachable Queue – опашка с обекти, които не се използват от програмата, но чакат да изпълнят метода си за финализация
* При почистване на паметта обектите, които имат метод за финализация, не се освобождават веднага
* Вместо това се прехвърлят от Finalization Queue във Freachable Queue
* Финализицията се извършва от отделна нишка за всеки обект от Freachable Queue

Как да ползваме финализация:

Като правило не трябва да се разчита на финализацията защото:

* не е ефективна – обектите, които се финализират, се унищожават с най-малко 2 преминавания на garbage collector-а
* ресурсите се заемат ненужно дълго време

Ако все пак се използва финализация, трябва да се внимава:

* по време на Finalize() да не се извикват други обекти, които изискват финализация
* редът на изпълнение на отделните финализации е неопределен
* възможни са синхронизационни проблеми – финализацията се изпълнява в отделна нишка
* финализацията не трябва да е времеотнемаща

Интерфейсът IDisposable

* Използването на Finalize() не гарантира в кой момент ще се освободят ресурсите
* Интерфейсът IDisposable позволява изрично (ръчно) освобождаване на ресурси
* Ръчното управление на ресурсите е по-ефективно
* Методът IDisposable.Dispose() трябва да освободи използваните ресурси и да извика GC.SuppressFinalize()
* Препоръчва се винаги IDisposable и Finalize() да се имплементират заедно

Реализация:

SomeResource resource = new SomeResource();

try

{// Use the resource}

finally {

if (resource != null) {

(IDisposable)resource.Dispose();

}

}

*Лекция 11*

1. Какво представляват потоците в .NET Framework? Кои са основните операции с тях? Кои са основните класове за работа с потоци?

Какво представляват:

Абстракцията "поток" е основният начин за осъществяване на входно-изходна активност в съвременните обектно-ориентирани езици (C#, C++, Java, Delphi)

Потоците:

* са подредени серии от байтове
* представляват абстрактни канали за данни, до които достъпът се осъществява последователно
* предоставят механизъм за четене и писане на поредица байтове от и към устройства за съхранение или пренос на данни

В .NET Framework повечето входно-изходни операции използват потоци

В .NET Framework потоците са два вида:

Базови потоци (base stream)

* четат и пишат данни от и към външен механизъм за съхранение на данни
* примери: FileStream, MemoryStream, NetworkStream

Преходни потоци (pass-through streams)

* четат и пишат в други потоци, като добавят допълнителна функционалност (напр. буфериране, кодиране и компресиране)
* например: BufferedStream и CryptoStream

Основните операции с потоци са:

конструиране (създаване)

* потокът се свързва с механизма за пренос/ съхранение на данни или с друг поток
* като параметър в конструктора на класа се подава информация за този механизъм
* например при файлов поток се посочва име на файл, а при низов поток – съответен низ

четене

* извличат се данни от потока
* в зависимост от типа на потока тези данни се извличат по различен начин
* например при файлов поток данните се прочитат от текущата позиция във файла

писане

* изпращат се данни в потока
* в зависимост от типа на потока тези данни се изпращат по различен начин
* например при писане във файл данните се записват във файла от текущата позиция

позициониране

* премества текущата позиция на потока (ако се поддържа позициониране)
* позиционирането става спрямо текущата позиция, началото или края на потока
* например при файлов поток се променя текущата позиция във файла

затваряне

* завършва се работата с потока и се освобождават използваните ресурси
* например при файлов поток се записват данните от вътрешните буфери, които не са все още записани на диска и се затваря файла

други операции

* изпразване на вътрешните буфери (flush)
* поддържа се и асинхронно четене и писане (което ще разгледаме в темата за работа с нишки и синхронизация)
* някои специални потоци поддържат и други специфични за тях операции

Базов клас за всички потоци е абстрактният клас System.IO.Stream

В него са дефинирани методи за извършване на основните операции

Не всички потоци поддържат операциите четене, писане и позициониране

* Кои операции се поддържат може да се провери чрез свойствата CanRead, CanWrite и CanSeek
* Потоците, които позволяват позициониране поддържат свойства Position и Length

Има специален поток Stream.Null, който игнорира всички опити за четене и писане

*Лекция 12*

1. Какво представлява езикът XML?

XML е:

* универсален език (нотация) за описание на структурирани данни
* данните се съхраняват заедно с мета-информация за тях
* прилича на HTML – текстово-базиран, използва тагове и атрибути
* с него се описват други езици (формати) за представяне на данни
* световно-утвърден стандарт, поддържан от W3C ([www.w3c.org](http://www.w3c.org/))
* независим от платформата, езиците за програмиране и операционната система

2. За какво служи?

Прилики между езиците XML и HTML:

* и двата са текстово базирани
* използват тагове и атрибути

Разлики между езиците XML и HTML:

* HTML е език, а XML е синтаксис за описание на други езици
* HTML описва форматирането на информацията, а XML описва структурирана информация
* XML изисква документите да са добре дефинирани (well-formed)

3. Кога се използва?

XML се използва:

* за обмяна на информация между различни системи
* за съхранение на структурирани данни
* за създаване на собствени езици за описание на информация

Недостатъци на XML:

* данните са по-обемисти
* намалена производителност
* често пъти е необходима много памет
* увеличаване на мрежовия трафик

4. Какво представляват пространствата от имена в XML документите? За какво служат? Кога се използват?

Пространствата от имена (namespaces) в XML документите позволяват дефиниране и използване на тагове с еднакви имена.Позволява се използването на пространства по подразбиране,

5. Какво представляват XML схемите? По какво си приличат и по какво се различават DTD, XSD и XDR схемите?

Схеми и валидация:

Схемите описват:

* допустимите тагове
* допустимите атрибути на таговете
* допустимите стойности за атрибутите и на елементите
* реда на поставяне на таговете
* стойности по подразбиране

Има няколко стандарта за XML схеми:

* DTD – Document Type Definition
* XSD – XML Schema Definition Language
* XDR – XML-Data Reduced

DTD (Document Type Definition) е:

* формален език за описaние структурата на XML документи
* съдържа съвкупност от правила за таговете в документа и техните атрибутите
* текстово-базиран език, но не е базиран на XML
* използва се рядко, защото е заместен от XSD

XSD (XML Scheme Definition Language) е:

* мощен XML-базиран език за описание структурата на XML документи
* съдържа съвкупност от правила за таговете в документа и техните атрибути

XDR (XML-Data Reduced) е:

* Език за описание на структурата на XML документи
* Компактен вариант на XML-Data схемите
* XML базиран език, подобен на XSD
* По-мощен от DTD
* По-слабо изразителен от XSD
* Въведен от Microsoft, използван често в техни продукти и технологии (напр. в SQL Server 2000, BizTalk, …)
* Може да описва съответствия между структурата на XML документи и релационни бази данни

6. Защо не можем да използваме XSLT за преобразуване на XML документ към произволен друг текстов формат, а само към друг XML?

* XSL трансформациите (XSLT) позволяват преобразуване на един XML документ в друг XML документ с различна структура
* XSLT разчита на XPath за извличане на части от входния документ, които се използват в изходния документ
* В частност XSLT може да се използва и за преобразуване на XML документи в XHTML

*Лекция 13*

1. Какви модели на базите от данни познавате?
   * йерархичен (дървовиден)
   * мрежови
   * релационен (табличен)
   * обектно-релационен
2. Кои са основните функции, изпълнявани от една система за управление на бази от данни (СУБД)?

* COUNT(…) – връща броя редове
* SUM(…) – сума на колона
* AVG(…) – средноаритметична стойност на колона
* MAX(…) – максимална стойност от колона
* MIN(…) – минимална стойност от колона

1. Дефинирайте понятието таблица в база от данни.

Таблиците представляват съвкупност от стойности, подредени в редове и колони. Пример (таблица PERSONS):

|  |  |  |
| --- | --- | --- |
| Ид | Име и Фамилия | Местност |
| 1 | Иван Иванов | Пловдив |
| 2 | Стефан Димитров | Кърджали |

Редовете имат еднаква структура

Колоните имат име и тип (число, символен низ, дата или др.)

1. Обяснете разликите между първичен и външен ключ.

Първичният ключ (primary key) е колона от таблицата, която уникално идентифицира даден неин ред (в таблицата по горе ид). Два записа (реда) са различни когато са различни първичните им ключове. Първичният ключ може да е съвкупност от няколко колони

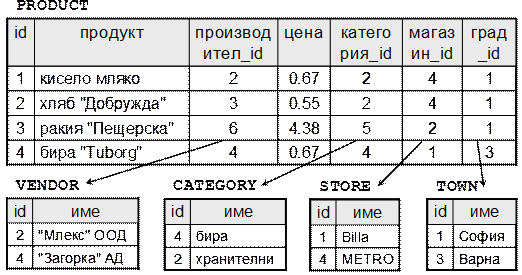
Външен ключ (foreign key constraint) -Стойността в дадена колона е ключ от друга таблица.

1. Посочете какви видове връзки между таблици познавате.

Връзките между таблиците се базират на взаимоотношения primary key / foreign key.

6. Кога дадена база от данни е нормализирана до четвърта нормална форма? Кои са предимствата на нормализираната база от данни?

Един по-завършен пример за база от данни, която е нормализирана до четвърта нормална форма е даден на следващата фигура:



Една нормализирана база от данни има сериозни предимства пред такава, в която не са спазени изброените по-горе правила. За пример нека да вземем база от данни, която моделира телефонен указател. Ако име, презиме и фамилия са представени чрез едно поле (т.е. не е спазено първото от правилата за нормализация), то търсенето в подобна база от данни по фамилия ще бъде затруднено.

7. За какво се използват ограниченията в една база от данни?

Ограниченията (constraints) задават правила, за данните, които не могат да бъдат нарушавани

Ограничение по първичен ключ (primary key constraint)

* Първичният ключ е уникален за всеки запис

Ограничение по уникален ключ (unique key constraint)

* Стойностите в дадена колона (или група колони) са уникални

Ограничение по външен ключ (foreign key constraint)

* Стойността в дадена колона е ключ от друга таблица

Ограничение по стойност (check constraint)

* Стойностите в дадена колона изпълняват дадено условие

Например:

(hour>=0) AND (hour<=24)

name = upper(name)

8. Посочете предимствата и недостатъците на използването на индекси в базите от данни.

Индексите представляват допълнителна информация за дадена таблица, която ускорява процеса на търсена на стойност в дадена колона или група колони. Най-често се реализират с B-дървета или хеш-таблици. Естест­вено реализацията е оставена на СУБД, като потребителят се възползва наготово от тях. Обикновено индексите се ползват при големи таблици, които често се претърсват за определени данни.

Предимството на индексите в процеса на търсенето е за сметка на моди­фицирането на данните. Наличието на индекс за дадена таблица забавя добавянето и изтриването на записи от нея заради нуждата от поддръжка на индекса.

9. Какво е основното предназначение на езика SQL?

SQL или Език за структурирани запитвания (Structured Query Language, SQL) е популярен [език за програмиране](http://bg.wikipedia.org/wiki/%D0%95%D0%B7%D0%B8%D0%BA_%D0%B7%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5), предназначен за създаване, модифициране, извличане и манипулиране на [данни](http://bg.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D0%BD%D0%B8) от [релационни системи за управление на бази данни](http://bg.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B8_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B8_%D0%B7%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BD%D0%B0_%D0%B1%D0%B0%D0%B7%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8). Стандартизиран е от [ANSI](http://bg.wikipedia.org/wiki/ANSI) / [ISO](http://bg.wikipedia.org/wiki/ISO).

10. За какво се използват транзакциите? Дефинирайте техните отговорности и обяснете нивата им на изолация.

Транзакциите са последователности от действия (заявки към базата данни), които се изпълняват атомарно:

* или се изпълняват всичките действия заедно (като едно цяло)
* или никое от действията не се изпълнява изобщо

Пример:

* Банково прехвърляне на пари от една сметка в друга (теглене + внасяне)
* Ако тегленето или внасянето на парите пропадне, пропада цялата операция

Отговорности:

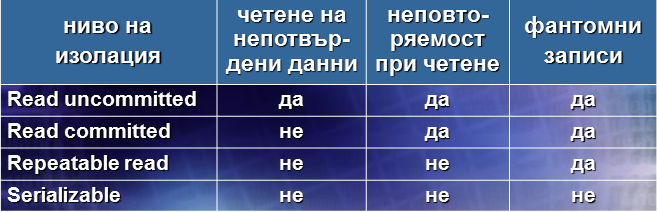
Транзакциите в повечето RDBMS системи имат 4 отговорности, заради които се наричат ACID транзакции:

* Atomicity – атомарност - Изпълнява се всичко или нищо
* Consistency – цялост на данните - Базата винаги остава консистентна
* Isolation – изолация на данните - Отделните транзакции са изолирани една от друга – не се виждат
* Durability – стабилност на данните - Ако една транзакция бъде потвърдена, тя не може да бъде изгубена

Нива на изолация:

Транзакциите могат да дефинират нива на изолация (isolation levels)

По-силната изолация осигурява по-добра консистентност, но работи по-бавно и заключва данните за по-дълго



11. Посочете основните системни компоненти на MS SQL Server.

MSSQLSERVER – самият сървър

* Основната услуга на SQL Server базата данни за инстанцията по подразбиране

SQLSERVERAGENT – SQL Server Agent

* Наблюдава SQL сървъра, извършва периодични действия и докладва за проблеми

MSSQLServerADHelper

* Служи за интеграция с Active Directory

MSSQLServerOLAPService - Служи за OLAP анализ на данни

Enterprise Manager

* Цялостна администрация на SQL сървър
* Менажиране на бази данни (създаване, backup, възстановяване, конфигуриране)
* Менажиране на обекти в базата (таблици, индекси, тригери, процедури, …)

Query Analyzer - Изпълнение и анализ на SQL заявки

DTS (Data Transformation Services) - Извличане, трансформация и импортиране на данни от и към външни източници

SQL Profiler - Проследяване на събития и оптимизация на производителността на SQL Server

SQL XML

* Публикуване на данни в Web среда (в Internet Information Services – IIS)
* Изпълнява SQL заявки и връща XML

Analysis Manager

* Компонент за Data Warehousing (OLAP)
* Извлича информация от многомерни аналитични модели

12. Избройте основните инструменти, които се използват при разработване на софтуер за SQL Server. За какво служи всеки от тях?

Програмни среди и инструменти

Visual Studio .NET 2002/2003

* SQL Servers в Server Explorer-а - Работа с таблици, изгледи, тригери, процедури, диаграми
* Създаване / изтриване / промяна и дебъгване на stored процедури
* DB проекти

Query Analyzer

* Работа със съхранени процедури
* Изпълнение на SQL заявки
* Профилиране на SQL заявки

13. Кои команди спадат към DDL? Опишете тяхното действие.

Видове команди:

Дефиниране / редактиране на обекти

* CREATE – създаване на обекти
* ALTER - Промяна / дефиниране на обекти
* DROP - Изтриване на обекти

14. Какви видове съединения на таблици познавате?

Какво е съединение?

* Комбиниране на колони на две или повече таблици и връщането им във виртуална таблица, при изпълнение на командата SELECT

Видове съединения

* Вътрешно съединение (INNER JOIN)
* Външно съединение (OUTER JOIN)
* Кръстосано съединение (CROSS JOIN)

15. Каква е употребата на агрегиращите функции в езика SQL?

Релациите често се използват съвместно с агрегатни функции, за създа­ва­не на колони с агрегатни изрази. Обикновено се агрегира инфор­ма­цията от редовете в таблицата-наследник, съответстващи на ред от роди­тел­ската таблица. Например в таблицата родител може да се създаде агре­гирана колона, която показва колко са на брой съответните записи в таблицата-наследник.

16. За какво се използват DBCC командите в SQL Server?

SQL Server предлага така наречените Database Console Commands (DBCC). Те се използват за проверка на физическата и логическата консис­тент­ност на базата от данни.

17. За какво се използват съхранените процедури? Посочете примери за използването им.

Както вече изяснихме, съхранените процедури пред­ставляват програмен код, състоящ се от последователност от SQL коман­ди, които се изпълняват в самия сървър за бази от данни. SQL Server поддържа следните типове съхранени процедури:

-     **системни** – това са съхранени процедури, които се използват за административни цели. Обикновено се извикват от инструменти като Enterprise Manager, но може да се използват и директно. Системните съхранени процедури започват с префикса sp\_.

-     **разширени** – представляват подпрограми, съдържащи се в дина­мични библиотеки (.dll файлове) и написани на езици като C и C++, които се зареждат и изпълняват като обикновени съхранени процедури. Обикновено започват с префикса xp\_.

-     **потребителски** – това са съхранени процедури, които са създадени от разработчиците или администраторите на един сървър. те се използват много често в практиката и затова трябва да им обърнем повече внимание.

Пример:

Силно се препоръчва да не използвате префикса sp\_ в името на създаваните от вас съхранени процедури. Ако SQL Server срещне такъв префикс в името на процедура, при извикването и той започва да претърсва за нея в някой специални бази от данни.

18. Кои команди се използват за управлението на транзакциите в T-SQL?

* Data Definition Language (DDL) - Дефиниция и управление на обектите в базата от данни (таблици, изгледи, тригери, ...)
* Data Manipulation Language (DML) - Работа с данните и обектите, съхранени в базата от данни (извличане, промяна, ...)
* DataBase Console Commands (DBCC) - Системни съхранени процедури

20. Какво е поведението на вложените транзакции?

От концептуална гледна точка за всички сървъри за бази от данни транзакциите се използват за успешното изпъл­нение или отказване на поредица от SQL заявки. Затова задължително тази поредица от заявки се изпъл­нява в try-catch блок, като последният ред в try частта трябва да съдържа код за потвърждаване на транзакцията, а в catch частта трябва да има код за отказването й при евентуално настъпва­не на изключение.

21. При какви случаи е възможно възникването на ситуацията "мъртва хватка"?

"мъртва хватка" (dead lock) може да настъпи пти ситуация, в която една транзакция изчаква освобождаването на запис, заключен от друга транзакция, а тя съответно изчаква първата да освободи някой друг запис. За избягването на такива ситуации трябва внимателно да се ана­лизират сценариите за паралелно изпълнение на транзакциите и да се проектира рабо­тата с общите записи така, че да не настъпват конфликтни заключвания.

22. Какви са начините за пренасяне на база от данни на друг компютър? Избройте предимствата и недостатъците им.

Начини за пренасяне на SQL Server база от данни:

* Чрез архивиране и възстановяване
* Чрез откачане и закачане
* метод за пренасяне на база данни чрез откачане и закачане работи единствено ако версиите на сървъра-източник на данните и сървъра-приемник са едни и същи.

23. Обяснете какво представляват свързаният и несвързаният модел за достъп до данни. Опишете в кои случаи се използва единият, и в кои – другият. Дайте примери. Опишете предимствата и недостатъците на двата модела.

Свързан модел (connected model) -Свързаният модел за работа с данни изисква постоянна връзка с базата от данни (online). При реализацията на този модел, която предлага ADO.NET, данните могат да бъдат четени само напред – връщане назад не е възможно. Промени не могат да се правят докато данните се четат – за тази цел трябва да се изпълняват отделни заявки към базата. Затова обикновено свързаният достъп се използва когато е необходимо да се прочетат данни, или да се направи единична промяна, но не се изисква сложна обработка на много информация от базата.

Предимства

* Свързаният модел работи директно със SQL заявки и е по-близък до релационните бази от данни. Това прави производителността му много добра.
* При него се изискват по-малко усилия от страна на разработчика.
* По-лесно се контролира конкурентният достъп на много потребители до данните в базата.
* Има специални механизми, чрез които може да се гарантира целостността на данните.
* Връзката с базата от данни е отворена през цялото време на работа и се предполага, че данните ще се извличат на по-малки части.

Недостатъци

* Един от основните недостатъци на свързания модел е необходимостта от постоянна мрежова връзка с източника на данните.
* При много организации връз­ката между отдалечени офиси и централния сървър не е надеждна и свързаността не е постоянна, поради което се налага offline работа.
* Друг недостатък е заемането на ресурси (отворени връзки към базата) за продължително време.

Несвързан модел (disconnected model) - При несвързания модел работата с данните се осъществява offline – данните се изтеглят от базата и се съхраняват на локалната машина. Първоначално се осъществява връзка към източ­ника на данни, за да се извлече необходимото подмножество от данни. То се съхранява в локал­ната система и връзката се затваря. След това върху тях се извършва необходимата обработка. През това време няма връзка с базата (затова моделът се нарича несвързан). След извършване на нужните операции с данните връзката може да се отвори отново за да се въведат направените промени и/или да се извлекат още данни.

Примери за използване на несвързан модел

Несвързаният модел може да се приложи при достъп до данни чрез уеб услуга. В този случай услугата извлича данните от източника и ги предава като несвързан обект към клиента. Клиентът ги обработва и след известно време изпраща до услугата направените от него промени, а услугата ги нанася в базата данни.

Предимства

* Основно предимство на несвързания модел е, че клиентът се свързва с основната база от данни само когато има нужда.
* Това намалява натоварването на сървъра на базата от данни и израз­ходва по-малко ресурси.
* Често се случва приложението да се нуждае само да чете и визуализира данните.
* При намаляване на броя отворени връзки към базата се подобрява значи­телно скалируемостта на приложението.
* Увеличаването на броя потреби­тели не води до много по-голямо натоварване на базата от данни, защото ресурсите за обслужване на всеки потребител са необходими само докато трая неговата заявка и след това се освобождават.
* Друго предимство на несвързания модел е възможността приложението да работи в offline режим, без да има постоянна физическа свързаност със сървъра на базата данни.

Недостатъци

* Недостатък на несвързания модел на достъп е, че данните при клиента не винаги са текущи.
* Така има опасност да се работи с остарели данни.
* Друг недостатък е, че трябва да се положат допълнителни усилия от разработчика за разрешаване на конфликтите между различните версии на данните.
* Подобен случай възниква, ако след запазване на данните на локалната машина потребителят промени някаква стойност в тях, а съще­временно друг потребител промени същата стойност в основната база. Тогава, когато потребителят се опита да обнови данните в базата, нас­тъпва конфликт.

24. Опишете еднослойните, двуслойните, трислойните и многослойните приложения – какво представляват съответните модели, техните предимства и недостатъци, случаи на използване. Дайте примери.

Еднослойни приложения - Най-често работи само един потребител

Предимства

* Всички компоненти са на едно място

Недостатъци

* Промяна на функционалността изисква преинсталация

Пример

* Приложение базирано на MS Access

Двуслойни приложения (клиент-сървър)- Потребителският интерфейс и бизнес правилата се дефинират на едно място. Данните се съхраняват във втория слой.

Предимства

* Има разделяне на функционалността

Недостатъци

* Лоша скалируемост – проблеми с поддръжката на голям брой клиенти

Примери

* MS SQL Server ↔ MS Query Analyzer
* MS Exchange ↔ MS Outlook

Трислойни приложения - Различните типове функционалност са в различни слоеве

Предимства

* Отделяне на функционалността между потребителски интерфейс, бизнес правила и съхранение / достъп до данните

Недостатъци

* По-трудна поддръжка
* Повече усилия за осигуряване на сигурността

Пример

* ASP.NET Web-приложение ↔ ASP.NET Web услуга ↔ MS SQL Server

Многослойни приложения - Системи с повече от 3 логически слоя. Възможност за добавяне на още слоеве за разширяване на функционалността.

Предимства

* Възможност за различни приложения да достъпват части от функционалността през отворени протоколи

Недостатъци

* Много труден процес по дефиниране и реализация на правила за сигурен достъп
* Изискват повече планиране и по-големи срокове за разработка

25. Обяснете накратко какво е ADO.NET и кои са неговите основни пространства от имена. За какво служат те?

ADO.NET е ориентирано към два основни сценария на обработка, които отразяват двата модела за достъп до данни (свързан и несвързан) и съответно предоставя различни класове за тяхната реализация (основно DataSet и DataReader). ADO (ActiveX Data Objects) е технология в Microsoft Windows, която предо­ставя единен стандарт за достъп до релационни бази от данни от Windows приложения посредством ActiveX обекти.

26. Какво представляват доставчиците на данни и кои са стандартните доставчици на данни в ADO.NET?

Доставчиците на данни (Data Providers) са съвкупности от класове, които осъществяват връзка с различни бази от данни. Те осигуряват възможност да се изпъл­няват команди и да се получават резултатите по начин, който е независим от източника на данни и неговата специфична функционал­ност. Те създа­ват тънък слой между ADO.NET приложението и източника на данните (базата данни). Доставчици­те на данни са проектирани да оси­гуряват ефективен достъп за промяна на данните или само за извличане.

За различните RDBMS се използват различни доставчици (Data Providers), като всеки е оптимизиран за работа със съответната база от данни. Това се налага, тъй като различните производители използват различни прото­коли за достъп до сървъра на базата. Различните доставчици на данни осигуряват сходна базова съвкупност от възможности, но въпреки това е възможно между тях да има разлика във функционалността. Това се дължи на разликите между различните източници, както и на разлики в имплементацията.

Всеки доставчик на данни (Data Provider) съдържа 4 основни класа, с помощта на които се осъществява достъпа до съответната база от данни. Те осигуряват връзка към базата (Connection), изпълнение на команди (Command), поточно извличане на данни (DataReader) и адаптери за работа при несвързан модел (DataAdapter). Допълнително в доставчиците на данни са реализи­рани и класове за работа с параметрични заявки (Parameter), за работа с транзакции (Transaction) и др.

27. Опишете класовете от SqlClient Data Provider.

-     SqlConnection –осъществява връзката с MS SQL Server. Предоставя методи за отваряне и затваряне на връзка, за започване на тран­закции и др. При използването на този клас програмистът не се инте­ресува от детайлите за това как точно се осъществява физи­ческата връзка с базата.

-     SqlCommand – изпълнява команди върху MS SQL Server през вече установена връзка. Той обвива текст на SQL заявка или извикване на съхранена процедура.

-     SqlDataReader – този клас служи за извличане на данните от SQL сървъра. Обект от него се създава имплицитно в резултат от изпъл­нение на команда.

-     SqlTransaction – използва се за работа с транзакции. Има методи Commit() и Rollback() за потвърждаване и отмяна на транзакции, съответно.

-     SqlDataAdapter – служи за "мост" между DataSet обект и SQL Server – през него се запълва DataSet и се обновява базата. Той сам управлява връзката с базата от данни – сам я отваря и затваря, като програмистът може да не се грижи за това.

28. Опишете начините за автентикация пред MS SQL Server. Дайте пример за символен низ за връзка с SQL Server (connection string)

Видове автентикация в SQL Server 2000- SQL Server предлага два механизма за автентикация на потребителите – Windows автентикация и смесена.

Windows автентикация- Windows автентикацията се базира на модела за сигурност в Windows NT 4.0 и Windows 2000 и се използва по подразбиране от SQL Server. При нея се разчита на автентикация на потребителя от операционната система. В този случай право на достъп се дава на Windows потребители и групи (или потребители от Microsoft Active Directory при работа в домейн). Когато се използва този модел на сигурност, администраторът на базата от данни дава достъп на потребителите до компютъра, на който работи базата и им дава права за влизане в SQL Server 2000.

Смесена автентикация- При този метод потребителите могат да бъдат автентикирани от Windows или от SQL Server 2000. Ако потребителят не може да използва стандар­тна Windows автентикация, автентикацията се извършва от SQL Server 2000, като за тази цел сървърът пази двойките име и парола на потре­бителите. Този модел се използва предимно за съвместимост с по-стари версии и в бъдещи версии на SQL Server може да спре да го поддържа. Има малка разлика в скоростта при двата модела на автен­тикация, като Windows автентикацията е по-бърза. Смесената автентикация е подходя­ща за хетерогенна среда, където не е наличен механизмът на Windows за автентикация.

Символен низ за връзка към база от данни (Connection String)

При създаване на връзка към база от данни е нужно да се зададат някои важни параметри, като например методът на автентикация, адресът на сървъра, името на базата и др. За целта се използва свойството ConnectionString на класа на връзката. То съдържа двойки име/стойност, които се разделят с точка и запетая (;). Редът им няма значение, както и малките и големи букви.

29. Опишете механизма на connection pooling. Защо е необходимо да се използва?

Механизмът на "connection pooling" поддържа "пул" от налични връзки към базата. Когато клиент се опита да отвори връзка се използва готова връзка от пула (разбира се, ако там има свободна), вместо да се създаде нова. Ако всички връзки от пула са заети се отваря нова. Ако пулът не съществува, той се създава при първото извикване на метода Open() на обект за връзка. При затваряне на връзка тя реално не се затваря, а се връща в пула, готова за повторна употреба.

31. Обяснете кои са основните класове и интерфейси за работа със свързан модел.

Свързаният модел­ се имплементира  чрез три основни класа:

-     клас за отваряне на връзка (SqlConnection)

-     клас за изпълнение на команда / команди (SqlCommand)

-     клас за обработка на редовете, получени като резултат от изпъл­нена заявка, който наричаме четец (SqlDataReader)

32. Опишете класовете, които се използват за реализация на несвързания модел в ADO.NET. За какво служи всеки от тях?

За реализация на несвързания достъп до данни се използват набор от класове, които са дефинирани в пространството от имена System.Data

-     DataSet – основен клас за представяне на данните в паметта.

-     DataTable – клас, представящ таблица в паметта.

-     XxxDataAdapter –осъществява достъпа до данните като изпол­зва XxxCommand и XxxConnection класове.

-     DataRelation – представя връзка между таблици, съхранявани в DataSet обект.

33. Каква е разликата между силно-типизиран и нетипизиран DataSet? Как се създават двата вида DataSet?

Силно типизираните DataSets наследяват класа DataSet и използват информацията от XML Schema файл, за да генерират нов клас. В този наследен клас обектите в колекциите, представящи таблиците, колоните, редовете и т.н. наследяват съответно от DataTable, DataColumn, DataRow и т.н., като се добавят специфични методи, свойства и събития съобразно използваната схема.

UsersDataSet dsUsers = ...;

string username = dsUsers.Users[0].username;

Ето типичен пример за достъп до нетипизиран DataSet обект:

|  |
| --- |
| DataSet dsUsers = ...;  string username = (string) dsUsers.    Tables["Users"].Rows[0]["username"]; |

34. Кои класове се използват при работа с DataSet? - основен клас за представяне на данните в паметта. Той може да се разглежда като абстракция на релационна база от данни, състояща се от таблици с връзки между тях. По същество DataSet е контейнерен клас, който съдържа таблици (DataTableCollection), релации (DataRelationCollection), ограничения (които са част от DataTableCollection) и някои други класове.

35. За какво служи класът DataTable? - клас, представящ таблица в паметта. Един DataSet може да съдържа множество такива обекти, като чрез тях представя цяла база от данни в паметта. DataTable съдържа обекти за колони (DataColumn), редове (DataRow) и ограничения (Constraints).

36. Как се добавят редове и колони в DataTable?

Силно типизираните DataSets наследяват класа DataSet и използват информацията от XML Schema файл, за да генерират нов клас. В този наследен клас обектите в колекциите, представящи таблиците, колоните, редовете и т.н. наследяват съответно от DataTable, DataColumn, DataRow и т.н., като се добавят специфични методи, свойства и събития съобразно използваната схема.

37. За какво служи класът DataRelation?

DataRelation класът се използва за създаване на връзки между таблици предимно с навигационна цел. Те се използват често при работа с таблици в отношение родител-наследник (master-detail таблици). Чрез тях лесно се извлича списъкът на редове от подчинената таблица (таблицата-нас­лед­ник), свързани с ред от главната таблица (таблицата-родител).

38. Какви са основните приложения на DataView?

DataView се използва за изпълнение на две основни операции върху данните в една таблица:

-     филтриране на редове;

-     сортиране на редовете.

39. Как се добавят ограничения в DataSet?

Схемата на източника на данни може да се извлече чрез метода FillSchema(…) на класа DataAdapter. Той извлича информация за схема­та, свързана с резултата от заявката в SelectCommand свойството на адап­тера. Методът създава таблица в DataSet обекта със съответните колони и конфигурира свойствата им според източника. Когато конфигурира AutoIncrement свойството, не задава стойности за AutoIncrementSeed и AutoIncrementStep. FillSchema(…) конфигурира първичния ключ и огра­ниченията по уникалност за таблиците в DataSet обекта.

40. За какво служат потребителските изрази в колоните на таблиците?

ADO.NET позволява създаване на колони, базирани на изрази. Тези изра­зи могат да се използват за извършване на пресмятания върху стойности на колони в един ред или изчисляване на агрегатни функции върху множество редове.

41. Опишете средствата за зареждане на DataSet от XML документ и записване на съдържанието на DataSet в XML документ.

Съдържанието на един DataSet обект може да бъде взето от XML поток или документ. При това .NET Framework дава голяма гъвкавост относно това каква информация да се зареди от XML източника, както и как точно да се създаде схемата на DataSet обекта. За това се използва методът ReadXml(…) на класа DataSet. Той е предефиниран за различните източ­ници на XML данните. Може да чете от следните източници:

-     Отворен за четене поток. В този случай методът приема като параме­тър обект от тип Stream или TextReader.

-     Символен низ. Този тип параметър означава път до файл или URL адрес. Ако файлът не съществува или потребителят няма право на достъп до него, методът предизвиква изключение.

-     XML четец. В този случай методът приема като параметър обект от тип XmlReader.

Ето кратък пример за зареждане на DataSet обект от XML файл:

|  |
| --- |
| DataSet dsOrders = new DataSet();  dsOrders.ReadXml("orders.xml"); |

42. Опишете предназначението на класа XmlDataDocument.

DataSet класът предоставя релационно представяне на данните, докато средствата за работа с XML работят с йерархични данни. .NET Framework дава възможност едновременно да се работи и с двете представяния на едни и същи данни. Това се постига чрез синхронизация на DataSet и XmlDataDocument, който осигурява йерархичния изглед на данните.

43. Опишете някои основни съображения относно сигурността при работа с базите от данни.

Сигурността при базите от данни има различни аспекти. В настоящата точка ще се спрем само на най-важните от тях, тъй като непознаването им може да доведе до сериозни проблеми в разработваните приложения.

Сигурност при динамични SQL заявки - Когато използваме SQL заявки, които трябва да включват стойност, въве­дена от потребителя, **в никакъв случай не трябва да сглобяваме заявката чрез долепване (конкатенация) на низове**.

Connection pooling и сигурност - Използването на connection pooling не позволява сигурността да се инте­грира в базата.

Съхраняване на connection string - Съхранението на низа за връзка към базата е важно за сигурността. Той често съдържа информация за автентикация, с която някой може да злоу­потреби. Низът за връзка трябва да може да се ползва от всички потре­бители на системата, но не трябва да е достъпен за тях, т.е. те не трябва да могат да го извличат в явен вид.

Защитна стена - Важно е, когато инсталираме сървъра за базата данни при клиента, да го защитим от нежелан външен достъп (от Интернет и от локалната мрежа). Това може да стане чрез най-обикновена защитна стена (firewall), която филтрира портовете, на които сървърът очаква своите клиенти.

Криптиране на комуникацията - Комуникацията между клиента и сървъра на базата данни обикновено не е криптирана и трафикът може да бъде подслушан. За повишаване на сигурността при обмяната на данни между клиента и сървъра може да се използва протоколът IPSec, който криптира целия мрежов трафик или да се използва VPN връзка, която също дава възможност за криптиране.

*Лекция 14*

1. Какво представлява библиотеката Windows Forms? Каква функционалност предоставя? Кога се използва?

Windows Forms

Библиотека за изграждане на прозоречно-ориентиран графичен потребителски интерфейс (GUI)

Поддържа концепцията за Rapid Application Development (RAD)

* компонентно-ориентирана архитектура
* управление, базирано на събития

Прилича на GUI средствата на Delphi и Visual Basic 6, с мощта на MFC

Съдържа богат набор от контроли

Data-aware компоненти

Вградена поддръжка на Unicode

Позволява наследяване и разширяване на форми и контроли

Поддържа ActiveX контроли

Поддържа печатане на принтер

Контролите могат да се изпълняват в Internet Explorer

* подобно на Java аплетите
* без предварителна регистрация (за разлика от ActiveX контролите)

Силна поддръжка на графика (GDI+)

Windows Forms е стандартната библиотека на .NET Framework за изграж-дане на прозоречно-базиран графичен потребителски интерфейс (GUI) за настолни (desktop) приложения.

2. Какво е компонент? Какво представлява компонентният модел .NET Framework? Какво е характерно за него?

Компоненти

Програмни единици (класове), които решават специфична задача

Преизползваеми (reusable)

Имат ясно дефиниран интерфейс, който описва техните:

* свойства
* методи
* събития

Компонентният модел на .NET Framework

* налага правила за създаване и използване на .NET компоненти (програмен модел)
* дефинира класове и интерфейси, които поддържат описанието на компоненти

Компонентният модел на .NET

позволява дефиниране на поведението на компонентите

* по време на дизайн (design-time behavior)
* по време на работа (runtime behavior)

осигурява лесна преизползваемост (reusability)

дефинира компоненти и контейнери

CLR осигурява междуезикова съвместимост на компонентите

Основната функционалност на компонентния модел се намира в System.ComponentModel

3. Опишете програмния модел на Windows Forms. Каква функционалност реализира той?

Програмният модел на Windows Forms дефинира:

* форми (прозорци и диалози)
* контроли (текстови полета, бутони, менюта, ленти с инструменти, ...)
* събития, които ги управляват
* жизнен цикъл на приложенията
* модел на пречертаване на контролите
* управление на фокуса и навигация

Жизненият цикъл на приложенията е базиран на съобщения

* Контролите получават съобщения за потребителските действия и реагират по специфичен начин

Главната нишка на всяко Windows Forms приложение работи така:

Постоянно слуша за съобщения

При получаване на съобщение (напр. преместване на мишката, натискане на клавиш или др.) го обработва така:

* намира контролата, за която се отнася съобщението
* предава й съобщението
* ако контролата е контейнер-контрола, тя търси в себе си за коя от нейните контроли е съобщението и й го предава

При затваряне на главната форма на приложението, то спира изпълнението си

4. Кои са най-важните класове от Windows Forms? Кои са най-важните им методи и свойства?

Библиотеката Windows Forms дефинира:

* съвкупност от базови класове за контролите и контейнер-контролите
* множество графични контроли

Основни базови класове:

* Component – .NET компонент
* Control – графична контрола (компонента с графичен образ)

Класът System.Windows.Forms.Control е основа на всички графични Windows Forms контроли

Неговите свойства са типични за всички Windows Forms контроли

По-важните свойства на класа Control:

* + Anchor, Dock – задават по какъв начин контролата се "закотвя" за контейнера си
  + Bounds – задава размера и позицията на контролата в нейния контейнер
  + BackColor – задава цвета на фона
  + ContextMenu – задава контекстно меню (popup menu) за контролата

По-важни методи на класа Control:

* + Focus() – фокусира контролата
  + Hide(), Show() – скрива/показва контролата

- ScrollableControl – контрола, която поддържа скролиране на съдържанието си. представлява контрола, която поддържа скролиране на съдържанието си. Може да съдържа в себе си други контроли.

- ContainerControl – контрола, която съдържа други контроли и управлява поведението на фокуса

5. Какво е характерно за формите в Windows Forms?

При настолните приложения графичният потребителски интерфейс позво-лява потребителят директно да взаимодейства с програмата чрез мишката и клавиатурата, а програмата прихваща неговите действия и ги обработва по подходящ начин.

В .NET Framework и особено в Windows Forms се поддържа концепцията за Rapid Application Development (RAD).

6. Как се поставят контроли в дадена форма? Как се прихващат събития, породени от даден контрол?

Поставянето на контроли във формата става чрез Controls.Add:

Form form = new Form();

Button button = new Button();

button.Text = "Close";

form.Controls.Add(button);

Редът на контролите (т. нар. z-order) се определя от реда на поставянето им (последната контрола е най-отгоре) Windows Forms дизайнерът на Visual Studio .NET се грижи за правилното поставяне на контролите

Прихващането на събития става така:

Form form = new Form();

Button button = new Button();

button.Click += new EventHandler(

this.button\_Click);

...

private void button\_Click(

object sender, EventArgs e)

{

*// Handle the "click" event*

}

Windows Forms дизайнерът на Visual Studio .NET генерира автоматично обработчиците на събития

*Лекция 17*

1. Опишете 7-те слоя от OSI мрежовия модел.

1. Физически слой
2. Канален слой
3. Мрежов слой
4. Транспортен слой
5. Сесиен слой
6. Представителен слой
7. Приложен слой

2. Опишете какво представляват понятията: IP адрес, DNS, порт, мрежов интерфейс, TCP, UDP и сокет връзка. Каква е разликата между TCP и UDP протоколите?

IP адрес

* Уникален 32-битов (64-битов) номер на компютър
* Всеки компютър в Интернет и локална TCP/IP мрежа си има IP адрес
* Записва се като 4 последователни 8-битови числа разделени с точки
* Някои компютри могат да имат по няколко IP адреса

Примери:

- 192.168.0.1

- 212.50.1.217

DNS (Domain Name Service)

* Услуга, която преобразува имена на компютри в IP адреси и обратното

Пример: На името www.google.com съответстват IP адресите:

* 216.239.59.104
* 216.239.59.99

Пример: На IP адрес 194.12.244.90 съответстват имената: zadnik.org, www.zemlemoria.org

Порт

* Уникален идентификатор на TCP или UDP връзка (канал за данни) в рамките на IP адрес
* Число между 0 и 65535
* Портовете от 0 до 1024 са заделени за специални (системни) приложения

Мрежов интерфейс

Възможно е една машина да е свързана към няколко мрежи

* локалната мрежа в сградата
* мрежата на доставчика на Интернет

Свързването към мрежа става посредством мрежов интерфейс

* Ethernet мрежов адаптер
* модем
* VPN връзка
* ISDN

Мрежовият интерфейс има IP адрес - Една машина може да има няколко мрежови интерфейса с няколко различни IP адреса

Loopback интерфейс

* Представлява служебен интерфейс за обратна връзка към локалната машина
* Има адрес 127.0.0.1 и име localhost

TCP

осигурява надежден двупосочен комуникационен канал (сокет)

Гарантира, че изпратените данни ще пристигнат без изменения и в същия ред

Ако не е възможно да се изпратят данните, ще възникне грешка

Използва се в приложения, в които редът на пристигане на данните и надеждността са важни

Комуникационният канал съществува докато някой от участниците в него не го прекъсне

UDP

Позволява изпращане и приемане на малки независими един от друг пакети с данни, наречени "datagram пакети"

UDP не гарантира реда на пристигане на datagram пакетите, нито че изобщо ще пристигнат

По-бърз от TCP, защото не създава комуникационен канал

Използва се в случаи, в които скоростта има по-голямо значение от надеждността

при някои игри, при приложения за разговори (chat), при видео и аудио разпространяването

Socket връзка

При TCP протокола всяка TCP връзка (сокет връзка, socket connection) се идентифицира с 4 елемента:

* Source\_IP – IP адрес на клиента
* Source\_Port – порт на клиента
* Dest\_IP – IP адрес на сървъра
* Dest\_Port – IP адрес на клиента

Пример: (192.168.0.2:2733 🡪 212.50.1.217:80)

При UDP протокола всеки UDP пакет се идентифицира със същите 4 елемента – (IP:port)-изпращач и (IP:port)-получател

3. Опишете основните мрежови услуги в Интернет, какви протоколи използват и кои TCP портове.

Услугите са сървърни приложения, които предоставят функционалност по стандартизиран протокол

* В Интернет работят много стандартни услуги
* Стандартните услуги си имат стандартен порт

|  |  |  |
| --- | --- | --- |
| Услуга | Порт | Описание |
| HTTP | 80 | Достъп до Web-ресурси |
| SMTP | 25 | Изпращане на e-mail |
| POP3 | 110 | Извличане на e-mail |
| FTP | 21 | Достъп до отдалечени файлове |
| DNS | 53 | Извличане на IP по име на сървър |
| SSH | 22 | Сигурен достъп до отдалечен терминал |

*Лекция 20*

1. Какви модели за разпределени приложения познавате? Каква е разликата между тях?

Повечето днешни приложения са разпределени

* Състоят се от няколко раздалечени компонента, които си взаимодействат

Модели за разпределени приложения

Модел "клиент/сървър"

Модел "разпределени обекти"

* DCOM – ползва се в Microsoft Windows
* CORBA – отворен стандарт, доста сложен
* Java RMI – базира се на Java технологията
* .NET Remoting – ползва се в .NET Framework

Модел "Web услуги"

2. Какво представляват Web услугите? Какъв проблем решава тази технология?

Web-услугите моделират услугите от реалния свят

* Програмни компоненти, достъпни отдалечено през Web
* Модел за изпълнение “заявка-отговор” - Клиентът поръчва, услугата изпълнява поръчката и връща резултат
* Използват отворени стандарти за комуникация – HTTP, XML и SOAP
* Сами описват интерфейса си за достъп чрез езика WSDL
* Могат да бъдаттърсени чрез UDDI

Моделът "разпределени обекти не е подходящ за употреба в Интернет

* Силна свързаност между услуга и клиент
* Трудности при хетерогенна инфраструктура
* Проблеми с версиите
* Защитните стени (firewalls) силно затрудняват комуникацията
* COM/DCOM се поддържа само в Windows
* CORBA не работи добре през Интернет
* Java RMI / EJB се поддържат само в Java

Решението: Web услуги

3. Какво представлява инфраструктурата на Web услугите? От какво се състои?

Инфраструктурата на Web услугите се състои от следните компоненти

* Директория (Directory) - UDDI
* Откриване (Discovery) - DISCO
* Описание (Description) - WSDL
* Формат на заявките (Wire Format) – SOAP ; XML, XSD, HTTP

4. Какво представляват UDDI директориите и за какво служат?

Universal Description, Discovery and Integration (UDDI) е платформено независима, Extensible Markup Language (XML), базирана регистър, чрез който бизнеса по целия свят могат да се изброяват в интернет, както и механизъм, който да се регистрират и да се намери приложения за интернет услуги , UDDI е отворена инициатива индустрия, финансиран от Организацията за развитие на стандартите за структурирана информация (OASIS), които позволяват на предприятията да публикуват обяви за услуги и да открият един друг, както и да определят как услугите или софтуерни приложения взаимодействат по интернет

Позволяват търсене на Web услуги

Организирани са по категории

Може да се търси по различни параметри

Използват UDDI стандарта

* UDDI (Universal Description, Discovery and Integration)
* Отворен стандарт на OASIS
* Представлява Web услуга, предлагаща регистрация и търсене на други Web услуги

Пример за UDDI директория: http://uddi.microsoft.com

5. Какво е DISCO и за какво служи?

DISCO (Discovery of Web Services) - XML базиран език, Описва Web услугата – връзка към WSDL файла, връзка към самата услуга и връзка към документация за услугата, Обикновено disco файлът стои в главната директория на услугата - Напр. http://www.mysite.com/math/math.disco

6. Какво е WSDL и за какво се използва?

WSDL (Web Services Description Language)

Описва интерфейса за достъп до дадена Web услуга

* имената на поддържаните методи
* входни и изходни параметри
* използвани типове данни

XML базиран отворен стандарт на W3C- ASP.NET Web услугите връщат своя WSDL при извикване с параметър ?wsdl http://localhost/MyService.asmx?wsdl

7. Какво е SOAP? От какво се състои? За какво се използва?

SOAP (Simple Object Access Protocol)

* Дефинира XML базиран формат за обмяна на съобщения
* Отворен стандарт на W3C

Едно SOAP съобщение се състои от:

* SOAP хедър – описва параметри на съобщението (метаданни)
* SOAP тяло – съдържа самото съобщение (данни – изпратената или върнатата заявка)

Обикновено SOAP съобщенията се предават по протокол HTTP - Това позволява да преминават през firewall

*Лекция 21*

1. Обяснете основните концепции на .NET Remoting инфраструктурата – канали, форматери, видове активация, видове маршализация и жизнен цикъл на обектите.

Remoting технологията в .NET Framework осигурява прозрачен достъп до отдалечени обекти

* Обекти от друг application domain
* Обекти от друг процес
* Обекти на отдалечена машина

Използва се за улесняване на комуникацията при разпределени приложения

.NET Framework предоставя специализирана инфраструктура за Remoting технологията

.NET Remoting инфраструктурата се състои от:

* Канали – пренасят съобщения от и към отдалечени обекти
* Форматери – кодират и декодират съобщенията в някакъв формат
* Прокси класове – предават извикванията на методи към отдалечените обекти
* Механизми за активация – осигуряват отдалечено инстанциране на обекти
* Маршализация – осигурява пренос на обекти, техните свойства, полета и т.н.

Каналите (channels) в .NET Remoting

Транспортират съобщения от и към отдалечени обекти. Могат да бъдат:

* TCP – използват чист TCP сокет
* HTTP – използват протокола HTTP
* Други – дефинирани от потребителя

Използват се както от клиентските, така и от сървърните приложения

Използват TCP портове

Трябва да бъдат регистрирани преди използване

Форматерите (formatters) форматират съобщенията, които се предават по Remoting каналите

Форматерите биват:

* Бинарен форматер: (Голяма производителност, Слаба съвместимост с други технологии (специфичен за .NET Remoting)
* SOAP форматер (XML базиран формат – добра съвместимост с други технологии и По-лоша производителност)
* Форматери по подразбиране:
  + HTTP каналите използват по подразбиране SOAP форматер
  + TCP каналите използват по подразбиране бинарен форматер

"Активация" означава създаване на отдалечен обект (инстанциране) и подготвяне на обекта за използване. Активацията бива два вида:

* + Активация от страна на сървъра (server-side activation)
  + Активация от страна на клиента (client-side activation)

Начини за активация:

* + Чрез оператора new
  + Чрез Activator.GetObject(…)
  + Чрез Activator.CreateInstance(…)

При активация на отдалечен обект Remoting инфраструктурата динамично създава прокси клас при клиента

Прокси класът изглежда като истинския клас (или интерфейс), но препраща обръщенията към отдалечения обект:

* + достъп до полета, свойства, събития и други членове
  + извикване на методи
  + възникналите изключения

При активация от сървъра (server-side activation)

* + Сървърът създава автоматично обекта при клиентска заявка

Има два режима на сървърна активация:

* + Single-Call – за всяка клиентска заявка се създава отделна инстанция
  + Singleton – една и съща инстанция обслужва всички клиенти

При активация от клиента (client-side activation)

* + Обектите се създават по заявка от клиента (ръчно, изрично)
  + Всяка заявка за създаване на обект създава нова, отделна инстанция
  + Всеки клиент работи със свои собствени обекти на сървъра
  + Регистриране на client-activated обект:

Маршализацията осигурява пренос на обекти между Remoting клиента и сървъра. Използва при:

* + предаване на параметри
  + връщане на стойност
  + достъп до свойства и полета

Има 3 типа обекти:

* + Marshal-by-Value обекти – пренасят се по стойност (сериализирани)
  + Marshal-by-Reference обекти – пренасят се по отдалечена референция (ObjRef)
  + Not-Marshaled обекти – не се пренасят

В .NET Remoting се използва техниката "lease-based lifetime" ("живот, отпуснат назаем")

* + При създаване на обект му се дава "време на живот"
  + След изтичане на това време обектът се унищожава от Garbage Collector-а
  + Времето за живот се настройва при създаване на обекта и не може да се променя след това.