Учебный курс

Технологии и средства разработки корпоративных систем

Лекция 4

Особенности платформы .NET для разработки корпоративных приложений

Лекции читает

кандидат технических наук, доцент

Зыков Сергей Викторович

Содержание лекции

Что такое .NET ?

.NET включает следующие основные аспекты :

1. Идеология проектирования и реализации программного обеспечения
2. Модель эффективной поддержки жизненного цикла прикладных систем
3. Унифицированная, интегрированная технологическая платформа
4. Современный, удобный в использовании, безопасный инструментарий для создания, размещения и поддержки программного обеспечения

.NET как идеология (vision)

1. Легкость развертывания приложений в глобальной среде Интернет
2. Экономичная разработка программного обеспечения
3. «Бесшовная», гибкая интеграция программных продуктов и аппаратных ресурсов
4. Предоставление программного обеспечения как сервиса
5. Новый уровень безопасности и удобства использования

.NET как вычислительная модель

1. Компонентный подход как развитие объектно-ориентированной модели
2. Универсальная система типизации: «всякая сущность есть объект»; унификация данных и метаданных
3. Строго иерархическая организация кода, пространств имен и классов
4. Универсальный интерфейс .NET Framework (включая поддержку различных подходов к программированию)
5. Высокая вариативность экземпляров реализации (в частности, на основе веб-сервисов)

.NET как технологическая платформа

1. Многоязыковая поддержка (десятки языков программирования)
2. Использование технологии веб-сервисов для обеспечения интероперабельности и масштабируемости в глобальной сетевой среде
3. Унификация доступа к библиотекам API-интерфейса независимо от языка и программной модели
4. Соответствие современным технологическим стандартам

.NET - универсальное инструментальное средство

1. Поддержка многоязыковой среды CLR (Common Language Runtime)
2. Возможность создавать компоненты проекта в единой среде на наиболее подходящем языке программирования
3. Доступность всех средств .NET для каждого из широкого спектра языков программирования
4. Сервисные возможности для разработчиков, (отладка, анализ кода, …) одинаковы для всех языков
5. Возможность облегченной самостоятельной разработки транслятора для любого языка программирования
   (**Microsoft** – VB, C#, … **другие** – APL, COBOL, Eiffel, Fortran, Haskell, **SML**, Perl, Python, Scheme, Smalltalk, …)

Архитектурная схема .NET Framework и Visual Studio.NET

Схема компиляции в Common Language Runtime

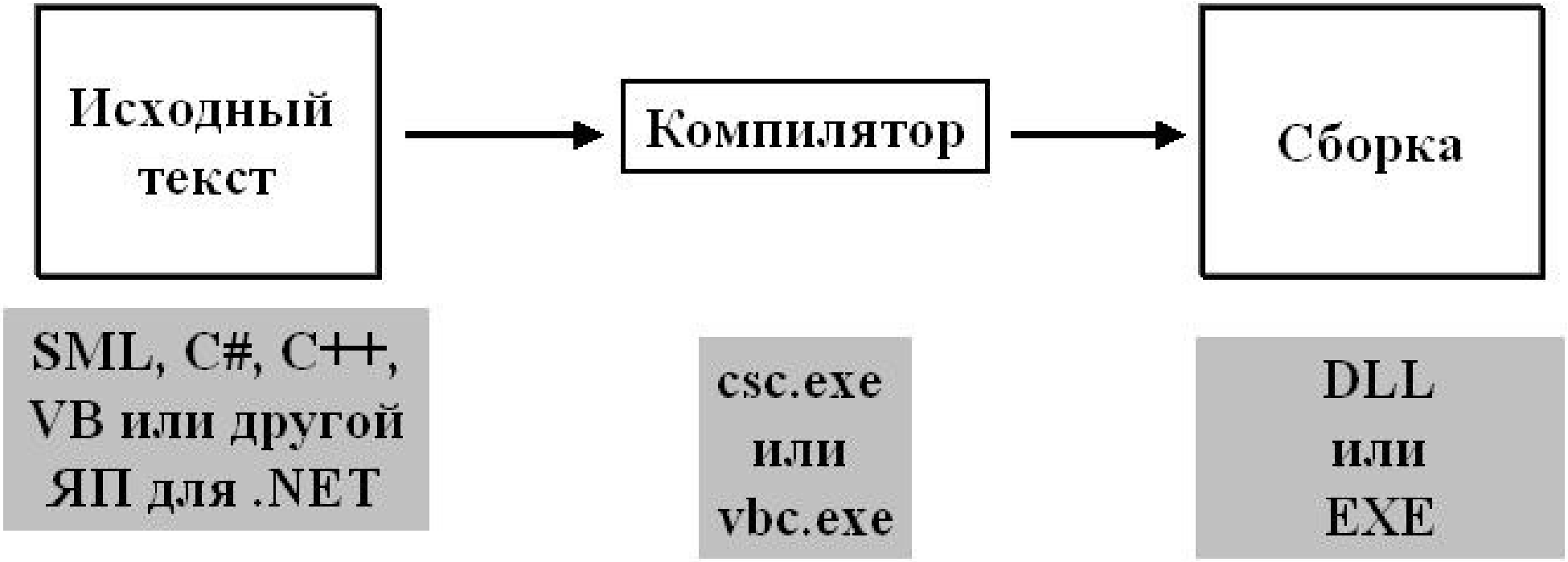
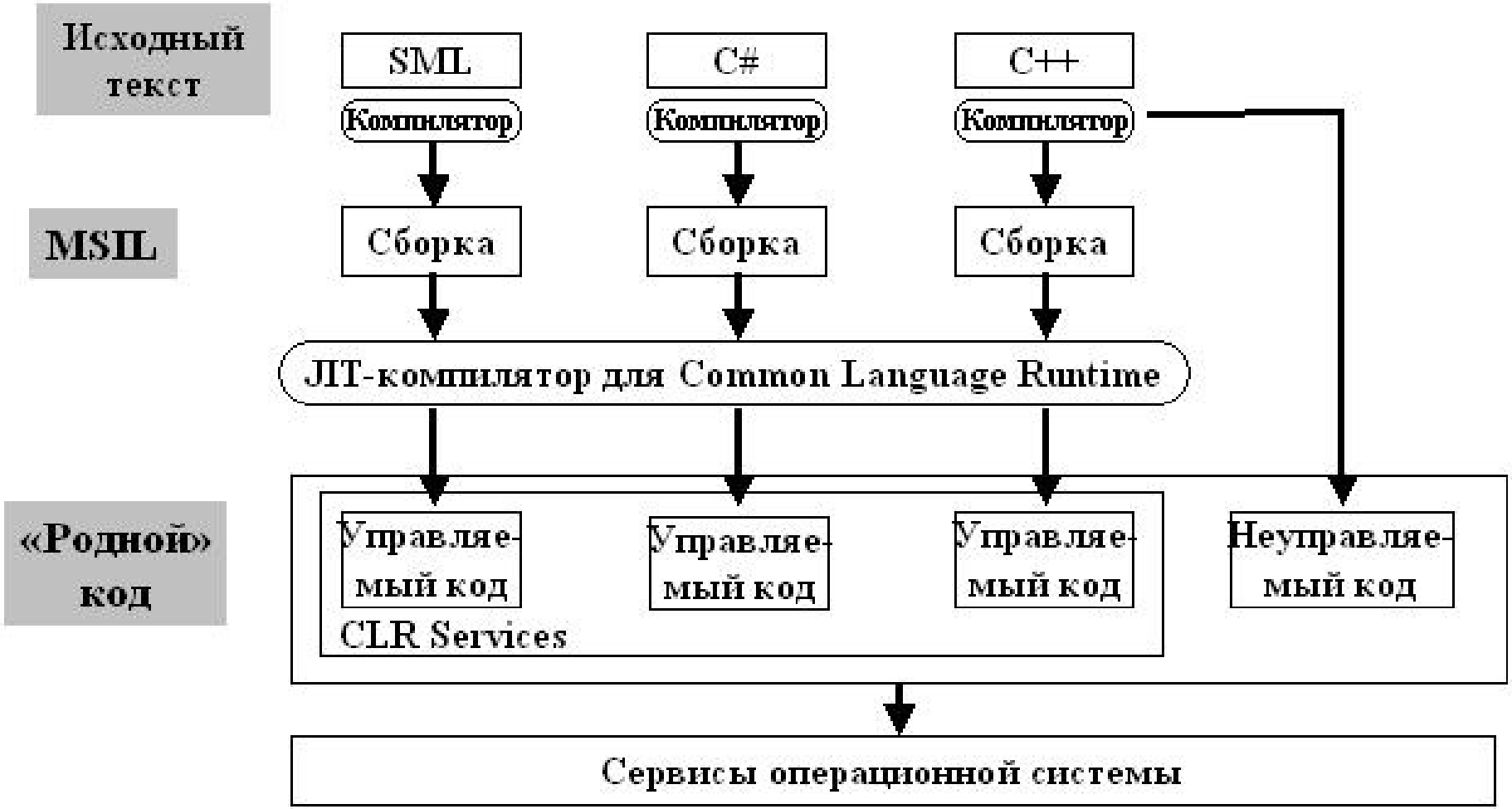


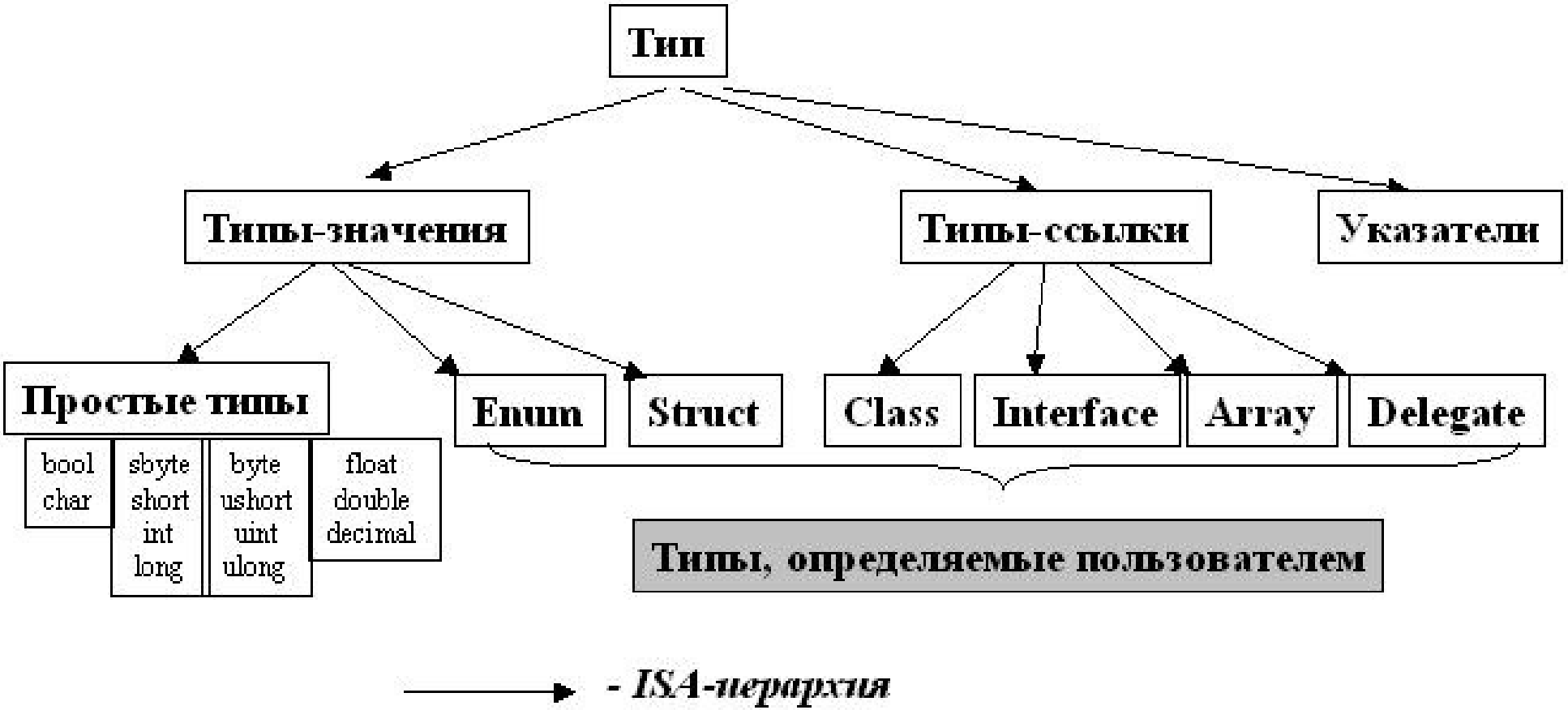
Схема выполнения CLR



CLR: Основные возможности

- поддержка стандартных типов и правил создания новых типов;
- межъязыковая интеграция
  - включение в код на одном ЯП классов на другом ЯП;
  - обработка исключений из программы на одном ЯП программой на другом ЯП;
  - ….
- единый набор библиотек классов для всех поддерживаемых ЯП;
- самоописываемые компоненты – не требуют дополнительных файлов (IDL, TLB, Proxy/Stub и т.п.);
- поддержка версий компонент и сборок кода;
- сервисы безопасности (запрет неавторизованного доступа к ресурсам для пользователей – role-based security, кода – code-based security и др.).

Универсальная система типизации (UTS)



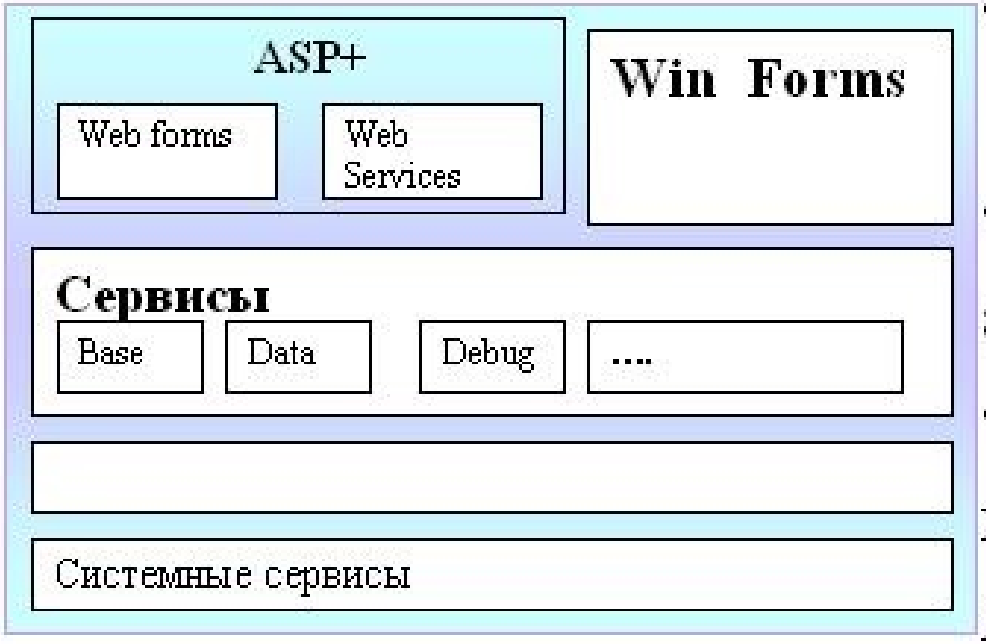
Архитектура .NET – «сборки» кода

Сборка кода (assembly) — группа ресурсов, типов и метаданные, описывающие эти ресурсы и типы.

Особенности:

- распространяется и реализуется как единое целое
- метаданные сборки содержат информацию о зависимостях между ресурсами, версиях и т.п.
- сборка характеризуется номером версии (последняя, специфичная, и т.п.)

Архитектура .NET – уровень сервисов



- Принцип .NET – «ПО как сервис»
- Следующий уровень арх-ры – уровень сервисов
- Сервисы доступны на уровне классов любого ЯП для .NET.

Арх-ра .NET – CLR, сервисы, компоненты

- CLR располагается над сервисами ОС (Windows CE, Windows ME, Windows 2000, Windows .NET)
- Системные сервисы - располагаются над CLR (доступ – через библиотеки классов):
  - доступ к функциям ОС
  - управление данными
  - отладка
  - другие сервисы и т.п.
- Выше – компоненты и сервисы для разработки:
  - Web-узлов
  - Web-сервисов
  - пользовательских интерфейсов (GUI)

Платформа .NET – Виды Интернет-приложений:

- **Web-приложения** – архитектура «клиент-сервер» с доступом пользователей к данным через Web-браузер (технология ASP .NET)
- **"Распределенные" приложения** – на основе иных механизмов удаленного взаимодействия компонент:
  - XML Web Services – на основе открытых стандартов
  - .NET Remoting – на основе внутренних протоколов Microsoft

.NET – виды базовых классов для сервисов:

- доступ к сервисам ОС (Windows CE, ME, 2000,.NET)
- доступ к графическим функциям (двумерная графика, обработка изображений, шрифты, в т.ч. технология ClearType, интеграция с GDI и DirectX)
- сетевые функции
- управление потоками
- глобализация
- криптография
- доступ к данным (библиотека классов ADO+ и OLE DB-драйверы)
- классы для средств разработки (отладка, трассировка, управление ресурсами, компиляция, установка ПО, протоколирование событий,…)
- другие классы (в т.ч. поддержка протокола SOAP)

Платформа .NET – уровень Windows Forms

Назначение – обеспечение разработки традиционных Windows-приложений на основе сервисов Microsoft .NET.

Особенности разработки – унификация доступа к:

- библиотекам классов
- механизмам распространения сервисов
- механизмам поддержки версий
- сервисам безопасности.

Вывод: создание Windows-приложений в архитектуре Microsoft .NET дает разработчикам существенные преимущества по сравнению с традиционным API-ориентированным подходом.

Платформа .NET – уровень Web Forms (1)

Назначение: Основа Web-сервисов и Web-приложений в архитектуре Microsoft .NET

Особенность:

- Программная модель основана на ASP+ — новом поколении активных серверных страниц, (эволюция технологии ASP –более 1 млн. разработчиков)

Идея веб-форм:

- (из Visual Basic 6): отделение логики Web-приложения от интерфейса (за счет объединения в рамках формы ASP- и HTML-кода)

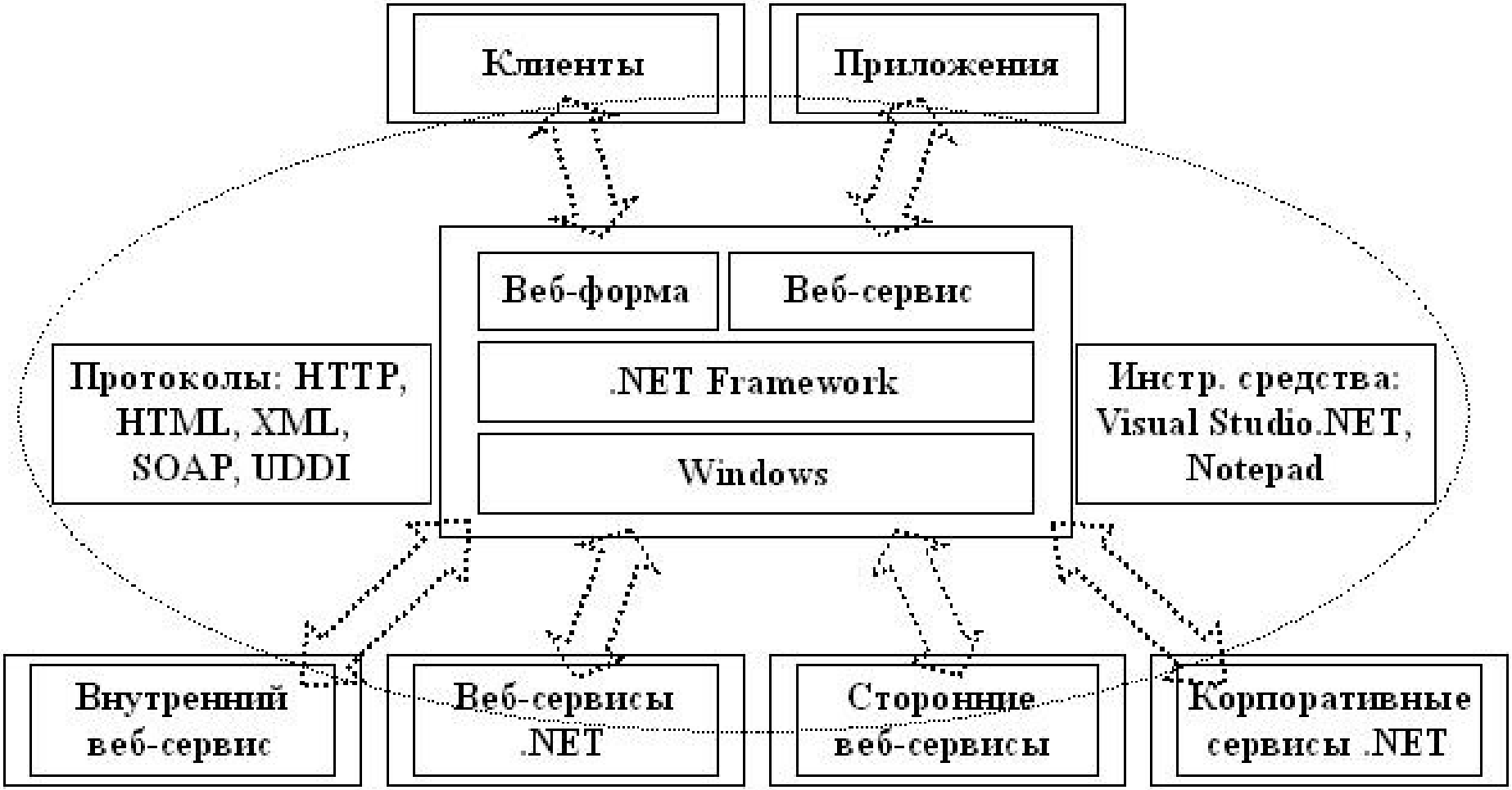
Платформа .NET – уровень Web Forms (2)

Преимущества:

- более строгая структурированность приложений,
- широкий спектр (серверных) интерфейсных элементов
- простая и мощная объектная модель
- легкость разработки (и масштабирования) Web-приложений

Основное средство для разработки приложений и сервисов в архитектуре .NET — Microsoft Visual Studio .NET. (современная версия Microsoft Visual Studio)

Веб-сервисы в .NET (1)



Веб-сервисы в .NET (2)

1. Программируемые компоненты приложений, доступные посредством стандартных Интернет-протоколов
2. Центральная часть архитектуры .NET
3. Распределяют функциональность по глобальной сети
4. Строятся на существующих и развивающихся стандартах: HTTP, XML, SOAP, UDDI, WSDL и др.

Компонентное программирование в .NET (1)

- Компоненты – это:
  - независимые повторно используемые и тиражируемые модули;
  - в целом более крупные, чем объект (объекты – конструкции уровня ЯП);
  - могут содержать множественные классы;
  - независимы от языка реализации.
- В общем случае, разработчик и пользователь компонента территориально разделены и пользуются разными языками в единой среде.

Компонентное программирование в .NET (2)

- Компонентная объектная модель (COM):
  - основной стандарт Microsoft для компонент;
  - содержит протокол для инициализации и использования компонентов внутри одного процесса, между процессами или между компьютерами;
  - основа для ActiveX, OLE и многих других технологий;
  - поддерживается в Visual Basic, C++, .NET и др.
- Модель Java Beans:
  - основной стандарт Sun Microsystems для компонент;
  - зависима от языка реализации.

Сравнение компонентно- и объектно-ориентированного программирования

1. Основные понятия объектно-ориентированного программирования:

- класс (class);
- интерфейс (interface)

2. Основные понятия компонентно-ориентированного программирования:

- свойство (property);
- событие (event);
- сборка (assembly)

.NET – наиболее существенные недостатки

1. Высокие требования к аппаратному обеспечению (минимум 256M RAM, 10G HDD для работы с Microsoft Visual Studio .NET)
2. Сложности работы с некоммерческими релизами программного обеспечения (некоторая неустойчивость, отсутствие полномасштабной документации);
3. Поддержка ряда теоретически интересных и практически полезных языков программирвоания не в полном объеме (SML для Visual Studio .NET – в процессе реализации);
4. Инструментарий .NET (и компиляторы для языков программирования) не ратифицированы по международным стандартам.

Платформа .NET – выводы (1)

1. Стратегическая идеология и технологическая платформа Microsoft на ближайшее десятилетие
2. Несомненное качественное превосходство над аналогами (Borland Delphi, Microsoft Visual Studio и др.) за счет:
   - интероперабельности и межъязыкового взаимодействия;
   - многоуровневой безопасности;
   - интеграции с веб-сервисами;
   - облегчения разворачивания и использования.
3. Некоторая незавершенность решения для широкого коммерческого использования в силу концептуальной новизны и грандиозности проекта.

Платформа .NET – выводы (2)

1. .NET – развитие платформы Windows
2. .NET – фундамент для создания корпоративных приложений нового поколения
3. основа. NET – компонентная интеграция приложений на уровне сервисов, взаимодействующих посредством языка XML и протокола SOAP
4. стратегическая цель .NET – создание инфраструктуры для разработки и функционирования распределенных приложений на базе Интернет-стандартов

Библиография

1. http://msdn.microsoft.com/net
2. Nathan A. .NET and COM: The Complete Interoperability Guide. Sams, 2002, 1608 pp.
3. Box D. Essential .NET, Vol.1: The Common Language Runtime. Addison Wesley, 2002, 432 pp.
4. Grimes F. Microsoft .NET for Programmers. Manning Publications, 2002, 386 pp.
5. J. Richter. Applied Microsoft .NET Framework Programming. Microsoft Press, 2002, 556 pp.
6. Зыков С.В. Проектирование корпоративных порталов.– М.: МФТИ, 2005.– 258 с.

Благодарю за внимание!

Вопросы?

- http://zykov.altweb.ru
- szykov@hotmail.com
- sergey.zykov@tekama.com