使用Delphi .NET开发

Using Delphi for Microsoft .NET Framework Development

主讲：周爱民
(Aimingoo)

- Delphi .NET的基本应用
- Delphi .NET的开发
- Delphi .NET中调用Win32代码
- Delphi .NET内核概要

Delphi .NET的基本应用

(10 分钟)

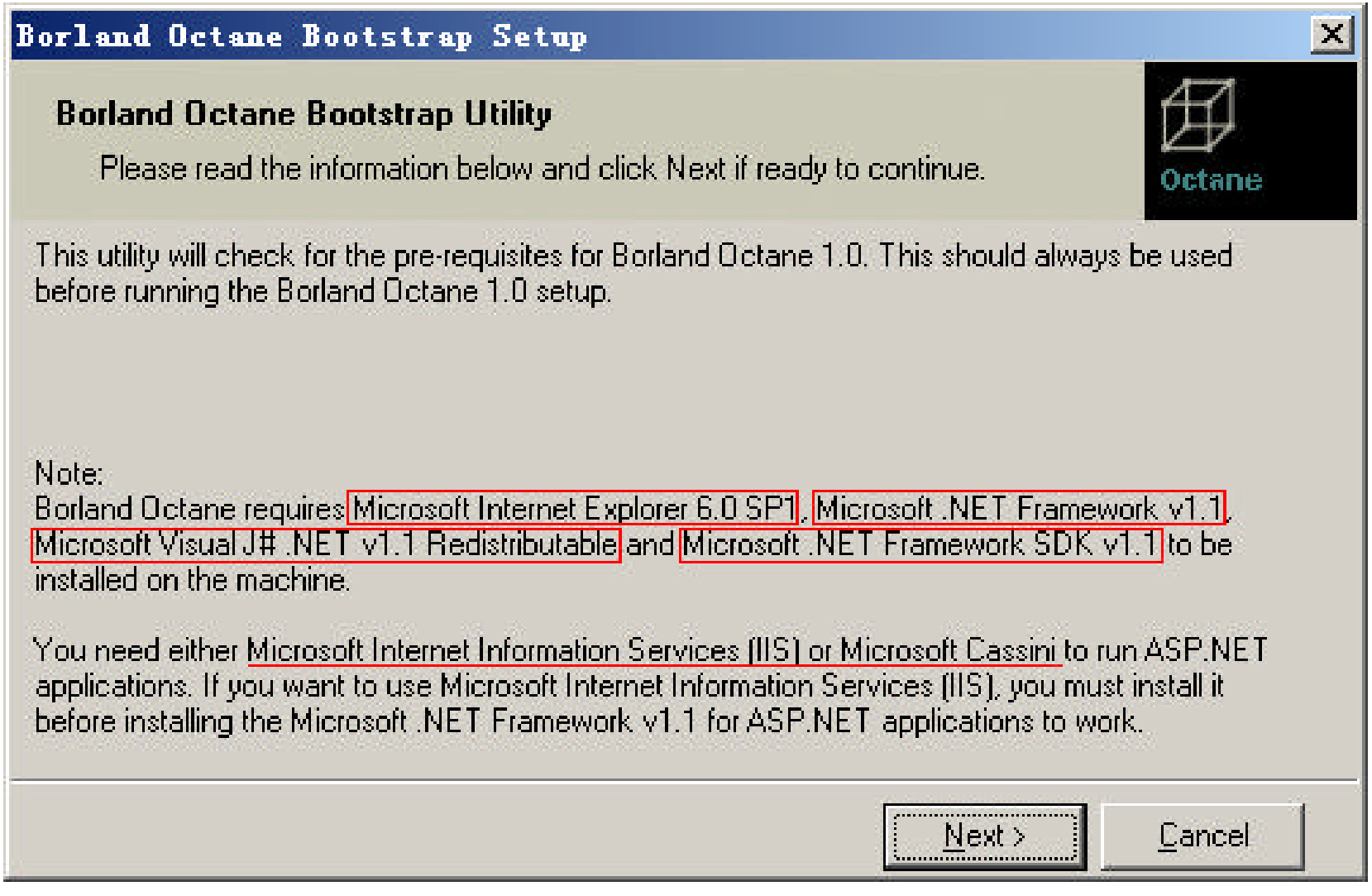
Delphi .NET的基本应用

- 安装Delphi .NET的基本需求
- Delphi .NET产品结构
- Delphi .NET的可选件
- Hello, World!
- Delphi .NET的调试环境

安装Delphi .NET的基本需求

- MS Internet Explorer 6.0 SP1
- MS .NET Framework v1.1
- MS .NET Framework SDK v1.1
- MS VJ# v1.1 Redistributable

- MS Internet Infomation Server (IIS)
- MS Cassini Web Server


Borland Octane Bootstrap Setup
Borland Octane Bootstrap Utility
Please read the information below and click Next if ready to continue.
Octane
This utility will check for the pre-requisites for Borland Octane 1.0. This should always be used before running the Borland Octane 1.0 setup.
Note:
Borland Octane requires Microsoft Internet Explorer 6.0 SP1, Microsoft .NET Framework v1.1, Microsoft Visual J# .NET v1.1 Redistributable and Microsoft .NET Framework SDK v1.1 to be installed on the machine.
You need either Microsoft Internet Information Services (IIS) or Microsoft Cassini to run ASP.NET applications. If you want to use Microsoft Internet Information Services (IIS), you must install it before installing the Microsoft .NET Framework v1.1 for ASP.NET applications to work.
Next >
Cancel

Delphi.NET产品结构

HTML 可视环境(Tidy)

Web Services

源码版本管理(Team)

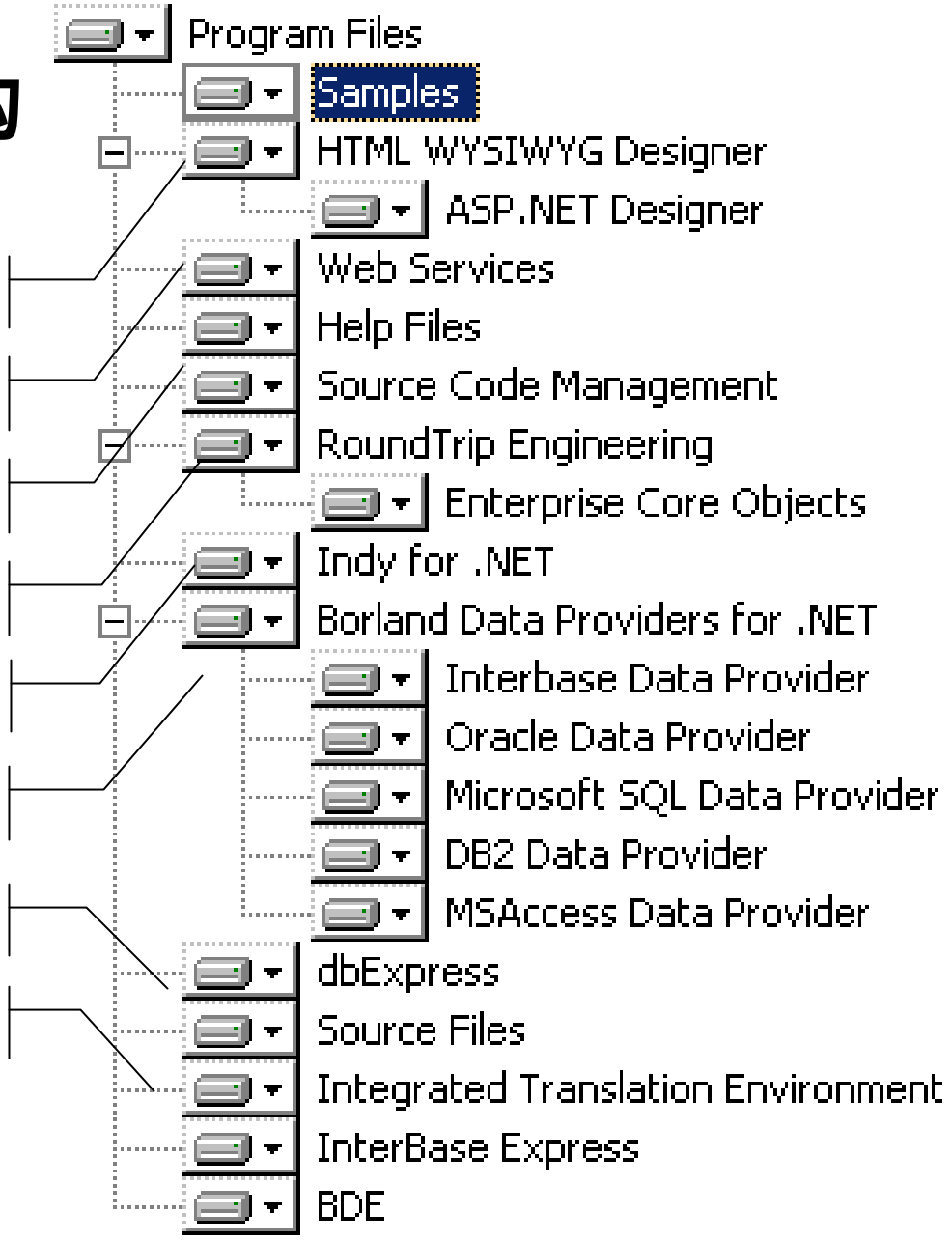
UML & MDA

Internet 应用开发

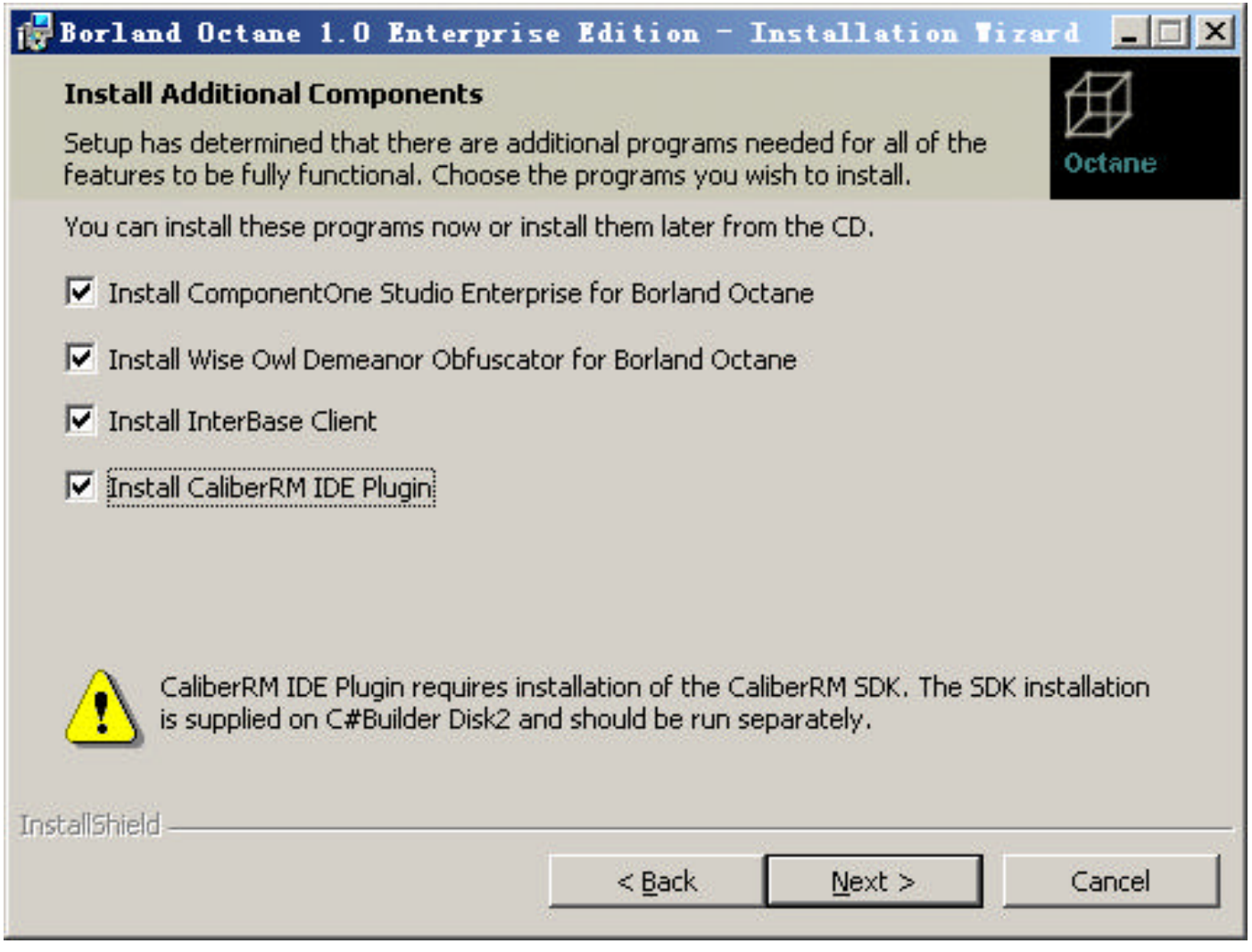
数据供应器

DBExperss接口层

多语言支持(ITE)



Delphi .NET的可选件



Hello, World!

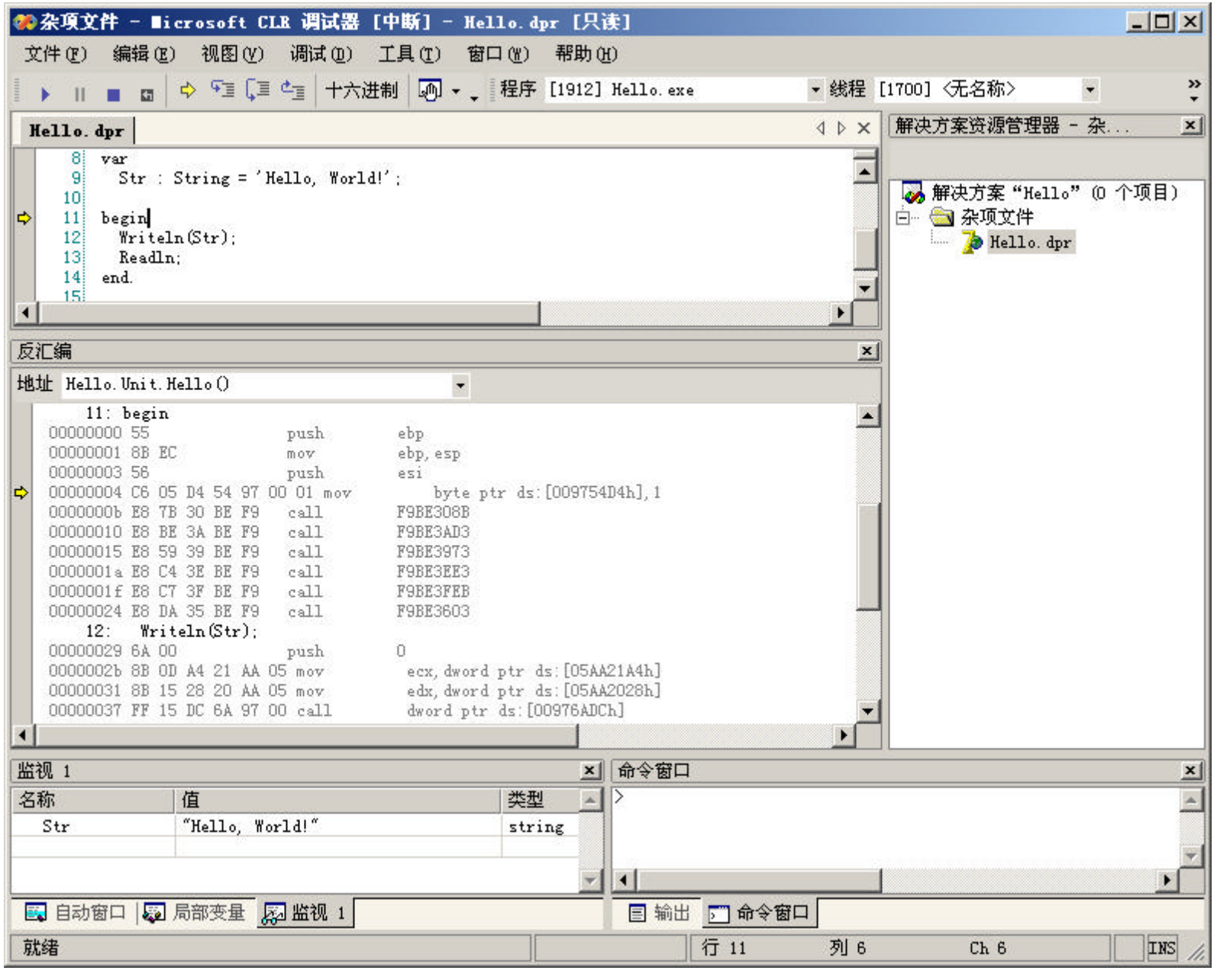
你能想象一下，.NET版本的
”Hello, World!”
程序会是怎样的吗？

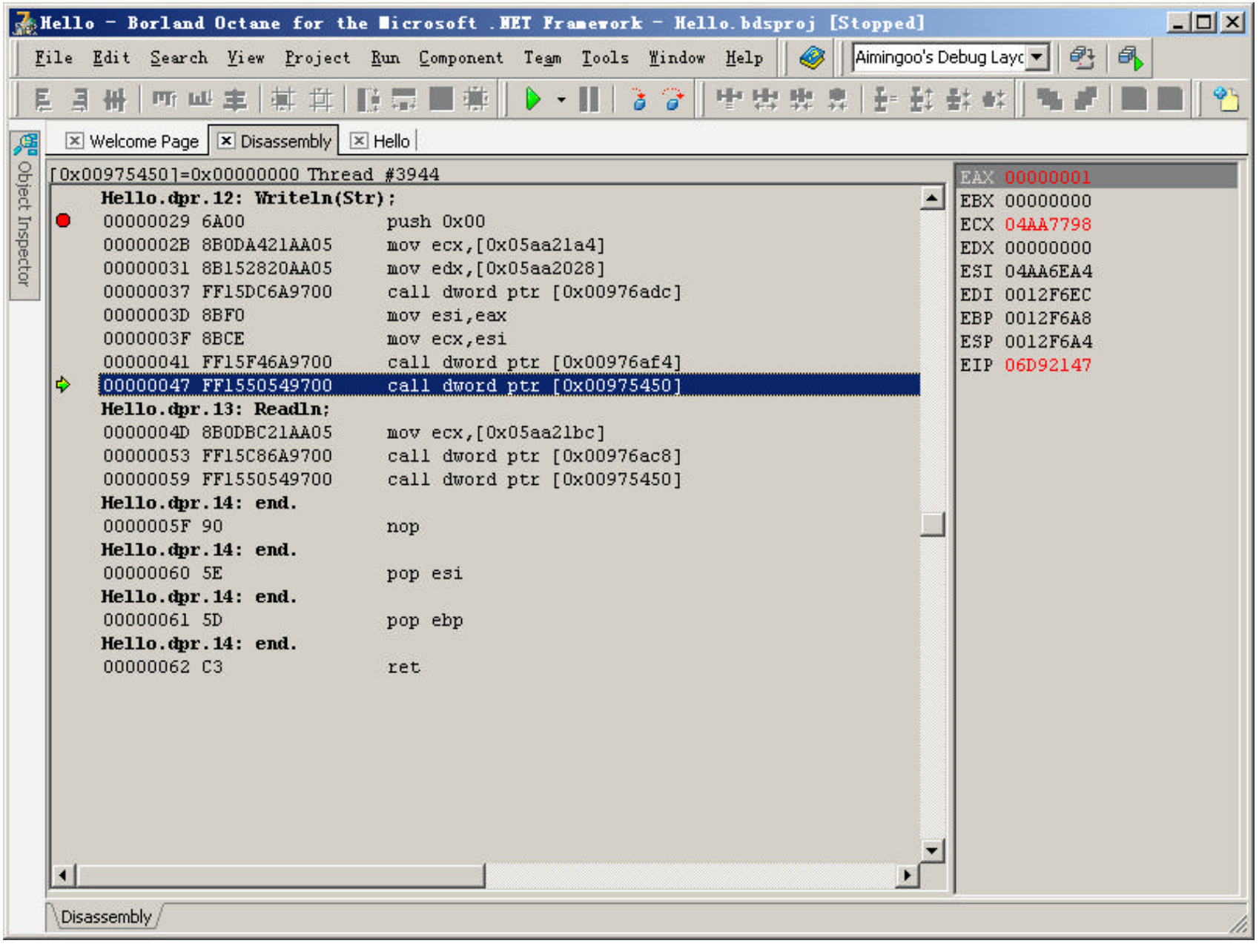
```
// .Net 版本的Hello, World源代码清单

program Hello;

{$APPTYPE CONSOLE}

var
  Str : String = 'Hello, World!';

begin
  Writeln(Str);
  Readln;
end.
```

Delphi .NET的调试环境

- 使用.NET SDK中的DbgCLR.EXE
- 使用Octane的集成调试环境

操作演示


杂项文件 - Microsoft CLR 调试器 [中断] - Hello.dpr [只读]
文件(F) 编辑(E) 视图(V) 调试(D) 工具(T) 窗口(W) 帮助(H)
十六进制
程序 [1912] Hello.exe
线程 [1700] <无名称>
Hello.dpr
8 var
9 Str : String = 'Hello, World!';
10
11 begin
12 Writeln(Str);
13 Readln;
14 end.
15
解决方案资源管理器 - 杂...
解决方案 "Hello" (0 个项目)
杂项文件
Hello.dpr
反汇编
地址 Hello.Unit.Hello()
11: begin
00000000 55 push ebp
00000001 8B EC mov ebp,esp
00000003 56 push esi
00000004 C6 05 D4 54 97 00 01 mov byte ptr ds:[009754D4h],1
0000000b E8 7B 30 BE F9 call F9BE308B
00000010 E8 BE 3A BE F9 call F9BE3AD3
00000015 E8 59 39 BE F9 call F9BE3973
0000001a E8 C4 3E BE F9 call F9BE3EE3
0000001f E8 C7 3F BE F9 call F9BE3FEB
00000024 E8 DA 35 BE F9 call F9BE3603
12: Writeln(Str);
00000029 6A 00 push 0
0000002b 8B 0D A4 21 AA 05 mov ecx,dword ptr ds:[05AA21A4h]
00000031 8B 15 28 20 AA 05 mov edx,dword ptr ds:[05AA2028h]
00000037 FF 15 DC 6A 97 00 call dword ptr ds:[00976ADCh]
监视 1
名称 值 类型
Str "Hello, World!" string
命令窗口
自动窗口 局部变量 监视 1
输出 命令窗口
就绪
行 11 列 6 Ch 6 INS


Hello - Borland Octane for the Microsoft .NET Framework - Hello.bdsproj [Stopped]
File Edit Search View Project Run Component Team Tools Window Help
Aimingoo's Debug Layc
Welcome Page
Disassembly
Hello
Object Inspector
[0x00975450]=0x00000000 Thread #3944
Hello.dpr.12: Writeln(Str);
00000029 6A00 push 0x00
0000002B 8B0DA421AA05 mov ecx,[0x05aa21a4]
00000031 8B152820AA05 mov edx,[0x05aa2028]
00000037 FF15DC6A9700 call dword ptr [0x00976adc]
0000003D 8BF0 mov esi,eax
0000003F 8BCE mov ecx,esi
00000041 FF15F46A9700 call dword ptr [0x00976af4]
00000047 FF1550549700 call dword ptr [0x00975450]
Hello.dpr.13: Readln;
0000004D 8B0DBC21AA05 mov ecx,[0x05aa21bc]
00000053 FF15C86A9700 call dword ptr [0x00976ac8]
00000059 FF1550549700 call dword ptr [0x00975450]
Hello.dpr.14: end.
0000005F 90 nop
Hello.dpr.14: end.
00000060 5E pop esi
Hello.dpr.14: end.
00000061 5D pop ebp
Hello.dpr.14: end.
00000062 C3 ret
EAX 00000001
EBX 00000000
ECX 04AA7798
EDX 00000000
ESI 04AA6EA4
EDI 0012F6EC
EBP 0012F6A8
ESP 0012F6A4
EIP 06D92147
Disassembly

Delphi .NET的开发

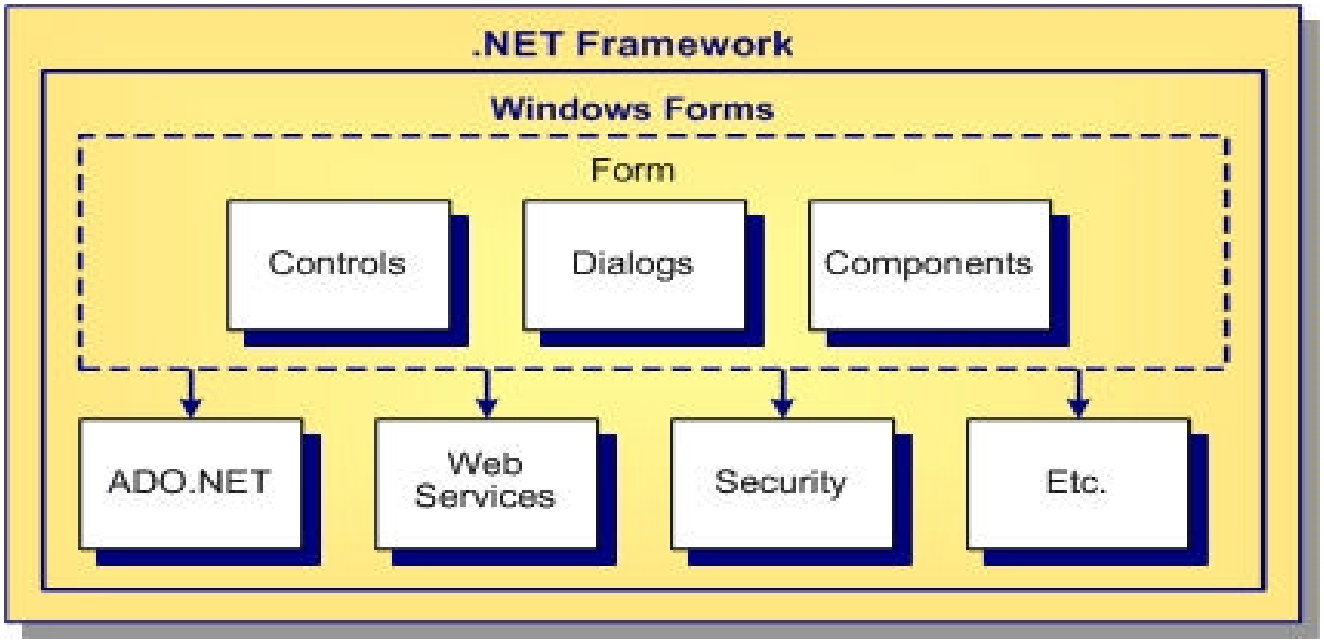
(30 分钟)

Delphi .NET的开发

- Console Application
- Windows Forms Application
- VCL Forms Application
- Data Providers Database Application
- ASP .NET Web Application
- ASP .NET Web Services Application
- Web Control Library
- Class、Component & Package

Windows Forms Application

- **透过**Forms，**可以轻松地访问**.NET**构架的其它部分。例如使用**Windows Forms**作为**Web Services**的**Client，**或者透过**ADO.NET**界面访问数据库。**

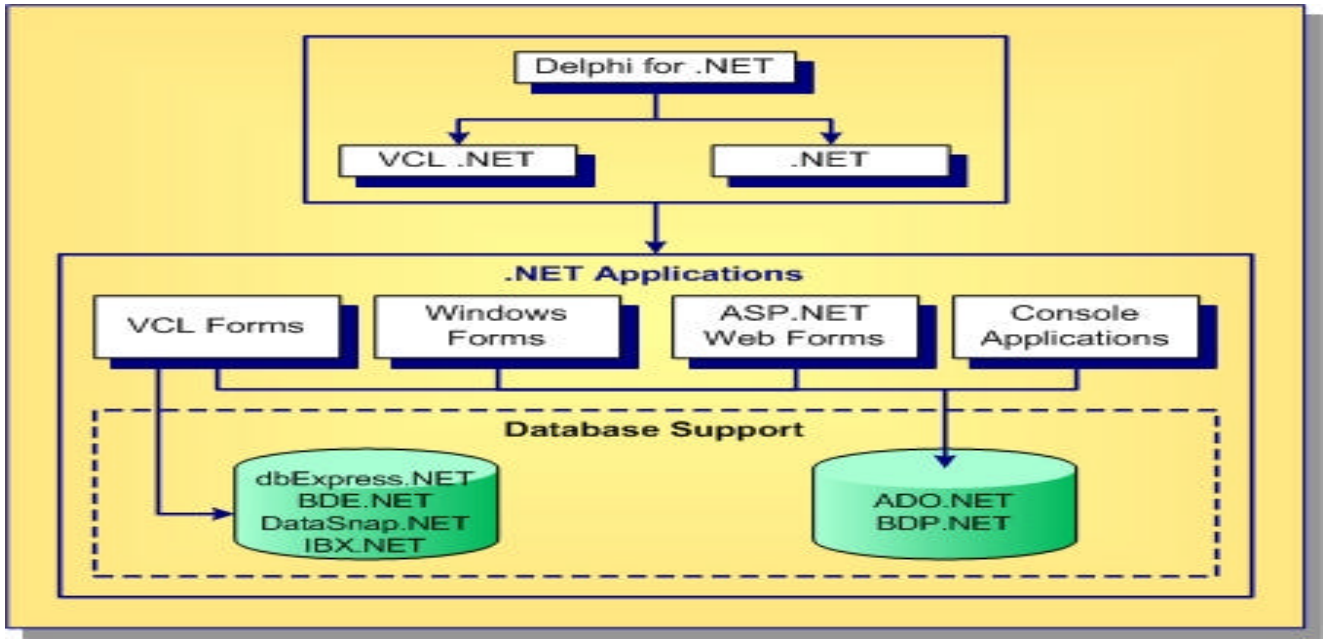


Windows Forms App. **体系结构**

程序示例

VCL Forms Application

- **通过将**Delphi VCL**映射到**.NET FCL，Borland**实现了在**Delphi .NET**上转化旧有系统的最简便方法。使旧有系统在**.NET**上的转化成本降至最低。**

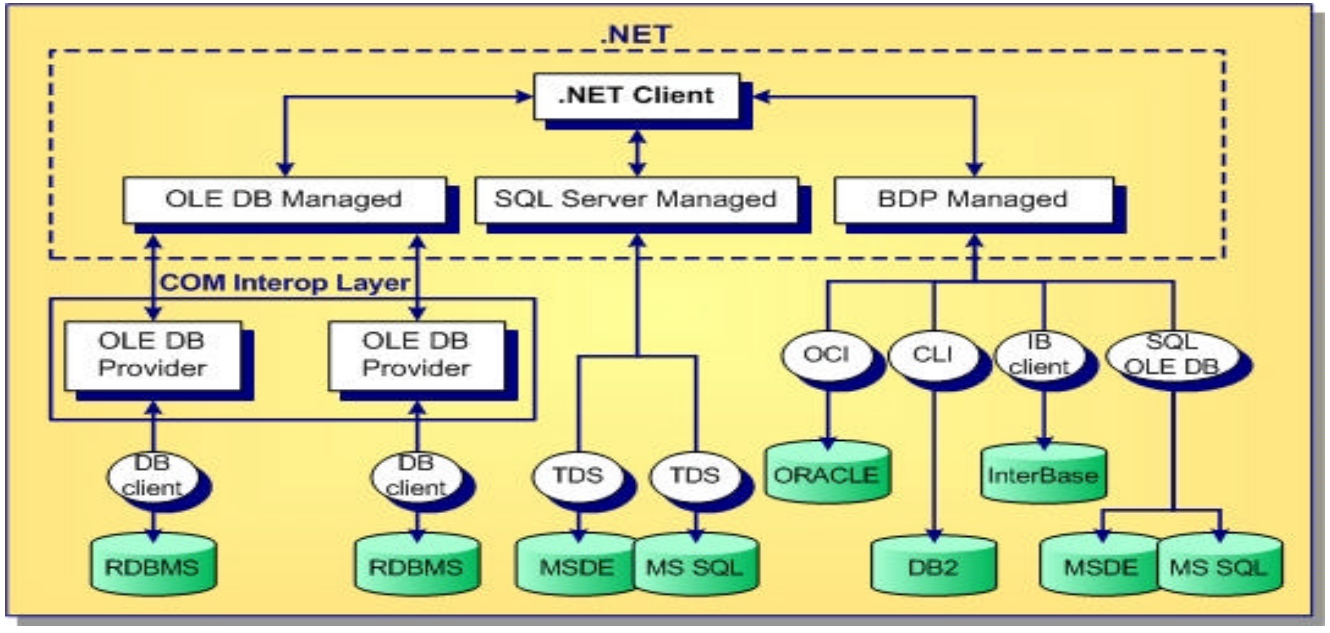


Delphi for .NET **的体系结构**

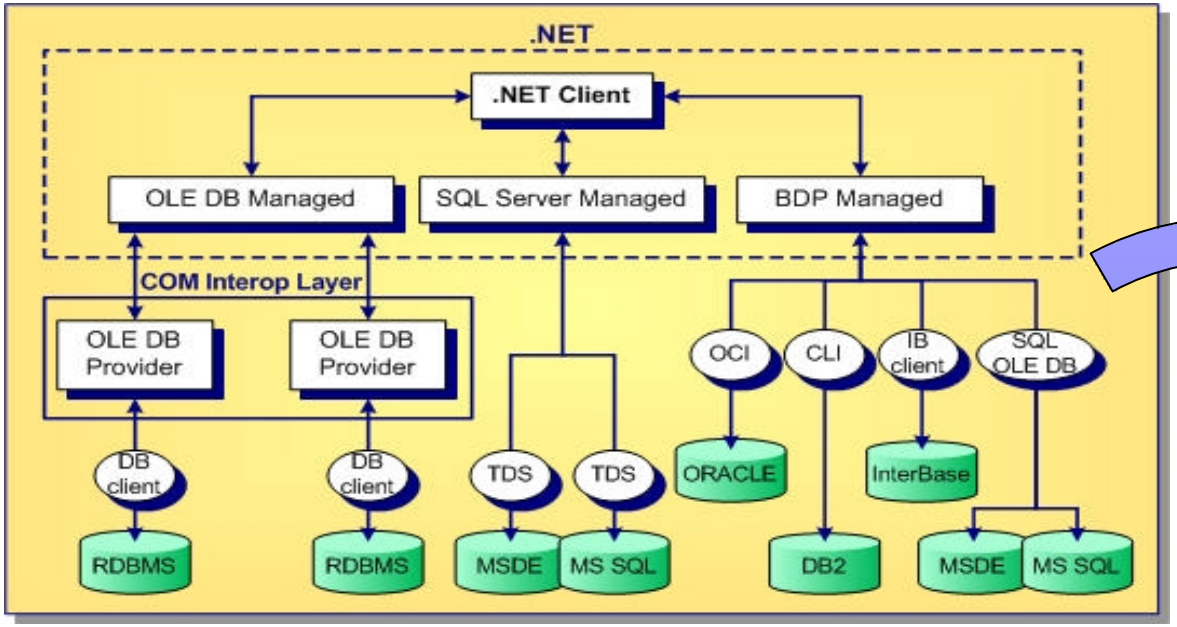
程序示例

Data Providers Database Application

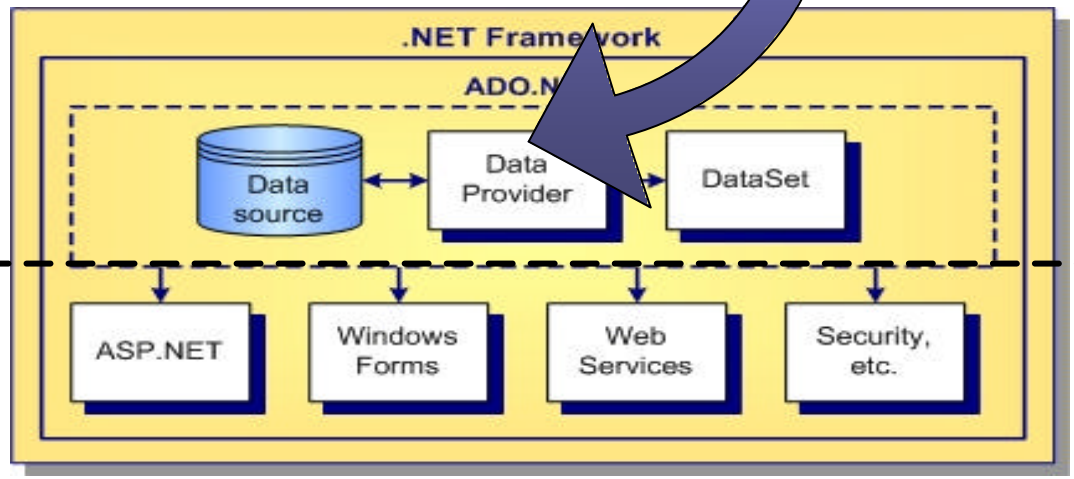
- BDP为Windows Forms App.、ASP.NET App.，以及Web Services提供同样的数据库接口界面。



BDP与其它.NET数据存取方案的比较



BDP并不改变ADO.NET的数据存取界面，而是对Data Provider的一个实现。

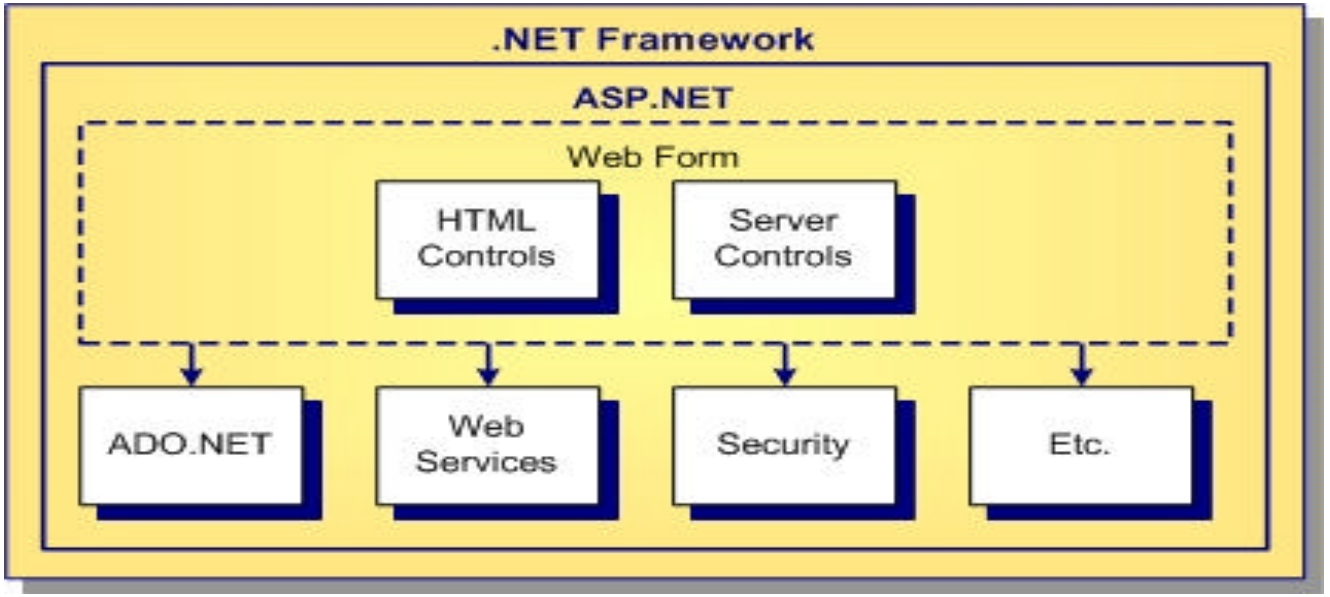


BDP在.NET数据存取体系中的地位

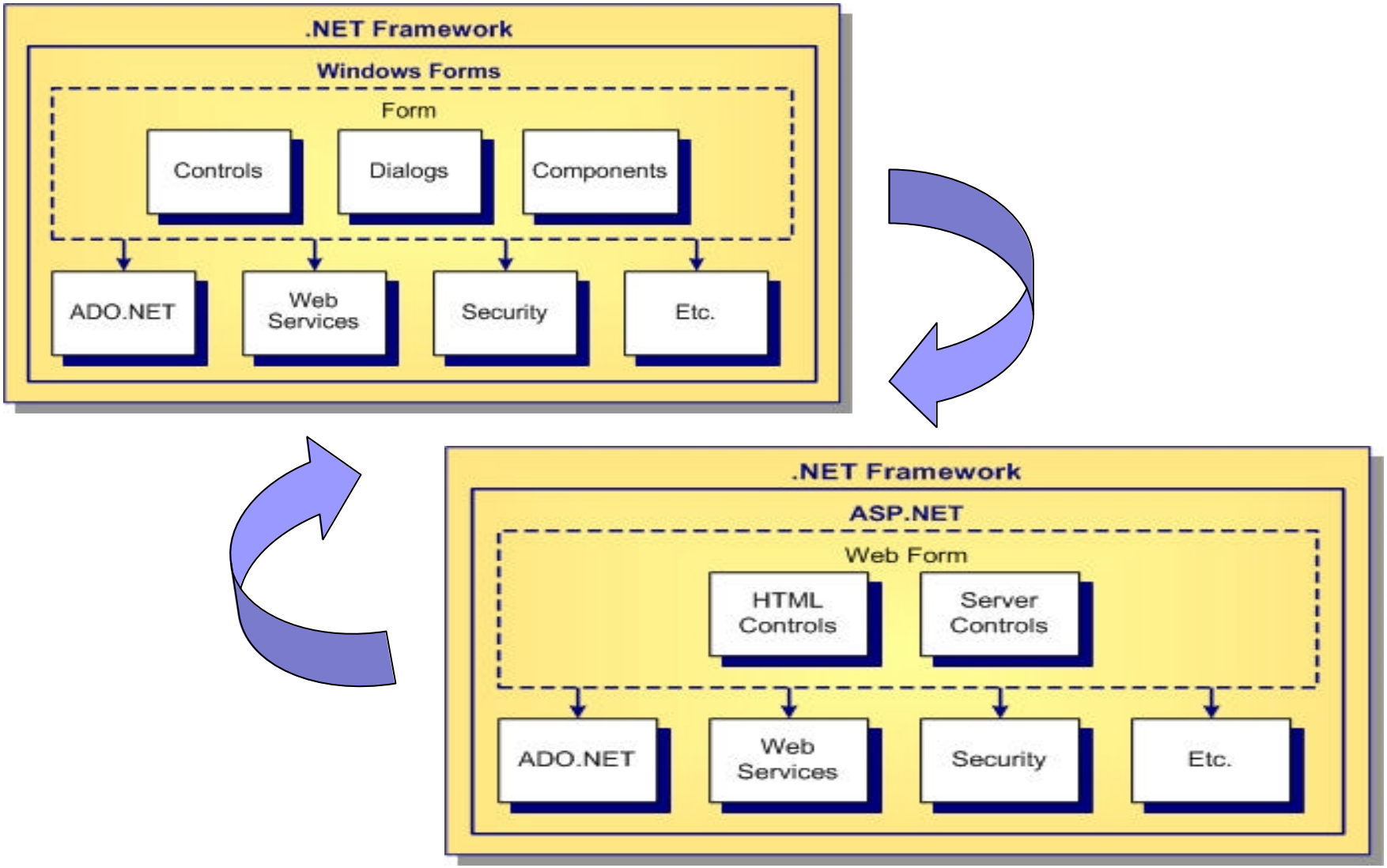
程序示例

ASP .NET Web Application

- HTML**控件与**Server**控件，享有与**Windows Forms**控件相同的设计界面。开发**ASP.NET Web App.**与**Windows Forms**方法类同，甚至更容易。**



ASP .NET Web App. **体系结构**

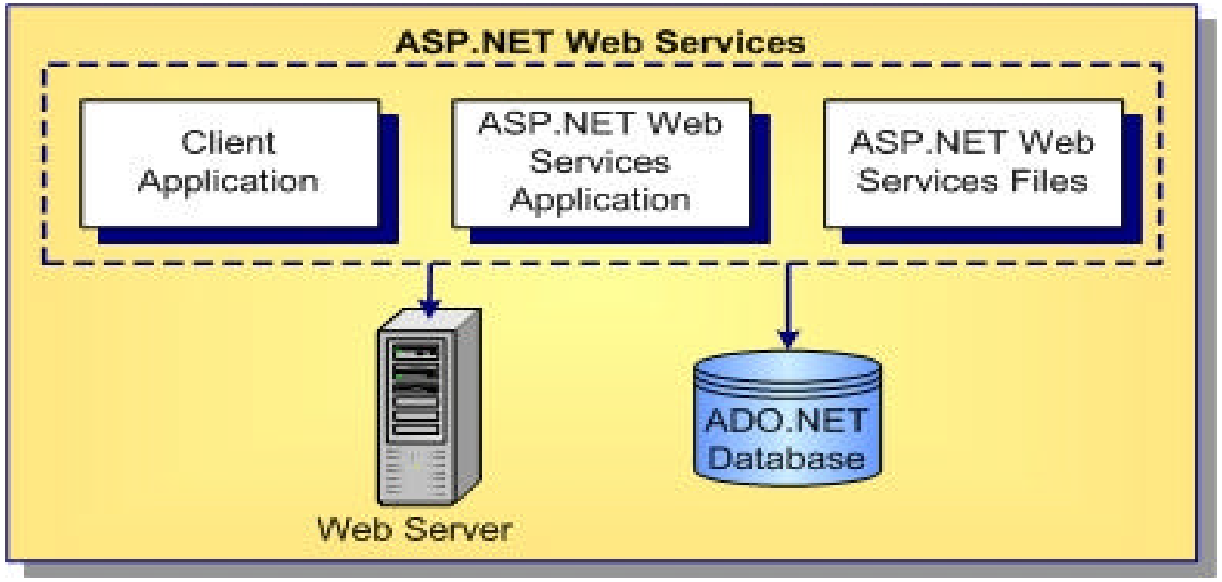


Windows Forms**与**ASP .NET Web App. **相似性**

程序示例

ASP .NET Web Services Application

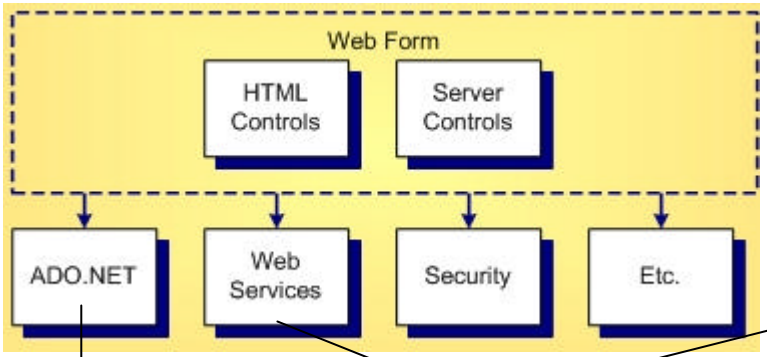
- **对于**Web Servers App.**来说，并不关心**Client**是谁，或怎样开发，而只需要提供一个访问界面。这看来象**COM**组件的机制，但却与**COM**无关。**

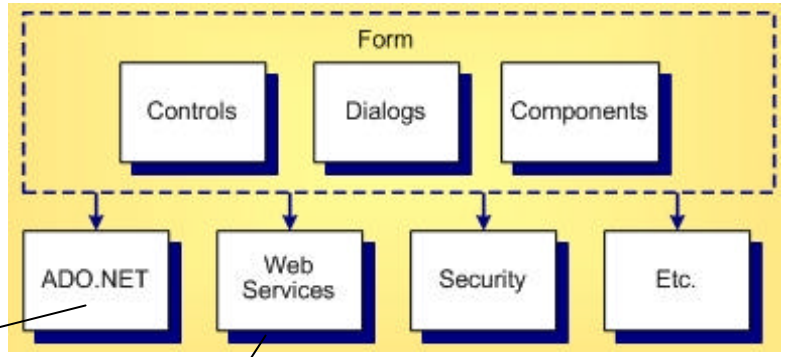


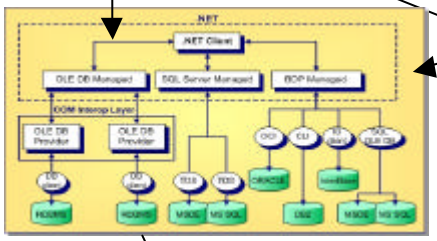
ASP .NET Web Services. **数据存取结构**

ASP .NET Application

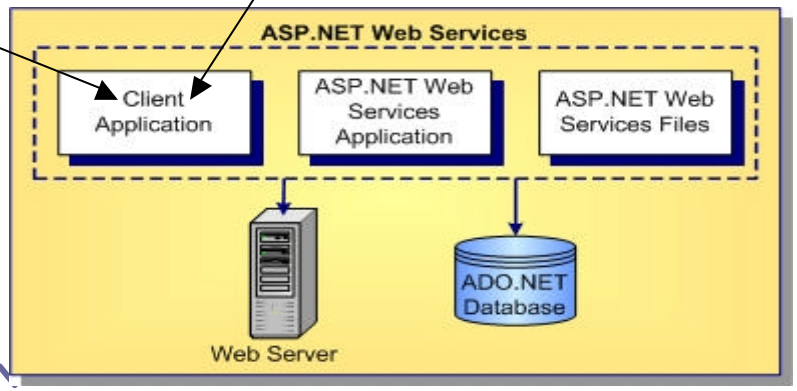
VCL .NET Application
FCL .NET Application



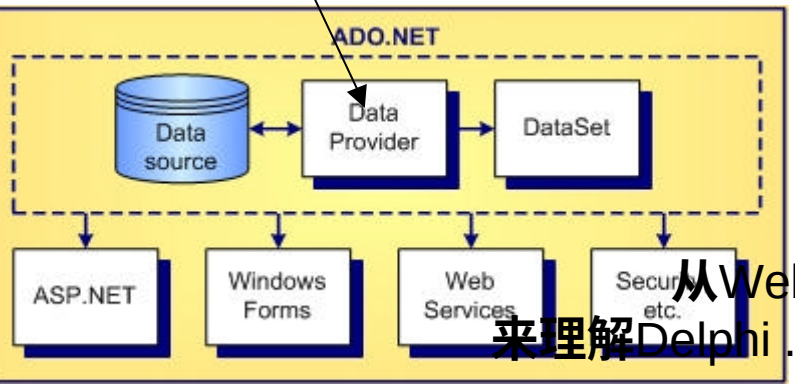




Borland Data Providers



ASP .NET Web Services

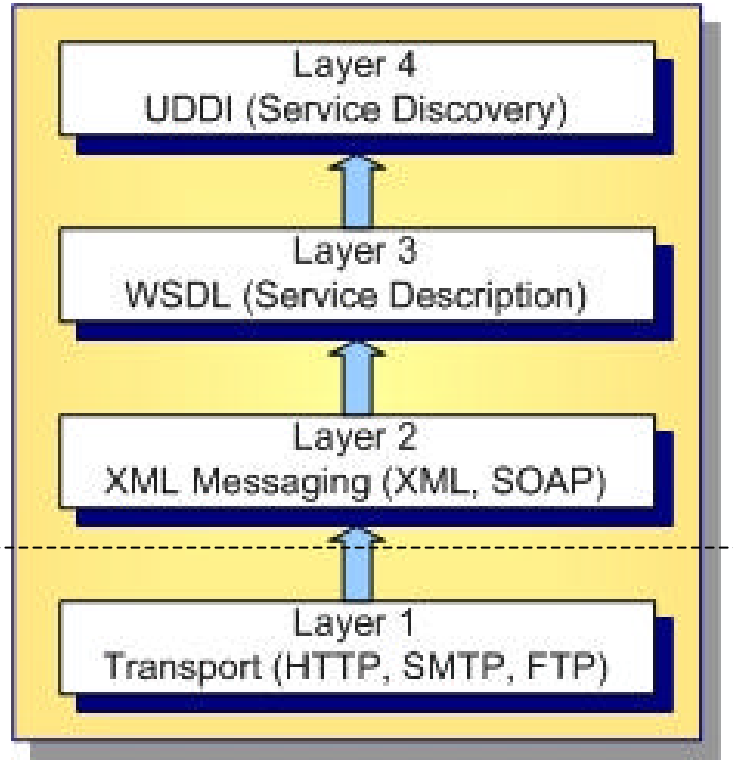


从Web Services的角度
来理解Delphi .NET 应用程序的实现构架

程序示例

使用Client访问Internet上的Web Servers

- **可以使用任何的Client应用来访问Web Servers。而Delphi .NET中的Client实现方案，比其它语言更加便捷。**



Delphi .NET自动实现

Client不必要实现

程序示例

Class、Component & Package

- **快速的实现一个新**Class
- **更好的组件设计器**
  (Windows Forms & ASP.NET User Control)
- Package**管理器**
- **程序组**(Assembly)**与包**(Package)

程序示例

Delphi .NET中调用Win32代码

(10 分钟)

Delphi .NET中调用Win32代码

- 托管(Managed) 与非托管(Unmanaged)代码
- 导入DLL例程
- InterOp实现对非托管代码的调用

托管(Managed) 与非托管(Unmanaged)代码

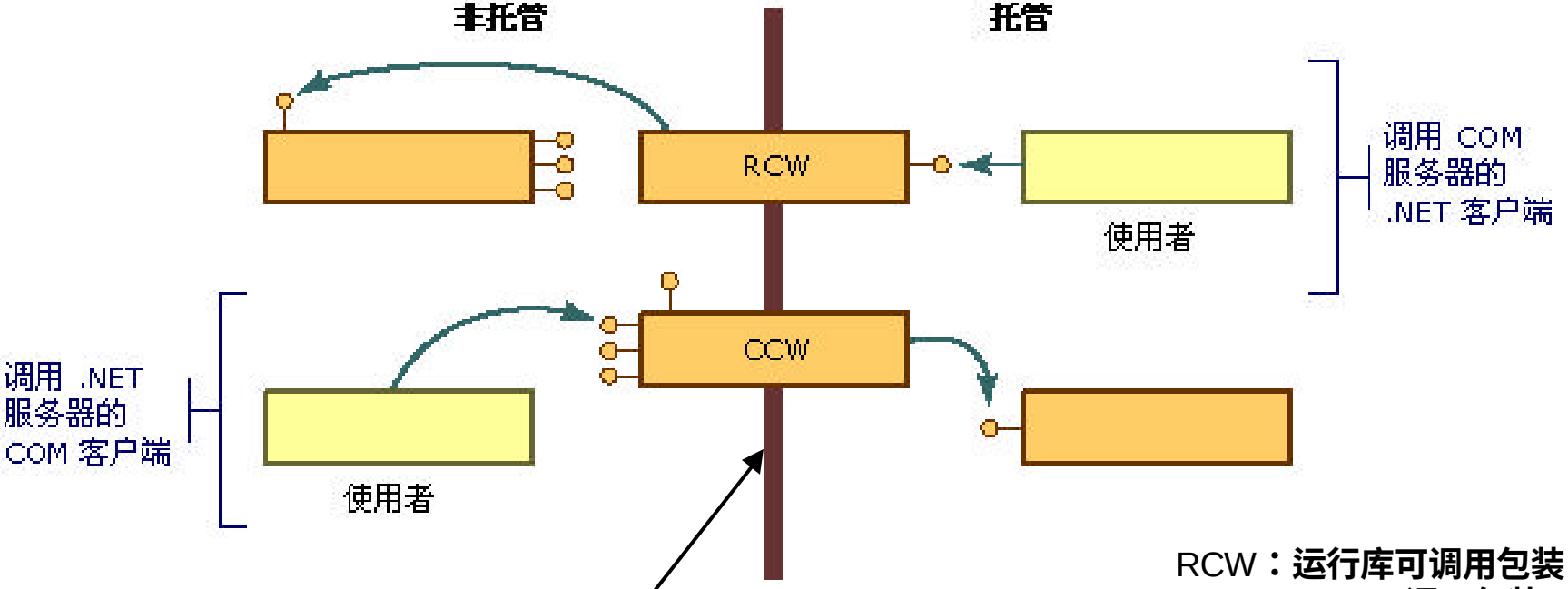
CLR提供的三种方案：

- 托管代码可以调用DLL中的非托管函数
- 托管代码可以使用现有的COM组件(服务器)
- 非托管代码可以使用托管代码实现的服务

导入DLL例程

```
program CurrentDir;
{$APPTYPE CONSOLE}

uses
  System.Text,                              // 支持 StringBuilder 类型
  System.Runtime.InteropServices;    // 支持 DllImport()

[DllImport('kernel32.dll', CharSet = CharSet.Auto, SetLastError = True,
 EntryPoint = 'GetCurrentDirectory')]

function GetCurrentDirectory(nBufferLength: LongWord;
  lpBuffer: StringBuilder): LongWord; external;

var
  Str : StringBuilder;

begin
  Str := StringBuilder.Create(256);
  Str.Length := GetCurrentDirectory(256, Str);
  Writeln(Str.ToString);
end.
```

托管代码 → COM组件
COM组件 → 托管代码实现的服务



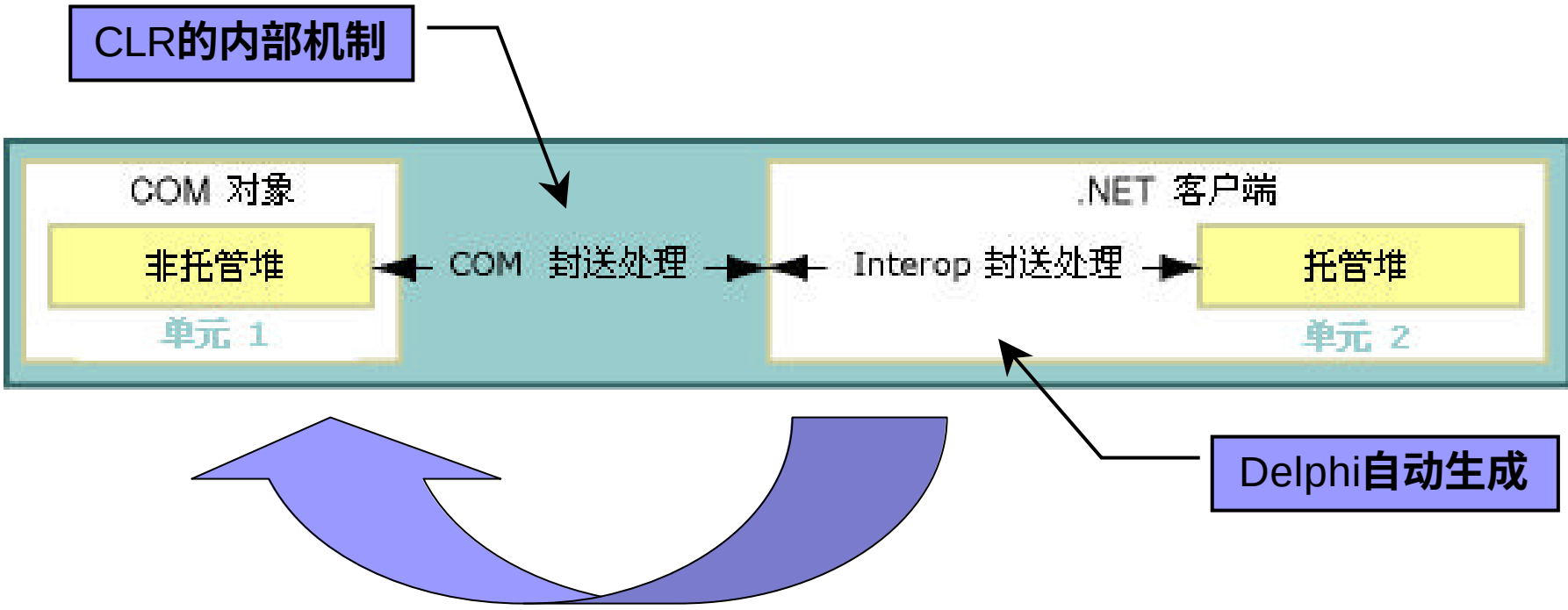
RCW：**运行库可调用包装**
CCW：**COM可调用包装**

Delphi **自动完成RCW和CCW的封包并生成调用代码**

CLR**中的**COM InterOp**机制**

InterOp实现对非托管代码的调用

(**示例：**.NET**版本**Microsoft Web Browser)



程序示例

Delphi .NET的内核概要

(10 分钟)

Delphi .NET的内核概要

- GetMem()、FreeMem()与ReallocMem()
- 消失的PChar、Pointer和过程指针
- 对象类型和类类型：Object Type & Class Type
- 基本类型映射：Types ←→ CTS
- 对象模型映射：VCL ←→ FCL
- 是真正的多重继承吗？
- 全新的RTL
- .NET Pascal语言特性

GetMem()、FreeMem() & ReallocMem()

- **内存是CLR管理的重要资源**
- **在Delphi .NET中，Memory Manager已经不复存在。**
- **必要的情况下，Dynamic Array类型，和New() & Dispose()例程可以作为替代。**

消失的PChar和Pointer

- 使用值类型的强制转换(装箱操作)来替代@
- 如果在API中使用PChar，可以使用String或者StringBuilder类型来替代
- 如果PChar用于操作一个内存块，则可用IntPtr类型或者动态数组(TBytes)类型替代
- 如果在API中使用内存块，可以使用Marshal class从非托管堆中分配并传值
  - GC(garbage collector)不能管理非托管堆，因此Marshal class的实例必须显式释放

真正消失的过程指针

```
type
  TAnimateWindowProc = function(...): BOOL; stdcall;
var
  AnimateWindowProc: TAnimateWindowProc = nil;
begin
  @AnimateWindowProc := GetProcAddress(aHandle, 'AnimateWindow');
  if AnimateWindowProc <> nil then
    AnimateWindowProc(Handle, 100, AW_BLEND or AW_SLIDE);
```

什么是非安全过程(Unsafe Procedure)

```
[DllImport('user32.dll', ..., EntryPoint = 'AnimateWindow')]
function AnimateWindow( ... ): BOOL; external;
```

```
var
  UserHandle: HMODULE;
  CanAnimate: Boolean;
begin
  CanAnimate := GetProcAddress(UserHandle, 'AnimateWindow') <> nil
  if CanAnimate then
    AnimateWindow(Handle, 100, AW_BLEND or AW_SLIDE);
```

对象类型和类类型：Object Type & Class Type

在Turbo Pascal 和 Delphi中：

- **对象类型**(Object Type)**是**

  TMyObject = Object

  ...

  在TP5.5 – Delphi 7.0**中，都被支持。**

  KOL**使用该类型实现自己的构架。**

- **类类型**(Class Type)

  TMyClass = Class

  ...

  Delphi 1.0 – 7.0**的面向对象都基于类类型系统**

- **接口**(Interface) **是编译支持的特殊的类型**

  IInterface = interface

  ...

对象类型和类类型：Object Type & Class Type

在Delphi .NET中：

- 从Turbo Pascal保留下来的原始的对象类型(Object Type)不存在了。

- Delphi .NET遵循CLR的约定，类型只能是：
  - 继承自TObject的引用类型(Class Type)
    - TObject = System.Object;
    - TClass = class of TObject;
    - TMyClass = TClass

      ...
  - 继承自System.ValueType的值(基元)类型
  - Interface类型

基本类型映射：Types ←→ CTS

Delphi Type	.NET Framework Type
ShortInt	System.SByte
Byte	System.Byte
SmallInt	System.Int16
Word	System.UInt16
Integer	System.Int32
Cardinal/LongWord	System.UInt32
Int64	System.Int64
UInt64	System.UInt64
Char/WideChar	System.Char
Single	System.Single
Double/Extended	System.Double
Boolean	System.Boolean
WideString/string	System.String

对象模型映射：VCL ←→ FCL

- TObject = System.Object
- Exception = System.Exception
- TComponent = System.ComponentModel.Component

是真正的多重继承吗？

- CLR不支持多重继承。
- 多重继承特性是基于Interface实现的。

```
TSomething = class(TInterfacedObject, IDescendant, IDesc)
  procedure P1;
  procedure P2;
  ...
end;
```

全新的RTL

- Borland.Delphi.System
- Borland.VCL
- RTL =
  - Data Type Support(Base & Interface & Object) ➔ CTS
  - Language ➔ .NET Pascal
  - Thread ➔ System.Threading, STA/MTA Thread Model
  - MemMgr ➔ New() & Dispose()
  - Exception ➔ System.Exception
  - Other OS Resource ➔ System.xxxxx
  - Debug Support ➔System.Diagnostics

.NET Pascal语言特性

- **这只是一份速食快餐，真正的内容：**

  $(BDS)\Source

  $(BDS)\Help

- Danny Thorpe**提供的**

  《Delphi 8 for .NET Language Enhancements and Portability Guidelines》

最常用的元素得以保留并重新实现

- Strings & Arrays
- Records
- Classes & Interfaces
- Exceptions
- Properties & Events
- Sets
- Text files
- Local Procedures
- Variants
- Components
- Streams
- New, Dispose
- Readln, Writeln
- Format
- Component Streaming
- Random
- Virtual Constructors

让它们从记忆中消失吧

- @, Addr(), Absolute directive
- Real48 six-byte floats
- File of <type>
- GetMem, FreeMem, ReallocMem
  - Use arrays or New() & Dispose()
- ExitProcs
- Old Object syntax (type foo = object)
- TVarData, Variant internals

与CLR不同的实现

Delphi syntax for:

- Virtual constructors
- Named constructors
- Virtual calls from class methods

- Unsafe types
  - PChars
  - Variant records

值类型

- 基元类型是值类型
- 值类型派生自TObject.ValueType
- 记录(record type)是值类型
- 记录可以实现接口
- 奇怪的语法(Boxing)：TObject(12).ToString()

新的语法(语义)元素

- **运算符重载**
- **嵌套类型定义**
- **类静态变量**
- **多投事件**(Multicast Events)

	多重继承	类(静态)字段	垃圾收集	运算符重载	泛型	内联函数
Delphi	通过接口		接口引用计数			
Java		Y	Y			
C++	Y	Y		Y	Y	Y
Delphi .NET	通过接口	Y	Y(CLR)	Y	Y(CLR)	

总是要跟着技术跑的。

只是在J2EE这个老兔子和.NET这个新兔子之间，必须要有个选择而已。

我不知道发展会怎样，但我觉得我只能选择.NET这只兔子。

如果你不仅仅想追着兔子，还想打着兔子，那么我建议选择Delphi .NET。

它是一把好猎枪。因为枪托上刻着：

Made In Borland.