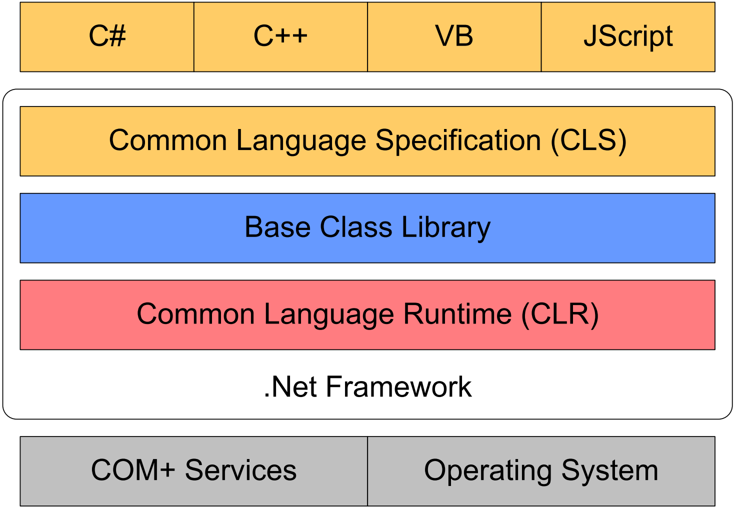
Part IV

.NET

.NET

- combination of frameworks, languages, tools, web services, . . .
  for the easy development of Windows applications
  (including web applications)
- common runtime system CLR (Common Language Runtime):
  abstract/virtual machine for all .NET languages
  (VB, C#, J#, F#, Managed C++, Java, Fortran, Haskell, Perl, . . .)
- based on common intermediate language CIL
  (Common Intermediate Language)
- common type system CTS (Common Type System)

Overview of .NET



C# C++ VB JScript

Common Language Specification (CLS)

Base Class Library

Common Language Runtime (CLR)

.Net Framework

COM+ Services Operating System

Further Properties of .NET

- (immediate) just-in-time compilation (→ security)
- comprehensive OO class library providing collections, threads, reflection, XML processing, Windows forms, web forms, ADO.NET, ASP.NET,. . .)
- versioning (“ending DLL hell”)
- very good support of web services

1. C#

- Java-like OO programming language with
    - (single) inheritance, interfaces
    - generic types (better than in Java, since supported by CLR)
    - garbage collection
    - code verification (e.g. `int` cannot be used as reference)
    - exceptions (need not be caught or declared)
    - reflection
    - iterators, foreach loops
    - assembly: (→ .jar file in Java)
        - collection of classes with manifesto (table of contents) and metadata about interfaces
        - provided as EXE or DLL (in directory or Global Assembly Cache)
        - no entry in Windows registry required

Differences from Java

- (besides reference types) value types (structs, enum); on stack rather than heap
- besides call-by-value also call-by-reference (`T m(ref T2 p){...}`, `o.m(ref a)`)
- multidimensional arrays as block of memory (rather than arrays of arrays)

  e.g. `int[,] a = new int[4,5]; ...   x = a[2,0];`

Differences from Java (2)

- property ≙ attribute with getter and setter, e.g.:

```
class C{
   private int number;
   public int Number{set{number = value;}
                     get{return number}}
}
C o; ...~ o.Number = o.Number + 1;
```

- namespace (≙ `package`) can be distributed among several directories
- partial classes
- a file may comprise several namespaces and classes
- typing rules can be ignored in unsafe program parts

Differences from Java (3)

- delegates and lambda expressions
- objects also on the stack (rather than on the heap only)
- (restricted) goto
- overridable methods declared as `virtual`
  (overriding methods as `override`, hiding ones as `new`)
- extension methods
- `sealed` classes have no subclasses (→ `final` in Java)

2. ASP.NET

- for dynamic generation of web pages with user interaction via forms
- Webform: HTML page with integrated controls
- is transformed into pure HTML with hidden form fields and server side code for event processing
- not only submitting a form but also using a control may cause an event, which is processed by the server
- filling a form and processing it may take several HTTP round trips between browser and web server

ASP.NET (continued)

- code behind: in a web form, one can use code, which is provided by the superclass of the class corresponding to the web form
- thus: HTML (web form) and C# code are cleanly separated
- difference from JSP: a web form describes the current web page (involved in round trips) rather than the next one
- for changing to another web page, a link or redirection is used
- master pages enable a uniform layout of a web site
- a session attribute stores information during a session (→ JSP)
- cookies can be set and read
- validators allow to check user input (internally translated to client side JavaScript)

Example: Web Form

```
<%@ Page Language="C#" AutoEventWireup="true"
  CodeBehind="LibLogin.aspx.cs" Inherits="Library.LibLogin" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
  <title>Welcome to the Library</title>
</head>
<body>
  <h1>Welcome to the Library</h1>
  <form id="form1" runat="server">
  <div>
    <asp:Label ID="PasswordLabel" runat="server" Text="Password"></asp:Label>
    <asp:TextBox ID="Password" TextMode="Password" runat="server"></asp:TextBox></br>
    <asp:RequiredFieldValidator ID="val" ControlToValidate="Password"
         ErrorMessage="Insert your password!" runat="server"/>
    <asp:Button ID="Ok" runat="server" Text="Ok" OnClick="Login"/>
  </div>
  </form>
</body>
</html>
```

HTML Page Transmitted to Browser (without Validator)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head><title>Welcome to the Library</title></head>
<body>
  <h1>Welcome to the Library</h1>
  <form name="form1" method="post" action="LibLogin.aspx" id="form1">
    <div>
    <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
      value="/wEPDwUKMTIxNDIyOTM0MmRkBODT+ziyM71eMrjNc9ap6hwWoCg=" />
    </div>
    <div>
       <span id="PasswordLabel">Password</span>
       <input name="Password" type="text" id="Password" /></br>
       <input type="submit" name="Ok" value="Ok" id="Ok" />
    </div>
    <div>
      <input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION"
        value="/wEWAwKclIP9BgLSxaDECgLh777vDG/iFwXGlUrZVFnVqg/27x7y75wb" />
    </div>
  </form>
</body>
</html>
```

Example: Corresponding Code Behind

```
using System;                      using System.Data;
...
using System.Web.UI.HtmlControls;  using Library.Web;

namespace Library{
  public partial class LibLogin : System.Web.UI.Page {

    public void Login(object sender, EventArgs ev){
        string passw = Password.Text;
        if (passw != "secret123"){
          Context.Session.Add("Login",false);
          Password.Text = "erroneous input";}
        else {Context.Session.Add("Login",true);
          Response.Redirect("Web/Library.aspx");}}
  }
}
```

- an additional partial class `LibLogin` is automatically generated
- here variables for accessing form fields such as `Password` are declared

Example 2: Web Form “Create User”

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="UserCreateWeb.aspx.cs"
  Inherits="Library.Web.UserCreateWeb" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" ...>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server"><title>Create User</title></head>
<body>
<h1>User Create</h1>
<form id="form1" runat="server">
<div>
  <asp:Label ID="NameLabel" runat="server" Text="Name"></asp:Label>
  <asp:TextBox ID="Uname" runat="server"></asp:TextBox><br/>
  <asp:Label ID="AddressLabel" runat="server" Text="Address"></asp:Label>
  <asp:TextBox ID="Uaddress" runat="server"></asp:TextBox><br/>
  <asp:Button ID="Ok" runat="server" Text="Ok" OnClick="Create"/>
</div>
</form>
<a href="UserManagementWeb.aspx">back</a>
</body>
</html>
```

Example: Code Behind for “Create User”

```
using Library.BusinessLogic; ...
namespace Library.Web {
  public partial class CreateUserWeb : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e) {
      if ((Context.Session["Login"] == null) ||
            (Convert.ToBoolean(Context.Session["Login"]) == false))
          Response.Redirect("LibLogin.aspx");}

    public void Create(object sender, EventArgs ev) {
      string name = Uname.Text;
      string address = Uaddress.Text;
      if ((name == null) || (name == ""))
         Response.Redirect("Error.htm");
      try{ UserManagement um = new UserManagement();
           um.CreateUser(name,address);
           Response.Write("User "+name+" created!");
           Uname.Text = "";  Uaddress.Text = "";}
      catch(Exception){Response.Redirect("Error.htm");}}
  }
}
```

3. ADO.NET

- framework for accessing databases and other data sources
- DB access with or without connection
- Steps for connection-oriented access:
  - establish connection to DB
  - repeat:
    - create SQL-statement with placeholders (`@placeholderName`)
    - fill placeholder with values,

      e.g. `AddWithValue("@placeholderName",value)`
    - execute SQL statement

      (`ExecuteScalar`, `ExecuteNonQuery`, `ExecuteReader`)
    - process results (possibly in a loop)
  - close connection to DB

Example: Business Logic with ADO.NET

```
using System.Data.SqlClient; ...
namespace Library.BusinessLogic{
  public class UserManagement{
    public void CreateUser(string name, string address){
      SqlConnection connection =
        new SqlConnection(ConfigurationManager.ConnectionStrings["Library"].ConnectionString);
      connection.Open();
      string sqltext = "SELECT COUNT(*) FROM User WHERE Name = @name";
        SqlCommand sql = new SqlCommand(sqltext, connection);
        sql.Parameters.AddWithValue("@name", name);
        int number = (int) sql.ExecuteScalar();
        if (number > 0)  throw new Exception("Name exists");
        else{ sqltext = "INSERT INTO User (Name, Address)"
                      + "VALUES (@name, @address);"
                      + "SELECT CAST(SCOPE_IDENTITY() AS INT);";
                sql = new SqlCommand(sqltext, connection);
                sql.Parameters.AddWithValue("@name", name);
                sql.Parameters.AddWithValue("@address", address);
                int uid = (int) sql.ExecuteScalar();}
        connection.Close();}}
}
```

Example 2: Processing Set-valued Queries (1)

```
public string ShowLoan(int clientNo) {
  SqlConnection connection =
    new SqlConnection(ConfigurationManager.
                        ConnectionStrings["Library"].ConnectionString);
  try { connection.Open(); }
  catch (Exception e) { Console.Write(e); }
  string sqltext =
     "SELECT Name, Uid, Title, InventoryNo, Date"
   + "FROM User U, Loan L, Copy C, Medium M"
   + "WHERE U.Uid = @uid AND U.Uid = L.User AND L.Copy = C.InventoryNo"
   + "      AND C.Medium = M.Id AND M.MediaType = 0;"
   + "SELECT Name, Uid, Title, InventoryNo, Date"
   + "FROM User U, Loan L, Copy C, Medium M"
   + "WHERE U.Uid = @uid AND U.Uid = L.User AND L.Copy = C.InventoryNo"
   + "      AND C.Medium = M.Id AND MediaType = 1";
  SqlCommand sql = new SqlCommand(sqltext, connection);
  sql.Parameters.AddWithValue("@uid", clientNo);
  SqlDataReader result = sql.ExecuteReader();  ...
```

Example 2: Processing Set-valued Queries (continued)

```
  string[] mediatext = { "Books", "CDs" };
  string output = "";  int i = 0;
  do  {
     output += "<h1>Loaned " + mediatext[i] + "</h1>";
     output += "<Table border="1">";
     output += "<tr><th>Name</th><th>UserNo</th><th>Title</th>";
     output += "<th>InventoryNo</th><th>Date</th></tr>";
    while (result.Read()) {
       object[] values = new object[result.FieldCount];
       result.GetSqlValues(values);
       output += "<tr>";
       foreach (object value in values)
         output += "<td>"+value.ToString()+"</td>";
       output += "</tr>";}
     output += "</Table>";
     i++;}
   while (result.NextResult());
  connection.Close();
  return output;
}
```

Remarks about the Examples

- rather than using SQL directly from the business logic one can interpose an OR-mapping layer (→ NHibernate)
- in example 2, presentation logic and business logic are mixed-up for simplicity
- much cleaner:
  - fill a data structure (DTO) with the result and deliver it to the presentation layer
  - only the presentation layer generates HTML
- transaction handling is still missing (see below)

Transaction Processing

- Steps: `BeginTransaction`, assign each `SqlCommand` to the t., `Commit` or `Rollback`
- local or distributed transactions with given isolation level
- Isolation Levels:
  - `ReadUncommitted`: no protection, locks ignored
  - `ReadCommitted`:
    - tuples can only be read after commit
    - the corresponding tables can change during the transaction
    - repeated reads of a table may produce different results
  - `ReadRepeatable`: repeated queries deliver the same results
  - `Serializable`
  - trade-off between effiziency and undesired effects

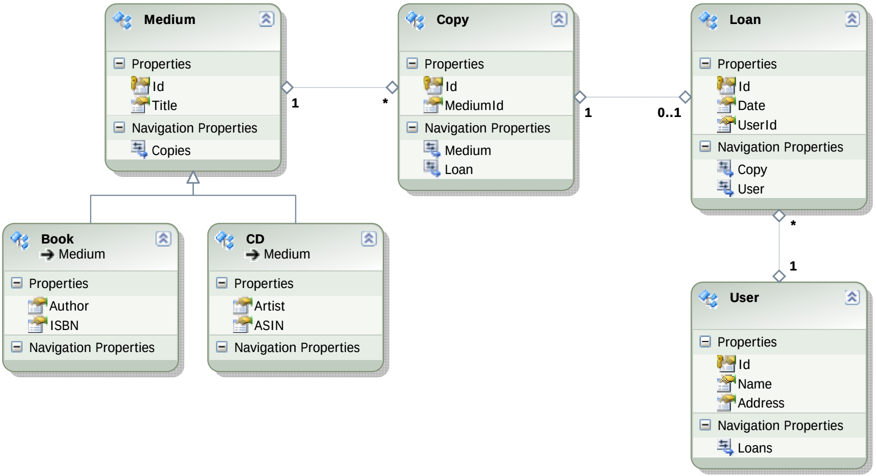
Example: Transaction Processing with ADO.NET

```
public void CreateUser(string name, string address){
  SqlConnection connection =
    new SqlConnection(ConfigurationManager.ConnectionStrings["Library"].ConnectionString);
  SqlTransaction trans = null;
  try {
    connection.Open();
    trans = connection.BeginTransaction(IsolationLevel.ReadCommitted);
    string sqltext = "SELECT COUNT(*) FROM User WHERE Name = @name";
    SqlCommand sql = new SqlCommand(sqltext, connection);
    sql.Parameters.AddWithValue("@name", name);
    sql.Transaction = trans;
    int number = (int) sql.ExecuteScalar();
    if (number > 0)  throw new Exception("Name exists");
    else{ sqltext = "INSERT INTO User (Name, Address) VALUES (@name, @address);"
                  + "SELECT CAST(SCOPE_IDENTITY() AS INT);";
          sql = new SqlCommand(sqltext, connection);
          sql.Parameters.AddWithValue("@name", name);
          sql.Parameters.AddWithValue("@address", address);
          sql.Transaction = trans;
          int bid = (int) sql.ExecuteScalar();
          trans.Commit();}}
  catch (Exception e) {if (trans != null) trans.Rollback();
                       Console.WriteLine(e);}
  finally {connection.Close();}
}
```

4. Entity Framework and LINQ

- ADO.NET Entity Framework: object-relational mapping
- inheritance strategies: TablePerType (default), TablePerConcreteType, TablePerHierarchy
- description of Entity Data Model in XML (graphical designer available)
- entities provided as OO classes
- mapping to storage schema (database)
- approaches: database-first, model-first (, code-first)
- queries (e.g.) with LINQ

Entity Data Model of Library



Example: Entity Framework

```
public User CreateUser(string name, string address){
  using (LibraryContainer db = new LibraryContainer()){
    User newUser = new User{
      Name = name,
      Address = address
    };
    db.UserSet.AddObject(newUser);
    db.SaveChanges();
    return newUser;
  }
}
```

LINQ

- type-safe queries over collections (IEnumerable) and other data sources (IQueryable))
- realized using extension methods, lambda expr. and closures
- all data sources with corresponding LINQ-provider can be handled analogously (rel. DB, XML, LDAP, collections, . . .)
- syntactic sugar: query expressions (`from ...where ...select ...`)
- additional syntax extends C# (and VB)
- thus: syntax errors in queries detected at compile time
- collections can be processed in parallel with `AsParallel()`

Example: LINQ

```
string name = "Bill";  var count;
```

Alternative formulations of equivalent queries:

1) method syntax:

a) `count = db.UserSet.Where(u => u.Name ==name).Count();`

b) `count = db.UserSet.Count(u => u.Name == name);`

2) query expression syntax: (here: mixed with method syntax)

```
count = (from u in db.UserSet
                  where u.Name == name
                  select u).Count();
```

Query Expression Syntax in EBNF

```
⟨query⟩   ::=
          from ⟨id⟩ in ⟨source⟩
          { from ⟨id⟩ in ⟨source⟩ |
          join ⟨id⟩ in ⟨source⟩ on ⟨expr⟩ equals ⟨expr⟩ [ into ⟨id⟩ ] |
          let ⟨id⟩ = ⟨expr⟩ |
          where ⟨condition⟩ |
          orderby ⟨ordering⟩ ( , ⟨ordering⟩*) }
          select ⟨expr⟩ | group ⟨expr⟩ by ⟨key⟩
          [ into ⟨id⟩ ⟨query⟩ ]
```

Selected LINQ-Commands (1)

Operation	Query Syntax	Semantics	in SQL
all		all elements fufill condition?	-
any		any element fulfills condition?	EXISTS
average		average	AVG
concat		union	UNION
contains		contains?	IN
count		# elements	COUNT
distinct		removes duplicates	DISTINCT
except		set difference	-
first		first element	-
groupby	group . . . by . . . into . . .	groups according to criterion	GROUP BY
intersect		intersection	-
join	join . . . in . . . on . . . equals . . . into . . .	(inner) join	JOIN
last		last element	-

Selected LINQ-Commands (2)

Operation	Query Syntax	Semantics	in SQL
max		maximal element	MAX
min		minimal element	MIN
orderby	orderby	sort in ascending order	ORDER BY
" descending	orderby . . . descending	sort in descending order	ORDER BY DESC
select	select	selects attributes	SELECT
sum		sum	SUM
take		first k elements	TOP
union		union	UNION
where	where	filters according to condition	WHERE