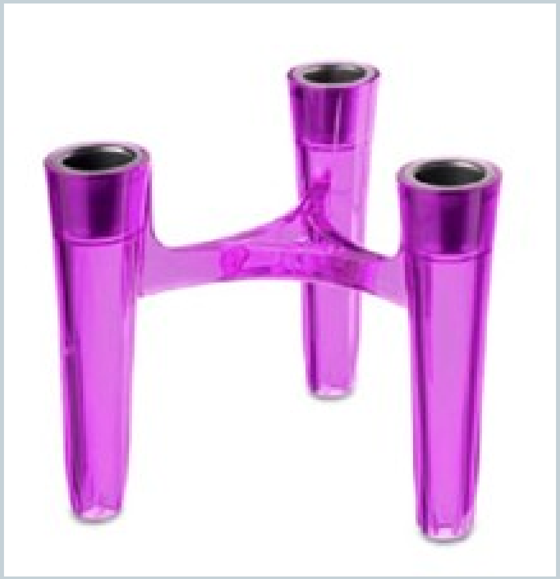
n-Tier Architecture

YANIV GATIGNO

DEC 2008

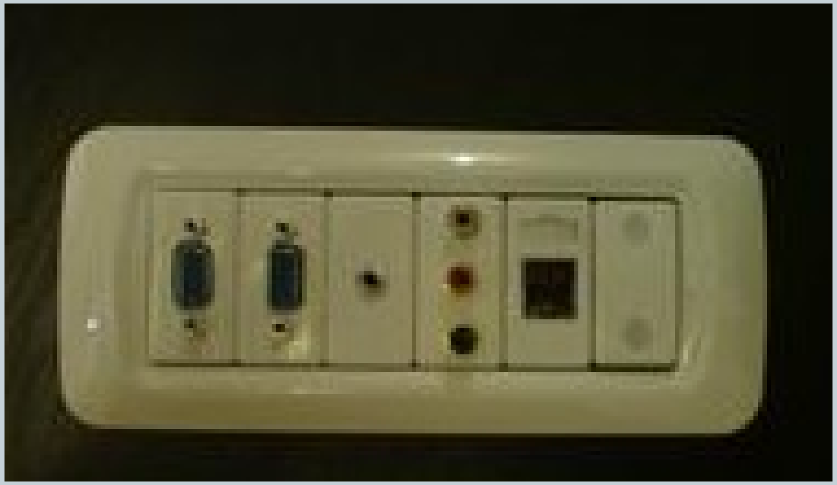
Intro

Modular X



Modular X

Modular X



Buzz Words

- Abstraction
- Modularity

Trade-Off

- Performance
- Complexity

Definition

Definition

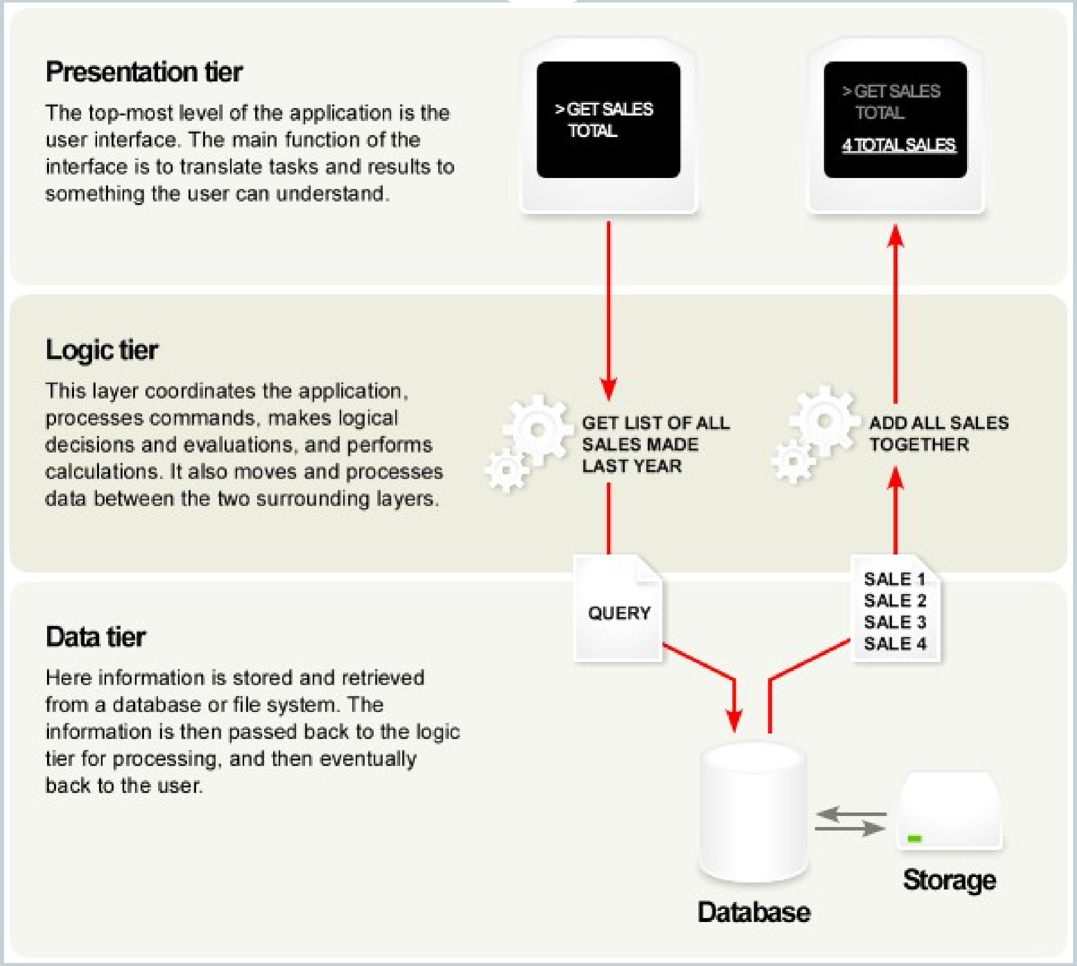
“

A CLIENT-SERVER ARCHITECTURE IN WHICH, THE PRESENTATION, THE APPLICATION PROCESSING AND THE DATA MANAGEMENT ARE LOGICALLY SEPARATE PROCESSES.

“

(WIKIPEDIA)

3-Tier Architecture



Presentation tier

The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

Logic tier

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Data tier

Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.

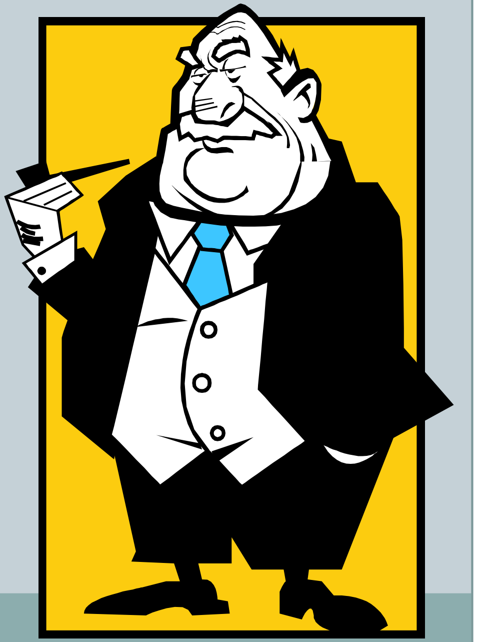
Example

Example

MY VACATION WITH MR NORTHWIDH

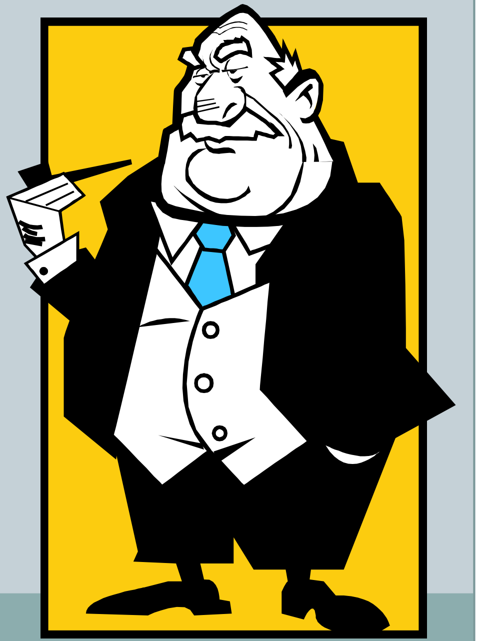
Northwind Management System Reqs

"I have a $200M/year trade company, but I want to start by managing my products list only"



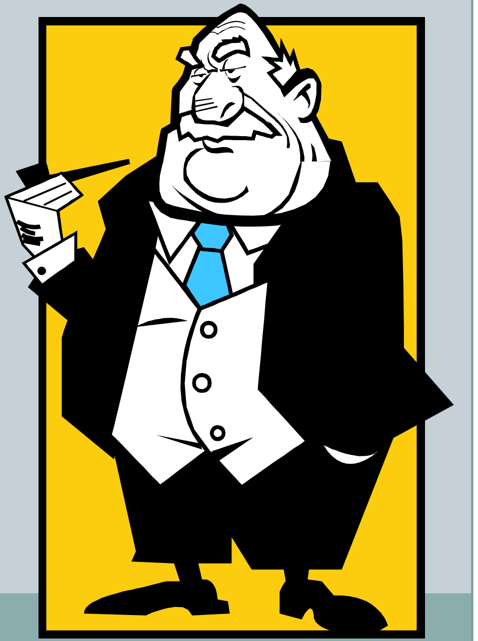
Northwind Management System Reqs

- Variants:
  - Scale
  - Predicted Ops over Data
  - Performance
  - Platform
  - Price



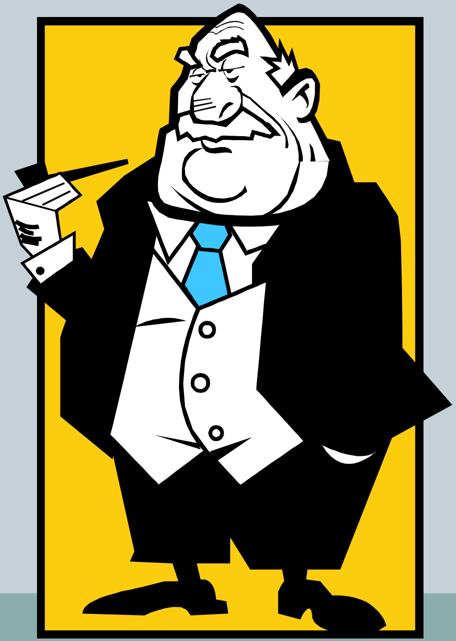
Northwind Management System Reqs

Q: How far can you see?



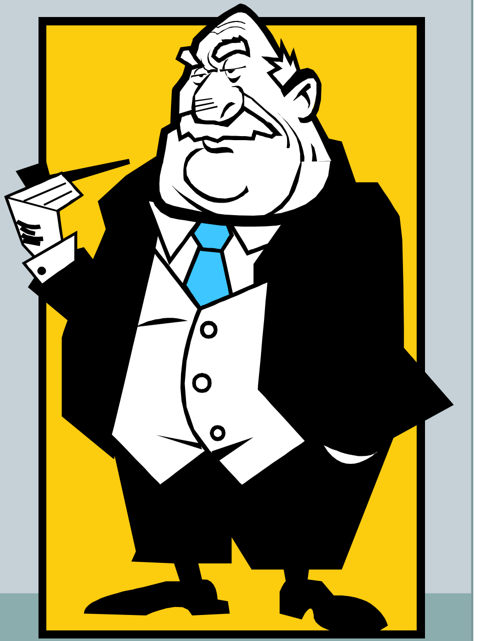
Northwind Management System Reqs

- Variants:
  - Scale
    - **Big**
  - Predicted Op over Data
    - **Mostly Viewing**
  - Performance
    - **DB Access is slow**
  - Platform
    - **Interchangeable**
  - Price
    - **Open Cheque**

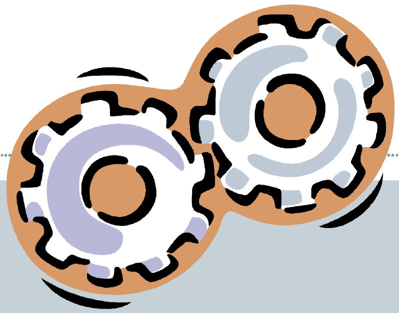


Northwind Management System Reqs

- 3-Tier

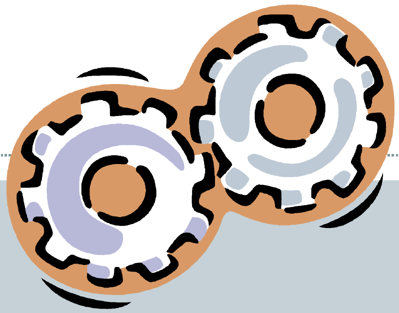


Specifications



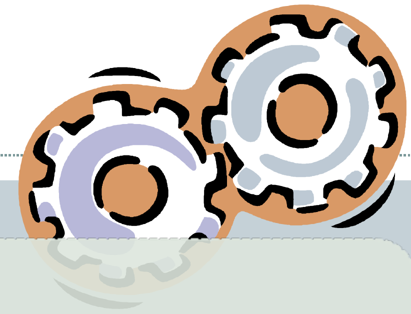
- User Scenarios
  - View all existing products
  - Add a new product
  - (Products are never removed)

Specifications



- System Modules:
  - Let’s do it together…

Specifications



Presentation Tier

- WinForm (/ WebForm)
- Handle Product **Objects**
- Communicates with BL over **SOAP**

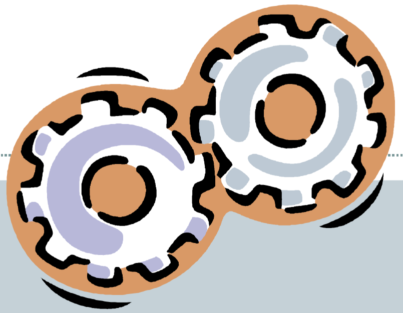
Business Logic Tier

- Web Service
- Exposes **Interface**
- Holds **Dataset**
- Converts Product <-> DataRow
- Communicates with Data Tier using **ADO.NET**

Data Tier

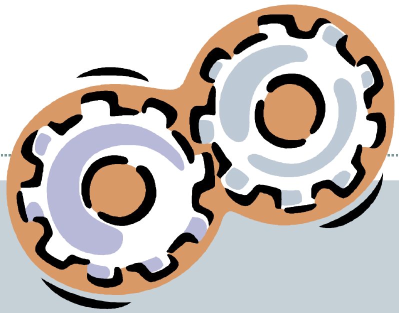
- Northwind DB
- Interchangeable Platform

Specifications



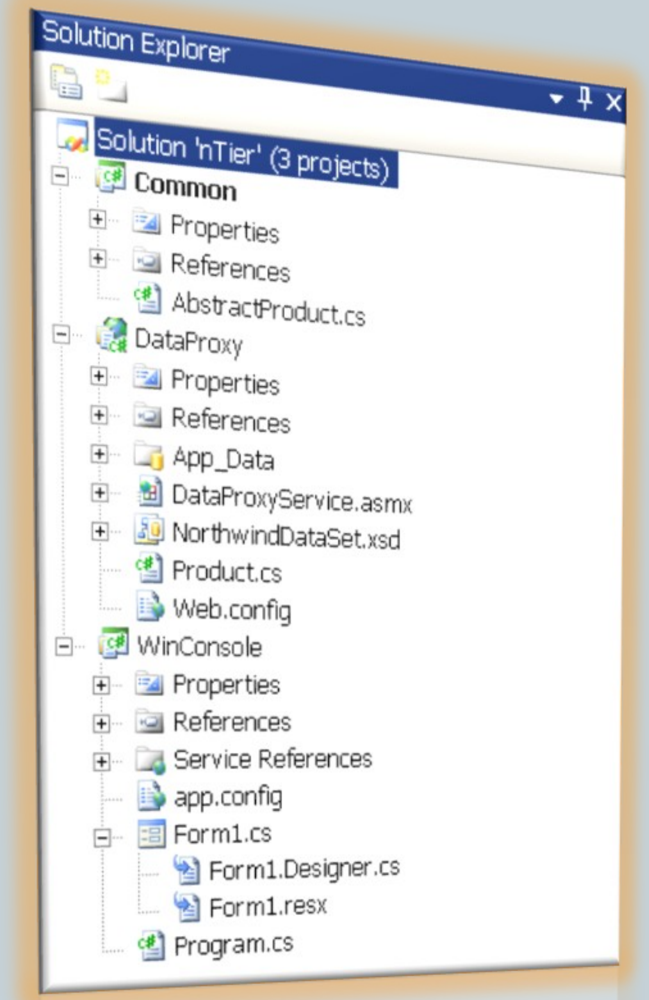
- Notes:
  - All application consoles interact with a single WebService
    - 'Proxy' Design Pattern
  - Presentation & BL are seperated
    - Winforms can be easily replaced by WebForm
  - Presentation & BL communicate via an Interface
    - BL implementation may be changed without touching the Presentation

Specifications



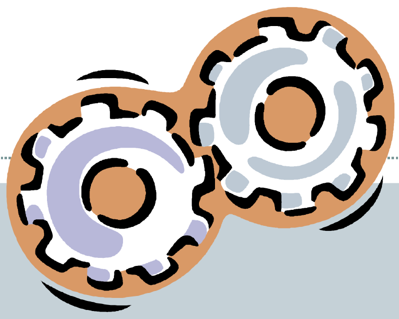
- Notes:
  - Presentation works with objects
    - It doesn't care about the source of the data or its type
  - Data is cached
    - Using the Typed DataSet
    - Presentation doesn't even know about it
  - BL communicates Data using ADO
    - DB can be replaced with hardly any changes in the BL

Specifications

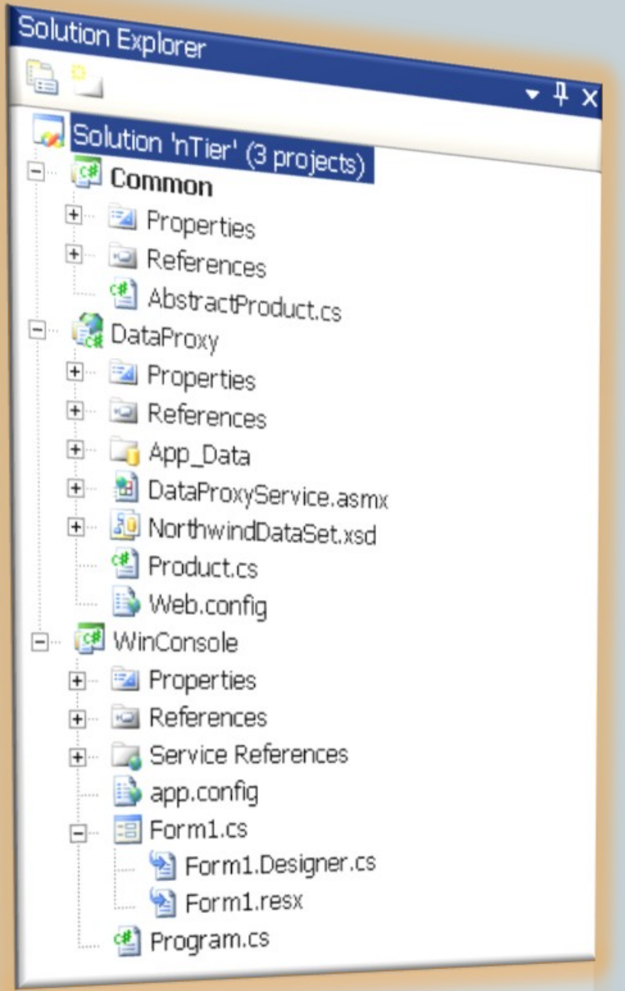


Solution Explorer

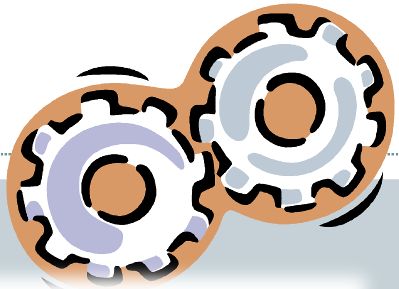
```
Solution 'nTier' (3 projects)
  Common
    Properties
    References
    AbstractProduct.cs
  DataProxy
    Properties
    References
    App_Data
    DataProxyService.asmx
    NorthwindDataSet.xsd
    Product.cs
    Web.config
  WinConsole
    Properties
    References
    Service References
    app.config
    Form1.cs
      Form1.Designer.cs
      Form1.resx
    Program.cs
```

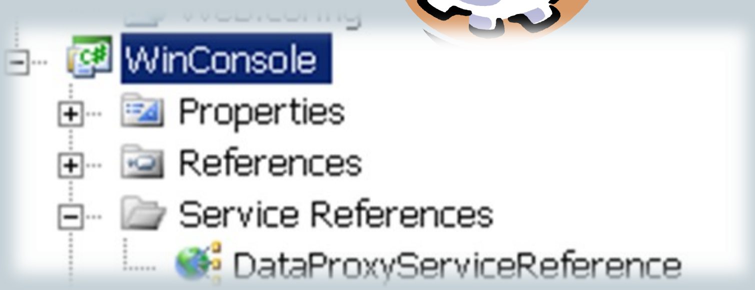


Specifications

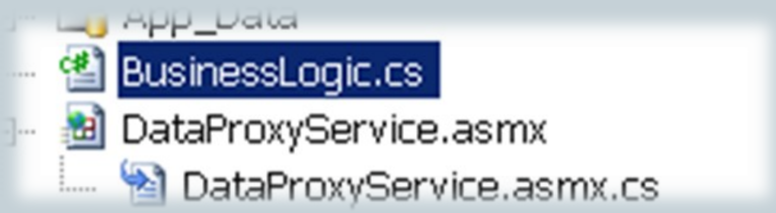


```
Solution Explorer
Solution 'nTier' (3 projects)
  Common
    Properties
    References
    AbstractProduct.cs
  DataProxy
    Properties
    References
    App_Data
    DataProxyService.asmx
    NorthwindDataSet.xsd
    Product.cs
    Web.config
  WinConsole
    Properties
    References
    Service References
    app.config
    Form1.cs
      Form1.Designer.cs
      Form1.resx
    Program.cs
```

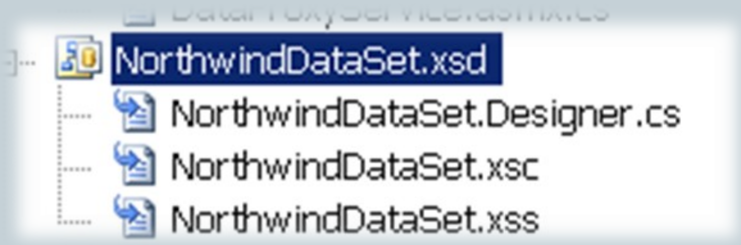




```
WinConsole
  Properties
  References
  Service References
    DataProxyServiceReference
```



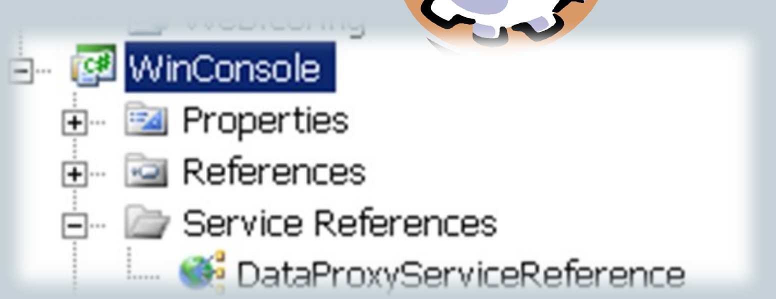
```
BusinessLogic.cs
DataProxyService.asmx
  DataProxyService.asmx.cs
```



```
NorthwindDataSet.xsd
  NorthwindDataSet.Designer.cs
  NorthwindDataSet.xsc
  NorthwindDataSet.xss
```

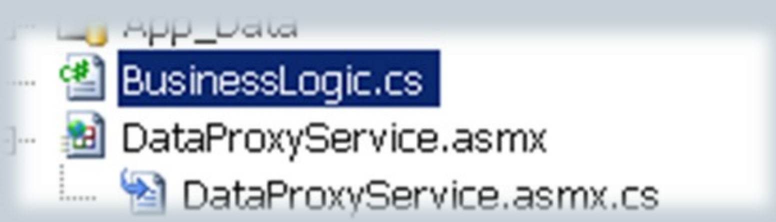
Specifications

Presentation Tier



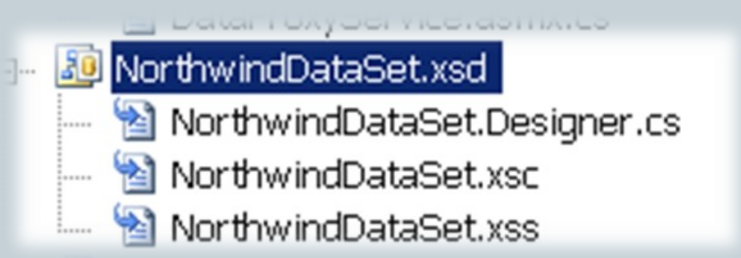
```
WinConsole
  Properties
  References
  Service References
    DataProxyServiceReference
```

Business Logic Tier



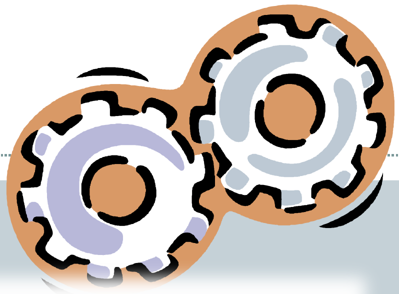
```
App_Data
BusinessLogic.cs
DataProxyService.asmx
  DataProxyService.asmx.cs
```

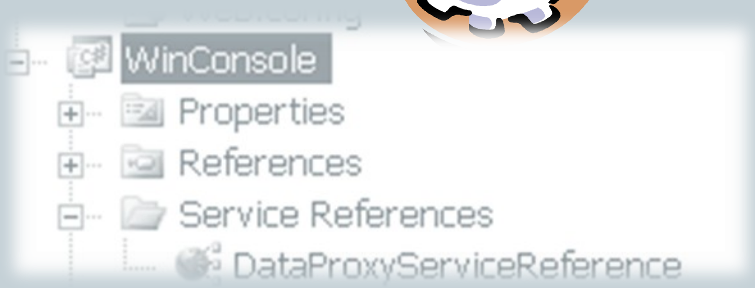
Data Tier

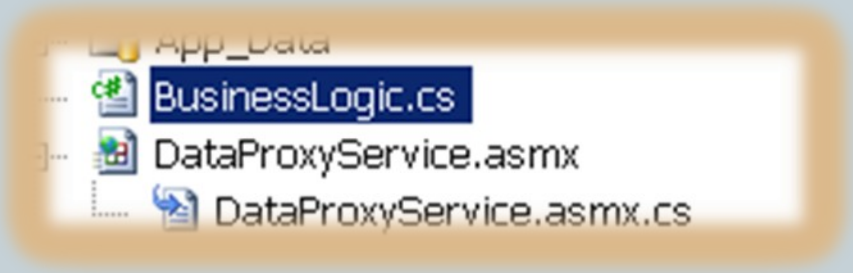


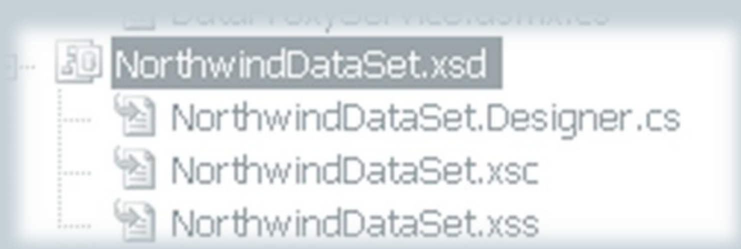
```
NorthwindDataSet.xsd
  NorthwindDataSet.Designer.cs
  NorthwindDataSet.xsc
  NorthwindDataSet.xss
```

Specifications

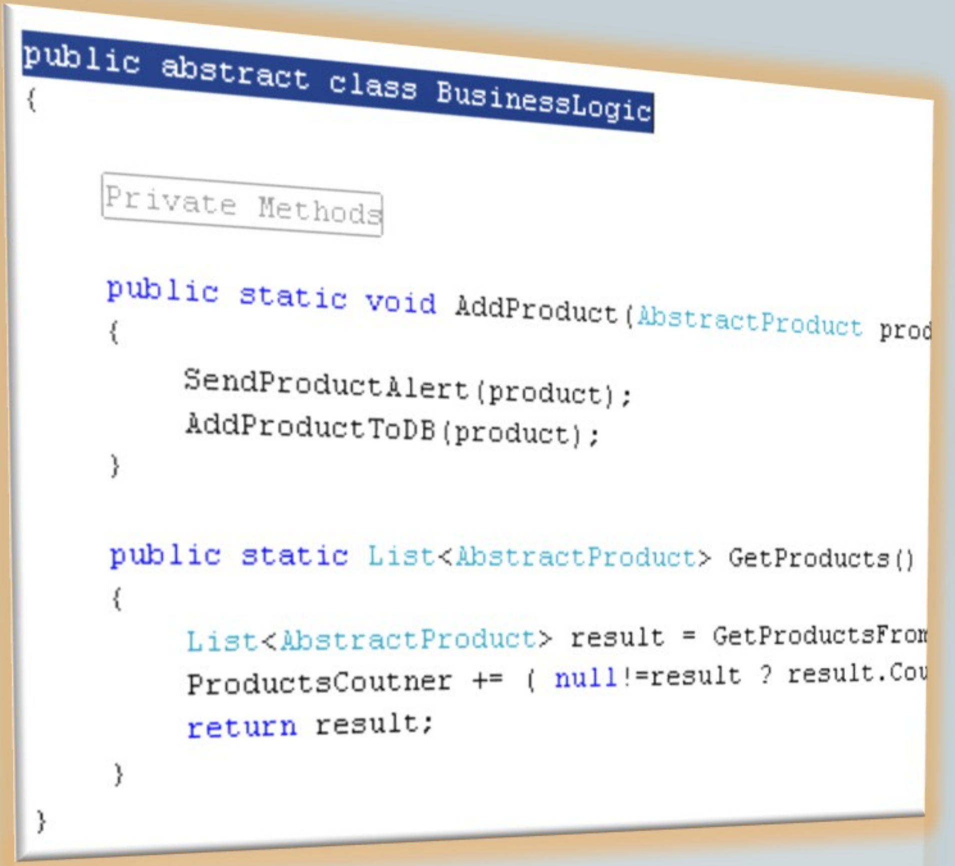



WinConsole
Properties
References
Service References
DataProxyServiceReference

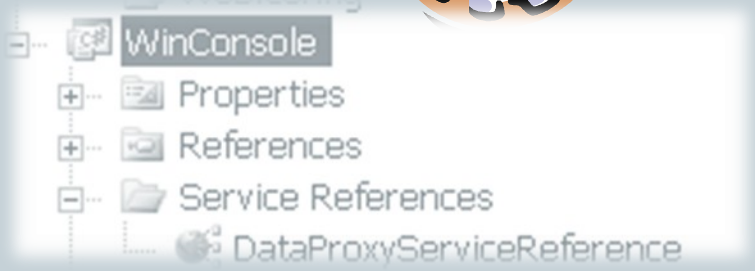

App_Data
BusinessLogic.cs
DataProxyService.asmx
DataProxyService.asmx.cs


NorthwindDataSet.xsd
NorthwindDataSet.Designer.cs
NorthwindDataSet.xsc
NorthwindDataSet.xss

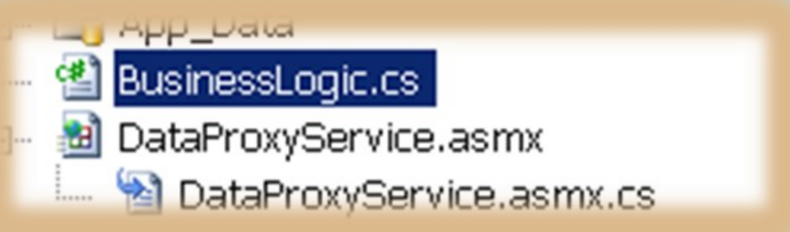
Specifications



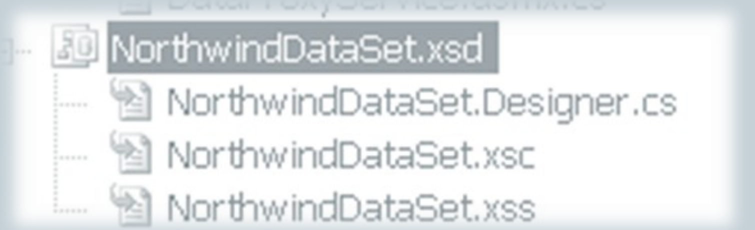
```
public abstract class BusinessLogic
{

    Private Methods

    public static void AddProduct(AbstractProduct prod
    {
        SendProductAlert(product);
        AddProductToDB(product);
    }

    public static List<AbstractProduct> GetProducts()
    {
        List<AbstractProduct> result = GetProductsFrom
        ProductsCoutner += ( null!=result ? result.Cou
        return result;
    }
}
```



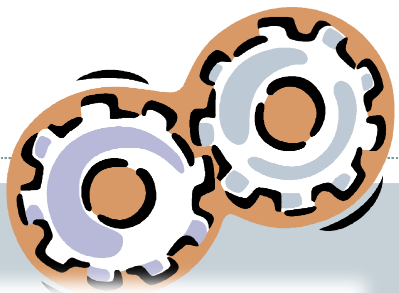
```
WinConsole
    Properties
    References
    Service References
        DataProxyServiceReference
```



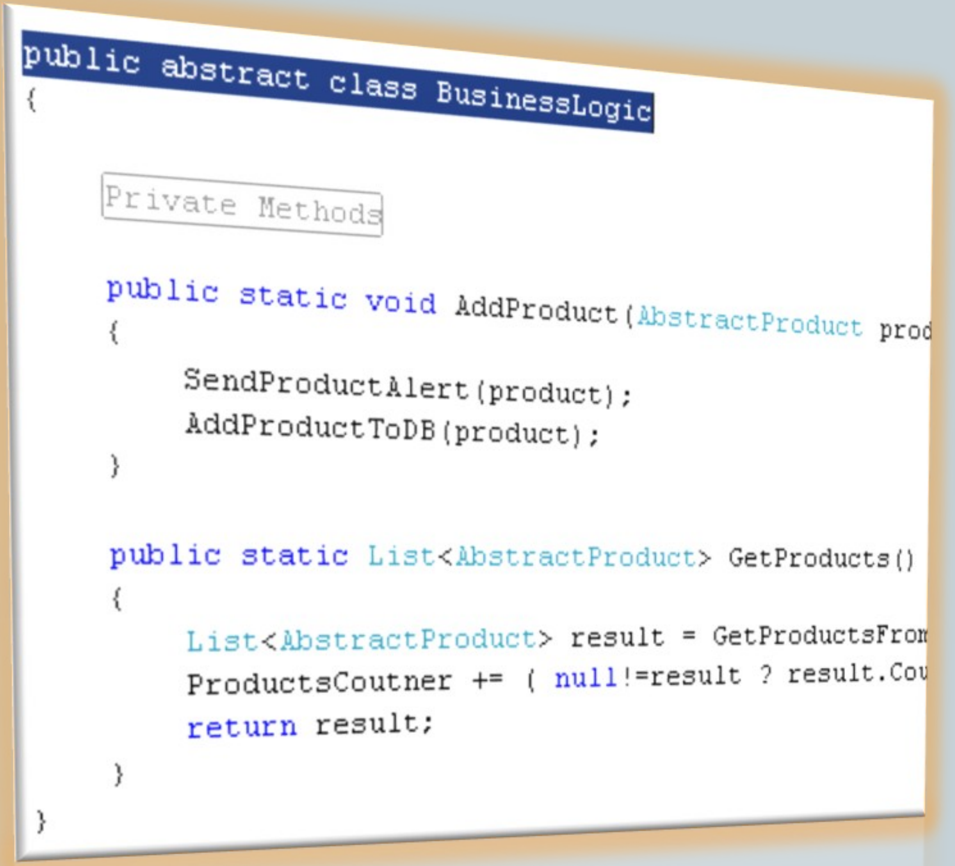
```
App_Data
BusinessLogic.cs
DataProxyService.asmx
    DataProxyService.asmx.cs
```



```
NorthwindDataSet.xsd
    NorthwindDataSet.Designer.cs
    NorthwindDataSet.xsc
    NorthwindDataSet.xss
```

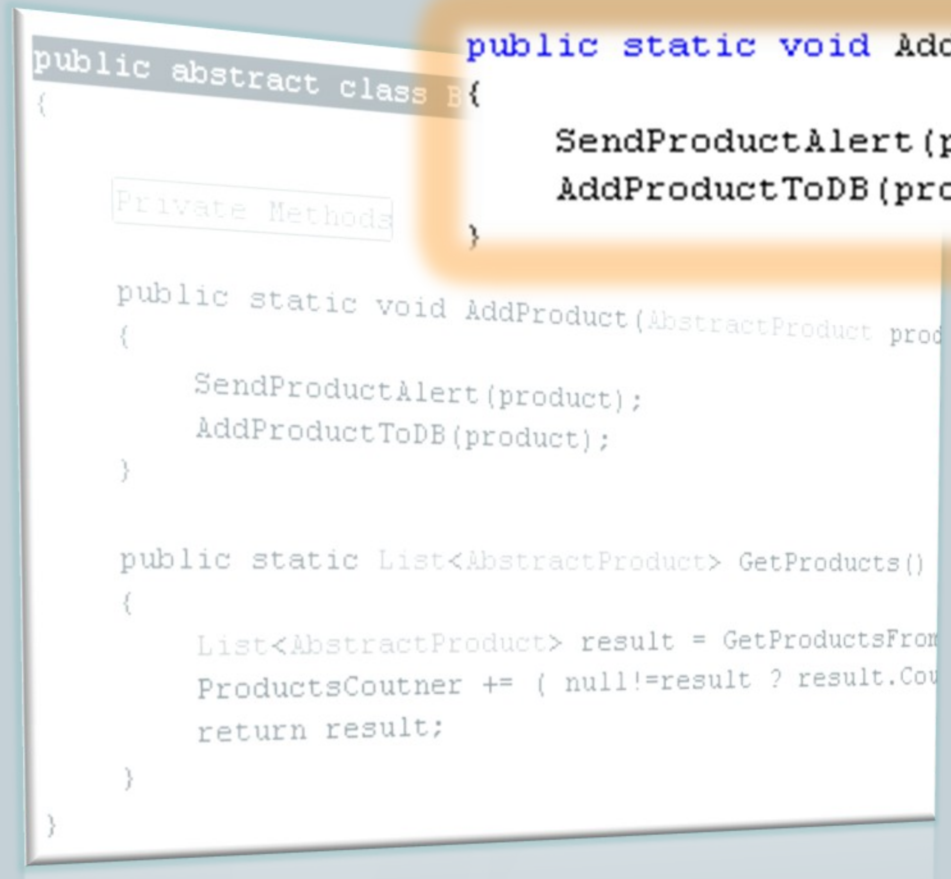


Specifications

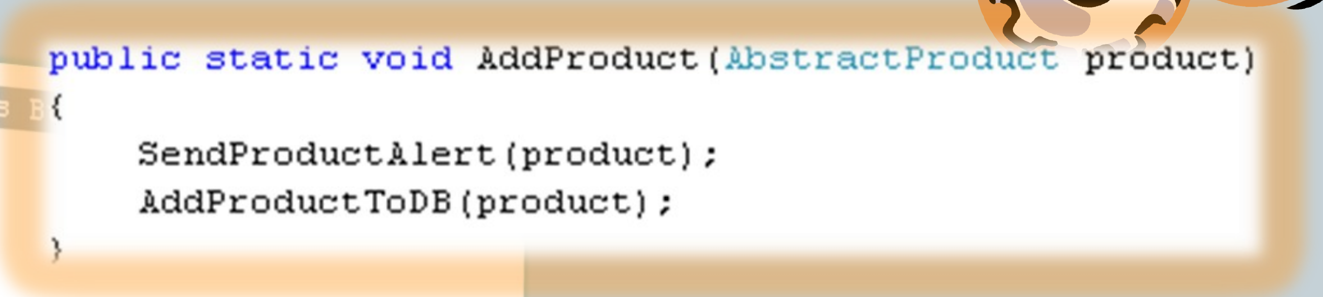


```
public abstract class BusinessLogic
{
    Private Methods

    public static void AddProduct(AbstractProduct prod
    {
        SendProductAlert(product);
        AddProductToDB(product);
    }

    public static List<AbstractProduct> GetProducts()
    {
        List<AbstractProduct> result = GetProductsFrom
        ProductsCoutner += ( null!=result ? result.Cou
        return result;
    }
}
```

Specifications

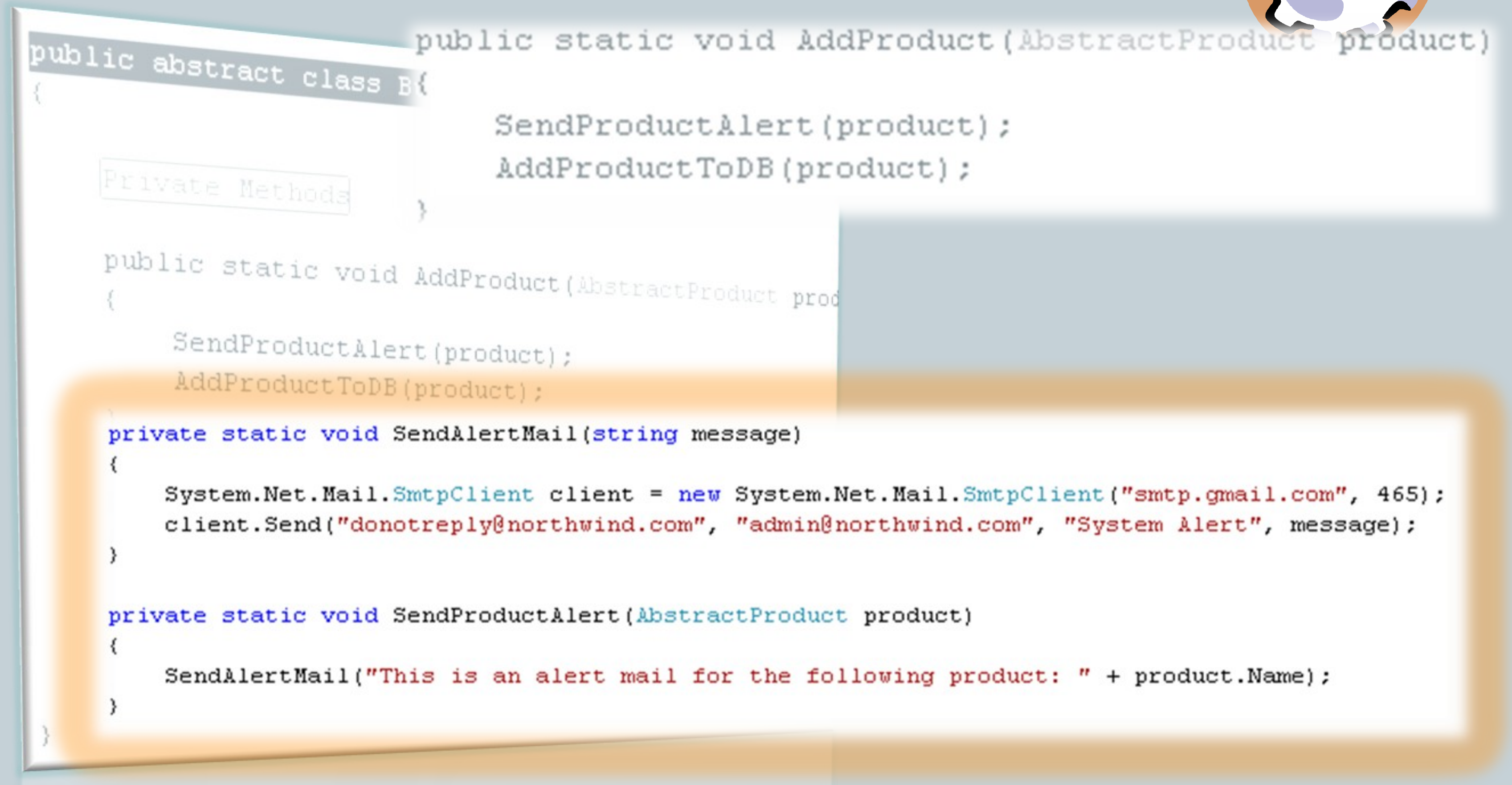


```
public abstract class B
{

    Private Methods

    public static void AddProduct(AbstractProduct prod
    {
        SendProductAlert(product);
        AddProductToDB(product);
    }

    public static List<AbstractProduct> GetProducts()
    {
        List<AbstractProduct> result = GetProductsFrom
        ProductsCoutner += ( null!=result ? result.Cou
        return result;
    }
}
```



```
public static void AddProduct(AbstractProduct product)
{
    SendProductAlert(product);
    AddProductToDB(product);
}
```

Specifications

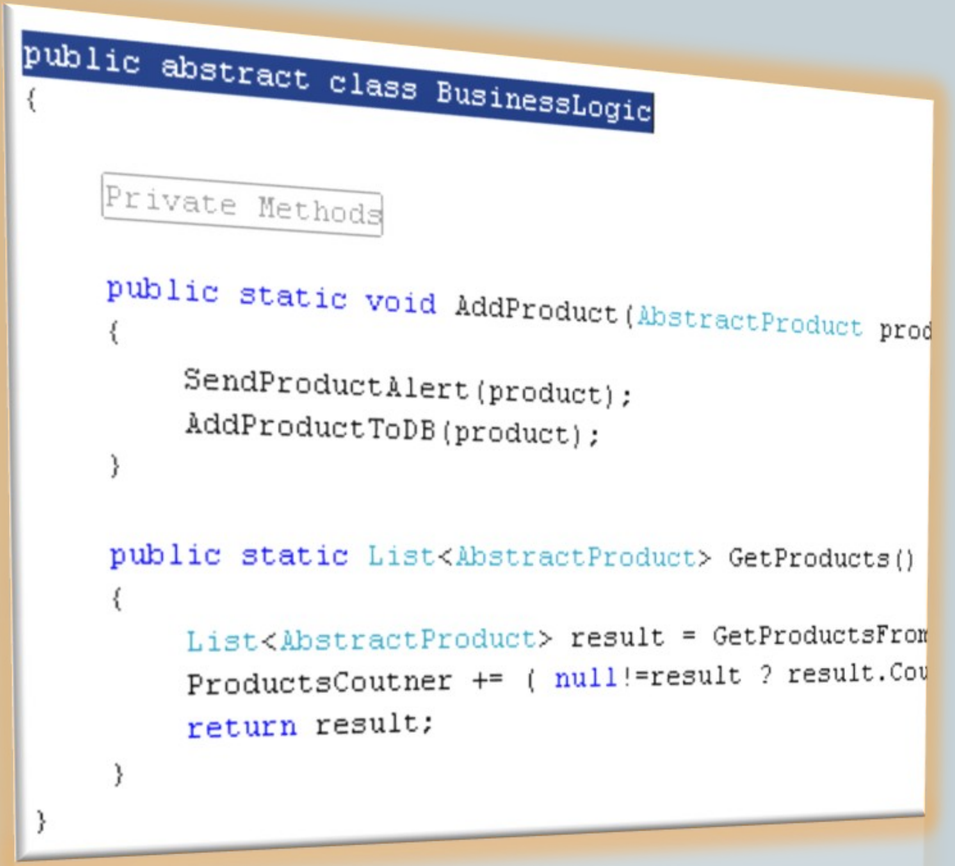


```
public abstract class B{
{

    Private Methods

    public static void AddProduct(AbstractProduct prod
    {
        SendProductAlert(product);
        AddProductToDB(product);
```

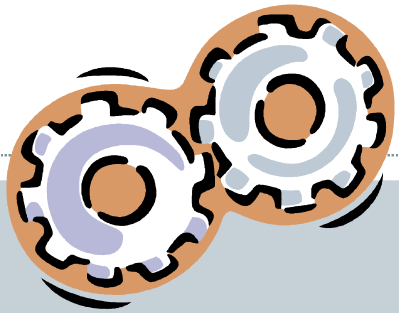
```
public static void AddProduct(AbstractProduct product)

    SendProductAlert(product);
    AddProductToDB(product);
}
```

```
private static void SendAlertMail(string message)
{
    System.Net.Mail.SmtpClient client = new System.Net.Mail.SmtpClient("smtp.gmail.com", 465);
    client.Send("donotreply@northwind.com", "admin@northwind.com", "System Alert", message);
}

private static void SendProductAlert(AbstractProduct product)
{
    SendAlertMail("This is an alert mail for the following product: " + product.Name);
}
}
```

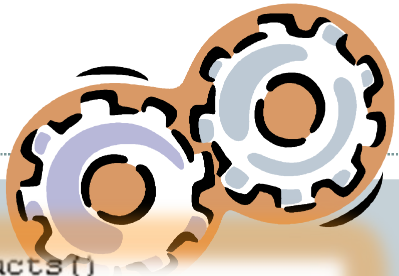
Specifications

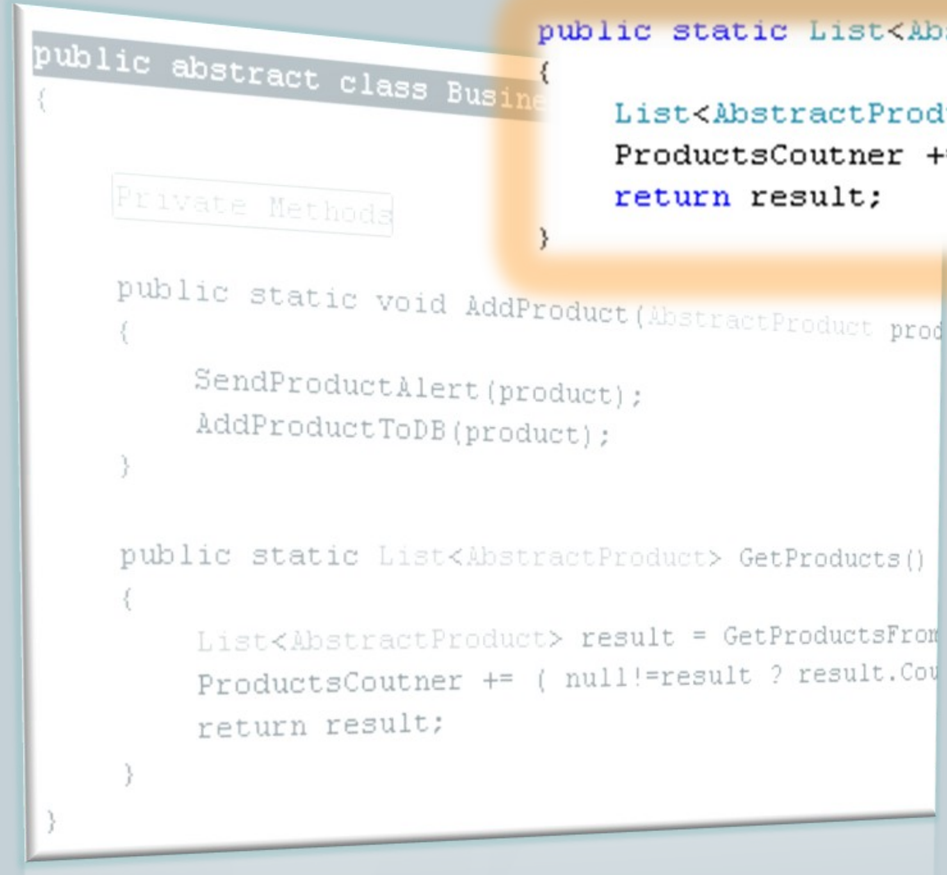


```
public abstract class BusinessLogic
{

    Private Methods

    public static void AddProduct(AbstractProduct prod
    {
        SendProductAlert(product);
        AddProductToDB(product);
    }

    public static List<AbstractProduct> GetProducts()
    {
        List<AbstractProduct> result = GetProductsFrom
        ProductsCoutner += ( null!=result ? result.Cou
        return result;
    }
}
```

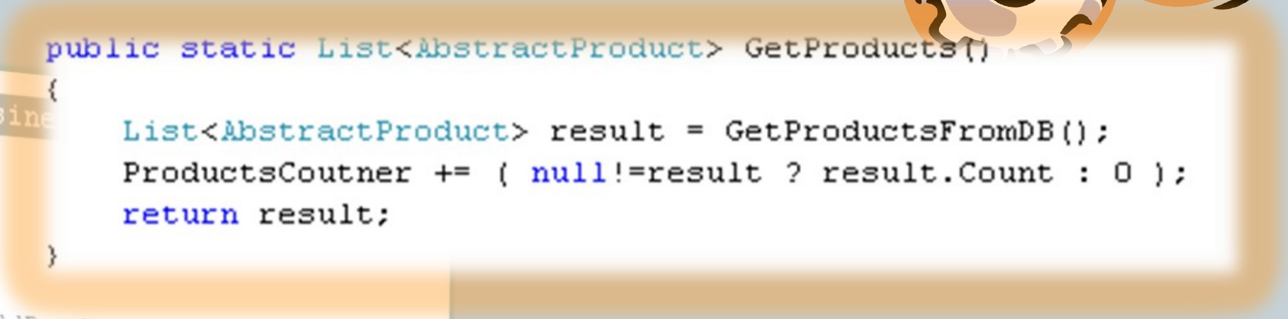


Specifications



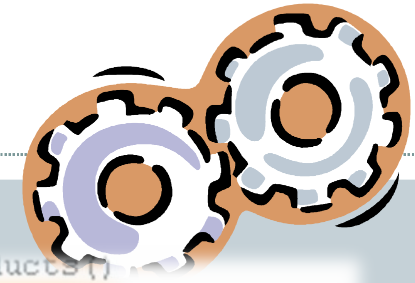


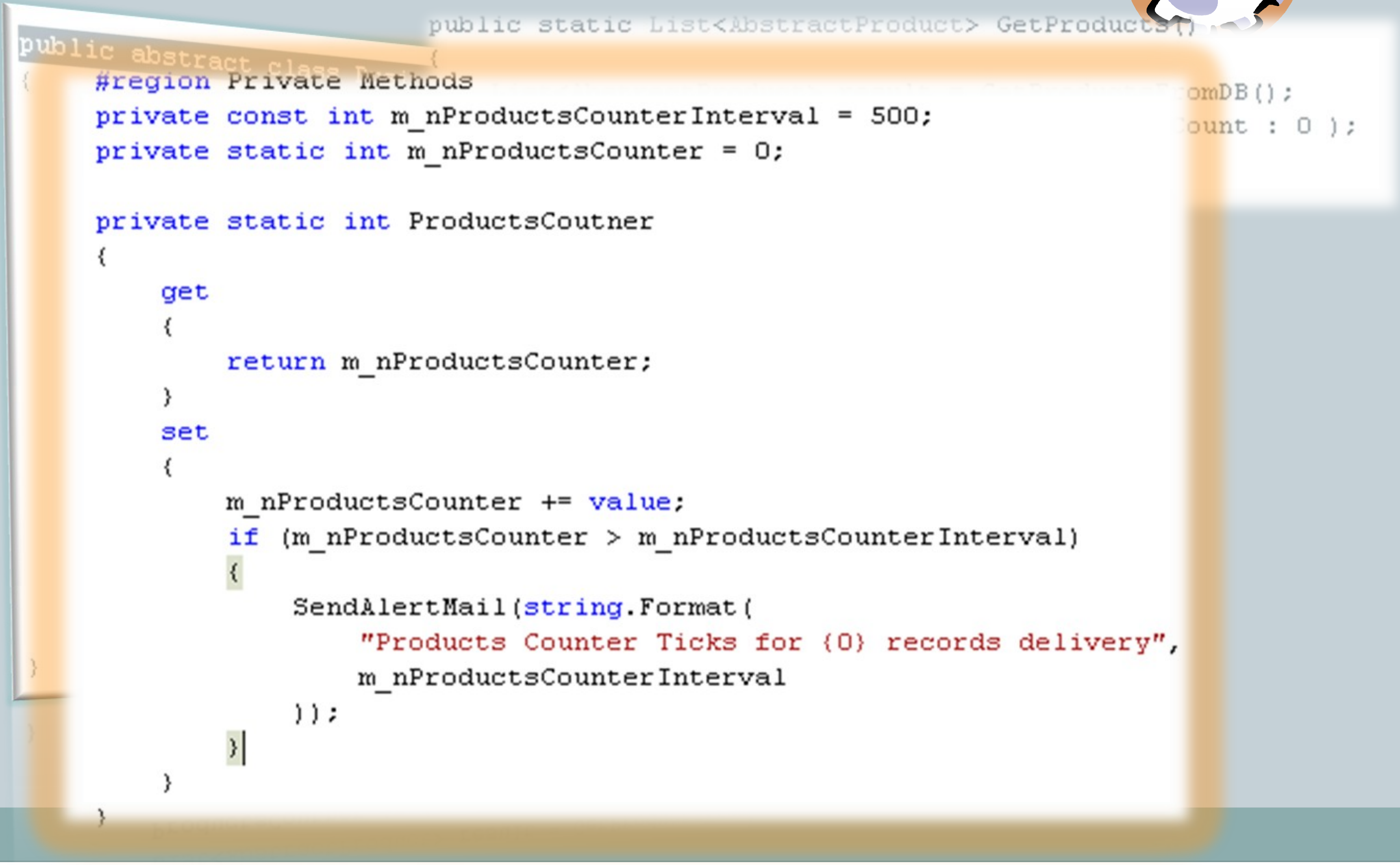
```
public abstract class Busine
{

    Private Methods

    public static void AddProduct(AbstractProduct prod
    {
        SendProductAlert(product);
        AddProductToDB(product);
    }

    public static List<AbstractProduct> GetProducts()
    {
        List<AbstractProduct> result = GetProductsFrom
        ProductsCoutner += ( null!=result ? result.Cou
        return result;
    }
}
```



```
public static List<AbstractProduct> GetProducts()
{
    List<AbstractProduct> result = GetProductsFromDB();
    ProductsCoutner += ( null!=result ? result.Count : 0 );
    return result;
}
```

Specifications

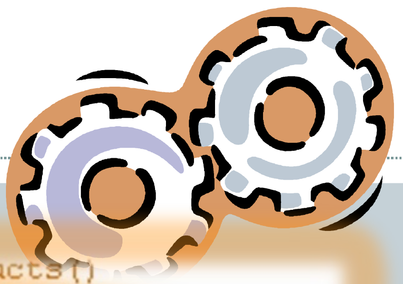




```
public abstract class Products
public static List<AbstractProduct> GetProducts()
```

```
#region Private Methods
private const int m_nProductsCounterInterval = 500;
private static int m_nProductsCounter = 0;

private static int ProductsCoutner
{
    get
    {
        return m_nProductsCounter;
    }
    set
    {
        m_nProductsCounter += value;
        if (m_nProductsCounter > m_nProductsCounterInterval)
        {
            SendAlertMail(string.Format(
                "Products Counter Ticks for {0} records delivery",
                m_nProductsCounterInterval
            ));
        }
    }
}
```

Specifications



```
public abstract class Busine
{

    Private M

    public
    {
        Ser
        Add
    }

    public
    {
        Lis
        Produ
        return result;
    }
}
```

```
public static List<AbstractPr                GetProducts()
{
    List<AbstractProduct> res
    ProductsCoutner += ( null
```

Solution Explorer
Solution 'nTier' (3 projects)
Common
References
app.config
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs

Business Logic Tier

Summary

Questions?