



# مقدمه ای بر معماری سه لایه

تالیف: ابوذر نوزاری

ناشر: شرکت نرم افزار سی الگوپارس

C#

[www.algopars.ir](http://www.algopars.ir)

*An Introduction to 3-Layer Architecture*

*By : Abozar Nozari*

*Publisher : AlgoPars Software Co.*

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

	:
	:
	:
	:
	:
( )	:
	<b>www.algopars.ir</b>
	<b>abozar_nozari@hotmail.com</b>

:

:

:

ADO .Net

Client/Server

UML

" "

Façade

Façade

Facade



*abozar\_nozari@hotmail.com*

"

"

UML

RUP

RUP

(Layering)

(Software Applications)

(Information Systems)

(OOP)

C#

ADO .Net

UML

) Rational Suite



:



" "

" "

## Software Architecture

---

<sup>1</sup> - Architecture  
<sup>2</sup> - Layering  
<sup>3</sup> - Software Developer

:  
( )

- 
- <sup>1</sup> - Data Structures
  - <sup>2</sup> - Computing
  - <sup>3</sup> - Complexity
  - <sup>4</sup> - Composition
  - <sup>5</sup> - Alternative
  - <sup>6</sup> - Structure
  - <sup>7</sup> - IEEE
  - <sup>8</sup> - System Integrity
  - <sup>9</sup> - Style

RUP

OSI

---

<sup>1</sup> - Architecture Based Methodology

<sup>2</sup> - Layering

<sup>3</sup> - ISO

<sup>4</sup> - Layered system

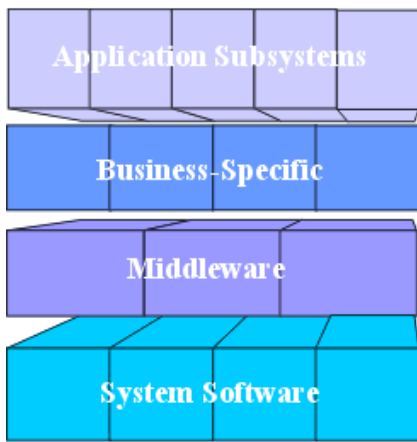
<sup>5</sup> - Hierarchy

<sup>6</sup> - Functionality

<sup>7</sup> - Application

<sup>8</sup> - Domain

<sup>9</sup> - Software Deployment



Distinct Application subsystems that make up an application - contain the value adding software developed by the organization.

Business Specific- contains a number of reusable subsystems specific to the type of business.

Middleware - offers subsystems for utility classes and platform-independent services for distinguished object computing in heterogeneous environments and so on.

System software - contains the software for the actual infrastructure such as operating systems, interfaces to specific hardware, device drivers and so on.

:( )

### Loosely Coupled

:( )


<sup>1</sup> - Subsystem ( )

<sup>3</sup> -Maintenance

<sup>4</sup> -Problem Solving

Client/Server

server      Client

Client/Server

Main Frame

( )

....

---

<sup>1</sup> - 3-Layered Architecture in Information Systems

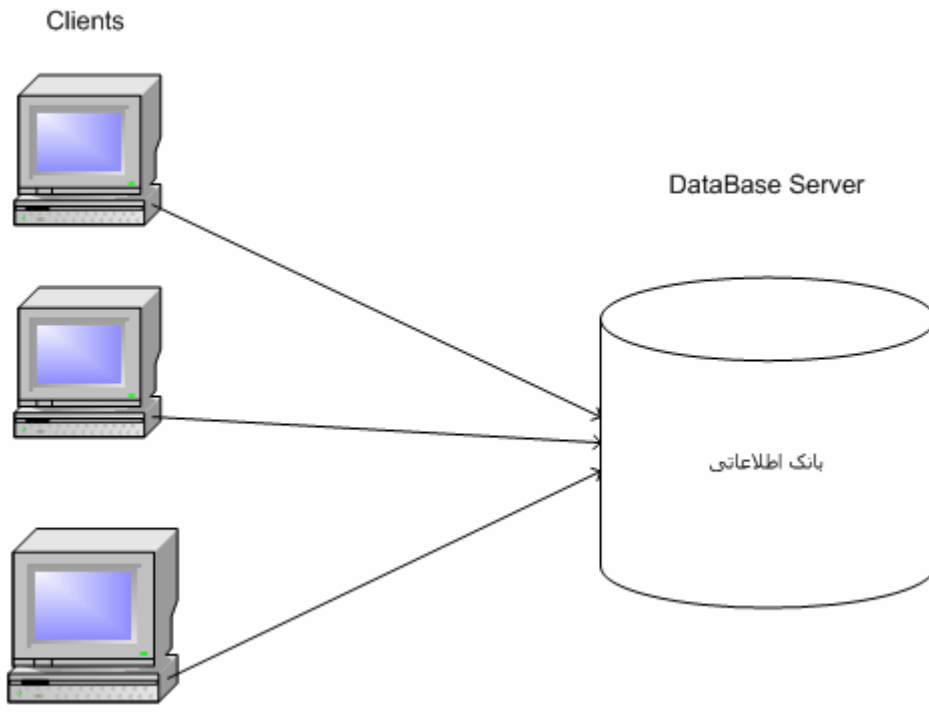
<sup>2</sup> - Information system (IS)

<sup>3</sup> - Data Base

<sup>4</sup> - Information

<sup>5</sup> - Data Base Management system (DBMS)

<sup>6</sup> - Static



Client/Server

:( )

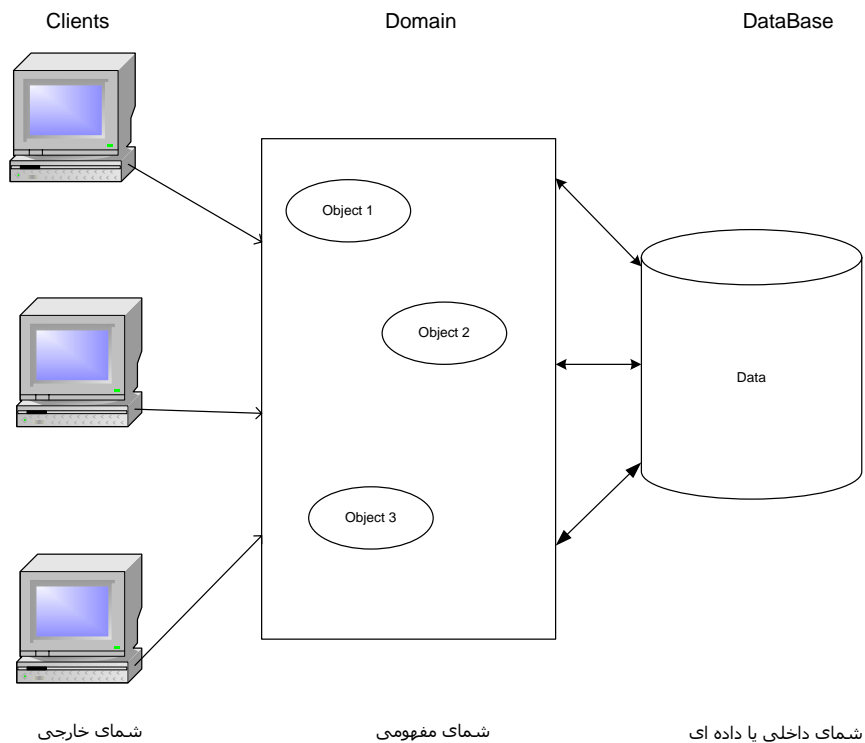
( )

Client/Server

Client

Server

- 
- 1 - Dynamic
  - 2 - Business
  - 3 - Approach
  - 4 - External
  - 5 - Conceptual
  - 6 - Internal
  - 7 - Schema
  - 8 - Semantic
  - 9 - Procedures



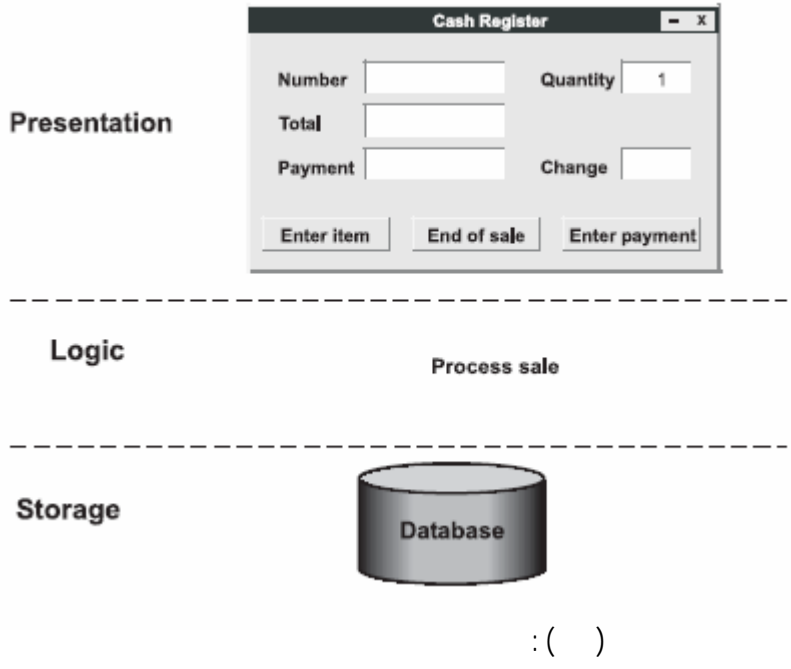
:( )

( )

- 
- <sup>1</sup> - Responsibilities
  - <sup>2</sup> - Presentation
  - <sup>3</sup> -Logic
  - <sup>4</sup> - Data Base (DB)

)

(



«Layer»

UML

UML 2.0

( )

UML

Business Presentation

Data Access

Logic Layer Domain Layer

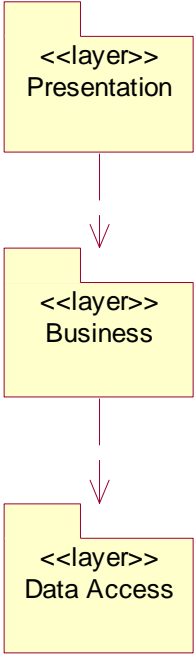
Business

:

:(Presentation Layer)

<sup>1</sup> - Extended Notation

<sup>2</sup> -Forms



UML

:( )

Domain

:(Business Logic Layer)

:(Data Access Layer)

---

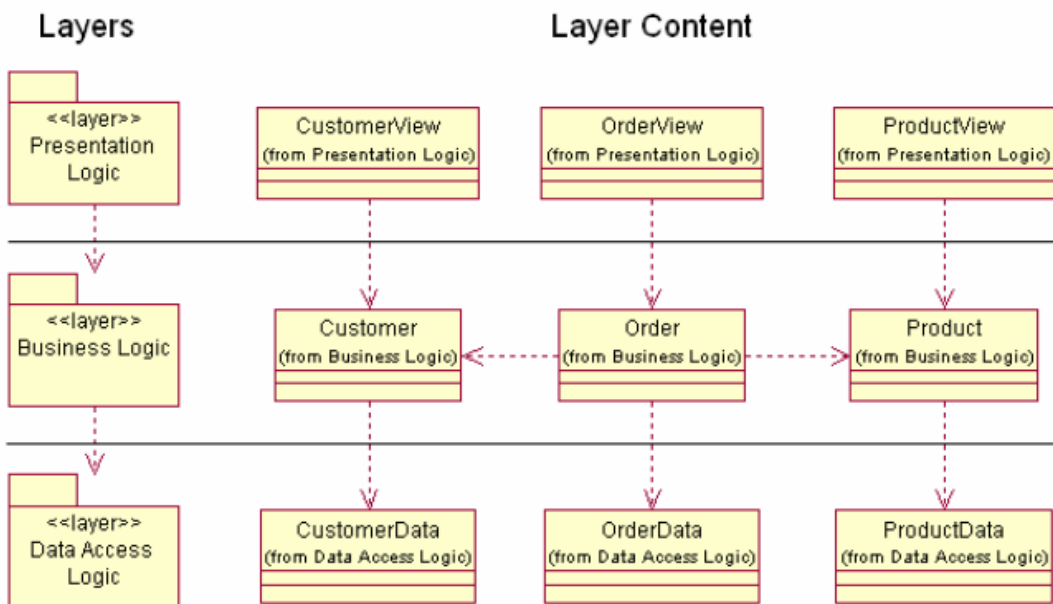
<sup>1</sup> - component  
<sup>2</sup> - Graphical User Interface(GUI)  
<sup>3</sup> - Dependency

- 
- <sup>1</sup> - Element
  - <sup>2</sup> - Peer
  - <sup>3</sup> - Event
  - <sup>4</sup> - Message
  - <sup>5</sup> - Message Passing
  - <sup>6</sup> - Directed Acyclic Graph (DAG)

```

: CustomerView
    ( )
: Customer
: CustomerData
)
( )

```



```
:( )
```

```
.( [4],[2] )
```

<sup>1</sup> - Persistent  
<sup>2</sup> - Execution Node

:( )

	:	Single System
:		Thin Client
	:	Fat Client

( )

- 
- <sup>1</sup> - Distribution
  - <sup>2</sup> - Nodes
  - <sup>3</sup> - Modularity
  - <sup>4</sup> - Reusability

( ):

( )

( )

GUI

String , Real ,Integer

( )

- 
- <sup>1</sup> - Fonts
  - <sup>2</sup> - Colors
  - <sup>3</sup> - Visibility

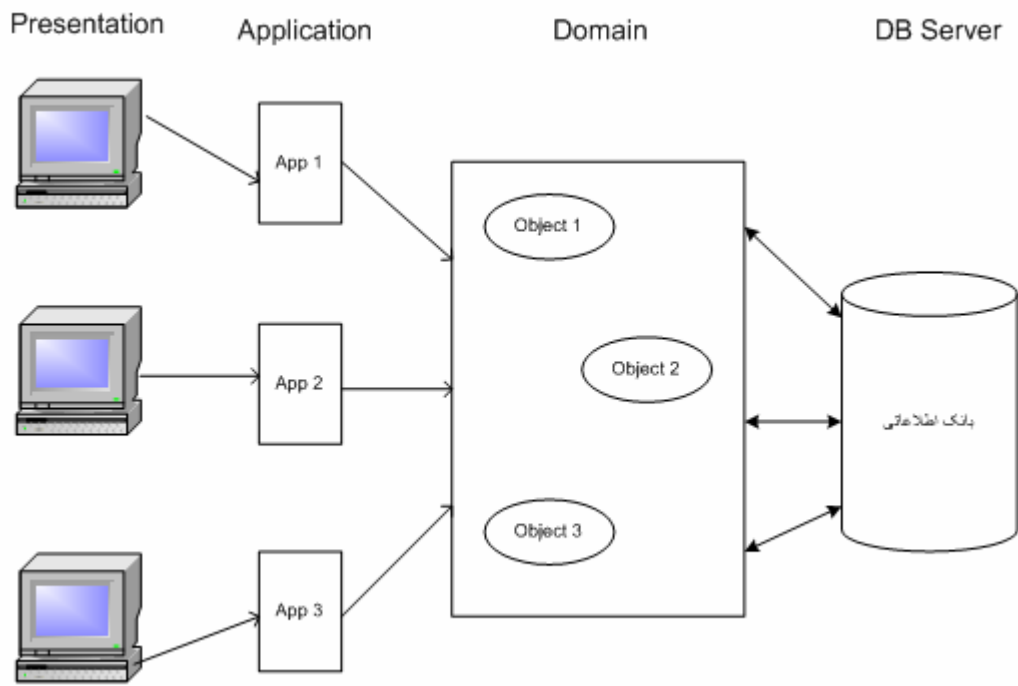
- <sup>5</sup> - Application Layer
- <sup>6</sup> - Format
- <sup>7</sup> - Interrelationship
- <sup>8</sup> - Data Type
- <sup>9</sup> - ADT
- <sup>10</sup> - Convert

Domain

Domain Layer

4

" "



:( )

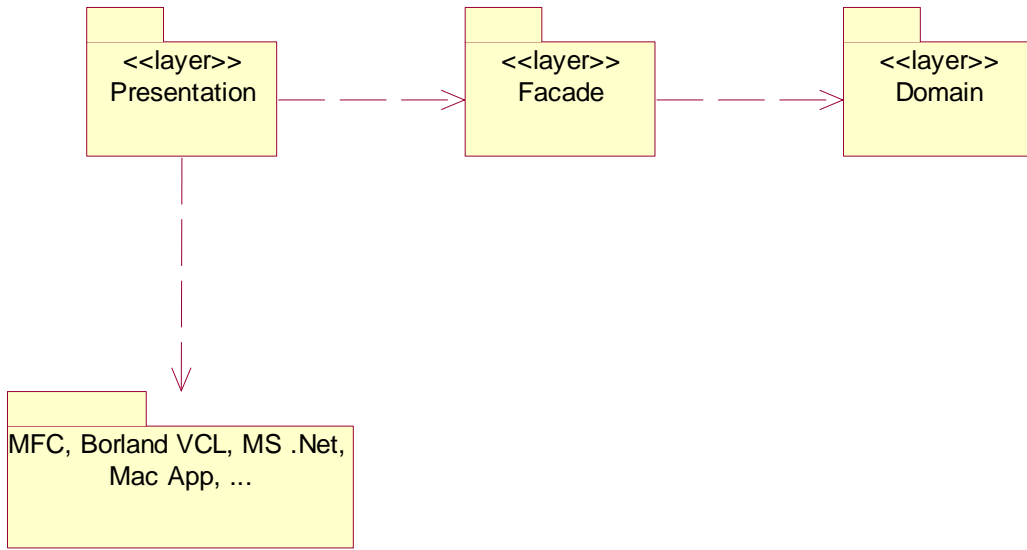
( )

UML

---

<sup>1</sup> - Libraries  
<sup>2</sup> -Design Patterns  
<sup>3</sup> - Facade

:



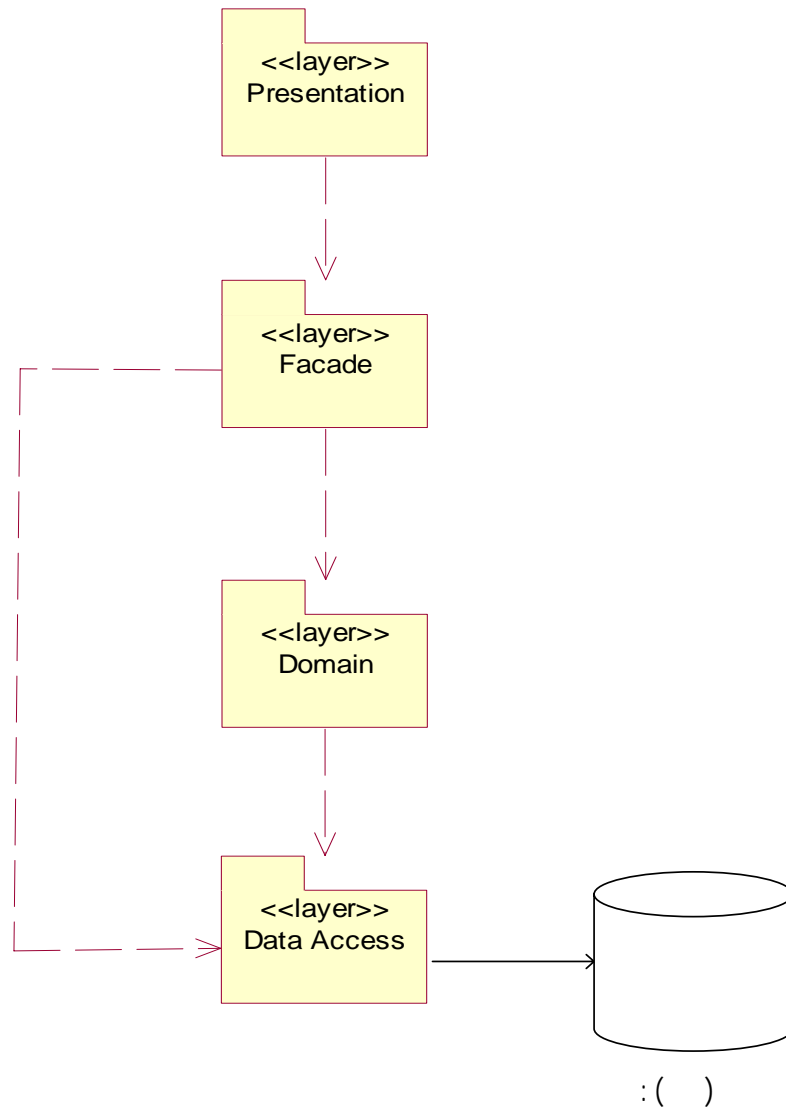
" " :( )

UML

( )

" " " "

( )



[1].Martin Fowler, *"Analysis Patterns: Reusable Object Models"*, Chapter 12, Addison Wesley.

[2].Pascal Roques, *"UML in Practice"*, Chapter 7, John Wiley, 2004.

[3].David Garlan, Mary Shaw, *"An Introduction to Software Architecture"*, School of Computer Science of Carnegie Mellon University, 1994.

[4].Peter Eeles, *"Layering Strategies"*, Rational Software Corporation, 2002.

[5].Rational Help System, *"Guidelines: Layering "*, Rational Software Corporation.



:



:

Packet Organization

Memory (POM)

[ ]

OOPLSA

Hillside Group

Design Patterns: Elements of Reusable :

Gang of Four

Object Oriented Software

---

<sup>1</sup> -Christopher Alexander

<sup>2</sup> -Ward Cunningham

<sup>3</sup> -Kent Back

<sup>5</sup> -Grady Boach

---

<sup>1</sup> -Class Interface

<sup>2</sup> -Hierarchal

<sup>3</sup> -Inheritance

:

.[ ]

( )

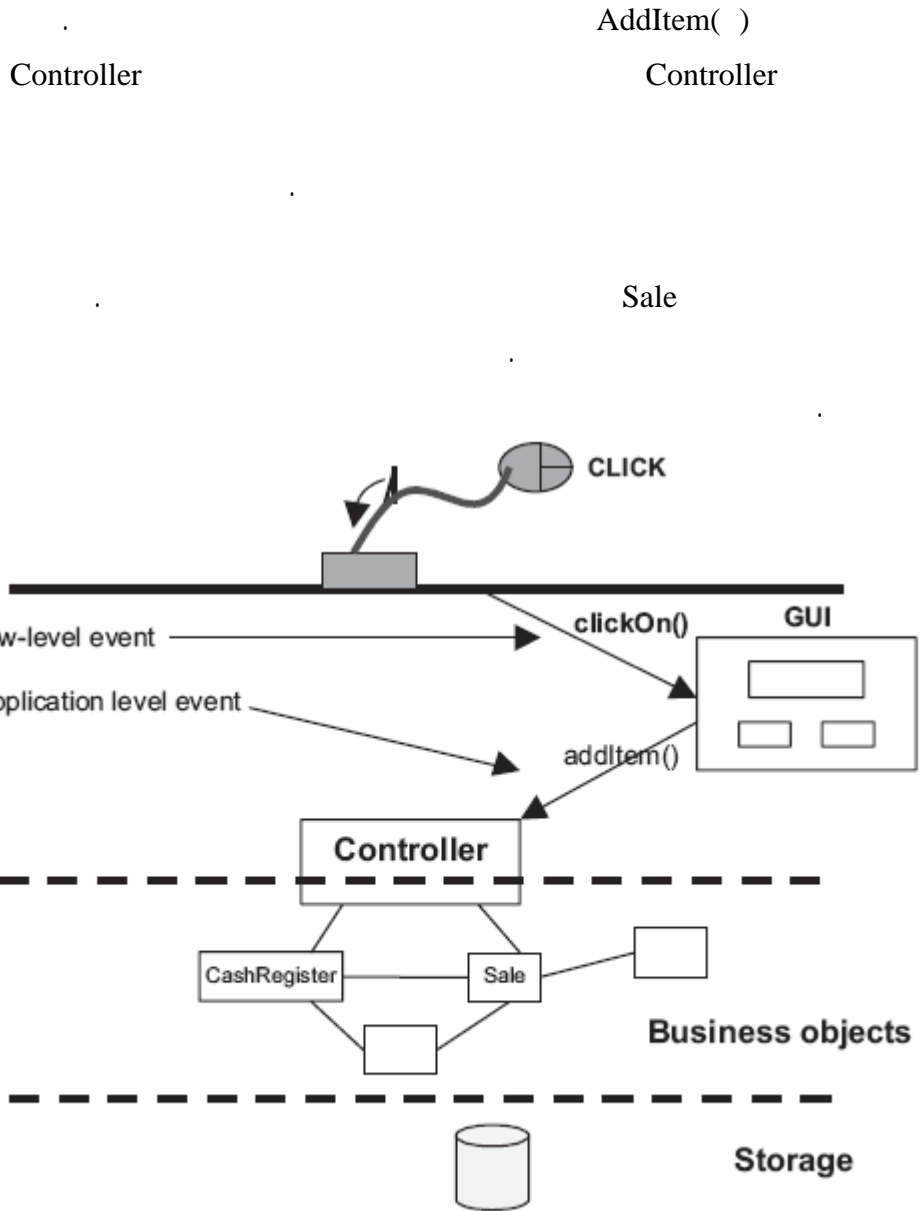
( )

ClickOn( )

Controller  
OnClick( )

Controller  
AddItem( )

- 
- <sup>1</sup> -Application Layer
  - <sup>2</sup> -Interface
  - <sup>3</sup> -Presentation
  - <sup>4</sup> -Domain
  - <sup>5</sup> -GUI
  - <sup>6</sup> -Message



AddItem( )

Controller

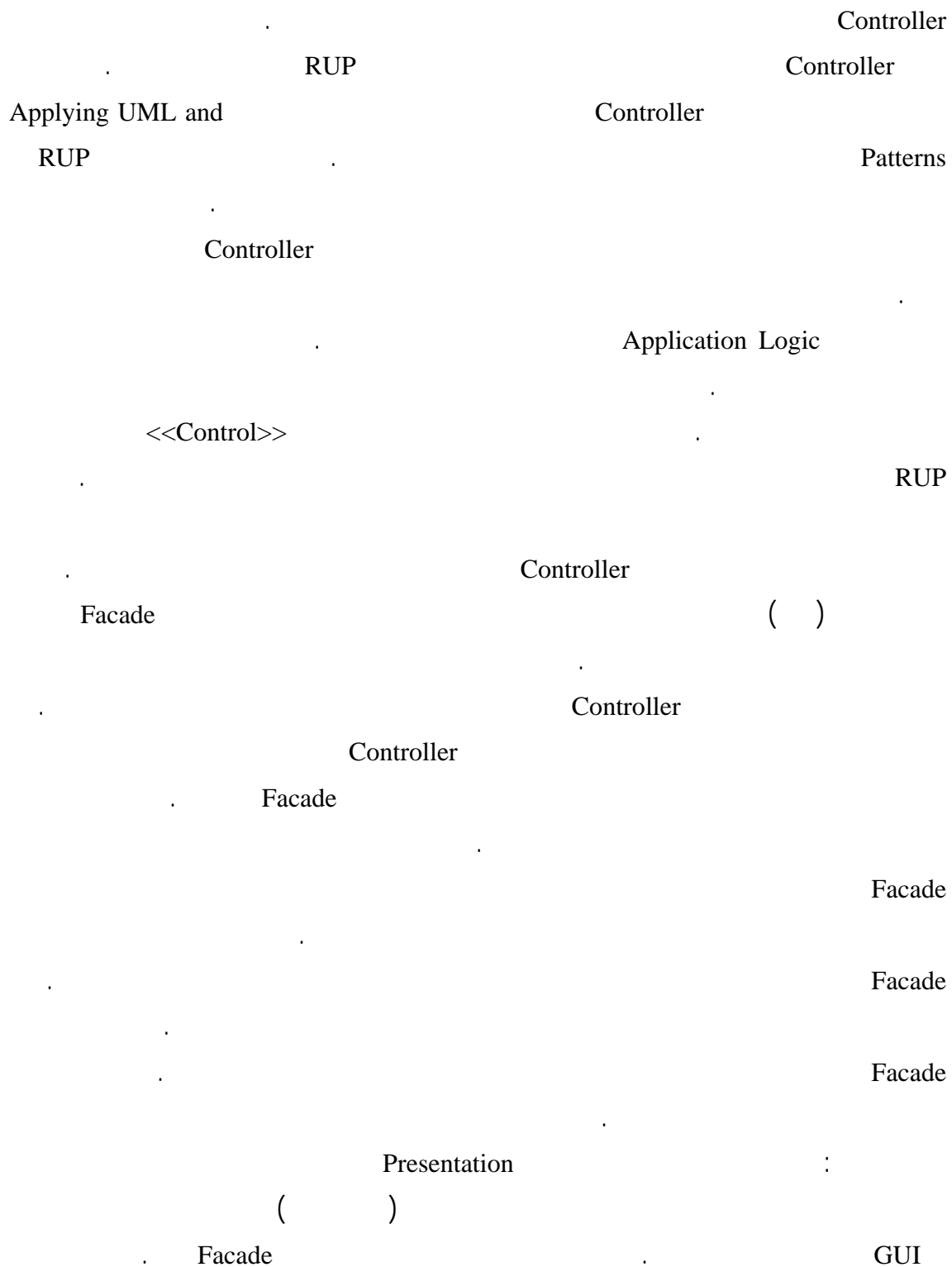
Controller

Sale

:( )

Controller

<sup>1</sup> -Low level event  
<sup>2</sup> -Application level event  
<sup>3</sup> -Business objects



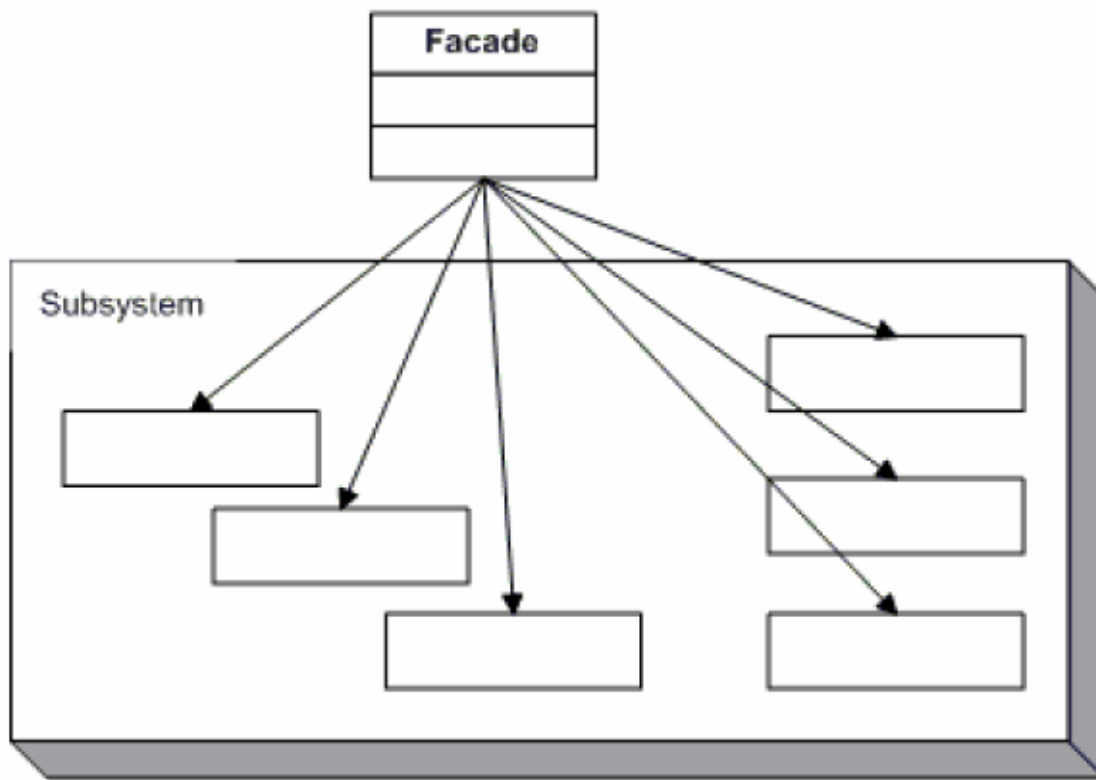

---

(fə'saɪd : )  
 2-Type

1  
 " " " "

Presentation

Facade



Façade : ( )

Facade

[ ]

Martin Fowler

Facade

App. Facade

Application Façade

App. Facade

Facade

Facade

Facade . (Type)

Facade

<sup>1</sup> -Attributes  
<sup>2</sup> -Operations  
<sup>3</sup> -Domain

:

Facade

...

Facade

:

- 
- 
- 
- 
- 

App. Facade

App. Facade

App. Facade

App. Facade

App.

Facade

( )

Facade

App. Facade

App. Facade

Facade Subject

Facade

Subject

Subject

Facade ( )

App. Facade

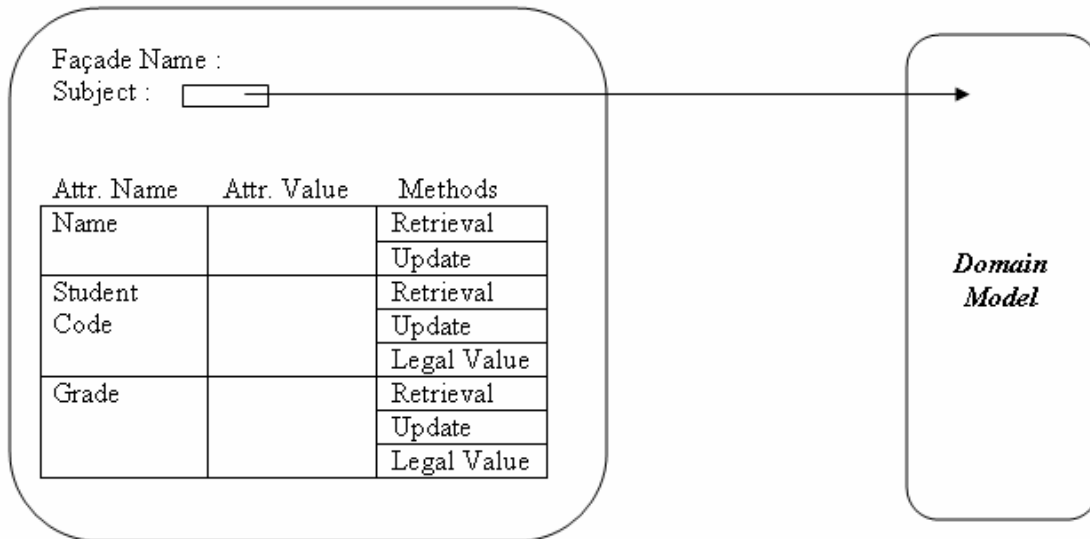
Subject

---

<sup>1</sup> -Validation  
<sup>2</sup> -Legal Values  
<sup>3</sup> -Subject

Subject type Facade  
 Subject App. Facade  
 Facade  
 Subject

App. Facade App. Facade



Façade : ( )

Facade ( )

Facade

Facade

Facade

Subject

<sup>1</sup> -Logical Window

<sup>2</sup> -Attribute



( )

---

<sup>1</sup> -Validation Checking  
<sup>2</sup> -Update

:

Facade : ( )


Facade

...

Facade

APP. Facade

Facade

APP. Facade

---

<sup>1</sup> -Common Method

CASE

Client / Server

Server Client

- 
- <sup>1</sup> -Domain Layer
  - <sup>2</sup> -Data Access
  - <sup>3</sup> -Semantic Modelers
  - <sup>4</sup> -Encapsulation
  - <sup>5</sup> -Applications

---

<sup>1</sup> -Reversion  
<sup>2</sup> -Concurrency  
<sup>3</sup> -Data Base Server

( )

Data Access

- 
- <sup>1</sup> -Load Routine
  - <sup>2</sup> -Map
  - <sup>3</sup> -Data Base Interface Tire
  - <sup>4</sup> -Data Source
  - <sup>5</sup> -Components

:

ADO

ADO .Net

... DB2 Oracle MS SQL Server

Façade

Façade

Façade

Client / Server

[1].Martin Fowler, "*Analysis Patterns: Reusable Object Models*", Chapters 12-13, Addison Wesley.

[2].Pascal Roques, "*UML in Practice*", Chapter 7, John Wiley, 2004.

[3].Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, "*Design Patterns : Elements of Reusable Object Oriented Software*", Chapter 1, Addison Wesley,1995.

journals.itorbit.net :

.[ ]



:



C#

ADO .Net

[ ]

**ADO .Net**

ADO .Net

ADO .Net

MSDN

---

<sup>1</sup> - Platform

:

: OleDbConnection  
Connection

SQL

: OleDbCommand

: OleDbDataAdapter

DataSet

: DataSet

: DataTable

DataTable

: DataRow

:

:

Access

```
string connectionString =  
    "Provider=Microsoft.Jet.OLEDB.4.0;" +  
    "Data Source=" + dbName;
```

OleDbConnection

```
OleDbConnection conn =  
    new OleDbConnection(connectionString);
```

---

Data Source

DataSet

DataSet

Cache

1

<sup>2</sup> - Query

<sup>3</sup> - Connection String

<sup>4</sup> - Run Time Error

```

private void openConnection() {
    if (conn.State == ConnectionState.Closed){
        conn.Open ();
    }
}

```

IDE

ADOCommand

SQL

Select

```

public DataTable openTable (string tableName) {
    OleDbDataAdapter adapter = new OleDbDataAdapter ();
    DataTable dtable = null;
    string query = "Select * from " + tableName;
    adapter.SelectCommand = new OleDbCommand (query, conn);
}

```

DataSet

```

DataSet dset = new DataSet ("mydata");
try {
    openConnection();
    adapter.Fill (dset);
}
catch(Exception e) {
    Console.WriteLine (e.Message );
}

```

DataSet

```

//get the table from the dataset
dtable = dset.Tables [0];

```

Select

Try

```

:

public DataTable openQuery(string query) {
    OleDbDataAdapter dsCmd = new OleDbDataAdapter ();
    DataSet dset = new DataSet ();
    //create a dataset
    DataTable dtable = null; //declare a data table
    try {
        //create the command
        dsCmd.SelectCommand =
            new OleDbCommand(query, conn);
    //open the connection
        openConnection();
        //fill the dataset
        dsCmd.Fill(dset, "mine");
        //get the table
        dtable = dset.Tables[0];
        //always close it
        closeConnection();
        //and return it
        return dtable;
    }
    catch (Exception e) {
        Console.WriteLine (e.Message);
        return null;
    }
}

```

## Delete \* Form Table

```

:

public void delete() {
    //deletes entire table
    conn = db.getConnection();
    openConn();
    if (conn.State == ConnectionState.Open ) {
        OleDbCommand adcmd =
            new OleDbCommand("Delete * from " + tableName, conn);
        try{
            adcmd.ExecuteNonQuery();
            closeConn();
        }
        catch (Exception e) {
            Console.WriteLine (e.Message);
        }
    }
}

```

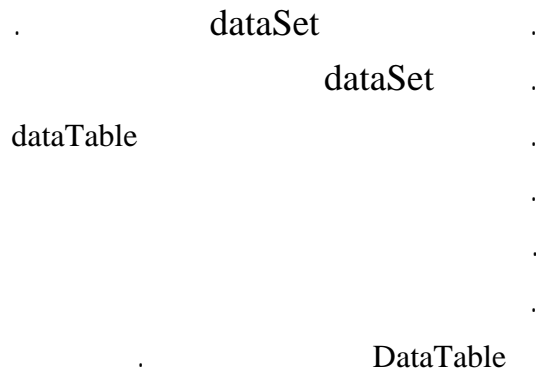
## ADO .Net

```

:

DataTable

```



```

DataSet dset = new DataSet(tableName); //create the data set
dtable = new DataTable(tableName); //and a datatable
dset.Tables.Add(dtable); //add to collection
conn = db.getConnection();
openConn(); //open the connection
OleDbDataAdapter adcmd = new OleDbDataAdapter();
//open the table
adcmd.SelectCommand =
new OleDbCommand("Select * from " + tableName, conn);
OleDbCommandBuilder olecb = new OleDbCommandBuilder(adcmd);
adcmd.TableMappings.Add("Table", tableName);
//load current data into the local table copy
adcmd.Fill(dset, tableName);
//get the Enumerator from the Hashtable
IEnumerator ienum = names.Keys.GetEnumerator();
//move through the table, adding the names to new rows
while (ienum.MoveNext()) {
    string name = (string)ienum.Current;
    row = dtable.NewRow(); //get new rows
    row[columnName] = name;
    dtable.Rows.Add(row); //add into table
}
//Now update the database with this table
try {
    adcmd.Update(dset);
    closeConn();
    filled = true;
}
catch (Exception e) {
    Console.WriteLine (e.Message);
}
  
```

---

<sup>1</sup> - Add  
<sup>2</sup> - Row

## Façade

Stop and Shop, Apples, 0.27  
Stop and Shop, Oranges, 0.36  
Stop and Shop, Hamburger, 1.98  
Stop and Shop, Butter, 2.39  
Stop and Shop, Milk, 1.98  
Stop and Shop, Cola, 2.65  
Stop and Shop, Green beans, 2.29  
Village Market, Apples, 0.29  
Village Market, Oranges, 0.29  
Village Market, Hamburger, 2.45  
Village Market, Butter, 2.99  
Village Market, Milk, 1.79  
Village Market, Cola, 3.79  
Village Market, Green beans, 2.19  
Waldbaum's, Apples, 0.33  
Waldbaum's, Oranges, 0.47  
Waldbaum's, Hamburger, 2.29  
Waldbaum's, Butter, 3.29  
Waldbaum's, Milk, 1.89  
Waldbaum's, Cola, 2.99  
Waldbaum's, Green beans, 1.99

1. Stores ( )
  - StoreName
  - StoreKey
2. Foods ( )
  - FoodName
  - FoodKey
3. Prices ( )
  - PriceKey
  - Price
  - StoreKey
  - FoodKey

:

Stores

Prices

Façade

Foods

DBTable

ADO.Net

Façade

ADO .Net

Façade

Dbase

```

public abstract class DBase {
    protected OleDbConnection conn;

    private void openConnection() {
        if (conn.State == ConnectionState.Closed){
            conn.Open ();
        }
    }
    //-----
    private void closeConnection() {
        if (conn.State == ConnectionState.Open ){
            conn.Close ();
        }
    }
    //-----
    public DataTable openTable (string tableName) {
        OleDbDataAdapter adapter = new OleDbDataAdapter ();
        DataTable dtable = null;
        string query = "Select * from " + tableName;
        adapter.SelectCommand = new OleDbCommand (query, conn);
        DataSet dset = new DataSet ("mydata");
        try {
            openConnection();
            adapter.Fill (dset);
            dtable = dset.Tables [0];
        }
        catch(Exception e) {
            Console.WriteLine (e.Message );
        }
        return dtable;
    }
    //-----
    public DataTable openQuery(string query) {
        OleDbDataAdapter dsCmd = new OleDbDataAdapter ();
        DataSet dset = new DataSet (); //create a dataset
        DataTable dtable = null; //declare a data table
        try {
            //create the command
            dsCmd.SelectCommand = new OleDbCommand(query, conn);
            openConnection(); //open the connection
        }
    }
}

```

- 
- <sup>1</sup> - Foreign Key
  - <sup>2</sup> - Abstract Class
  - <sup>3</sup> - Encapsulate

```

        //fill the dataset
        dsCmd.Fill(dset, "mine");
        //get the table
        dtable = dset.Tables[0];
        closeConnection(); //always close it
        return dtable; //and return it
    }

    catch (Exception e) {
        Console.WriteLine (e.Message);
        return null;
    }
}
//-----
public void openConnection(string connectionString) {
    conn = new OleDbConnection(connectionString);
}
//-----
public OleDbConnection getConnection() {
    return conn;
}
}
}

```

SQL Server Access

```

public class AxsDatabase :Dbase {
    public AxsDatabase(string dbName) {
        string connectionString =
            "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + dbName;
        openConnection(connectionString);
    }
}

```

: SQL Server

```

public class SQLServerDatabase:DBase {
    string connectionString;
    //-----
    public SQLServerDatabase(String dbName) {
        connectionString = "Persist Security Info = False;" +
            "Initial Catalog =" + dbName + ";" +
            "Data Source = myDataServer;User ID = myName;" +
            "password=";
        openConnection(connectionString);
    }
    //-----
    public SQLServerDatabase(string dbName, string serverName,
        string userid, string pwd) {
        connectionString = "Persist Security Info = False;" +
            "Initial Catalog =" + dbName + ";" +
            "Data Source =" + serverName + ";" +
            "User ID =" + userid + ";" +
            "password=" + pwd;
        openConnection(connectionString);
    }
}

```

:

: DbTable

DBTable

DBase

```
public class DBTable {
    protected DBase db;
    protected string tableName;
    private bool filled, opened;
    private DataTable dtable;
    private int rowIndex;
    private Hashtable names;
    private string columnName;
    private DataRow row;
    private OleDbConnection conn;
    private int index;
//-----
public DBTable(DBase datab, string tb_Name) {
    db = datab;
    tableName = tb_Name;
    filled =false;
    opened = false;
    names = new Hashtable();
}
//-----
public void createTable() {
    try {
        dtable = new DataTable(tableName);
        dtable.Clear();
    }
    catch (Exception e) {
        Console.WriteLine (e.Message );
    }
}
//-----
public bool hasMoreElements() {
    if(opened)
        return (rowIndex < dtable.Rows.Count) ;
    else
        return false;
}
//-----
public int getKey(string nm, string keyname){
    DataRow row;
    int key;
    if(! filled)
        return (int)names[ nm];
    else {
        string query = "select * from " + tableName + " where " +
            columnName + "=\'' + nm + '\''";
        dtable = db.openQuery(query);
        row = dtable.Rows[0];
        key = Convert.ToInt32 (row[keyname].ToString());
    }
}
```

---

<sup>1</sup> - Single Values

```

        return key;
    }
}
//-----
public virtual void makeTable(string cName) {
    //shown below
//-----
private void closeConn() {
    if( conn.State == ConnectionState.Open) {
        conn.Close();
    }
}
//-----
private void openConn() {
    if(conn.State == ConnectionState.Closed ) {
        conn.Open();
    }
}
//-----
public void openTable() {
    dtable = db.openTable(tableName);
    rowIndex = 0;
    if(dtable != null)
        opened = true;
}
//-----
public void delete() {
    //shown above
}
}

```

DBTable

Stores

Foods

C#

Hashtable

Hashtable

Hashtable

Hashtable

C#

Hashtable

## Hashtable

```
public void addTableValue(string nm) {
//accumulates names in hash table
    try {
        names.Add(nm, index++);
    }
    catch (ArgumentException) {}
//do not allow duplicate names to be added
}
```

## Hashtable

### Hashtable

### Enumerator

```
public virtual void makeTable(string cName) {
    columnName = cName;
//stores current hash table values in data table
    DataSet dset = new DataSet(tableName); //create dataset
    DataTable dt = new DataTable(tableName); //and a datatable
    dset.Tables.Add(dt); //add to collection
    conn = db.getConnection();
    openConn(); //open the connection
    OleDbDataAdapter adcmd = new OleDbDataAdapter();
//open the table
    adcmd.SelectCommand =
    new OleDbCommand("Select * from " + tableName, conn);
    OleDbCommandBuilder olecb = new OleDbCommandBuilder(adcmd);
    adcmd.TableMappings.Add("Table", tableName);
//load current data into the local table copy
    adcmd.Fill(dset, tableName);
//get the Enumerator from the Hashtable
    IEnumerator ienum = names.Keys.GetEnumerator();
//move through the table, adding the names to new rows
    while (ienum.MoveNext()) {
        string name = (string)ienum.Current;
        row = dt.NewRow(); //get new rows
        row[columnName] = name;
        dt.Rows.Add(row); //add into table
    }
//Now update the database with this table
    try {
        adcmd.Update(dset);
        closeConn();
        filled = true;
    }
    catch (Exception e) {
        Console.WriteLine (e.Message);
    }
}
```

}

```
public class Stores :DBTable {
    public Stores(DBase db):base(db, "Stores"){
    }
    //-----
    public void makeTable() {
        base.makeTable ("Storename");
    }
}
public class Foods: DBTable {
    public Foods(DBase db):base(db, "Foods"){
    }
    //-----
    public void makeTable() {
        base.makeTable ("Foodname");
    }
    //-----
    public string getValue() {
        return base.getValue ("FoodName");
    }
}
public virtual string getValue(string cname) {
    //returns the next name in the table
    //assumes that openTable has already been called
    if (opened) {
        DataRow row = dtable.Rows[rowIndex++];
        return row[cname].ToString().Trim ();
    }
    else
        return "";
}
}
```

GetValue

DBtable

: Prices

Prices

Prices

DBtable

---

<sup>1</sup> - Overload

:

" "

Prices

DBTable

Price FoodKey StoreKey

StoreFoodPrice

```
public class StoreFoodPrice {
    private int storeKey, foodKey;
    private float foodPrice;
    //-----
    public StoreFoodPrice(int sKey, int fKey, float fPrice) {
        storeKey = sKey;
        foodKey = fKey;
        foodPrice = fPrice;
    }
    //-----
    public int getStore() {
        return storeKey;
    }
    //-----
    public int getFood() {
        return foodKey;
    }
    //-----
    public float getPrice() {
        return foodPrice;
    }
}
```

```
public class Prices : DBTable {
    private ArrayList priceList;
    public Prices(DBase db) : base(db, "Prices") {
        priceList = new ArrayList ();
    }
    //-----
    public void makeTable() {
        //stores current array list values in data table
        OleDbConnection adc = new OleDbConnection();

        DataSet dset = new DataSet(tableName);
        DataTable dtable = new DataTable(tableName);

        dset.Tables.Add(dtable);
        adc = db.getConnection();
        if (adc.State == ConnectionState.Closed)
            adc.Open();
        OleDbDataAdapter adcmd = new OleDbDataAdapter();

        //fill in price table
        adcmd.SelectCommand = new OleDbCommand("Select * from " + tableName, adc);
    }
}
```

---

<sup>1</sup> - StoreFoodPrice

<sup>2</sup> - Instance

:

```
OleDbCommandBuilder custCB = new OleDbCommandBuilder(adcmd);
adcmd.TableMappings.Add("Table", tableName);
adcmd.Fill(dset, tableName);
IEnumerator ienum = priceList.GetEnumerator();
//add new price entries
while (ienum.MoveNext()) {
    StoreFoodPrice fprice =
        (StoreFoodPrice)ienum.Current;
    DataRow row = dtable.NewRow();
    row["foodkey"] = fprice.getFood();
    row["storekey"] = fprice.getStore();
    row["price"] = fprice.getPrice();
    dtable.Rows.Add(row); //add to table
}
adcmd.Update(dset); //send back to database
adc.Close();
}
//-----
public DataTable getPrices(string food) {
    string query=
        "SELECT Stores.StoreName, " +
        "Foods.Foodname, Prices.Price " +
        "FROM (Prices INNER JOIN Foods ON " +
        "Prices.Foodkey = Foods.Foodkey) " +
        "INNER JOIN Stores ON " +
        "Prices.StoreKey = Stores.StoreKey " +
        "WHERE(((Foods.Foodname) = \" + food + "\") " +
        "ORDER BY Prices.Price";
    return db.openQuery(query);
}
//-----
public void addRow(int storeKey, int foodKey, float price)
    priceList.Add (new StoreFoodPrice (storeKey, foodKey, price));
}
}
```

Foods Stores

FoodKey StoreKey

Prices

prices

```
public class DataLoader {
    private csFile vfile;
    private Stores store;
    private Foods fods;
    private Prices price;
    private DBase db;
    //-----
    public DataLoader(DBase datab) {
        db = datab;
        store = new Stores(db);
        fods = new Foods (db);
    }
}
```

```

        price = new Prices(db);
    }
    //-----
    public void load(string dataFile) {
        string sline;
        int storekey, foodkey;
        StringTokenizer tok;
        //delete current table contents
        store.delete();
        fods.delete();
        price.delete();
        //now read in new ones
        vfile = new csFile(dataFile);
        vfile.OpenForRead();
        sline = vfile.readLine();
        while (sline != null){
            tok = new StringTokenizer(sline, ",");
            store.addTableValue(tok.nextToken()); //store
            fods.addTableValue(tok.nextToken()); //food
            sline = vfile.readLine();
        }
        vfile.close();
        //construct store and food tables
        store.makeTable();
        fods.makeTable();
        vfile.OpenForRead();
        sline = vfile.readLine();
        while (sline != null) {
            //get the gets and add to storefoodprice objects
            tok = new StringTokenizer(sline, ",");
            storekey = store.getKey(tok.nextToken(), "Storekey");
            foodkey = fods.getKey(tok.nextToken(), "Foodkey");
            price.addRow(storekey, foodkey, Convert.ToSingle(tok.nextToken()));
            sline = vfile.readLine();
        }
        //add all to price table
        price.makeTable();
        vfile.close();
    }
}

```

## Listbox

```

private void loadFoodTable() {
    Foods fods = new Foods(db);
    fods.openTable();
    while (fods.hasMoreElements()){
        lsFoods.Items.Add(fods.getValue());
    }
}

```

```

private void lsFoods_SelectedIndexChanged(object sender, System.EventArgs e) {
    string food = lsFoods.Text;
    DataTable dttable = prc.getPrices(food);

    lsPrices.Items.Clear();
    foreach (DataRow rw in dttable.Rows) {
        lsPrices.Items.Add(rw["StoreName"].ToString().Trim() + "\t" + rw["Price"].ToString());
    }
}

```

( )

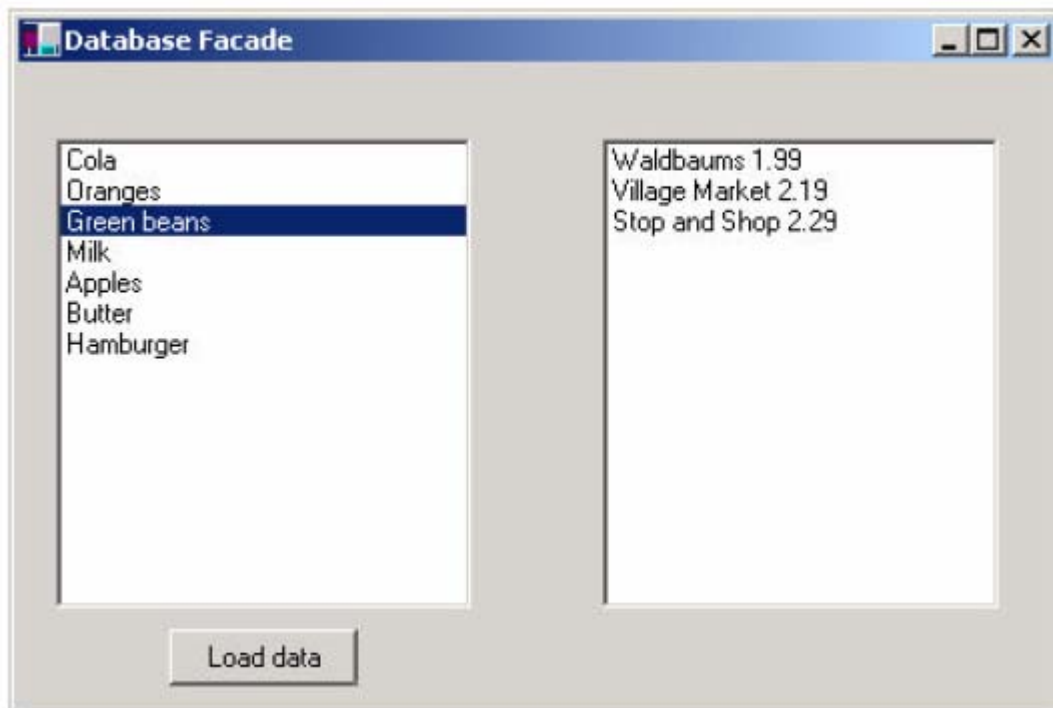
ADO .Net

ADO .Net

Façade

ADO .Net

Hashtable



( )

:

[1]. James W. Cooper, *"Introduction to Design Patterns in C#"*, IBM T J Watson Research Center, Chapter 18, 2002