Data Access Layer (DAL)
Contains methods for accessing the database that are not specific to the project. The only thing specific to the project here is the connection string in the settings.

TEMPLATE CONNECTION STRING
server=(local);database=[DBNAMEHERE];integrated security= SSPI;

SQL Layer
Handles the stored procedures specific to the project.

Business Logic Layer (BLL)
Applies business – such as taxing American goods- rules between the NWNTier and SQL layer.

NWNTier
The front-end containing the form.

Connects to Business layer

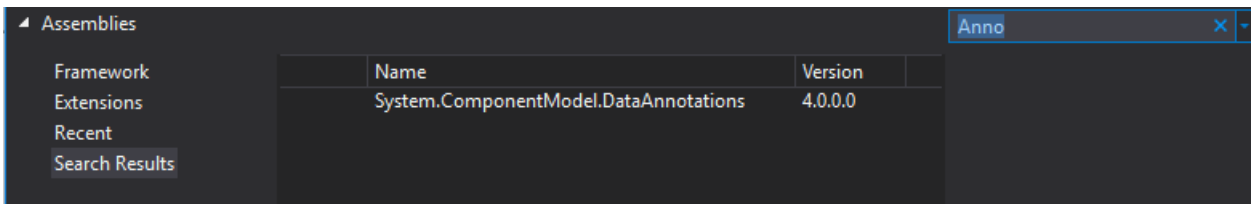
User

Model
Contains classes representing rows in SQL tables (product, user, etc). Remember to store them in an entities folder.

Snippits:

Annotations

Looking up Data Annotations



Required and String Length

```
[Required]
[StringLength(40, ErrorMessage = "Product name is too long!")]
public String ProductName { get; set; }
```

Range (Enforced min and max value)

```
[Range(0, int.MaxValue, ErrorMessage = "Unit Price cannot be negative!")]
public double UnitPrice { get; set; } = 0;
```

Regex

```
[Required]
[RegularExpression("[A-Z][A-Z][A-Z][1-9][FM]", ErrorMessage = "Employee ID is not in the proper format.")
public string EmpId { get; set; }
```

Custom

```
[PubId]
public string PubId { get; set; }
```

```
internal class PubIdAttribute : ValidationAttribute
{
    protected override ValidationResult IsValid(object value, ValidationContext validationContext)
    {
        string pattern = "99[0-9][0-9]";
        Match m = Regex.Match(value.ToString(), pattern);

        if (value.ToString() == "1756"
            || value.ToString() == "1622"
            || value.ToString() == "0877"
            || value.ToString() == "0736"
            || value.ToString() == "1389"
            || m.Success)
        {
            return ValidationResult.Success;
        }

        return new ValidationResult("Please enter a correct value.");
    }
}
```

Checking for Annotation Violations

```
public List<String> problems { get; set; }
```

```
private bool IsValidEntry(Product pr)
{
    ValidationContext context = new ValidationContext(pr, null, null);
    List<ValidationResult> results = new List<ValidationResult>();

    bool toSend = Validator.TryValidateObject(pr, context, results, true);

    foreach(ValidationResult r in results)
    {
        problems.Add(r.ErrorMessage);
    }
    return toSend;
}
```

Unit Testing

Unit Test Example + Assert.AreEqual();

```
[TestMethod]
//Ensures that ProductDB's HowManyMuricans class is working correctly.
public void ProductDBMuricanAccuracy1()
{
    ProductDB pdb = new ProductDB();
    int muricans17 = pdb.HowManyMuricans(56);

    Assert.AreEqual(muricans17, 17);
}
```

Expecting 1 error (validation context)

```
[TestMethod]
public void Validate_InvalidEmpId_Expect1Error()
{
    Employee e = new Employee()
    {
        emp_id = "gravy",
        fname = "Charles",
        lname = "LeBlanc",
        job_lvl = 40,
        pub_id = "9911",
        minit = "F",
        job_id = 15,
        hire_date = DateTime.Now.AddYears(-1)
    };

    ValidationContext context = new ValidationContext(e);
    List<ValidationResult> results = new List<ValidationResult>();
    bool actual = Validator.TryValidateObject(e, context, results, true);

    Assert.AreEqual(1, results.Count);
}
```

Enums

```
public enum EmployeeType
{
    None,
    PartTime,
    FullTime,
    Casual
}
```

Binding a combo box with an enum

```
private void LoadEmployeeTypes()
{
    cmbEmployeeType.DataSource = Enum.GetValues(typeof(EmployeeType));
}
```

Inheritance

Abstract class with abstract method example

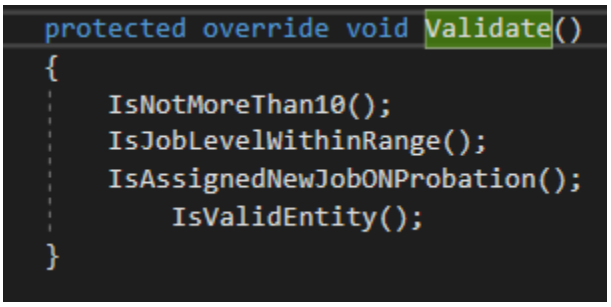
```
public abstract class BaseBL
{
    public List<ValidationError> ValidationErrors { get; set; }

    protected abstract void Validate();
}
```

Extend it

```
public class EmployeeBL : BaseBL
{
    private Employee _emp;
    //public List<ValidationError>
```

Implement the method (MANDATORY)



```
protected override void Validate()
{
    IsNotMoreThan10();
    IsJobLevelWithinRange();
    IsAssignedNewJobONProbation();
        IsValidEntity();
}
```

Misc

Regex example:

```
string pattern1 = "[A-Z][A-Z][A-Z][1-9][0-9][0-9][0-9][0-9][FM]";
string pattern2 = "[A-Z][-][A-Z][1-9][0-9][0-9][0-9][0-9][FM]";

Match m1 = Regex.Match(val, pattern1);
Match m2 = Regex.Match(val, pattern2);
```

Binding a list of objects to a combobox:

```
cmbPublisher.DataSource = publishers;
cmbPublisher.DisplayMember = "PubName";
cmbPublisher.ValueMember = "PubId";
```

Adding an empty “please select one” item to a combo box

```
private void stapleRaces()
{
    DataSet tempRaces = Login.GETDATA("Select RaceId, Name FROM Race ORDER BY Name;", "races");
    DataRow allRow = tempRaces.Tables[0].NewRow();
    tempRaces.Tables[0].Columns["RaceId"].AllowDBNull = true;
    allRow["RaceId"] = DBNull.Value;
    allRow["Name"] = "Select All";
    tempRaces.Tables[0].Rows.InsertAt(allRow,0);

    cboRaces.DataSource = tempRaces;
    cboRaces.DisplayMember = "races.Name";
    cboRaces.ValueMember = "races.RaceId";

}
```