ASP Page Data Binding

by Dmitry Pavlov

ASP Page Data Binding

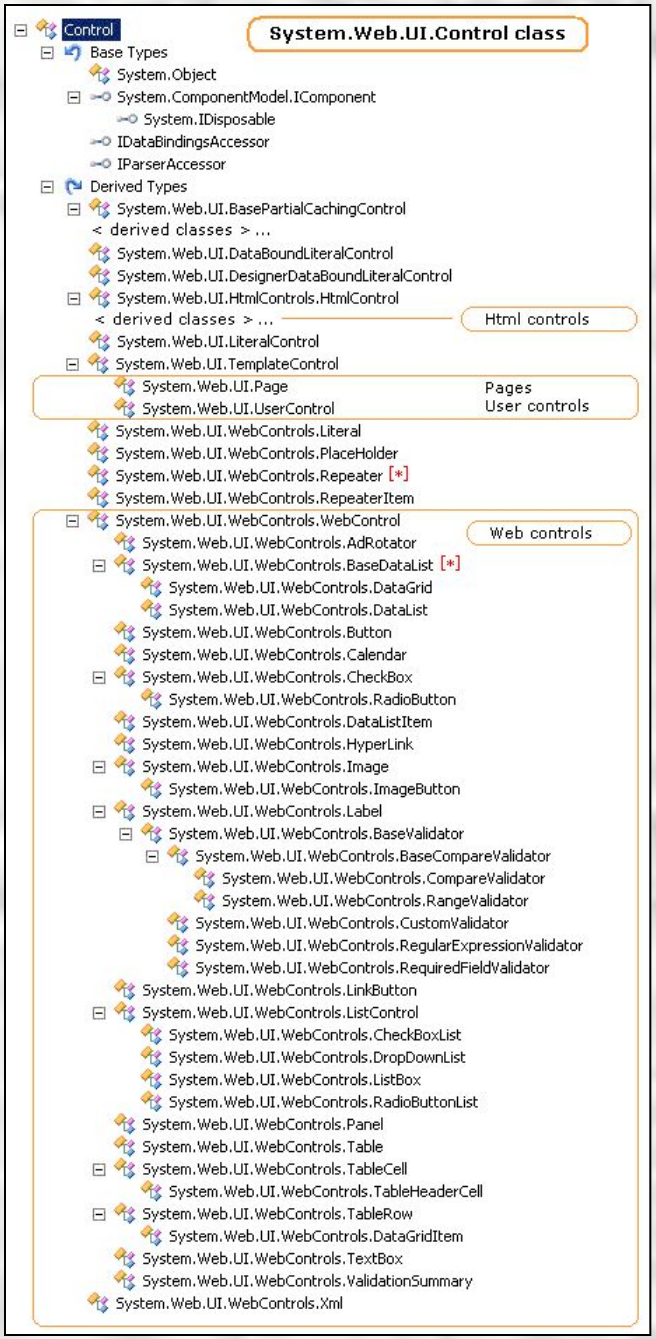
Data Binding – механизм для связывания данных с элементами управления, которые эти данные отображают.

Элементами управления пользовательским интерфейсом являются все классы, унаследованные непосредственно от класса **System.Web.UI.Control** или от его потомков.

В MSDN все эти классы имеют примерно такое описание для метода DataBind:

DataBind (inherited from **Control**)	Binds a data source to the invoked server control and all its child controls.

ASP Page Data Binding



Иерархия наследников класса System.Web.UI.Control

Все производные классы (за исключением классов, помеченных *) наследуют реализацию метода DataBind определенную в базовом классе без изменений:

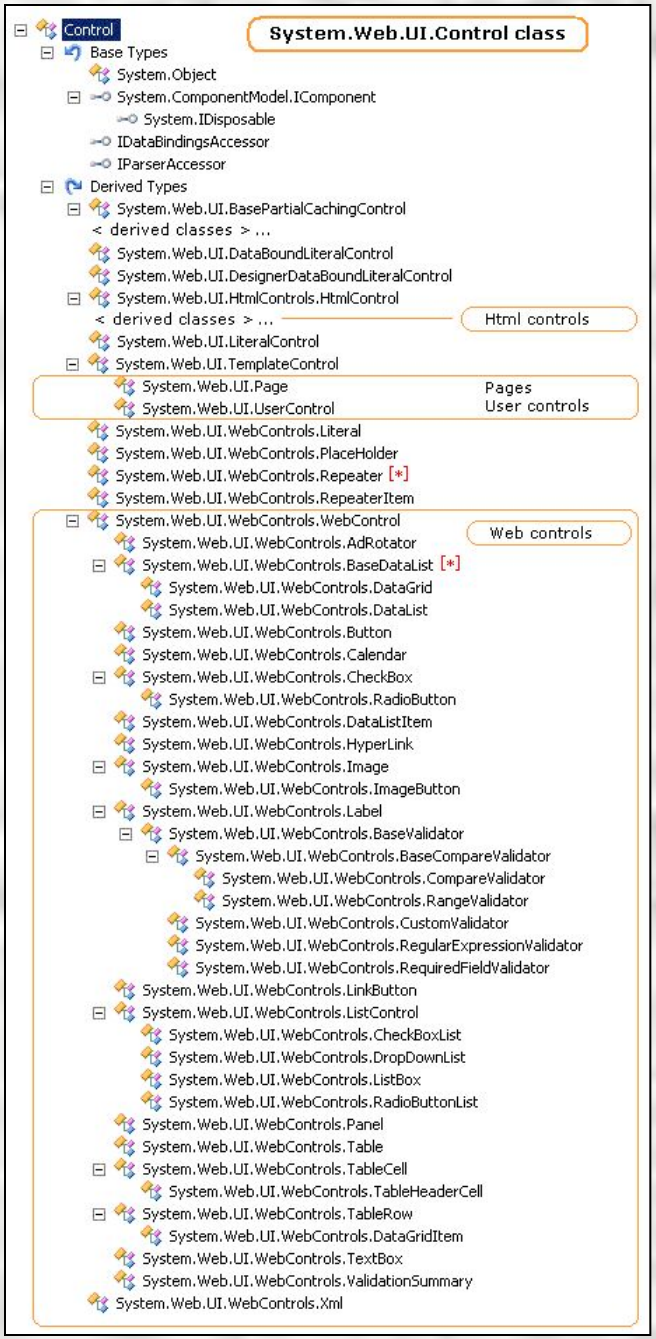
```
public class Control : IComponent, IDisposable, IParserAccessor, IDataBindingsAccessor {

    // ...

    private static readonly object EventDataBinding;
    private ControlCollection   _controls;
    private EventHandlerList    _events;

    // ...

    public virtual void DataBind() {
        this.OnDataBinding(EventArgs.Empty);
        if (this._controls == null) {
            return;
        }
        string errorMsg = this._controls.SetCollectionReadOnly("Parent collections readonly");

        for (int i = 0; i < this._controls.Count; i++) {
            this._controls[i].DataBind();
        }

        this._controls.SetCollectionReadOnly(errorMsg);
    }

    // ...

    protected virtual void OnDataBinding(EventArgs e) {
        if (this._events != null) {
            EventHandler handler = this._events[Control.EventDataBinding] as EventHandler;
            if (handler != null) {
                handler(this, e);
            }
        }
    }

    // ...
}
```

Контрол инициирует событие DataBinding

Метод DataBind вызывается для каждого контрола в коллекции дочерних контролов текущего.

* - метод DataBind() переопределен

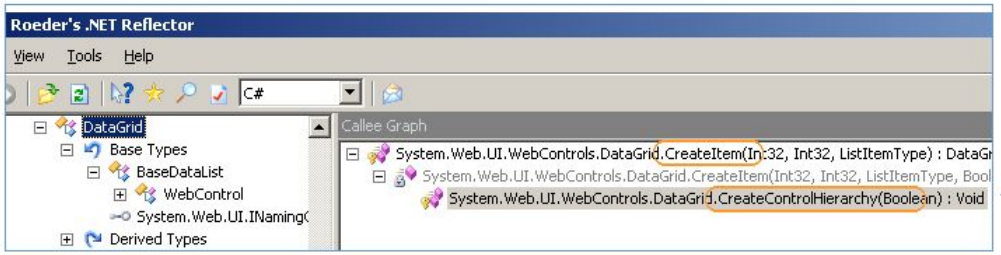
ASP Page Data Binding



Repeater, BaseDataList (родитель для классов DataGird и DatList**)**
В этих классах DataBind переопределен следующим образом:

```
public override void DataBind() {
    this.OnDataBinding(EventArgs.Empty);
}

protected override void OnDataBinding(EventArgs e) {
    base.OnDataBinding(e);
    this.Controls.Clear();
    base.ClearChildViewState();
    this.CreateControlHierarchy(true);
    base.ChildControlsCreated = true;
    this.TrackViewState();
}
```

Вызов абстрактного метода, определенного в наследниках

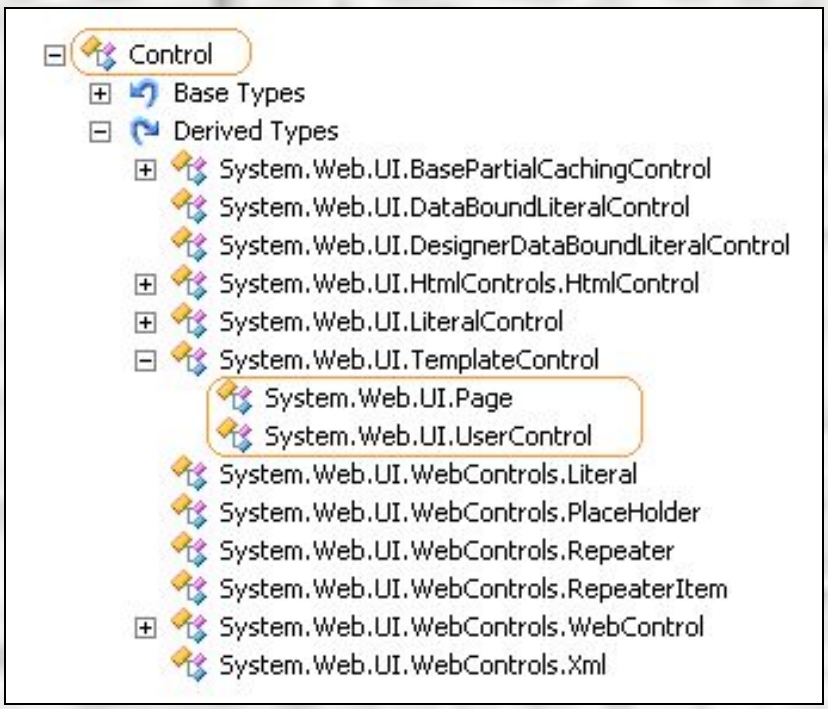



```
private DataGridItem CreateItem(
    int  itemIndex,     int dataSourceIndex, ListItemType     itemType,
    bool dataBind,      object dataItem,     DataGridColumn[] columns,
    TableRowCollection rows, PagedDataSource pagedDataSource
    ) {

    DataGridItem item = this.CreateItem(itemIndex, dataSourceIndex, itemType);
    DataGridItemEventArgs args = new DataGridItemEventArgs(item);
    if (itemType != ListItemType.Pager) {
        this.InitializeItem(item, columns);
        if (dataBind) {
            item1.DataItem = dataItem;
        }
        this.OnItemCreated(args);
        rows.Add(item);
        if (dataBind) {
            item.DataBind();
            this.OnItemDataBound(args);
            item.DataItem = null;
        }
        return item;
    }
    this.InitializePager(item, columns.Length, pagedDataSource);
    this.OnItemCreated(args);
    rows.Add(item);
    return item;
}
```

Все равно происходит вызов метода DataBind определенного в базовом классе System.Web.UI.Control

ASP Page Data Binding

Как мы видим, ВСЕ наследники System.Web.UI.Control связываются с данными по единой схеме – вызов DataBind для родителя будет означать также вызов этого метода для ВСЕХ дочерних контролов. Эту схему удобно использовать для управления данными в жизненном цикле страниц и контролов:

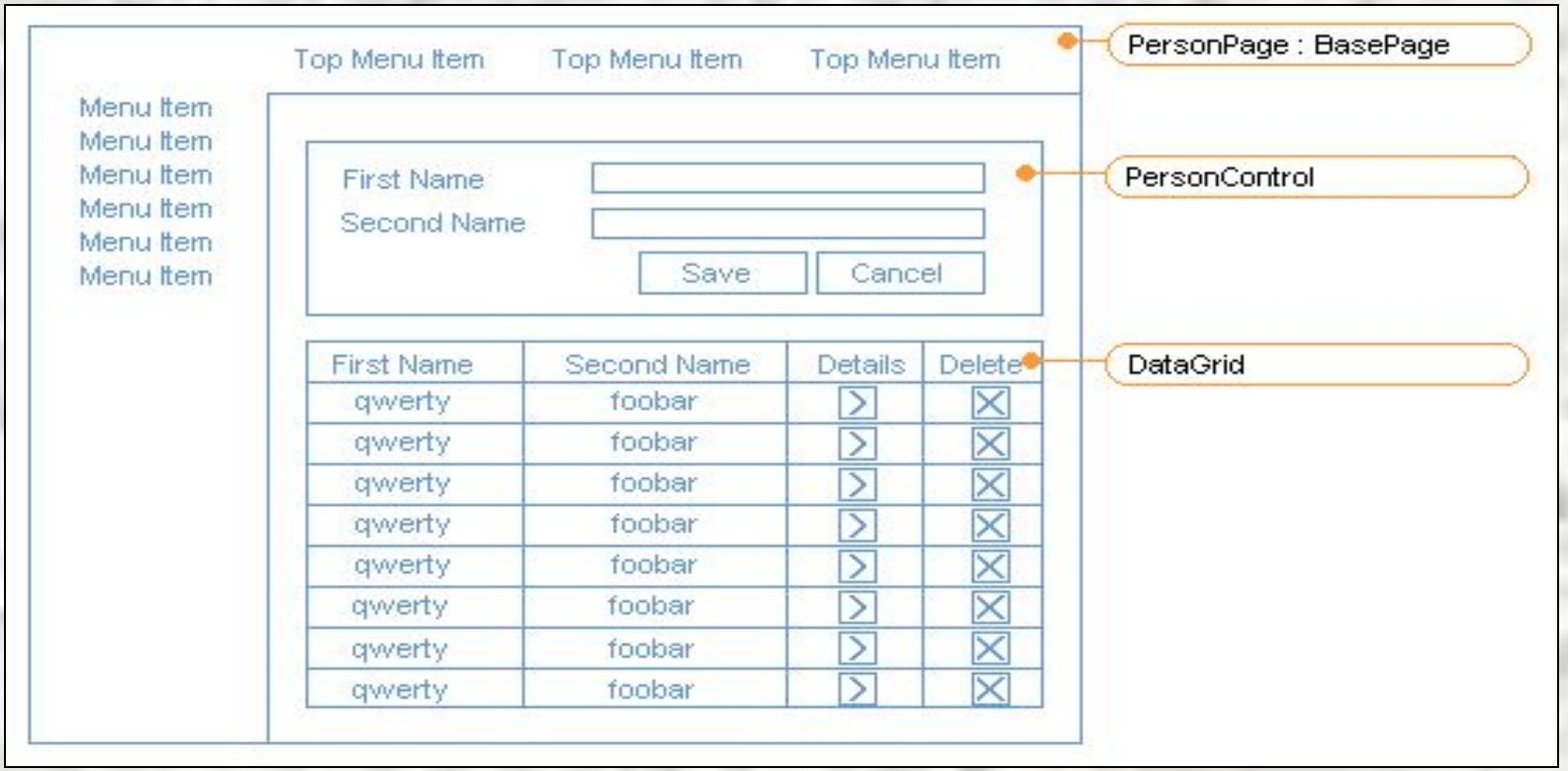


ASP Page Data Binding

Поскольку мы знаем, что вызов DataBind() для страницы, повлечет за собой вызовы DataBind() для всех дочерних контролов (как «обычных», так и реализованных нами), мы может сосредоточить основную логику работы всех страниц приложения в базовом классе.

Это позволит не тратить массу усилий и времени на то, чтобы разобраться, что делает очередной метод «**bindMyControlIfAllOk(true)**» и как расставить if-ы, чтобы заставить «**этого монстра**» работать. ☺ ☺ ☺

Достаточно определить в базовом классе набор удобных виртуальных методов и последовательность их возовов. А затем переопределять их в классах страниц.



ASP Page Data Binding

Например можно реализовать базовый класс следующим образом:

```
public class BasePage : Page {

    // This method can be used to load static data (dictionaries ect.) e.g. populate data
    // in DropDownList. Called once on not post back.
    protected virtual void Initialize() {}

    // This method can be used to load target data (list of entities, entity details ect.)
    // Called once on not post back. Can be called manually to reload data.
    protected virtual void DataLoad() {}

    // This method can be used to set DataSource property
    // for DropDownList, DataGrid ect. Only for controls holding target data.
    protected virtual void SetDataSource() {}

    // This method can be used to setup page layout. Only target data should be used as a source.
    // DataSource property can be used as target data.
    protected virtual void SetLayout() {}

    protected override void OnInit(EventArgs e) {
        base.OnInit(e);
        if (!IsPostBack) {
            Initialize();
        }
    }

    protected override void OnLoad(EventArgs e) {
        if (!IsPostBack) {
            DataLoad();
        }
        SetDataSource();
        base.OnLoad(e);
    }

    protected override void OnPreRender(EventArgs e) {
        SetDataSource();
        SetLayout();
        DataBind();
        base.OnPreRender(e);
    }
}
```

ВСЕ ЗДОРОВО И СРАЗУ?..

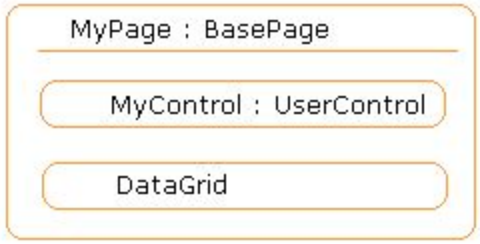
ТАКОГО НЕ БЫВАЕТ!!! ☺️

ASP Page Data Binding

И если в Белорибо еще **bindMyControlIfAllOk()**, то в Белобаджо уже - **DataBind()!**

Или как мигрировать с наименьшими потерями?

Схематично эту страницу можно изобразить так:



Переход должен происходить от «листьев» к «корню» (от контролов к странице). При этом важно помнить о том, **когда** можно **«безопасно»** обрабатывать данные:

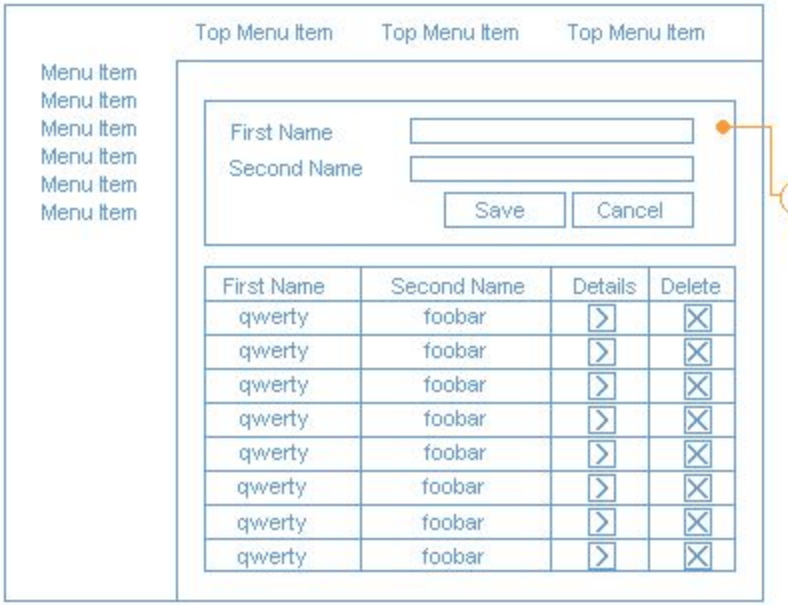
- **Request** — получаем запрос от клиента
- **Load** — подаем данные в контролы (по принципу установки свойства DataSource)
- **Events** — обрабатываем события (меняем данные)
- **PreRender** — снова подаем данные после изменений и вызываем Page.DataBind()
- **Response** — возвращаем клиенту обработанные данные

!!! Контролы не должны хранить данные между PostBack-ами. Данные им подаются со страницы. !!!

ASP Page Data Binding

И если в Белорибо еще **bindMyControlIfAllOk()**, то в Белобаджо уже - **DataBind()!**

Или как мигрировать с наименьшими потерями?



PersonControl 1

```
public override void DataBind() {
    // Обеспечить, чтобы все дочерние
    // контролы получили необходимые данные
    base.DataBind ();
}
```

Переопределив DataBind важно не забыть вызвать base. DataBind, а также обеспечить чтобы к моменту его вызова все необходимые данные были переданы в контрол сверху и обработаны по ссылке.

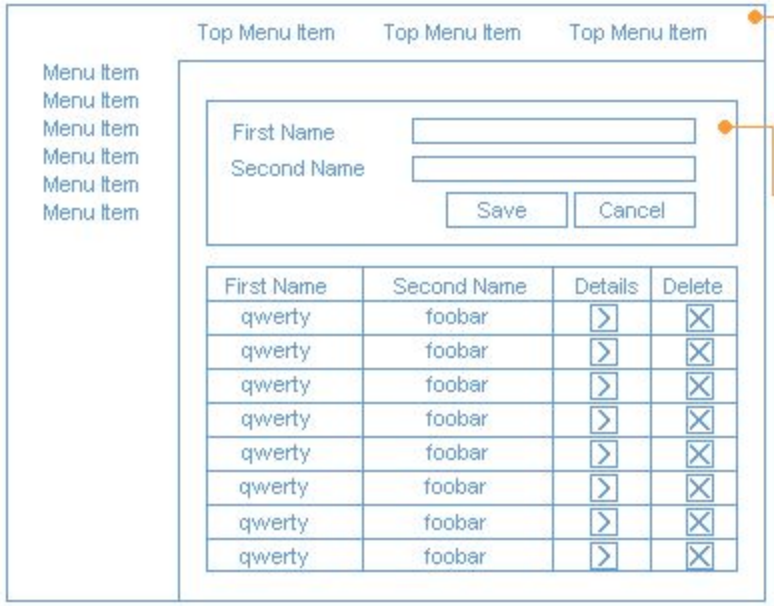
- **Request** — получаем запрос от клиента
- **Load** — подаем данные в контролы (по принципу установки свойства DataSource)
- **Events** — обрабатываем события (меняем данные)
- **PreRender** — снова подаем данные после изменений и вызываем Page.DataBind()
- **Response** — возвращаем клиенту обработанные данные

!!! Контролы не должны хранить данные между PostBack-ами. Данные им подаются со страницы. !!!

ASP Page Data Binding

И если в Белорибо еще **bindMyControlIfAllOk()**, то в Белобаджо уже - **DataBind()!**

Или как мигрировать с наименьшими потерями?



PersonPage (2)

```
public override void DataBind() {
    // Сделать то же самое в классах страниц
    base.DataBind ();
}
```

PersonControl (1)

```
public override void DataBind() {
    // Обеспечить, чтобы все дочерние
    // контролы получили необходимые данные
    base.DataBind ();
}
```

Переопределив DataBind важно не забыть вызвать base. DataBind, а также обеспечить, чтобы его вызов осуществлялся после отработчиков событий (как правило в PreRender- е).

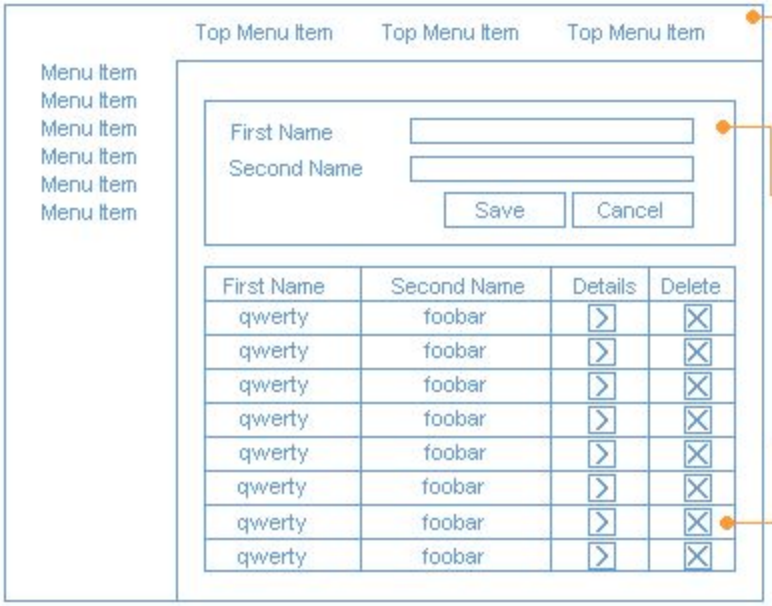
- **Request** — получаем запрос от клиента
- **Load** — подаем данные в контролы (по принципу установки свойства DataSource)
- **Events** — обрабатываем события (меняем данные)
- **PreRender** — снова подаем данные после изменений и вызываем Page.DataBind()
- **Response** — возвращаем клиенту обработанные данные

!!! Контролы не должны хранить данные между PostBack-ами. Данные им подаются со страницы. !!!

ASP Page Data Binding

И если в Белорибо еще **bindMyControlIfAllOk()**, то в Белобаджо уже - **DataBind()!**

Или как мигрировать с наименьшими потерями?



PersonPage (2)

```
public override void DataBind() {
    // Сделать то же самое в классах страниц
    base.DataBind ();
}
```

PersonControl (1)

```
public override void DataBind() {
    // Обеспечить, чтобы все дочерние
    // контролы получили необходимые данные
    base.DataBind ();
}
```

DataGrid

```
// Для "стандартных" контролов метод
// DataBind() будет вызван автоматически
// в процессе работы DataBind() родителя
// (старницы или контрола)
```

Когда страница и все контролы переведены на «новую» схему Binding-а, количество вызовов DataBind-а равно 1-му (в PreRender-е) или 2-м (при первой загрузке в PageLoad-е и в PreRender-е)

- **Request** — получаем запрос от клиента
- **Load** — подаем данные в контролы (по принципу установки свойства DataSource)
- **Events** — обрабатываем события (меняем данные)
- **PreRender** — снова подаем данные после изменений и вызываем Page.DataBind()
- **Response** — возвращаем клиенту обработанные данные

!!! Контролы не должны хранить данные между PostBack-ами. Данные им подаются со страницы. !!!

ASP Page Data Binding

И если в Белорибо еще **bindMyControlIfAllOk()**, то в Белобаджо уже - **DataBind()!**

Или как мигрировать с наименьшими потерями?

Когда все страницы имеют схожую структуру, вынести вызовы DataBind в BasePage класс займет 1 день работы одной пары.

Поскольку рассматриваемый подход естественным образом приводит к тому, что установка данных в контролы выделяется в отдельный метод, также около 1 дня потребуется чтобы определить в базовом классе необходимые виртуальные методы (SetDataSource и т.п.) и использовать их в дочерних классах.

BasePage 3

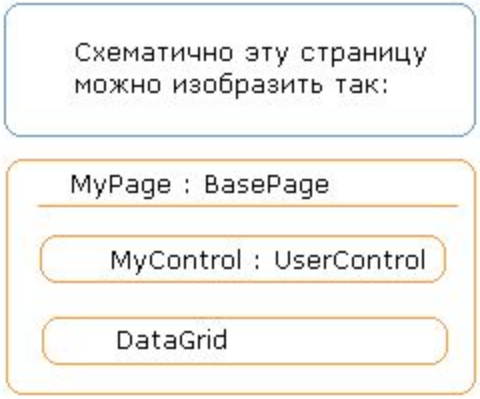
```
// Теперь можно оперативно вынести всю стройную
// схему в базовый класс для страниц.
protected virtual void Initialize() {}
protected virtual void DataLoad() {}
protected virtual void SetDataSource() {}
protected virtual void SetLayout() {}
protected override void OnInit(EventArgs e) {
    base.OnInit(e);
    if (!IsPostBack) {
        Initialize();
    }
}
protected override void OnLoad(EventArgs e) {
    if (!IsPostBack) {
        DataLoad();
    }
    SetDataSource();
    base.OnLoad(e);
}
protected override void OnPreRender(EventArgs e) {
    SetDataSource();
    SetLayout();
    DataBind();
    base.OnPreRender(e);
}
```

ASP Page Data Binding

И если в Белорибо еще **bindMyControlIfAllOk()**, то в Белобаджо уже - **DataBind()!**

Или как мигрировать с наименьшими потерями?

Схематично эту страницу можно изобразить так:



Ура! Мы в Белобаджо!!! ☺

Request — получаем запрос от клиента
Load — подаем данные в контролы (по принципу установки свойства DataSource)
Events — обрабатываем события (меняем данные)
PreRender — снова подаем данные после изменений и вызываем Page.DataBind()
Response — возвращаем клиенту обработанные данные

!!! Контролы не должны хранить данные между PostBack-ами. Данные им подаются со страницы. !!!

ASP Page Data Binding and Validation

Использование стандартной схемы для связывания данных:

- облегчает написание кода страниц и контролов с нуля
- СИЛЬНО облегчает модификацию уже имеющегося GUI кода
- позволяет вынести большую часть общей логики в базовые классы
- «очищает» GUI код от многочисленных и часто избыточных вызовов методов типа «bindMyControlIfAllOk()»
- облегчает чтение и понимание GUI кода
- позволяет четко разделить такие блоки как, например, валидация, установка DataSource-ов, разграничение прав доступа и т.п.
- значительно упрощает рефакторинг

Поговорим?