Getting Started with xRP Framework

Cloud xRP Summit

Virtual Developer Conference

Sergey Marenich

Acumatica

The Cloud ERP

Marenich Sergey

Email: smarenich@acumatica.com

Blog: http://asiablog.acumatica.com

Experience: 12 years of experience at Acumatica as a developer, solution architect, commerce team lead.

Specialization: As a developer, he specializes in C# with deep expertise in Microsoft technologies. Apart from development work, Sergey is also experienced in the field of training & development consulting as well as customer relationship management.

Agenda

Getting Started

Acumatica Platform Architecture

Acumatica Framework Essentials

- Data Access Classes, Business Logic Controllers
- Data Querying & Events Model
- User Interface
- Customizations

Further Steps

Getting Started

Getting Started Guide

https://www.acumatica.com/blog/quick-start-acumatica-developer-guide/

Steps:

- Get Prerequisites
  - SQL, IIS, Git, Visual Studio
- Learn
  - C#, .NET, SQL, ASP.Net
- Download Acumatica Installer & Install
  - with Debugger Tools for Acumatica
- Deploy Acumatica Instance with Demo Data

Nota Bene!

No License for Development with Trial Mode

- All Features are Accessible
- 2-Concurent users Per Instance

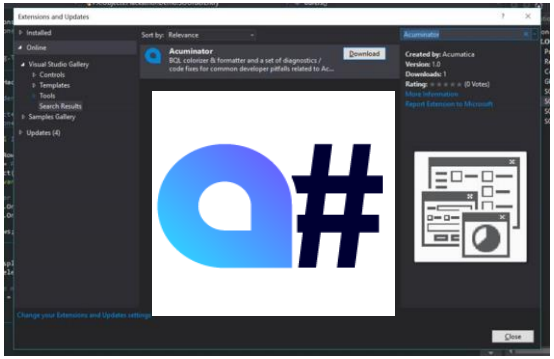
Acumatica ADN Subscription

- Development support
- More development licenses

Acuminator Visual Studio extension for Acumatica Framework

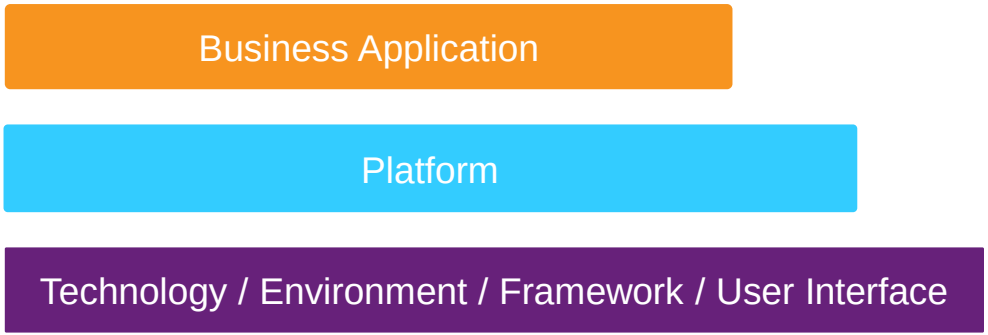
- Static code analysis
- Colorizer
- Suggestions tool
- Code Map

Acumatica Developer Network



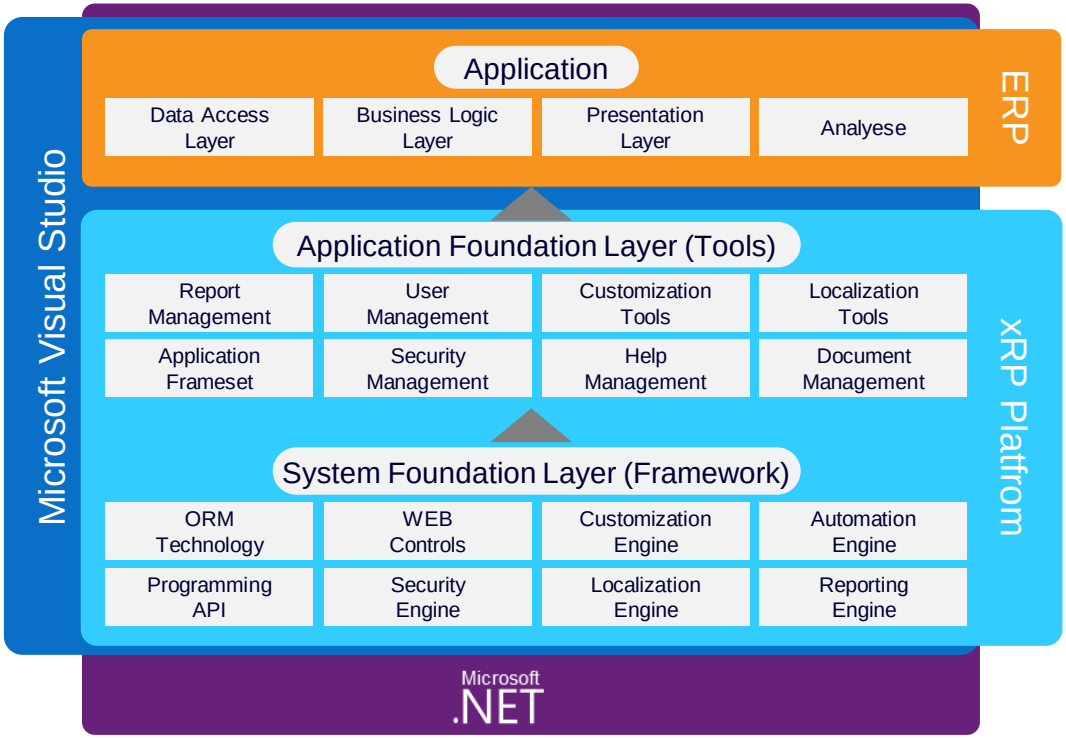
Acumatica Platform Architecture

xRP Platform Origins



- **Code Standardization –** it is easier to support standardized code by any developer who knows it
- **Speedup Development –** with high level primitives and tools
- **Protect form Technology changes –** platform hides underlaying technology and allows to change it without touching business logic code.
- **Share Tools with Partners –** involve all other companies to build new world together

Acumatica Platform Architecture



- **Single Runtime Environment** – Microsoft .NET
- **Single Development Environment** – Microsoft Visual Studio.
- **System Foundation Layer** – Set of high quality components and libraries used for the web application development.
- **Application Foundation Layer** – application frameset, development tools and set of pre-build web applications on top of system foundation layer.
- **Acumatica Application** – web application developed using system foundation layer and integrated into application foundation layer.

Acumatica 6-tiers Web Architecture

Users

Internet

Application Server

Database Server

Infrastructure

Web Browser / Mobile / API Client

TLS 1.2

Network

Over HTTPS

IIS (Web Server)

ASP.Net Web Forms REST/SOAP Web API

Entity Business Logic Controller

Data Access Classes (DAC, BQL)

MSSQL / MYSQL / Aurora / Azure

Tiers

Client

Web

Presentation

Business

Data Access

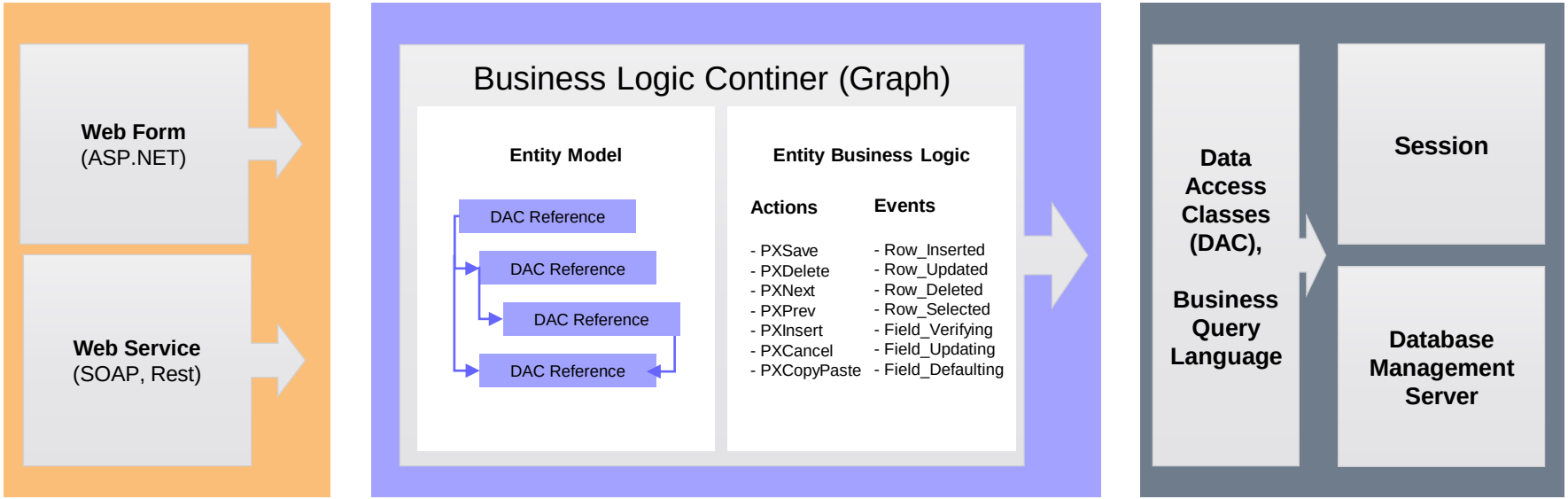
Database

HTML 5
JavaScript

Acumatica Cloud xRP Platform
Microsoft .Net Framework
C# Language

DB Tables,
Keys, Indexes
SQL

Acumatica Framework Architecture

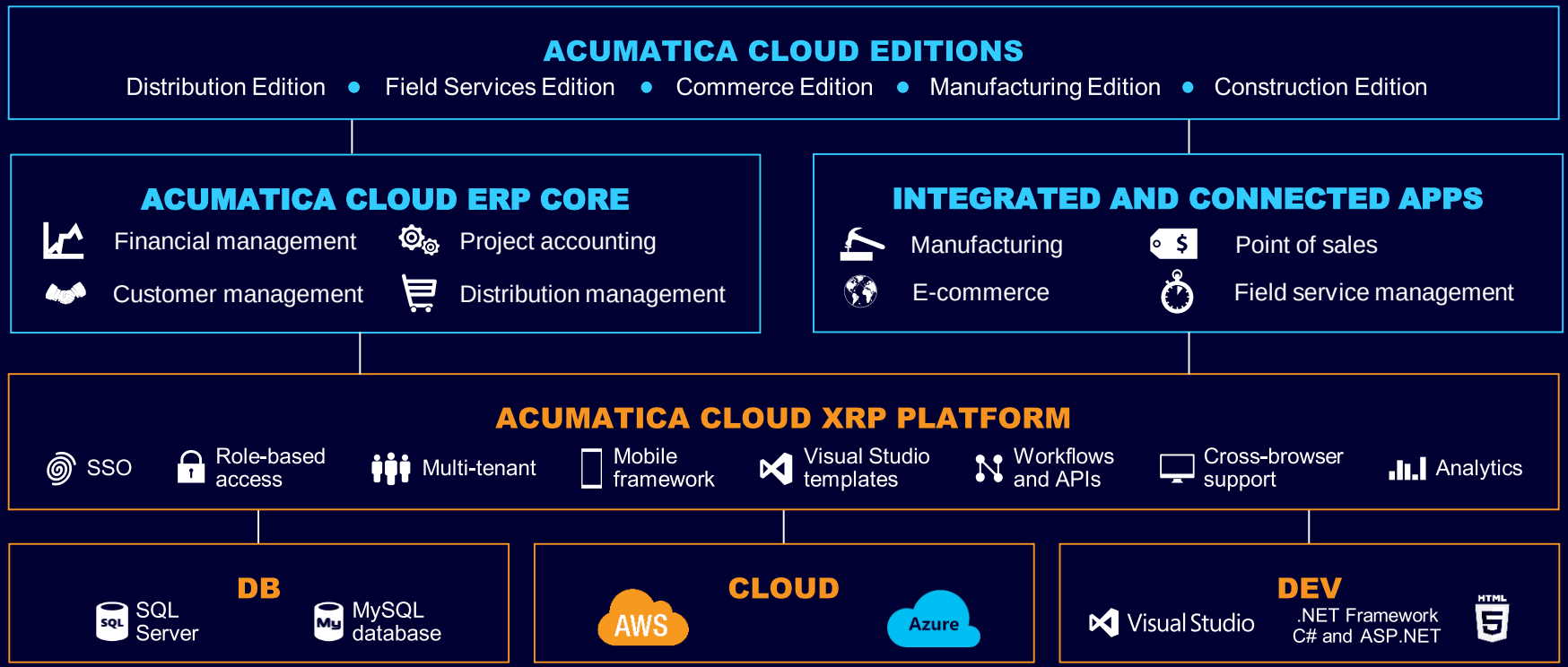


Presentation Layer

Business Logic Layer

Data Access Layer

Acumatica Cloud xRP Platform



Acumatica
The Cloud ERP

Acumatica Framework Essentials

Acumatica Platform Essentials

- Data Access Classes (DAC)
- Business Logic Containers (BLC or Graph)
- Data Views
- Business Query Language (BQL)
- User Interface (ASP.NET)
- Events
- Customizations (Extensions)

Data Access Classes (DAC)

DAC - Object-Relational Mapping

SQL

```
CREATE TABLE [dbo].[Product]
(
  //...
  [ProductCD] [nvarchar](15) NOT NULL,
  //...
  [Active] [bit] NOT NULL,
  //...
  CONSTRAINT [Product_PK]
    PRIMARY KEY CLUSTERED
  (
    [ProductCD] ASC
  )
)
```

DAC

```
public class Product : PX.Data.IBqlTable
{
        //...
        public abstract class productCD : PX.Data.IBqlField {}
        [PXDBString(15, IsUnicode = true, IsKey = true)]
        [PXDefault]
        [PXUIField(DisplayName = "Product ID")]
        [PXSelector(
               typeof(Search<Product.productCD>),
               typeof(Product.productCD),
               typeof(Product.unitPrice))]
        public string ProductCD { get; set; }
        //...
        public abstract class active : PX.Data.IBqlField { }
        [PXDBBool()]
        [PXDefault(true)]
        [PXUIField(DisplayName = "Active")]
        public bool? Active { get; set; }
        //...
}
```

DAC - Definition

Every DAC:

- Mapped to table
- Has field properties
- Has field abstract classes

```
public class Product : PX.Data.IBqlTable
public string ProductCD { get; set; }
public abstract class productCD : PX.Data.IBqlField {}
```

Fields:

- Have keys “IsKey = true”
- Have attributes

```
[PXDBString(15, IsUnicode = true, IsKey = true)]
[PXDefault(true)]
[PXUIField(DisplayName = "Active")]
```

Attributes – Declarative Programming

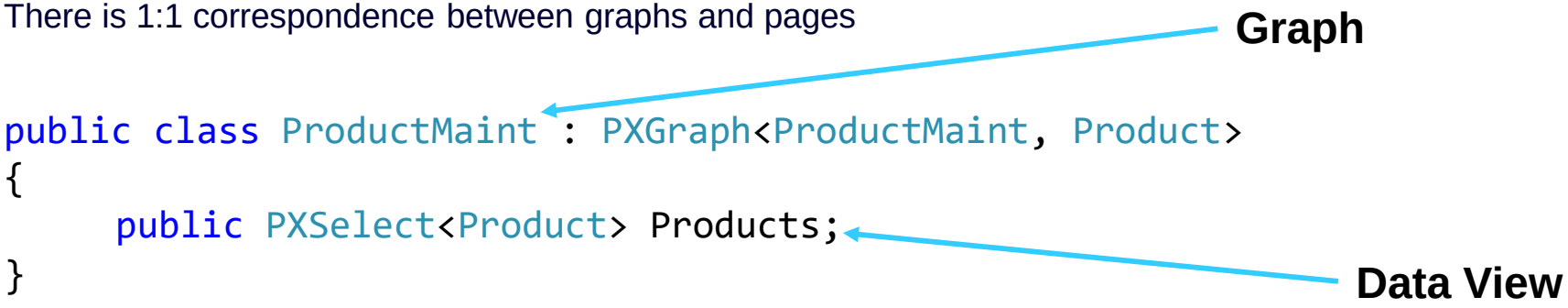
Attribute	Meaning
[PXDBString(3, IsKey = true, InputMask="CCC")]	Database mapped field of type string
[PXBool()]	Virtual field of type bool
[PXDefault("Open")]	Provides default value for field
[PXDefault(PersistingCheck = PXPersistingCheck.NullOrBlank)]	Makes field required
[PXUIField(DisplayName = "Product ID", Enabled = true)]	Defines name of the field in UI
[PXUIField(Enabled = false)]	Makes field disabled in UI
[PXSelector(typeof(Search<Product.productCD>))]	Defines foreign key reference and selector
[PXStringList(new string[] { "H", "C" }, new string[] { "On Hold", "Canceled" })]	Defines list of predefined values (drop-down)
[PXFormula(typeof(Mult<Transaction.qty, Transaction.unitPrice>))]	Brings auto-calculation rules that will be executed automatically.

Business Logic Containers (BLC or Graph)

Business Logic Containers (BLC or Graph)

Controllers encapsulating Business Logic and Data Views

There is 1:1 correspondence between graphs and pages



Graphs controls Business logic with:

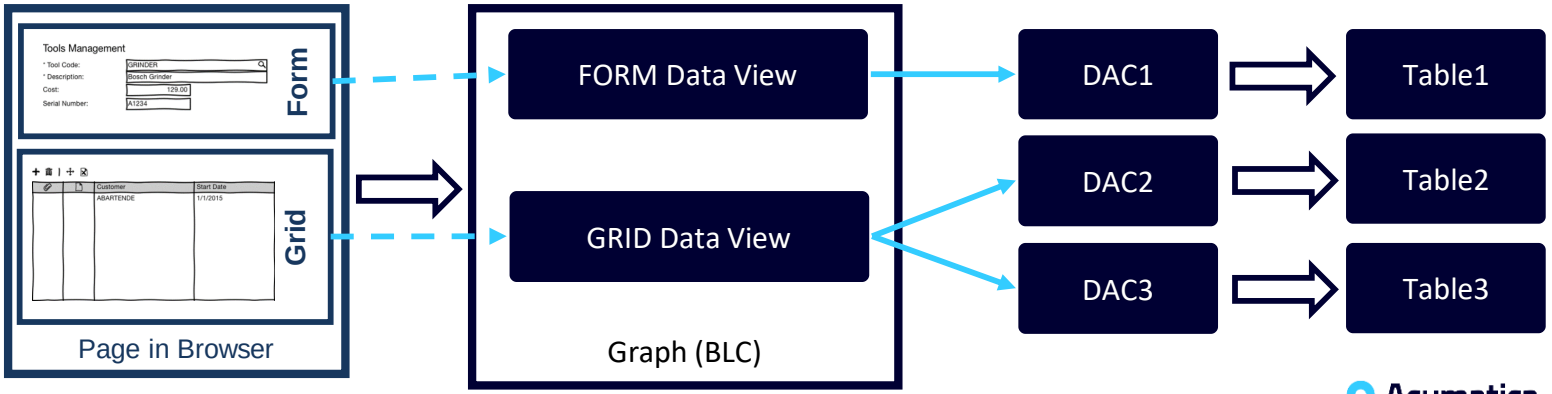
- Actions (Buttons)
- Events

Data Views

Data Views

Data Views:
```
public PXSelect<Product> Products;
public PXSelect<DocTran, InnerJoin<Product, On<...>>> Trans;
```

- Maintain changes in the Session with PXCache<Batch>
- Selects data from the Database with PXView<Batch>
- Provides data for User Interface

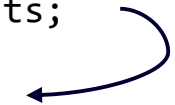


Business Query Language (BQL)

Business Query Language (BQL)

Data View - `PXSelect<Product, Where<Product.active, Equals<True>>> Products;`

BQL Query - `typeof(Select<Product, Where<Product.active, Equals<True>>>)`



- Is a part of Acumatica Data Access Layer
- Is mapped to SQL queries
- Hides the underlying database engine
- Is checked at compile time
- Comes with a variety of clauses allowing to express most DB queries

BQL – Business Query Language

A simple query can look like:

```
public PXSelect<Product> Products;
```

Result SQL:

```
SELECT Product.ProductCD, Product.Active, ...
FROM Product Product
ORDER BY Produc.ProductCD
```

BQL – Business Query Language

```
public PXSelectJoin<SupplierProduct,

        InnerJoin<Supplier,
                On<Supplier.supplierID, Equal<SupplierProduct.supplierID>>>,

        Where2<
                Where<Current<SupplierFilter.countryCD>, IsNull,
                        Or<Supplier.countryCD, Equal<Current<SupplierFilter.countryCD>>>>,
                And<Where<Current<SupplierFilter.minOrderQty>, IsNull,
                        Or<SupplierProduct.minOrderQty,
                                GreaterEqual<Current<SupplierFilter.minOrderQty>>>>>>,

        OrderBy<Asc<SupplierProduct.productID,
                Asc<SupplierProduct.supplierPrice,
                Desc<SupplierProduct.lastPurchaseDate>>>>> Products;
```

SELECT … FROM …

JOIN … ON …

WHERE (… OR …) AND (… OR …)

(…)

… > "VALUE"

ORDER BY … ASC, … DESC

BQL – Querying Data

Passing Parameters – Required:

```
foreach(Product record in PXSelect<Product,
      Where<Product.isActive,
            Equal<Required<Product.isActive>>>>
      >.Select(this, true));
```

Using Context - Current:

```
Product record = PXSelect<Product,
      Where<Product.productID,
            Equal<Current<Tran.productID>>>>
      >.Select(this));
```

Acuminator

- Validation
- Formatting
- Colorizing

```
public PXProcessingJoin<
    BalancedARDocument,
    LeftJoin<ARInvoice,
        On<ARInvoice.docType, Equal<BalancedARDocument.docType>,
        And<ARInvoice.refNbr, Equal<BalancedARDocument.refNbr>>>,
    LeftJoin<ARPayment,
        On<ARPayment.docType, Equal<BalancedARDocument.docType>,
        And<ARPayment.refNbr, Equal<BalancedARDocument.refNbr>>>,
    InnerJoinSingleTable<Customer,
        On<Customer.bAccountID, Equal<BalancedARDocument.customerID>>,
    LeftJoin<ARAdjust,
        On<ARAdjust.adjgDocType, Equal<BalancedARDocument.docType>,
        And<ARAdjust.adjgRefNbr, Equal<BalancedARDocument.refNbr>,
        And<ARAdjust.adjNbr, Equal<BalancedARDocument.adjCntr>,
        And<ARAdjust.hold, Equal<boolFalse>>>>>>>>>,
    Where2<Match<Customer, Current<AccessInfo.userName>>,
        And<ARRegister.hold, Equal<boolFalse>,
        And<ARRegister.voided, Equal<boolFalse>,
        And<ARRegister.scheduled, Equal<boolFalse>,
        And<Where<BalancedARDocument.released, Equal<boolFalse>,
            And<BalancedARDocument.origModule, Equal<GL.BatchModule.moduleAR>,
            Or<BalancedARDocument.openDoc, Equal<boolTrue>,
            And<ARAdjust.adjdRefNbr, IsNotNull>>>>>>>>>>
    ARDocumentList;
```

Next Generation of BQL

Fluent BQL:

```
SelectFrom<Detail>.
InnerJoin<Document>
        .On<Detail.docNbr.IsEqual<Document.docNbr>>.
Where<Document.bAccountID.IsEqual<@P.AsInt>.
        And<AccessInfo.businessDate.FromCurrent.
                Diff<Document.date>.Days.IsGreater<TwentyEight>>>.
OrderBy<Detail.itemID.Asc, Document.date.Desc>
```

Linq to BQL

```
var results = graph
    .Select<CRCase>()
    .FirstOrDefault(c => c.CaseCD == “000123”);
```

User Interface (ASP.NET)

User Interface - ASPX

User Interface in Acumatica controls by ASPX files and LayoutRules

```
<px:PXFormView ID="form" DataMember="Products">
<Template>
      <px:PXLayoutRule StartRow="True"/>
      <px:PXSelector ID="edProductCD" DataField="ProductCD" />
      <px:PXTextEdit ID="edProductName" DataField="ProductName" />

      <px:PXLayoutRule StartColumn="True"/>
      <px:PXCheckBox ID="edActive" DataField="Active"/>
      <px:PXNumberEdit ID="edUnitPrice" DataField="UnitPrice"/>
</Template>
</px:PXFormView>
```

Data View

Relative Positioning Rules

DAC Field

Attributes defines UI Controls

Control	Attributes on the DAC Field	ASPX Definition
Text box	[PX(DB)String]	<px:PXTextEdit ID= ... />
Number edit box	[PX(DB)Int] or [PX(DB)Decimal]	<px:PXNumberEdit ID=... />
Mask edit box	[PX(DB)String(InputMask =...)]	<px:PXMaskEdit ID=... />
Drop-down list	[PXStringList] or [PXIntList]	<px:PXDropDown ID=... / >
Selector	[PXSelector]	<px:PXSelector ID=... />
Check box	[PX(DB)Bool]	<px:PXCheckBox ID=... />
Date-time picker	[PX(DB)Date]	<px:PXDateTimeEdit ID=... />
Time span edit box	[PXDBTimeSpan]	<px:PXDateTimeEdit ID=... TimeMode="True" />

Commit Changes

Normally server doesn't get notified of every change to data made by user
You can make certain controls post changes to server once they occur:

```
<px:PXFormView ID="form" ... >
      <Template>
            <px:PXCheckBox ID="chkHold"
                           runat="server"
                           DataField="Hold"
                           CommitChanges="True" />
```

The same for grid columns:

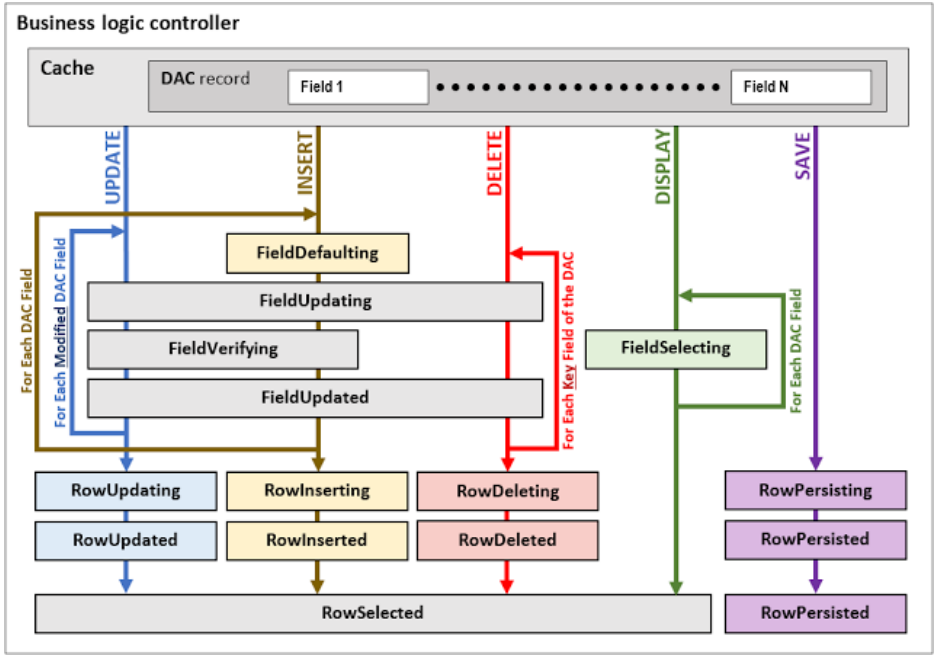
```
<px:PXGridLevel ... >
      <RowTemplate>
            <px:PXGridColumn DataField="ProductID"
                             CommitChanges="True" />
```

Events

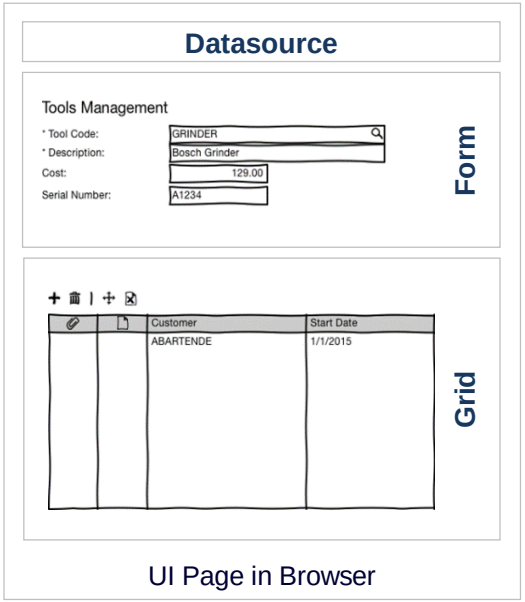
Understanding The Event Model

Acumatica events:

- 12 Row-level events
- 5 Field-level events



Where do events come from?



UI Page in Browser

Get (HTML, Scripts)

Callbacks (Data, States)

Get, Post (XML)

Here is no events, just data.

```
<ctl00_phF_form_edBatchNbr_pnl_gr>
<Rows Level="0" HashCode="">
 <Row i="0">
 <Cells>
 <Cell Value="GL"/>
 <Cell Value="GL00090“ ReadOnly="False"/>
 <Cell Value="STAT" ReadOnly="False"/>
 <Cell Value="032013" Text="03-2013"/>
 <Cell Value="P" Text="Posted"/>
 </Cells>
 </Row>
</Rows>
```

```
<px:PXDataSource ID="ds“ TypeName="JournalEntry"
Visible="True" PrimaryView="BatchModule“/>
```

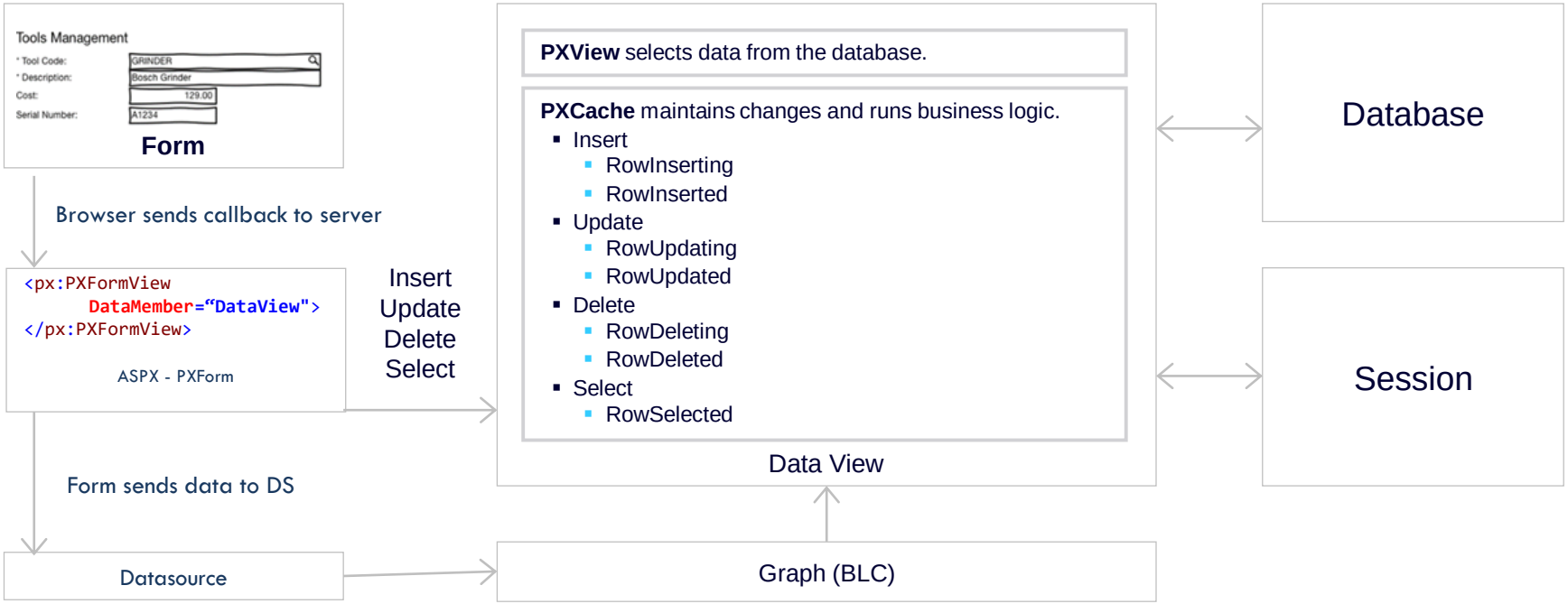
```
<px:PXFormView ID="PXFormView1" DataMember="BatchModule" >
 <Template>
 <px:PXLayoutRule StartColumn="True"/>
 <px:PXDropDown ID="DropDown1" DataField="Module"/>
 <px:PXSelector ID="Sele1" DataField="BatchNbr"/>
 <px:PXCheckBox ID=“cbReverse" DataField="Reverse"/>
 </Template>
</px:PXFormView>
```

```
<px:PXGrid ID="PXGrid1" SkinID="Details" >
 <px:PXGridLevel DataMember="GLTranModuleBatNbr">
 <Columns>
 <px:PXGridColumn DataField="LineNbr"/>
 <px:PXGridColumn DataField="BranchID"/>
 <px:PXGridColumn DataField="AccountID"/>
 </Columns>
 <Mode InitNewRow="True" AllowFormEdit="True"/>
 </px:PXGridLevel>
</px:PXGrid>
```

ASPX Page on Server

Where do events come from?

Where do events come from?



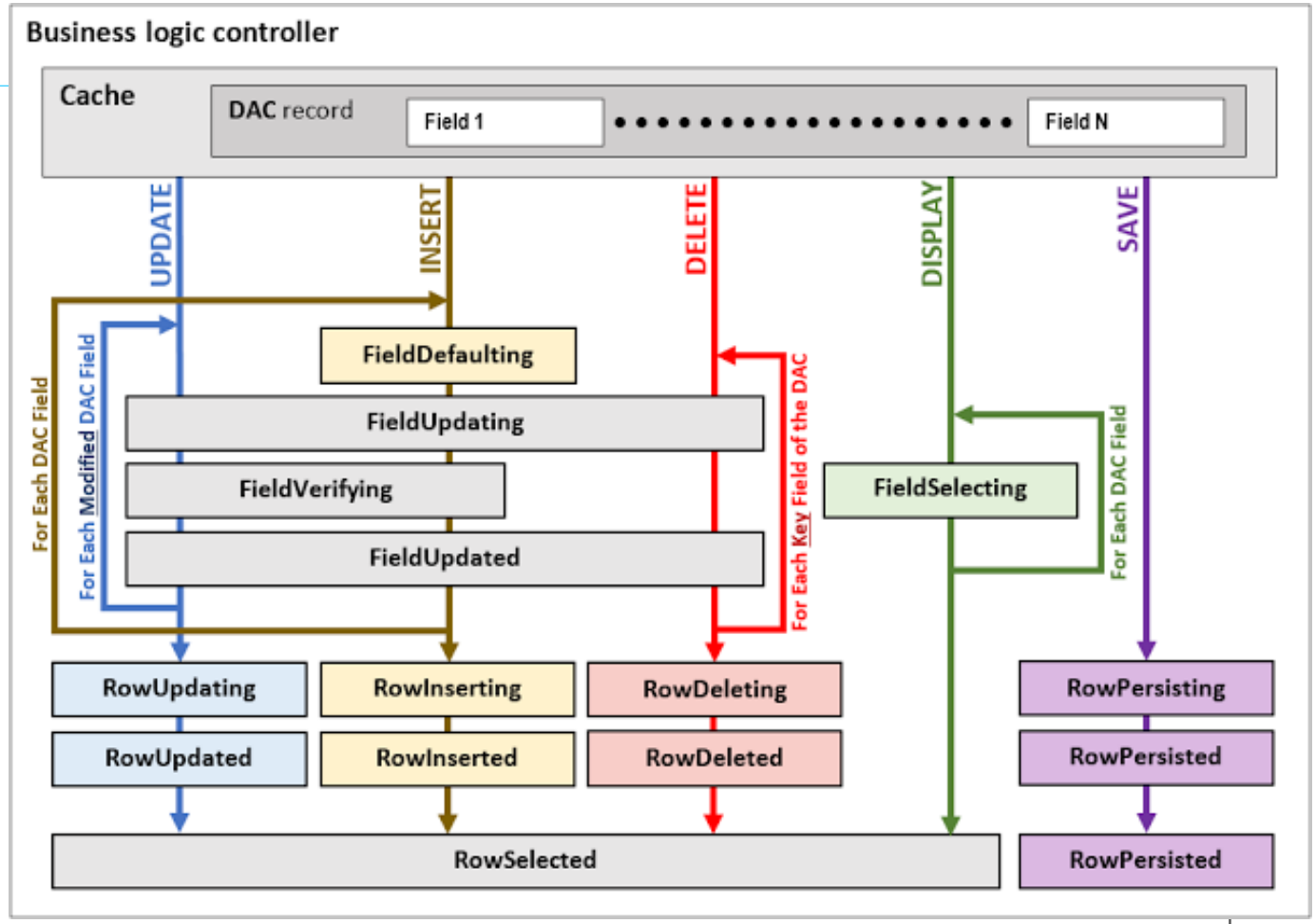
Commonly Used Events

`RowUpdating` – occurs before updating of cached record
`RowUpdated` – occurs after updating of cached record

`RowSelecting` – occurs system reads record from database reader
`RowSelected` – occurs when system should show record to the UI

`FieldDefaulting` – occurs when system calc default value for the field.
`FieldVerifiyng` – occurs when need to validate input value.
`FieldUpdated` – occurs when the field value is changed.

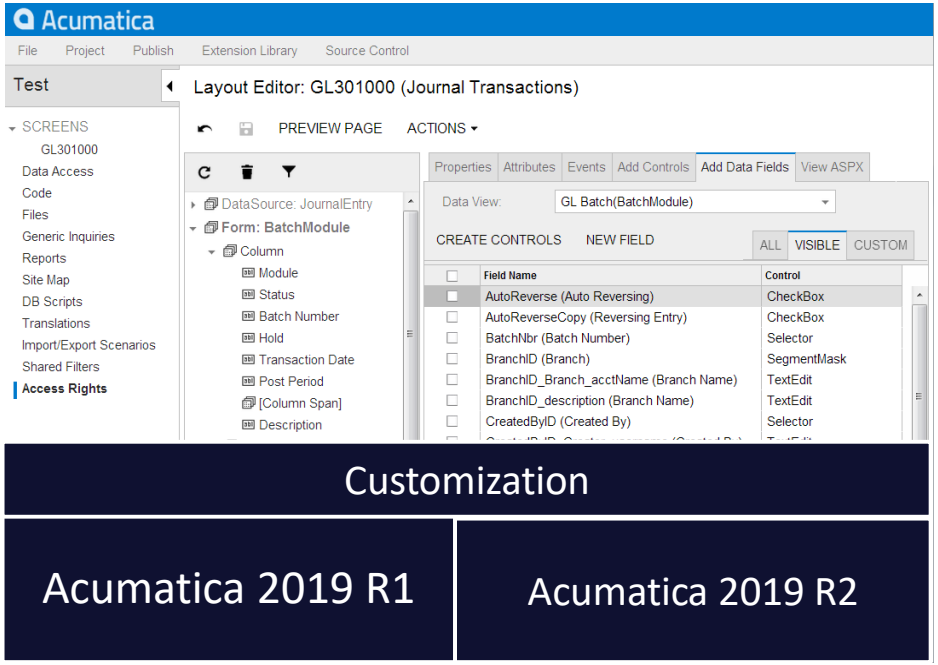
The Event Model



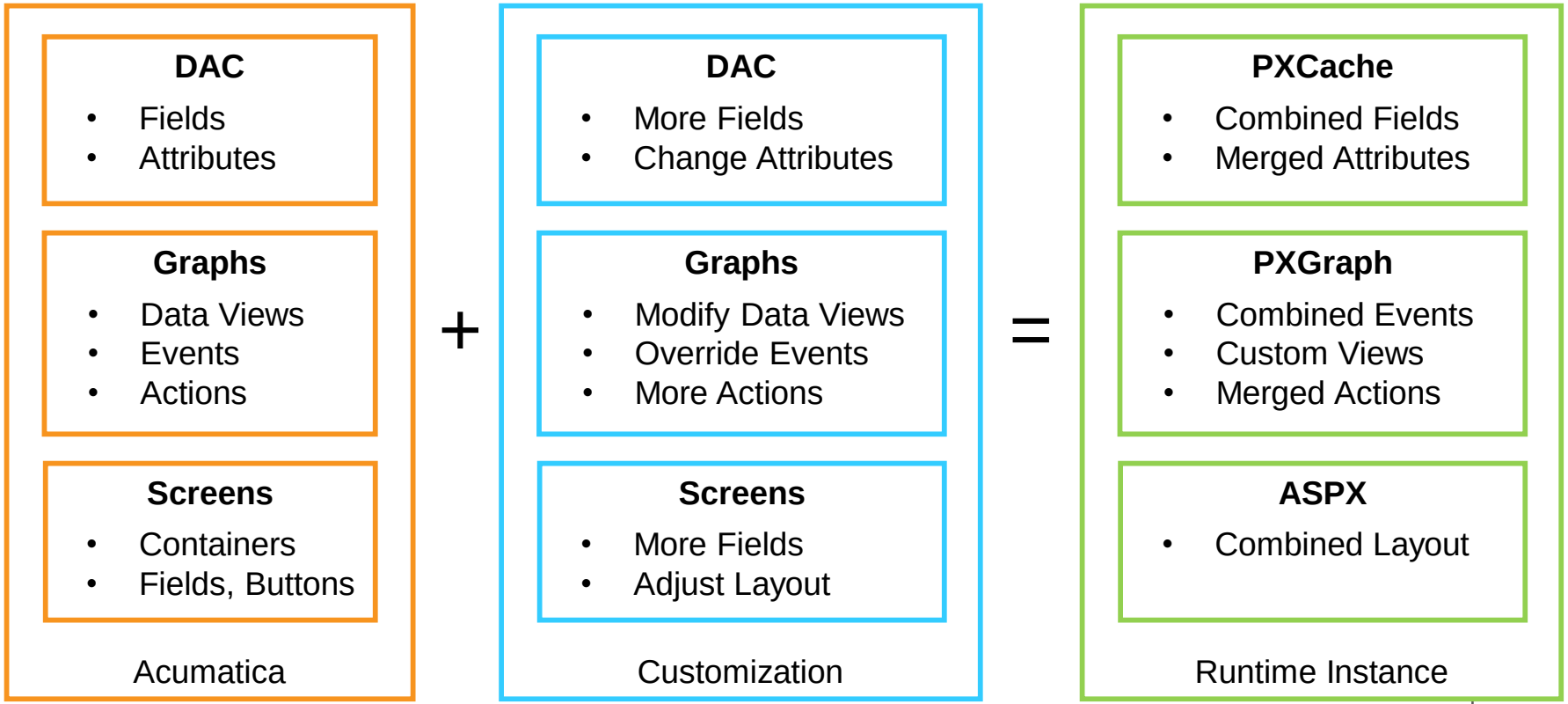
Customizations

Customization as Plugins

- Easy way to Deploy
  - Attach and Detach
  - Upgrade Proof
- Develop Independently from Acumatica
  - Use Visual Studio
  - Keep in Source Control
- Customize everything:
  - User Interface
  - Business Logic
  - Database



Need Customize? – Extend!



Acumatica Extensibility Framework

```
public class MyDACExtension : PXCacheExtension<MyDAC>
{
	//Put new fields definition here
	//Customize existing attributes and fields
}

public class MyGraphExtension : PXGraphExtension<MyGraph>
{
	//Put new event handlers, actions, data views or methods here
	//Customize existing logic with defining new one with the same name
}
```

Most Interesting Usage of Extensions

Getting Extensions

```
PXCache<Product>.GetExtension<ProductExt>(row);
sender.GetExtension<ProductExt>(row);
row.GetExtension<ProductExt>();
```

Overriding Methods

```
[PXOverride]
public bool Validate(string value, Func<string, bool> baseMethod)
{ return baseMethod(value); }
```

Overriding Events

```
public void DAC_Filed_Event(PXCache sender, PXEventTypeEventArgs e, PXEventType del)
```

Override Action / Data View

```
public PXAction<DAC> ValidateObjects;
public PXSelect<DAC> Objects
```

Further Steps

What to do Next?

Learn Acumatica Framework and Tools:

- https://www.acumatica.com/acumatica-developer-training/
- Takes about 2-3 weeks for a developer

Customization Guide:

- https://help.acumatica.com/Main?ScreenId=ShowWiki&pageid=316b14fa-f406-4788-993c-7b043b1c5bd9

Learn from others, participate!

- https://github.com/Acumatica/
- http://stackoverflow.com/questions/tagged/acumatica
- http://asiablog.acumatica.com/

Acumatica
The Cloud ERP

Sergey Marenich

smarenich@acumatica.com
http://asiablog.acumatica.com

No Reliance
This document is subject to change without notice. Acumatica cannot guarantee completion of any future products or program features/enhancements described in this document, and no reliance should be placed on their availability.

Confidentiality: This document, including any files contained herein, is confidential information of Acumatica and should not be disclosed to third parties.