APPEON®

PowerBuilder Roadmap

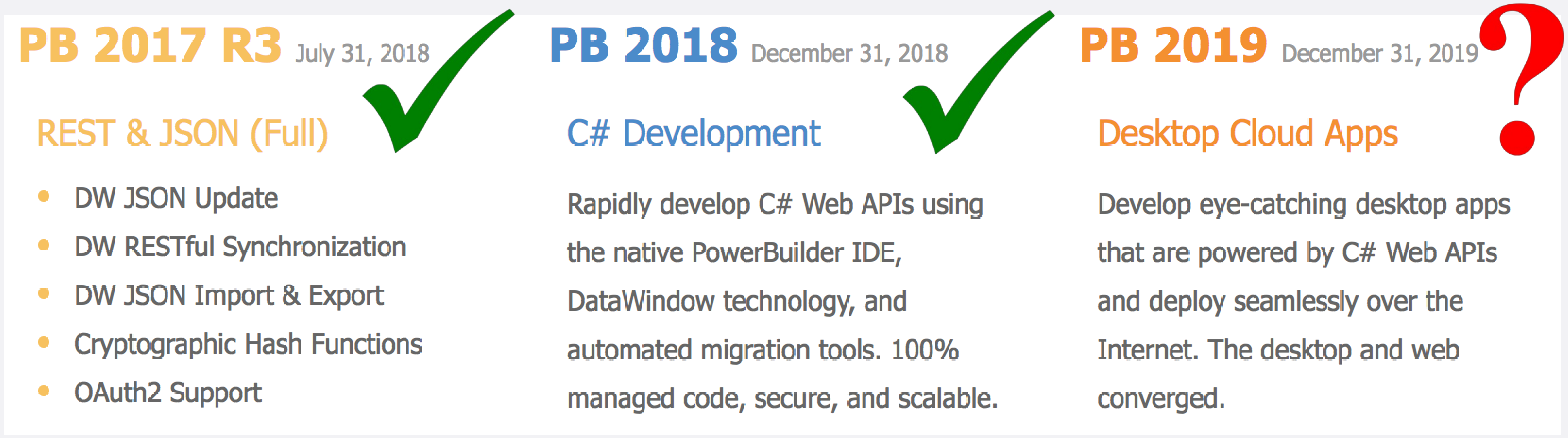
A simpler, faster, open-standards approach to the cloud for .NET!

Armeen Mazda, Appeon CEO
June 11, 2018

©2018 Appeon Limited and its subsidiaries. All rights reserved.

Roadmap
Status Update

A simpler, faster, open-standards approach to the cloud for .NET!



PB 2017 R3 July 31, 2018 ✓

REST & JSON (Full)

- DW JSON Update
- DW RESTful Synchronization
- DW JSON Import & Export
- Cryptographic Hash Functions
- OAuth2 Support

PB 2018 December 31, 2018 ✓

C# Development

Rapidly develop C# Web APIs using the native PowerBuilder IDE, DataWindow technology, and automated migration tools. 100% managed code, secure, and scalable.

PB 2019 December 31, 2019 ?

Desktop Cloud Apps

Develop eye-catching desktop apps that are powered by C# Web APIs and deploy seamlessly over the Internet. The desktop and web converged.

C# Development in PowerBuilder 2018

Low Learning Curve

As little as 2 weeks.

Easy Migration

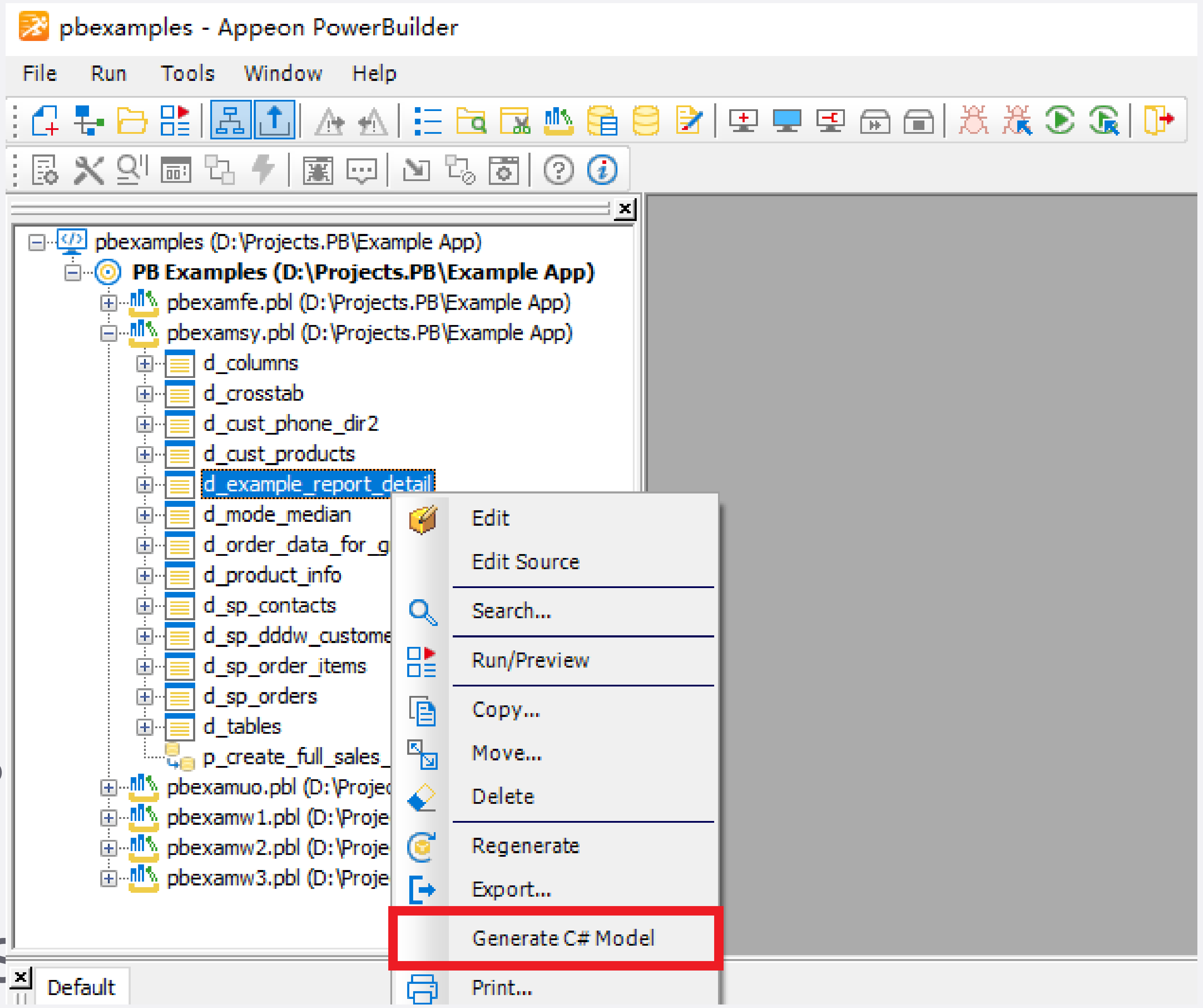
Migration tools provided.

Less Coding & Testing

Even less than PowerScript.

Learning Curve and Migration

- Using the familiar native PowerBuilder IDE
- C# programming centered around the DataWindow
- Offers practically the same properties & functions
- SQL annotations does not require strong SQL expertis
- Automated migration of DataWindows to C# objects



APPEON
www.appeon.com

Less Coding, Less Testing

- ✓ Visual like before, but less coding than PowerScript
  - More powerful data access object layer
  - Separation of concerns (SQL in data model)
- ✓ Automatic validation of SQL syntax <u>and</u> operations
- ✓ Simpler code makes it easy to write fast code
- ✓ No recoding when switching databases (SQL annotations)

C# Programming with DataWindow

```
public function datastore of_retrieve (date ad_start, date ad_end, decimal adec_amt);
Datastore lds
lds = Create Datastore
lds.dataobject = "d_order_customer"
lds.SetTransObject(SQLCA)
lds.Retrieve(ad_start, ad_end, adec_amt)
Return lds
end function
```

C#

```
public IDataStore GetOrderCustomerInfo(DateTime startDate, DateTime endDate, decimal amount)
{
    IDataStore dataStore = new DataStore("d_order_customer", _context);
    dataStore.Retrieve(startDate, endDate, amount);
    return dataStore;
}
```

C# Programming with Entity

```csharp
public IEnumerable<D_Order_Customer> GetOrderCustomerInfo(DateTime startDate, DateTime endDate, decimal amount)
{
    var query = from c in _context.Set<EFCore_Customer>()
                from p in _context.Set<EFCore_Person>()
                from s in _context.Set<EFCore_SalesOrderHeader>()
                where c.PersonId == p.Businessentityid &&
                      s.CustomerId == c.CustomerId &&
                      s.OrderDate >= startDate &&
                      s.OrderDate <= endDate
                group s.SubTotal by new
                {
                    p.Title,
                    p.Firstname,
                    p.Middlename,
                    p.Lastname,
                    c.ModifiedDate,
                    c.CustomerId
                }
                into g
                let avg = g.Average()
                where avg > amount
                orderby avg
                select new D_Order_Customer
                {
                    Person_Title = g.Key.Title,
                    Person_Firstname = g.Key.Firstname,
                    Person_Middlename = g.Key.Middlename,
                    Person_Lastname = g.Key.Lastname,
                    Customer_Modifieddate = g.Key.ModifiedDate,
                    Customer_Customerid = g.Key.CustomerId,
                    Sumamt = g.Sum(),
                    Avgamt = avg
                };

    return query.ToList();
}
```

C# Programming with Entity

```
//code of model
public class D_Order_Customer
{
    [SqlColumn("Title")]
    public String Person_Title { get; set; }

    [SqlColumn("FirstName")]
    public String Person_Firstname { get; set; }

    [SqlColumn("MiddleName")]
    public String Person_Middlename { get; set; }

    [SqlColumn("LastName")]
    public String Person_Lastname { get; set; }

    [SqlColumn("ModifiedDate")]
    public DateTime Customer_Modifieddate { get; set; }
    [Key]
    [Identity]
    [SqlColumn("CustomerID")]
    public Int32 Customer_Customerid { get; set; }

    [SqlColumn("sumamt")]
    public Decimal Sumamt { get; set; }

    [SqlColumn("avgamt")]
    public Decimal Avgamt { get; set; }
}
```

Data Access Layer in PowerScript

```
public subroutine of_get_customerrank (date ad_startdate, date ad_enddate, ref str_customerrank astr_rank[])

Long ll_row
Long ll_customerId
Dec lde_AvgAmt
Dec lde_SumAmt

DECLARE cur CURSOR FOR
    SELECT TOP 10000
         Sales.Customer.CustomerID,
         Sum(Sales.SalesOrderHeader.SubTotal) as SumAmt,
         Avg(Sales.SalesOrderHeader.SubTotal) as AvgAmt
    FROM Sales.Customer,
         Sales.SalesOrderHeader
    WHERE Sales.SalesOrderHeader.CustomerID = sales.customer.customerID AND
         Sales.SalesOrderHeader.OrderDate between :ad_startdate and :ad_enddate
    GROUP BY Sales.Customer.CustomerID
    ORDER BY Sum(Sales.SalesOrderHeader.SubTotal) DESC
    USING SQLCA;

OPEN cur;

FETCH cur INTO :ll_customerId , :lde_AvgAmt, :lde_SumAmt;
DO WHILE SQLCA.SqlCode = 0
    ll_row++

    astr_rank[ll_row].CustomerId = ll_customerId
    astr_rank[ll_row].AvgAmt = lde_AvgAmt
    astr_rank[ll_row].SumAmt = lde_SumAmt

    FETCH cur INTO :ll_customerId , :lde_AvgAmt, :lde_SumAmt;
LOOP

CLOSE cur;

Return

end subroutine
```

Data Access Layer in C# (with PowerBuilder)

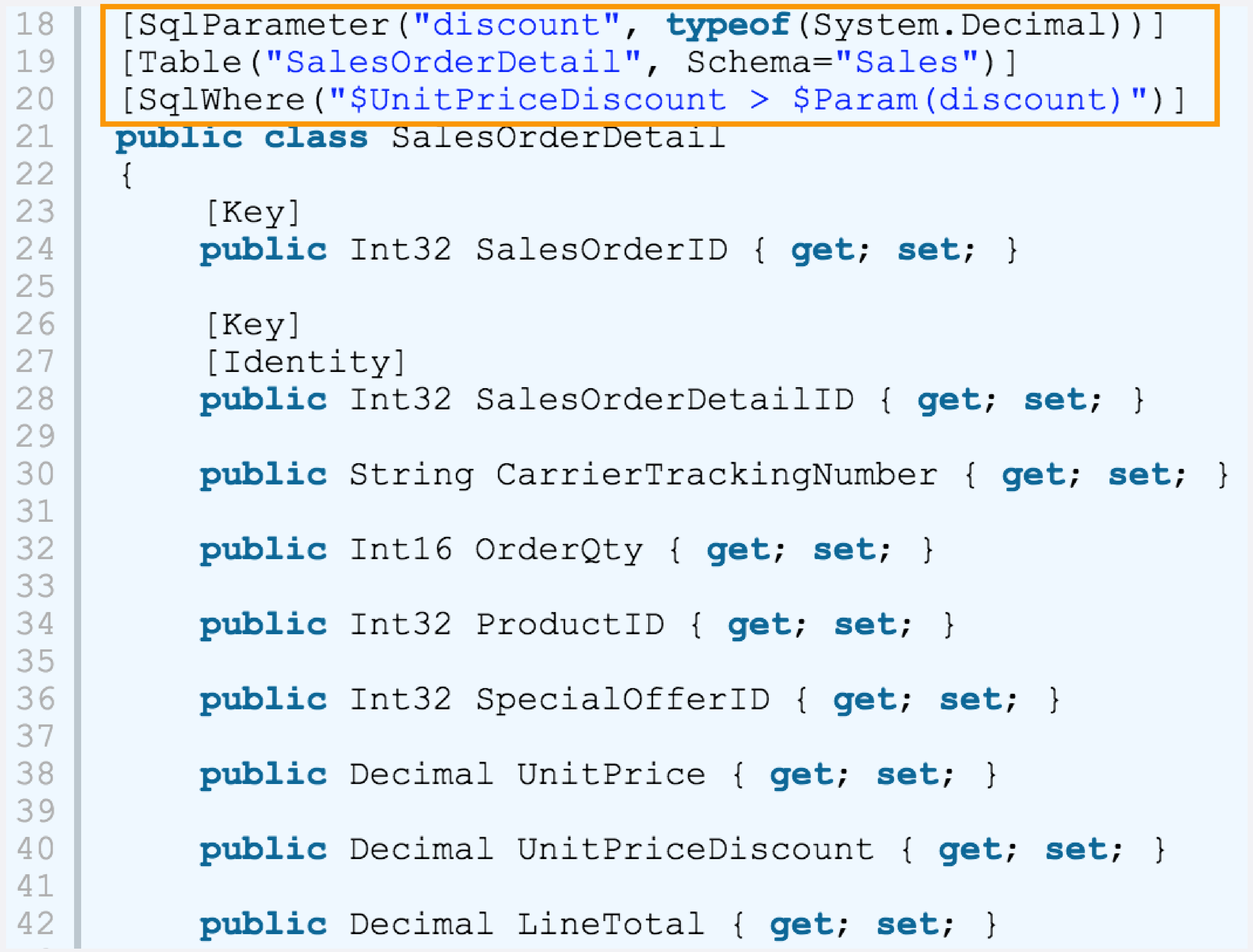
```
public IList<CustomerRank> GetCustomerRank(DateTime startDate, DateTime endDate)
{
    return _context.SqlExecutor.Select<CustomerRank>(
                @"SELECT TOP 10000
                    c.CustomerID,
                    Sum(h.SubTotal) as SumAmt,
                    Avg(h.SubTotal) as AvgAmt
                FROM Sales.Customer c,
                    Sales.SalesOrderHeader h
                WHERE h.CustomerID = c.customerID AND
                    h.OrderDate between @start and @end
                GROUP BY c.CustomerID
                ORDER BY Sum(h.SubTotal) DESC",
                startDate,
                endDate);
}
```

SQL Separation & Annotations in PowerBuilder 2018

```
public void BasicSqlGenerate()
{
    //No sqlText parameter, SqlExecutor will auto generate SQL for you
    var orderDetails = _context.SqlExecutor.SelectToStore<SalesOrderDetail>(
                                ParamValue.New("discount", 0));

    this.AppendResult("Row count: " + orderDetails.Count);
```

SQL Separation & Annotations in PowerBuilder 2018



```
[SqlParameter("discount", typeof(System.Decimal))]
[Table("SalesOrderDetail", Schema="Sales")]
[SqlWhere("$UnitPriceDiscount > $Param(discount)")]
public class SalesOrderDetail
{
    [Key]
    public Int32 SalesOrderID { get; set; }

    [Key]
    [Identity]
    public Int32 SalesOrderDetailID { get; set; }

    public String CarrierTrackingNumber { get; set; }

    public Int16 OrderQty { get; set; }

    public Int32 ProductID { get; set; }

    public Int32 SpecialOfferID { get; set; }

    public Decimal UnitPrice { get; set; }

    public Decimal UnitPriceDiscount { get; set; }

    public Decimal LineTotal { get; set; }
```

PowerBuilder 2018
C# Features

- PowerBuilder NVOs + ModelStore
- REST/JSON interface (Web API)
- OAuth Server
- C# Unit Testing Framework

Migration Benefits

Besides saving time/money and getting tons of features?

1. Current .NET targets deprecated
2. Big boost in performance & scalability
3. Portable – Linux/Windows, any Web server*, any C# IDE

**.NET Core compatible*

Quick Poll

1. Currently have PowerBuilder 2017? Standard Edition?
2. Currently or in near future using C#?
3. How soon migrating PowerBuilder to C# or cloud?
   within 6 months, 12 months, 24 months, or no need?

Thank You

www.appeon.com/pb2018.html