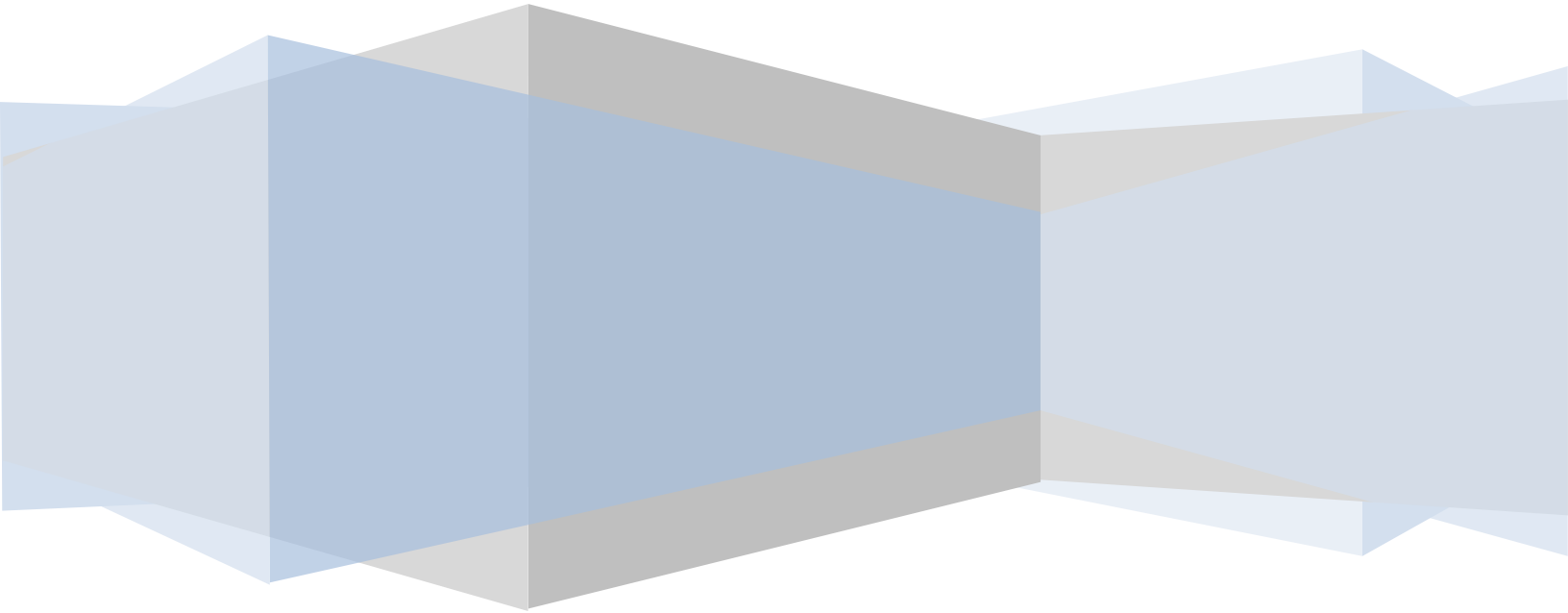
ASP.NET Dynamic Data Filtering

Overview Document

Joshua A. Heyse

Last update: 9/30/2008



Introduction

ASP.NET Dynamic Data introduced data scaffolding to the ASP.NET framework. This provides the ability to dynamically generate web pages based on data or object schemas. Field templates are used to render data based on the type of data element to provide usability improvements and support validation. While working with Dynamic Data a need for enhanced filtering support was identified. ASP.NET Dynamic Data Filtering is an open source project which builds on Dynamic Data to provided dynamic querying and filtering capabilities.

What you should know before continuing

Consumers of this library should be familiar with the ASP.NET Dynamic Data and writing LINQ expressions. More advanced users should be comfortable with creating Lambda Expressions.

Why is this library needed

In the version of ASP.NET Dynamic Data released with Visual Studio 2008 SP1 limited filtering support is provided. The out of the box Dynamic Data framework provides support for filtering based on foreign keys and boolean fields only. This is largely limited due to the filtering capabilities provided by the `LinqDataSource`.

Goal

The goal of this library is to extend ASP.NET Dynamic Data to provided support for filtering scenarios seen in data driven web applications. This library will provide common filters and filtering controls along with an extensibility model designed to allow developers to create custom filters.

Getting Started

To get started using Dynamic Data Filtering download and install DynamicDataFiltering.msi from: http://www.codeplex.com/DynamicDataFiltering/Release/ProjectReleases.aspx?ReleaseId=16483. This installer will create a directory under %PROGRAMFILES%\Dynamic Data Filtering which includes 3 directories:

- Bin – The binaries, pdbs and xml documentation files for the controls.
- Samples – C# and Visual Basic reference implementations which demonstrate the capabilities of Dynamic Data Filtering.
- Templates – The default templates for C# and Visual Basic which are copied to the DynamicData\FilterItems folder when a DynamicFilterForm or DynamicFilterRepeater is first dragged onto the design surface.

The installer also adds the Dynamic Filter User Control item templates in the Visual Studio “Add New Item…” window.

Reference Implementation

The Adventure Works solution is a reference implementation of the dynamic filtering library. The key points of interest are:

- The FilterTemplates folder located in the DynamicData directory contains default filter templates for the standard data types (ex. Integer, Boolean, DateTime)
- The Products/Search.aspx page demonstrates the use of the DynamicFilterForm and filter user controls to add filter criteria to a DynamicLinqDataSource and bound GridView control.
- The Products/Current.aspx page demonstrates the use of a static expression parameter defined in the DynamicLinqDataSource WherePredicateParameters. The parameter defined in Parameters/CurrentProductParameter.cs is used in this page.
- The DynamicData/List.aspx page demonstrates the DynamicFilterRepeater control which dynamically creates Dynamic Filter controls based on the metatdata defined on the Product class in BusinessObjects/Metadata/Product.cs.

Design

The filtering library has several key components which allow it to work; this section gives a high level design overview.

DynamicLinqDataSource

The `Catalyst.Web.DynamicData.DynamicLinqDataSource` is used in place of the `System.Web.UI.WebControls.LinqDataSource` to allow for advanced filtering features. The `DynamicLinqDataSource` is designed to maintain backwards compatibility with the `LinqDataSource`.

The `LinqDataSource` supports filtering by building a parameterized query expression string in the Where property and providing parameter values in the `ParameterCollection` `WhereParameters` property. If no where clause is specified an AND operation is performed among all values in the `WhereParameters` property. This method only supports limited where clause logic or forces the user to edit and manipulate messy where clauses.

The `DynamicLinqDataSource` adds the idea of chaining where parameter expressions to the LINQ query. This is accomplished in code using extension methods on an object that implements `IEnumerable<T>` and looks like:

```
Products.Where(p => p.Color == "Red").Where(p => p.ListPrice > 50 &&
p.ListPrice > 100)
```

During the `OnSelecting` method of the `DynamicLinqDataSource` the `WherePredicateParameters` are iterated through and responsible for returning a `LambdaExpression` which is appended to the LINQ query.

IDynamicExpressionParameter

The Catalyst.Web.DynamicData.`IDynamicExpressionParameter` defines two methods which must be implemented for every parameter that is used to chain complex expressions on to the `IEnumerable<T>`. If a standard equality comparison is needed the default ASP.NET parameters will also work. The two methods defined on `IDynamicExpressionParameter` are:

```
LambdaExpression GetLambdaExpression(Type itType)
IQueryable AppendQuery(IQueryable query)
```

Common functionality, including the default implementation of `AppendQuery` is included in `DynamicExpressionParameterBase`.

DynamicFilterForm

The `Catalyst.Web.DynamicData.DynamicFilterForm` is the container for the dynamic filter control. The `DynamicFilterForm` will add parameters to the `WherePredicateParameters` property of a `DataSource` which implements `IPredicateDynamicDataSource`.

DynamicFilterRepeater

The `Catalyst.Web.DynamicData.DynamicFilterRepeater` interigates the MetaModel of the object and creates a DynamicFilterControl for each property that is attributed with the FilterAttribute. The `DynamicFilterRepeater` will add parameters to the `WherePredicateParameters` property of a `DataSource` which implements `IPredicateDynamicDataSource`.

FilterTempalteUserControlBase

The `Catalyst.Web.DynamicData.FilterTempalteUserControlBase` is the user interface representation for a `DynamicExpressionParameter`. The user control which extends `FilterTempalteUserControlBase` must implement `GetWhereParameters` which returns an `IDynamicExpressionParameter` to the querying consumer.