

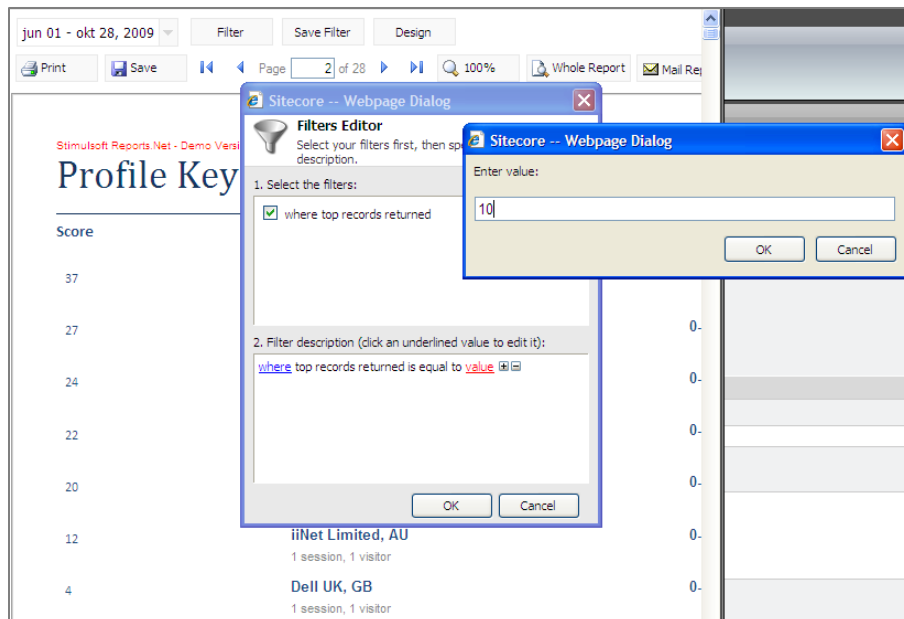
Creating a Top Records Report Filter in Sitecore OMS

Introduction

Welcome to my third Sitecore blog on creating OMS reports.

In this post, I will create a “top” filter that limits the records displayed in a report. To do this you would normally hard code a SQL TOP clause into your SQL query but in this blog I will demonstrate how you can achieve the same result by creating a filter.

A “top” filter could allow an end-user to enter any numerical value of their own, such as top 10 or top 100. This would make it easier for a marketer to see at a glance which leads are the most important to follow up and is better for performance than displaying every record in the database.



We realized that this would be something “nice to have” in the product and the solution presented here is one way of achieving this. However, it is not a typical approach. For example, we don’t recommend the use of TSQL in your SQL queries as this can cause problems when migrating from a SQL Server to an Oracle database. This blog merely demonstrates some of the possibilities available to developers when creating Sitecore OMS reports. As a report analyst, if you have enough in-depth knowledge of the SQL query language and some programming experience, you can create almost any filter or report you need. The only limitations are the contents of your database tables and your own imagination!

Prerequisites

When you create a new filter, you place the logic behind the filter in a C# class file. Therefore to create new filters it is necessary to have some knowledge of C# and Visual Studio.

You also need to install the following:

- Sitecore OMS
- Stimulsoft Report Designer (Stimulsoft Reports.Net 2009.2)
- Visual Studio 2005 or 2008

Optional step: You can add a filter to any existing report but in this blog I will duplicate the *ProfileKeyFilter* report I created in my previous blog and rename it *ProfileKeyFilterTop*. This is an optional step which makes it easier to test your filter without affecting you other existing reports. It also means that you are working with the data sources and SQL query.

Creating a Top Filter

To create a new filter, you need to complete the following steps:

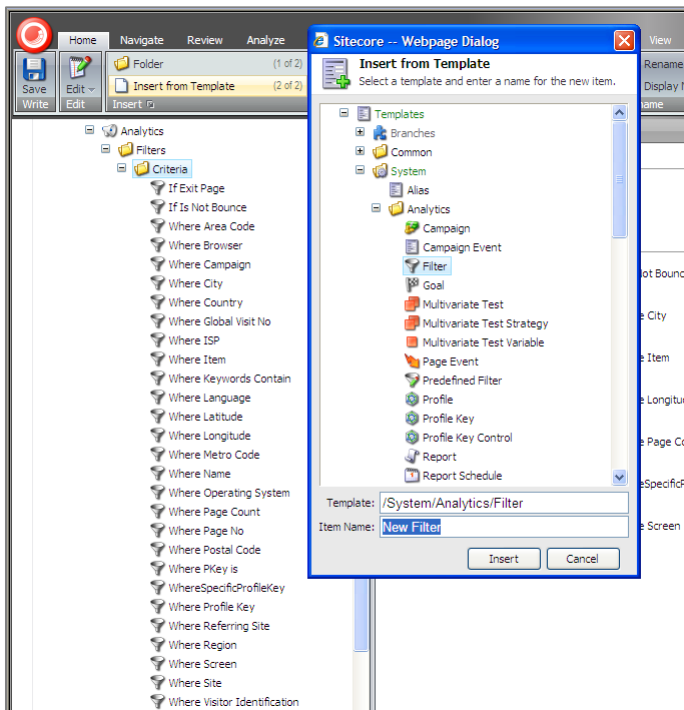
- Create a Filter Definition Item
- Create your Visual Studio Filter Implementation
- Add a Top Clause to your SQL Query
- Configure your Sitecore Definition Item
- Test your Top Filter

Step 1: Create a Filter Definition Item

A Sitecore report filter consists of a filter definition item, stored in the filter criteria folder and a C# class containing the filter implementation.

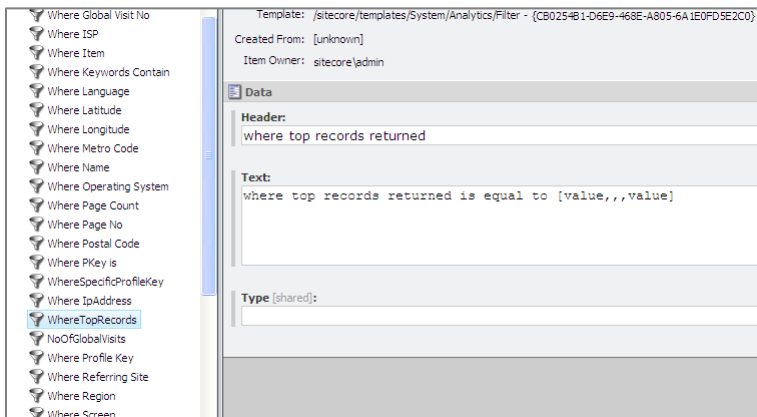
To create a filter definition item:

1. Open the Content Editor and in the content tree navigate to the following location.
/sitecore/system/Settings/Analytics/Filters/Criteria
2. In the ribbon, click *Insert from Template*. Select the filter template: */System/Analytics/Filter*



3. Name your filter *WhereTopRecords*.
4. In the **Header** field, enter:
where top records returned
5. In the **Text** field, enter:
where top records returned is equal to [value,,value]

6. The **Type** field contains a reference to your Visual Studio filter implementation. Leave this field blank for the moment.



7. Save your changes.

Step 2: Create your Filter Implementation in Visual Studio

To create a new report filter, you must use Visual Studio and create a new C# class. Alternatively, re-use an existing filter, such as *WhereSpecificProfileKey* from my last blog. In this case, re-name it *WhereTopRecords* and edit the existing C# class file.

- Make sure that your class includes the following references and inherits from the *Operator Filter Base* class:
 - Sitecore.Kernel
 - Sitecore.Analytics

```
public class WhereTopRecords : OperatorFilterBase
```

- Your filter class, uses the following two methods:
 - a. *ApplyFilter*
 - b. *IsApplicable*

These methods are the same for all filter implementations.

C# code used in this filter

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using Sitecore.Analytics.Data.Filters.Filters;
using Sitecore.Analytics.Data.Filters;
using Sitecore.Diagnostics;
```

```

namespace <your project name>.sitecore_modules.<your project folder>
{
    public class WhereTopRecords : OperatorFilterBase
    {
        public string value;

        // Methods
        public override void ApplyFilter(SqlCommand sqlCommand)
        {
            Assert.ArgumentNotNull(sqlCommand, "sqlCommand");
            string @operator = base.GetOperator();
            sqlCommand.AddWhereClause("MaxRecords", Value, false);
        }

        public override bool IsApplicable(SqlCommand sqlCommand)
        {
            Assert.ArgumentNotNull(sqlCommand, "sqlCommand");
            return (sqlCommand.HasPlaceholder("MaxRecords"));
        }

        //Properties
        public string Value
        {
            get
            {
                return (this.value ?? "1000");
            }
            set
            {
                Assert.ArgumentNotNull(value, "value");
                int defaultValue = 1000;
                Int32.TryParse(value, out defaultValue);
                this.value = defaultValue.ToString();
            }
        }
    }
}

```

You can either copy and paste this code into your C# class file or use the *ApplyFilter* and *IsApplicable* methods to construct your own code solution.

Explaining the code:

In my previous blog post, I explained the standard code to include in a filter implementation. In this post, I will only include the sections that are important to this filter.

Variables

Declare the following variable. This variable is used in the Properties section of your class and holds a value for the TOP clause entered by an end-user.

```
// Fields
```

```
public string value;
```

Methods

ApplyFilter

This method inserts your filter into the WHERE clause of your SQL query. It also passes parameters, such as a value for the TOP clause to the report engine.

```
// Methods
public override void ApplyFilter(SqlCommand sqlCommand)
{
    Assert.ArgumentNotNull(sqlCommand, "sqlCommand");
    string @operator = base.GetOperator();
    sqlCommand.AddWhereClause("Top", Value, false);
}
```

Edit the *AddWhereClause* parameters:

(<Placeholder name>, <where clause>, <isExceptCondition>)

Use the following syntax:

Parameter	Data type	Value
Placeholder name	string	MaxRecords – Appears in the SQL query as {MaxRecords}. Use MaxRecords to avoid confusion with SQL TOP.
Where clause	string	Value –This is a variable containing a top value entered by an end-user.
isExceptCondition	boolean	False – this determines what should be displayed if there is an error.

IsApplicable

This method determines where to insert your filter using a placeholder. The placeholder must be inserted into the WHERE clause of your SQL query using the following syntax:

{placeholder}

Normally a SQL TOP clause can only be added to the SELECT part of a SQL query. To make it possible to insert it in the right place, you need to use Transact SQL to create a workaround.

```
public override bool IsApplicable(SqlCommand sqlCommand)
```

```

    {
        Assert.ArgumentNotNull(sqlCommand, "sqlCommand");
        return (sqlCommand.HasPlaceholder("MaxRecords"));
    }

```

You can either use the placeholder name I have used {MaxRecords} or replace this with a placeholder name of your own. Both methods must include the same placeholder name.

Properties

Create the following properties.

Use the *get* and *set* properties to process user input. If a user enters a value, such as *10* the filter returns the top 10 records. If the value is left empty then the top 1000 records are returned by default.

```

//Properties
public string Value
{
    get
    {
        return (this.value ?? "1000");
    }
    set
    {
        Assert.ArgumentNotNull(value, "value");
        int defaultValue = 1000;
        Int32.TryParse(value, out defaultValue);
        this.value = defaultValue.ToString();
    }
}

```

Compile your code and ensure that it is added to your Sitecore bin folder.

C:\inetpub\wwwroot\\WebSite\bin

Step 3: Add a TOP Clause to your SQL Query

To add a filter implementation to a report, you normally add a placeholder to the WHERE clause in your SQL query. However, since the TOP clause appears immediately after the SELECT clause, you need a workaround to implement a Top filter. Use Transact SQL (TSQL) to create a CASE WHEN ELSE statement and embed this into the SELECT clause of your SQL query.

To add a TOP clause to your SQL query.

1. Open your *ProfileKeyFilterTop* report in the report designer
2. Right click the *IPOwners* data source and click Edit
3. Copy the following TSQL code into your SQL query directly after the SELECT statement.

```

top (
    CASE WHEN '{MaxRecords}' = ''

```

```

        THEN 1000
        ELSE CAST(REPLACE( REPLACE( '{MaxRecords}', 'and (' ,
'' ), ')', '' ) as integer )
        END
    )

```

Notice that the placeholder {MaxRecords} is included as part of this embedded code.

The Workaround

Our placeholder {MaxRecords} contains the three parameters added by the *AddWhereClause* method call (described in step 2). Our workaround allows us to insert this clause into the SELECT statement at the top of the SQL query even though this is not actually part of the WHERE clause. However, this will only work if we remove some of the unnecessary parameters and characters generated by the *AddWhereClause* method, because the *AddWhereClause* method generates text that looks something like:

and (10)

where “10” is the user entered value, but the TOP clause just requires:

10

The code in the parentheses following the TOP statement contains a TSQL CASE WHEN statement. In Transact SQL this can be used instead of the classic IF THEN statement. We use it here to add one of two alternative numerical values to the TOP clause:

- A default value of 1000 (stated in the THEN clause)

OR

- A user entered value (stated in the ELSE clause)

If an end-user fails to enter a value, then the first 1000 records are displayed by default. The ELSE part of the code takes the text generated by the *AddWhereClause* method, strips out unwanted code generated by the *AddWhereClause* method using REPLACE, and converts the remaining user entered value to an integer using CAST.

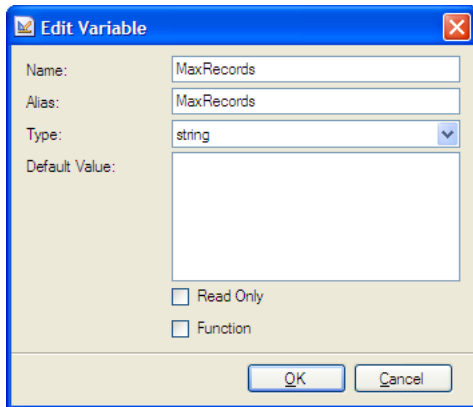
```

CAST(REPLACE( REPLACE( '{MaxRecords}', 'and (' , '' ), ')', '' ) as
integer )

```

So if a user enters a value such as “10” then *TOP 10* is inserted into the SELECT part of the SQL query.

4. Next, create a report designer variable for your placeholder {MaxRecords}. To do this, in the **Dictionary** panel, click *New Item, New Variable*.
5. In the **Name** field enter *MaxRecords* and enter the same value in the **Alias** field. Keep the Type as string and click OK.



Make sure your full SQL query matches the code below:

```

SELECT
    top (
        CASE WHEN '{MaxRecords}' = ''
            THEN 1000
            ELSE CAST(REPLACE( REPLACE( '{MaxRecords}', 'and (' , '' ),
                ')', '' ) as integer )
            END
        )
    Count(DISTINCT IpOwner.IPOwnerId) AS IPOwnerIdObject, IpOwner.Name,
    IpOwner.Country,
    Count(DISTINCT Session.GlobalSessionId) AS VisitorCount,
    Count(DISTINCT Session.SessionId) AS VisitCount,
    Sum(DISTINCT Profile.Total) AS Score,
    ProfileKeyDefinition.Name AS Prof

FROM Session
INNER JOIN Profile ON Session.SessionId = Profile.SessionId
INNER JOIN GlobalSession ON GlobalSession.GlobalSessionId =
Session.GlobalSessionId
INNER JOIN Ip ON Ip.IpId = Session.IpId
INNER JOIN IpOwner ON IpOwner.IpOwnerId = Ip.IpOwnerId
INNER JOIN Browser ON Browser.BrowserId = Session.BrowserId
INNER JOIN ProfileKey ON ProfileKey.ProfileId = Profile.ProfileId
INNER JOIN ProfileKeyDefinition ON
ProfileKeyDefinition.ProfileKeyDefinitionId =
ProfileKey.ProfileKeyDefinitionId

WHERE
Session.Timestamp >= @StartDate and
Session.Timestamp <= @EndDate
{MyProfileKey}

GROUP BY IpOwner.Name, IpOwner.Country, ProfileKeyDefinition.Name

ORDER BY Prof, Score DESC, IpOwner.Name

```

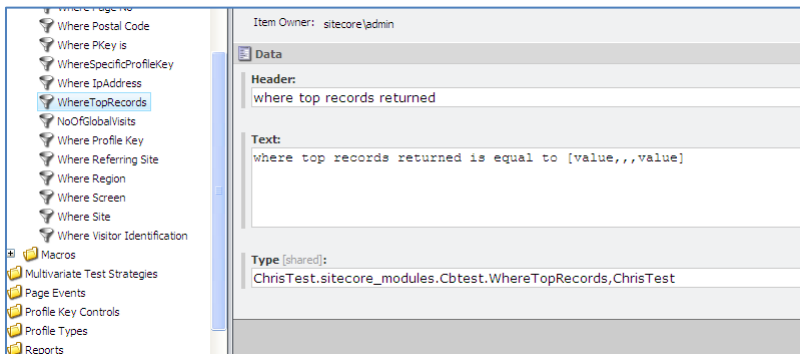
Note: This query also contains the placeholder for the profile key filter *MyProfileKey* that I created in my previous blog.

6. Click **OK** and save your changes in the report designer.

Step 4: Configure your Sitecore Filter Definition Item

When you have created your filter class and updated your SQL query you then configure your Sitecore filter definition item to point to your C# assembly.

1. In the Content Editor, Criteria folder, select your filter definition item.
/sitecore/system/Settings/Analytics/Filters/Criteria/WhereTopRecords



2. In the **Type** field, enter the correct path to your C# class and assembly.

```
ChrisTest.sitecore_modules.Cbtest.WhereTopRecords,ChrisTest
```

└────────────────────────────────┘ └────────────────┘ └──────────┘
namespace class name assembly

3. Save your changes.

Step 5: Test the Top Filter

View the *ProfileKeyFilterTop* report and test your new top filter in Sitecore Analytics.

1. In Sitecore Analytics, to refresh your reports, close and then expand the Reports node.
2. Click on your new report: *ProfileKeyFilterTop*

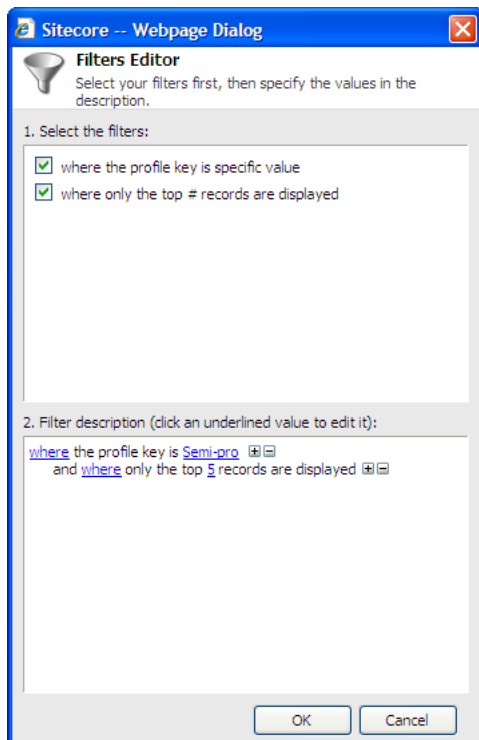
3. Select a profile key such as *Semi-pro* using the filter created in my previous blog. All records for the chosen profile key are displayed in your report.

Stimulsoft Reports .Net - Demo Version

Profile Key: Semi-pro

Score	Organization	Profile Key
483	Vasakronan Service Partner AB, SE 5 sessions, 2 visitors	Semi-pro
453	Adorama, US 5 sessions, 2 visitors	Semi-pro
429	Nicam Corporation, DK 3 sessions, 3 visitors	Semi-pro
388	PopPhoto, US 3 sessions, 2 visitors	Semi-pro
271	Dell UK, GB 4 sessions, 4 visitors	Semi-pro
231	Canon NL, NL 1 session, 1 visitor	Semi-pro
202	Amazon Inc., US 4 sessions, 1 visitor	Semi-pro
168	TPG Internet Pty Ltd., AU 1 session, 1 visitor	Semi-pro
127	iNet Limited, AU 1 session, 1 visitor	Semi-pro
92	SBC Internet Services, US 1 session, 1 visitor	Semi-pro
72	Google, VN 12 sessions, 12 visitors	Semi-pro
71	Ebay, US 1 session, 1 visitor	Semi-pro

4. Open the **Filters Editor** window again. You should see your new top filter displayed in the **Filters Editor** window.
5. Select your top records filter.



6. Click *value* in the **Filter description** panel and enter a value such as 5. Click **OK** to close the **Filters Editor** window. You should now only see the top 5 records returned for the *Semi-pro* profile key.

Stimulsoft Reports Net - Demo Version

Profile Key: Semi-pro

Score	Organization	Profile Key
483	Vasakronan Service Partner AB, SE 5 sessions, 2 visitors	Semi-pro
453	Adorama, US 5 sessions, 2 visitors	Semi-pro
429	Nicam Corporation, DK 3 sessions, 3 visitors	Semi-pro
388	PopPhoto, US 3 sessions, 2 visitors	Semi-pro
271	Dell UK, GB 4 sessions, 4 visitors	Semi-pro