

Sitecore CMS 6.6 Segment Builder Developer's Guide

A developer's guide to using the Segment Builder API.



Introduction

The Segment Builder component is used to get a list of visitors that matches certain criteria from the Analytics database. This component is used by the Engagement Automation to add a segment to a state of an automation plan. For more information, see the *Engagement Automation Cookbook* — section *Adding a Segment to a State*.

This guide is a combined API reference and a developer's guide. It describes how the Segment Builder works with Sitecore Engagement Automation functionality.

Developers can use the Segment Builder component to:

- Use the dialog to get visitors from selected segments.
- Create new Segment Builder rules.



Using the Segment Builder Dialog

This chapter describes how to:

- Programmatically open the dialog.
- Get the visitors from a segment that is described by a collection of rules (result of the dialog). This is the first part of the document.

Accessing the Segment Builder

To open the Segment Builder dialog and allow end users to get a list of visitors from the Analytics database that matches a certain criteria, you must:

1. Use the following code to open the dialog:

```
var url = new UrlString(Sitecore.Context.Site.XmlControlPage);
url["xmlcontrol"] = "Sitecore.Shell.Applications.Analytics.SegmentBuilder";
var handle = new UrlHandle();
handle["Value"] = <rules value>; //when updating an existing rule, pass the XML
created previously, otherwise - use string.Empty.
handle.Add(url);
SheerResponse.ShowModalDialog(url.ToString(), "800px", "600px", string.Empty, true);
args.WaitForPostBack();
```

- 2. On the dialog, select various criteria and click OK.
- 3. You can then implement the <u>post back action</u> and get the dialog result. The result is a string that describes the rules. You can use the Segment Builder to convert these rules to a set of criteria and later on to an SQL statement. The following code snippet describes this action:

```
if (args.IsPostBack)
{
    if (args.HasResult)
    {
        var value = args.Result == "-" ? string.Empty : args.Result;
        //processing the value code here. Use SegmentBuilder class to get number of
        visitors of list of Visitor IDs:
        //var segmentBuilderHelper = new
        Sitecore.Shell.Applications.Analytics.SegmentBuilder.SegmentBuilder();
        //int count = segmentBuilderHelper.GetVisitorCount(value);
    }
    return;
}
```

4. Based on this result, you can then implement the code to get the number of visitors or visitor IDs that match the specified user rules.

Sitecore® is a registered trademark. All other brand and product names are the property of their respective holders. The contents of this document are the property of Sitecore. Copyright © 2001-2013 Sitecore. All rights reserved.



The following example describes <code>SegmentBuilderRules</code> class. This class describes how to open Segment Builder dialog and save its result. The result is saved in the regular *Rules* format.

```
namespace Sitecore.Shell.Applications.ContentEditor
{
    using Sitecore.Diagnostics;
    using Sitecore.Text;
    using Sitecore.Web;
    using Sitecore.Web.UI.Sheer;
    /// <summary>
    /// The segment builder rules.
    /// </summary>
    public class SegmentBuilderRules : Rules
    {
        /// <summary>
        /// Edits the text.
        /// </summary>
        /// <param name="args">The arguments.</param>
        protected override void Edit (Sitecore.Web.UI.Sheer.ClientPipelineArgs args)
            Assert.ArgumentNotNull(args, "args");
            if (this.Disabled)
            {
                return;
            }
            if (args.IsPostBack)
            {
                if (args.HasResult)
                {
                    this.Value = args.Result == "-" ? string.Empty : args.Result;
                    SheerResponse.SetAttribute(this.ID, "value", this.Value);
                    SheerResponse.SetModified(true);
                    this.Refresh();
                }
                return:
            }
            var value = this.Value;
            if (value == EditorConstants.NoValue)
            {
                value = string.Empty;
            }
            Assert.IsNotNull(Sitecore.Context.Site, "site");
            var url = new UrlString(Sitecore.Context.Site.XmlControlPage);
            url["xmlcontrol"] =
            "Sitecore.Shell.Applications.Analytics.SegmentBuilder";
            var handle = new UrlHandle();
            handle["Value"] = this.Value;
            handle.Add(url);
            SheerResponse.ShowModalDialog(url.ToString(), "800px", "600px",
            string.Empty, true);
            args.WaitForPostBack();
       }
    }
```



Getting the Visitor from a Rule Collection

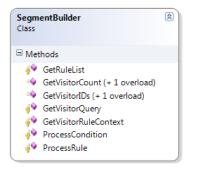
This section describes the important classes, functionalities, parent classes, implementation, methods, and properties in the Segment Builder. Each section presents an important module in the API. Each section contains a description of the class and some tables that describe the properties and methods of this class.

You can also use the following classes to get the visitor's information.

Sitecore.Shell.Applications.Analytics.SegmentBuilder.SegmentBuilder

The SegmentBuilder class contains the methods that you can use to get the VisitorIds that match to selected rules.

It is a class-helper that gets a list of visitor IDs from the result that you get from the Segment Builder dialog. This result is a string value with serialized list of segments defined in the dialog. Each segment is a combination of segment builder rules.



The following table describes the public methods in the SegmentBuilder class and their overloads:

Method	Description	
int GetVisitorCount(string rules, Database database)	Gets the number of the site visitors. The rules parameter is a string that represents segment builder rule The database parameter contains the definitions of the rules items. For example: E17C9}\"> 	
	34DB6FCE2F9E}\" uid=\"2EE7B61586224E01AC71CAB9C5E819DF\" automationids=\"{D036CAB9-B2F5-4176-A67F- 82081463B78E} {1A9E3116-8D42-46C5-BAD1- FD4D9B342826}\" /> To process the rule, the method gets the item from the specified database	
	and executes the rule. In the content tree, you can navigate to the Segment Builder rule items using the following path: Master/sitecore/system/Settings/Rules/Segment Builder For more information about the rule engine, see the <i>Rule Engine</i>	



Method	Description
	Cookbook.
<pre>int GetVisitorCount(string rules)</pre>	This is an overload of the previous method. It gets the number of visitors. It assigns the database value to Client.ContentDatabase that is equal to the context database.
Enumerable <guid> GetVisitorIDs(string rules, Database database)</guid>	This method is the similar to previous two methods. The only difference is that it returns list of visitor IDs that match specified rules.
<pre>IEnumerable<guid> GetVisitorIDs(string rules)</guid></pre>	This is an overload of the previous method. It takes the rules and gets the IDs of the visitors. It assigns the database value to Client.ContentDatabase that is equal to the context database.

Note

To get other visitor related information, you should use the following class in the Engagement Analytics API: Sitecore.Analytics.Data.DataAccess.VisitorFactory.GetVisitor.



Sitecore.Analytics.Data.DataAccess.DataAdapters.DataAdapterManager

The DataAdapterManager class defines the data adapter manager that redirects method calls to the DataAdapterProvider class.

DataAdapterManager 🛞 Class	that belongs to the Segment Builder: F see the Engagement Analytics API Re	For information about the other properties
≡ Fields	Property	Description
 helper Properties Format Lookups Maintenance Provider Providers ReportingSql Sql Status Subscription Testing TrafficType VisitorClassification VisitorQuery Visitors Visitors Visitors Lood(+ 1 overload) LoadSharedData LoadVisitAttributes LoadVisitor Update 	VisitorQuery VisitorQuery	 This class has two functionalities: It is the base class for the GetVisitorIDsSql and GetVisitorCountSql methods that you can use to convert the visitor Query to a statement that can be executed in the DBMS to get the number of visitors and their IDs. Gets the visitor query. In the Analytics database, it accesses rule items in the that reference the rule classes. These classes are used to construct the query.
Nested Types ConnectionStrings Static Class Properties Analytics AnalyticsSettings Automations		Note You can use inherited classes like the SqlVisitorQuery class to convert data that is stored in this object into the appropriate SQL or Oracle statements.
AutomationSettings Reporting ReportingSettings	VisitorBase Visitors	Gets the visitors. You can also use this property to get the number of the visitors or the visitors IDs from the constructed VisitorQuery object.

The following table describes the properties of the DataAdapterManager class

The methods of this class are not part of the segment builder. They are part of the Engagement Analytics API.



Creating a Segment Builder Rule

This chapter describes how to create a new rule class. It presents all relevant classes in the Sitecore.Analytics.Data.DataAccess.DataAdapters namespace.

Segment Builder Rule API

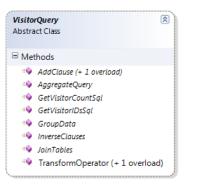
The following sections present the classes that exist in the DataAdapters namespace. This namespace is part of the Engagement Analytics API. However, you should use it to create the Segment Builder rules.

Sitecore.Analytics.Data.DataAccess.DataAdapters.VisitorQuery

VisitorQuery is an abstract class that contains the main methods that manipulate the visitor's query. It contains the methods that are invoked by the rule implementations to construct the condition query. This query that is executed on the Analytics database.

You can use it to:

- Aggregate the criteria to select the visitors and group them in a segment. This is done using the methods: AddClause, AggregateQuery, and JoinTables.
- Generate SQL statements that are based on the aggregated criteria. You can execute these statements in the Analytics database. For example, you can:
 - Use GetVisitorCountSql to get the SQL statement that is executed in the DBMS to get the number of visitors
 - Use GetVisitorIDsSql to get the SQL statement that is executed in the DBMS to get a list of the visitor IDs.





The following table describes the methods of the $\ensuremath{\texttt{VisitorQuery}}$ class:

Method	Description	
<pre>void AddClause(ClauseBase left, ClauseBase right, ClauseOption option, string group)</pre>	Takes the left and right operands of the clause, the logical operator, the group name, and adds the clause to the query. The group parameter is a unique name for the query that is generated by every rule. All the queries that have the same group name are combined using parenthesis.	
	<pre>Example: AddClause(``a=b", ``c=d", ``and", ``group1"); AddClause(``e=f", ``g=h", ``and", ``group2");</pre>	
	In this example, you have different groups. Therefore, the SQL query is: Select from where (a=b and c=d) or (e=f and g=h)	
	<pre>Example: AddClause(``a=b", ``c=d", ``and", ``group1"); AddClause(``e=f", ``g=h", ``and", ``group1");</pre>	
	In this example, you have the same groups. Therefore, the SQL query is: Select from where (a=b and c=d or e=f and g=h)	
<pre>void AddClause(ClauseBase clause, string group)</pre>	Takes the clause, the group name, and adds the clause.	
<pre>void AggregateQuery(VisitorQuery query, ClauseOption option)</pre>	Takes the visitor query clauses, logical operator and generates a clause that contains all the registered clauses associated with the operation.	
<pre>string GetVisitorCountSql(out object[] parameters)</pre>	Takes the parameter parameters and returns the SQLstatement that counts the visitors.Parameters is an array of parameters that are passed with theSQL statement to the database to get the result.	
<pre>string GetVisitorIDsSql(out object[] parameters)</pre>	Takes the parameters and returns a string that contains an SQL statement. This SQL statement returns a list of Visitor IDs. These Visitor IDs match the criteria that is aggregated according to the specified rules.	
<pre>void GroupData(AnalyticsTable table, string column)</pre>	Takes the analytics table, column name, and groups data by this column name.	



Method	Description
<pre>void InverseClauses()</pre>	Takes all clauses that are added to the VisitorQuery object, aggregates them and then negates them. Inversing means negating or adding the logical operator Not.
	Example: Suppose you have the letters: A, B, C, D and the clause letter > B that returns C and D. The inverted clause is then: Not(letter > B) or letter <= B and consequently returns A and B.
	<pre>Example: Suppose you have the same letters: A, B, C, D, and the clause: letter = A OR letter > C. The inverted clause is then: Not(letter = A OR letter > C) and consequently returns B and C. This is used when you have the except where rule. For example, except where visitor from UK The previous rule means NOT (user from UK).</pre>
<pre>void JoinTables(AnalyticsTable tableLeft, string columnLeft, AnalyticsTable tableRight, string columnRight)</pre>	Takes the analytics left table, left column, analytics right table, right column, and joins the tables.
QueryConditionOperator TransformOperator(ConditionOperator	Takes the rule related logical operator and transforms it into an operator that VisitorQuery supports.
@operator)	For example, the rules engine supports two types of logical operators: greater and equal
QueryConditionOperator TransformOperator(StringConditionOperator @operator)	Takes the rule engine related string operator and transforms it into an operator that VisitorQuery supports. For example, the rules engine supports two types of string operators: contains and startwith. The developers use this method if they are writing their own conditions and need to convert the operators that are selected in the rules into query operations.



Sitecore.Analytics.Data.DataAccess.DataAdapters.Sql.SqlVisitorQuery

You can use the SqlVisitorQuery class to generate queries as SQL statement.

Since the base class <code>VisitorQuery</code> is abstract, you always will work with <code>SqlVisitorQuery</code> or <code>OracleVisitorQuery</code>.

The OracleVisitorQuery class has the same functionality as SqlVisitorQuery but for Oracle databases.

	JUNE	sitorQuery
Fields Fields I aliaseChars aliases clauses groupingList havingClauses Forperties Aliase Forpertis Aliase Autor Properties Autor ParameterPostfix AggregateClauses AggregateClauses GetVolupClause GetVolupClause GetVisitorClause GetVisitorClause GetVisitorClause ProcessColumn ProcessColumn ProcessColumn	Class	
aliaseChars aliases clauses groupingList havingClauses joinReferences Troperties Aliases GroupingList HavingClauses NamePostfix NamePostfix NamePrefix ParameterPrefix ParameterPrefix AddClause (+ 1 overload) AddClause (+ 1 overload) AddHavingClause AddJoinReference AdgregateQuery BuildFromClause GetFromClause GetFromClause GetSelectIDsClause GetSelectIDsClause GetVisitorIDsSql GetUmPReference GetIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII		
aliases clauses clauses clauses clauses groupingList havingClauses joinReferences CoupingList HavingClauses NamePostfix AddClause (+ 1 overload) GetClauseSql GetSelectCountClause GetSelectDosClause GetSelectDosClause GetSelectDosClause GetVisitorDosClause GetVisitorDosClause GetVisitorCountSql GetVisitorCountSql GetVisitorCountSql GetVisitorQuery TranslateOperatorToString ProcessValue QueryTable SqlVisitorQuery TranslateOperatorToString Table Methods GroupReference GetSelectCounn ForcessColumn GroupReference GetSelectCounn ProcessValue ColumnName Table Methods GroupReference GetSelectCounn Properties ColumnRight Properties ColumnRight Properties ColumnRight Properties ColumnRight TableLeft		
clauses groupingList havingClauses joinReferences foroupingList havingClauses Ailases GroupingList HavingClauses NamePostFix NamePostFix NamePostFix NamePostFix NamePostFix ParameterPostFix ParameterPostFix ParameterPostFix AddClause (+ 1 overload) GetVoinenClause GetFactingClause GetSelectCountClause GetSelectCountClause GetVisitorCountSql GetVisitorQuery TranslateOperatorToString ProcessColumn ProcessColumn ProcessColumn ProcessColumn Protecties ColumnName Table Methods GroupReference Get GetUsitorCountSql GetUsitorCountSql GroupReference PointTableReference GetSelectIos GroupReference GetSelectIos GolumnLeft for columnRight Properties ColumnRight TableLeft		
groupingList havingClauses joinReferences faises GroupingList HavingClauses GroupingList HavingClauses NamePostfix NamePostfix NamePostfix NamePostfix ParameterPostfix ParameterPrefix AddClause (+ 1 overload) AddHavingClause AddLoinReference AggregateClauses AggregateClauses AggregateClauses AggregateClause GetSelectDountClause GetSelectDountClause GetSelectDountClause GetSelectDountClause GetVisitorClause GetSelectDountClause GetSelectDountClause GetVisitorClause GotUmnAume Table Methods GroupReference Get GetUmnKight Properties ColumnRight Properties ColumnRight Properties ColumnRight Methods TableLeft ColumnRight	_	
Properties JoinReferences JoinReferences JoinReferences JoinReferences Aliases GroupingList HavingClauses NamePostfix NamePostfix NamePrefix ParameterPostfix ParameterPostfix ParameterPostfix AddLoinReference AddLoinReference AggregateQuery BuildFromClause GetFromClause GetFromClause GetForuClause GetSelectIDsClause GetVisitorCountSql GetVisitorCountSql GetVisitorCountSql GetVisitorDsSql ForcessColumn ProcessColumn ProcessColum GroupReference SqlVisitorQuery TranslateOperatorToString Nested Types GroupReference SqlVisitorQuery TranslateCoperatorToString Nested Types GroupReference SqlVisitorQuery TranslateCoperatorToString Properties ColumnName Table Methods GroupReference SqlVisitorCountRight Properties ColumnRight Properties ColumnRight		
joinReferences foroupingList GroupingList HavingClauses NamePostfix NamePostfix NamePostfix NamePostfix NamePrefix ParameterPostfix ParameterPrefix AddClause (+ 1 overload) AddgregateQuery BuildFromClause GetClauseSql GetFromClause GetSelectCountClause GetSelectDosClause GetSelectDosClause GetSelectDosClause GetSelectDosClause GetSelectDosClause GetSelectDosClause GetSelectDosClause GetSelectDosClause GetSelectDosClause GetVisitorCountSql GetVisitorCountSql GetVisitorQuery ForcessColumn ProcessValue QueryTable SqlVisitorQuery TranslateOperatorToString Nested Types GroupReference Get Gas ColumnName Table Methods GroupReference SqlVisitorQuery TableReference GetSelectCountRight Properties ColumnRight Properties ColumnRight Properties ColumnRight TableLeft		
Properties Aliases GroupingList HavingClauses NamePostfix ParameterPostfix ParameterPostfix ParameterPostfix AddHavingClause AddUoinReference AggregateQuery BuildFromClause GetClauseSql GetFromClause GetSelectDourtClause GetSelectDourtClause GetSelectDourtClause GetSelectDourtClause GetSelectDourtClause GetSelectDourtClause GetVisitorDoSql GetWhereClauses JoinTables ProcessColumn ProcessValue SqlVisitorQuery ProcessValue Clause GetSelectDourtString ProcessValue GetYable SqlVisitorQuery TranslateOperatorToString ProcessValue GetSelect SqlVisitorQuery TranslateOperatorToString DoinTables ProcessColumn ProcessValue GetYable SqlVisitorQuery TranslateOperatorToString Deterties ColumnName Table Hethods GroupReference Sql GetSelectDourtClause GetSelectDourtClause GetSelectDourtSel GetClauseState Clause GetSelectDourtSel GetUmNAme Table Hethods GroupReference Claus GetUmNLeft ClaumRight Heroperties ColumnRight TableLeft CableAct Cabl	-	-
Aliases GroupingList HavingClauses NamePostfix NamePostfix ParameterPostfix ParameterPostfix ParameterPostfix ParameterPostfix ParameterPostfix ParameterPostfix AddUause (+ 1 overload) AddHavingClause AdgregateClauses AggregateClauses AggregateQuery BuildFormClause GetClauseSql GetClauseSql GetGroupByClause GetGroupByClause GetSelectCountClause GetSelectDoutClause GetVisitorClause GetVisitorClause] GetVisitorClause] GetVisitorClause] GetVisitorClause] GetVisitorClause] GetVisitorClause] GetVisitorClause] GetVisitorClause] GetVisitorClause] GetVisitorClause] GetVisitorClause] GetVisitorClause] GetVisitorClause] GetVisitorClause] ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn Process		
GroupingList HavingClauses NamePostfix NamePostfix NamePostfix ParameterPostfix ParameterPostfix AddHavingClause AddUainReference AggregateClauses AddUainReference AggregateClauses AdgregateClauses AdgregateClauses GetCrountClause GetCrountClause GetSelectDoutClause GetSelectDoutClause GetVisitorClause GetVisit		
HavingClauses HavingClauses NamePostfix NamePrefix ParameterPrefix ParameterPrefix AddDinReference AdgregateClauses AddDinReference AggregateClauses AddDinReference GetVisitorCountSql GetSelectIDsClause GetVisitorCountSql		
NamePostfix NamePrefix ParameterPostfix ParameterPostfix AddClause (+ 1 overload) AddClause (+ 1 overload) GetClauseSql GetClauseSql GetClauseSql GetClauseSql GetClause GetSelectCountClause GetSelectCountClause GetVisitorCountSql GetVisitorCountSql GetVisitorCountSql GetVisitorCountSql GetVisitorCountSql GetVisitorCountSql GetVisitorCountSql GetVisitorCountSql GroupData FrocessColumn ProcessValue QueryTable GroupDate Class ProcessColumn ProcessValue Class Properties ColumnName Table Hethods GroupReference Class Fields ColumnRight Properties ColumnRight Properties ColumnRight FableIf		HavingClauses
NamePrefix ParameterPostfix ParameterPostfix ParameterPostfix ParameterPostfix ParameterPostfix ParameterPostfix Add/Clause (+ 1 overload) Add/LavingClause AdgregateQuery AdgregateQuery BuildFromClause GetClauseSql GetForupByClause GetSelectCountClause GetSelectCountClause GetSelectCountClause GetSelectCountClause GetSelectCountSql GetVisitorDoSql GetVisitorDoSql GetVisitorQuery ProcessColumn ProcessValue QueryTable SqlVisitorQuery TranslateOperatorToString Nested Types GroupReference Get GetSelectCountSql GetSelectCountSql GetVisitorQuery TranslateOperatorToString Nested Types GroupReference GetSelectCountRight Properties ColumnRight Properties ColumnRight Properties ColumnRight Table.eft		NamePostfix
ParameterPostfix ParameterPrefix ParameterPrefix AddUause (+ 1 overload) AddHavingClause AddUoinReference AggregateClauses AggregateClauses GetClauseSql GetFramClause GetGroupParameterName GetVisitorClause JoinTables ProcessColumn ProcessValue SqlVisitorQuery TranslateOperatorToString Properties ColumnName Table Hethods GroupReference SqlVisitorClause GetSqlVisitorClause GetSqlVisitorQuery TranslateOperatorToString FroessColum ForcessValue GetVisitorToString FroessColum ForcestValue GetVisitorClause GetClause Get		NamePrefix
AddClause (+ 1 overload) AddMavingClause AddDioiReference AdgregateQuery AdgregateQuery BuildFromClause GetClauseSql GetFromClause GetSelectCountClause GetSelectCountClause GetSelectCountClause GetSelectCountClause GetSelectCountSql GetVisitorCountSql GetVisitorCountSql GetVisitorCountSql GetVisitorCountSql GetVisitorQuery ProcessColumn ProcessColumn ProcessValue QueryTable SqlVisitorQuery TranslateOperatorToString Nested Types GroupReference GetS GoumnName Table Methods GroupReference GetSelectCountRight Properties ColumnRight Properties ColumnRight TableAft ColumnRight TableAft	1	ParameterPrefix
AddHavingClause AddUoinReference AdgregateQuery BuildFromClause GetClauseSql GetFromClause GetSelectCountClause GetSelectCountClause GetSelectCountClause GetSelectCountSql GetVisitorCountSql GetVisitorCountSql GetVisitorCountSql GetVisitorDsSql GetVisitorQuery ProcessColumn ProcessColumn ProcessColumn ProcessColumn SqlVisitorQuery TranslateOperatorToString Nested Types GroupReference GroupReference Table Hethods GroupReference Table Hethods GroupReference Gas Fields ColumnRight Properties ColumnRight TableLeft	⊟ Met	hods
AddHavingClause AddUoinReference AdgregateQuery BuildFromClause GetClauseSql GetFromClause GetSelectCountClause GetSelectCountClause GetSelectCountClause GetSelectCountSql GetVisitorCountSql GetVisitorCountSql GetVisitorCountSql GetVisitorDsSql GetVisitorQuery ProcessColumn ProcessColumn ProcessColumn ProcessColumn SqlVisitorQuery TranslateOperatorToString Nested Types GroupReference GroupReference Table Hethods GroupReference Table Hethods GroupReference Gas Fields ColumnRight Properties ColumnRight TableLeft	-9	AddClause (+ 1 overload)
AddJoinReference AggregateClauses AggregateClauses AggregateQuery BuildFromClause GetClauseSql GetFromClause GetGroupAyClause GetGroupAyClause GetSelectCountClause GetSelectDoxIclause GetVisitorClausEql GroupReference SqlVisitorQuery TranslateOperatorToString Properties ColumnName Table Methods GroupReference Sql GiumnLeft GolumnRight Properties ColumnRight Properties ColumnRight TableLeft TableLeft ColumnRight SelveStevEquerEquerEquerEquerEquerEquerEquerEquer		
AggregateQuery AggregateQuery BuildFromClause GetClauseSql GetFromClause GetSefectCountClause GetSefectDocClause GetSefectDocClause GetVisitorCountSql GetVisitorCountSql GetVisitorDoSql GetVisitorDoSql GetVisitorQuery GetVisitorQuery ProcessColumn ProcessValue QueryTable SqlVisitorQuery TranslateOperatorToString Nested Types GroupReference SqlVisitorQuery Table Methods GroupReference Sql SoinTableReference Sql	- 7 9	AddJoinReference
BuildFromClause GetClauseSql GetClauseSql GetClauseSql GetClauseSql GetSelectCountClause GetSelectCountClause GetSelectCountClause GetSelectCountSql GetVisitorCountSql GroupTable SqlVisitorQuery TranslateOperatorToString Nested Types GroupReference Get Fields GolumnName GolumnRight Properties ColumnRight Properties ColumnRight Table.eft ColumnRight Table.eft		
GetClauseSql GetFromClause GetFromClause GetGroupPs/clause GetSelectDountClause GetSelectDountClause GetSelectDountClause GetSelectDoutClause GetVisitorCountSql GroupPate SqlVisitorQuery TranslateOperatorToString Nested Types GroupReference Gass GroupReference GetVisitorCountSql GetVisitorCountSql GroupReference SqlVisitorCountSql GroupReference SqlVisitorCountSql GroupReference SqlVisitorCountSql GroupReference SqlVisitorCountSql GroupReference SqlVisitorCountSql Fields ScloumnLeft ScloumnRight Properties ColumnRight TableLeft TableLeft TableLeft ScloumRight SqlVisitorCountSql TableLeft ScloumRight SqlVisitorCountSql TableLeft ScloumRight SqlVisitorCountSql TableLeft ScloumRight SqlVisitorCountSql SolumRight SqlVisitorCountSql SqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCountSqlVisitorCoun	- 9	AggregateQuery
GetFramClause GetGroupByClause GetGroupByClause GetHavingClause GetSelectCountClause GetSelectDorClause GetVisitorCountSql GetVisitorCountSql GetVisitorCountSql GetVisitorCountSql GetVisitorCountSql GroupData InverseClauses JoinTables ProcessColumn ProcessValue QueryTable GroupReference Gas ClaumnName Table Methods GroupReference GetSelectCountLeft ColumnRight Properties ColumnRight Properties ColumnRight Properties ColumnRight Properties ColumnRight TableLeft ColumnRight TableLeft		
GetGroupByClause GetArvingClause GetSelectCountClause GetSelectCountClause GetSelectCountClause GetSelectCountClause GetViltorCountSql GetViltorCountSql GetViltorCountSql GetViltorCountSql GetViltorCountSql GetViltorCountSql GetViltorCountSql GetViltorCountSql GetViltorCountSql GroupData InverseClauses JoinTables ProcessValue QueryTable SqlVisitorQuery TranslateOperatorToString Nested Types GroupReference SelVistorDost GroupReference SelVisitorQuery Table Methods GroupReference SelVistorDost GetViltorCountEft getColumnRight Properties ColumnLeft getColumnRight TableLeft TableLeft		
GetHavingClause GetSelectCountClause GetSelectCountClause GetSelectDosClause GetSelectDosClause GetSelectDosClause GetVisitorDosCol GetVisitorDo		
GetSelectCountClause GetSelectDoxClause GetSelectDoxClause GetSelectDoxClause GetVisitorCountSql GetVisitorCountSql GetVisitorCountSql GetVisitorDoSql GetVisitorDoSql GetVisitorDoSql GetVisitorQuess JoinTables ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn CouryTable SqlVisitorQuery TranslateOperatorToString Rested Types GroupReference SqlVestorCountSql Hethods GroupReference SqlVestorCounnRight Properties ColumnRight Properties ColumnRight TableIeft		
 GetSelectIDsClause GetVilionCountSql GetVilionCountSql GetVilionCountSql GroupData InverseClauses JoinTables ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn ProcessColumn SqlVisitorQuery TranslateOperatorToString Nested Types GroupReference GroupReference GroupReference GroupReference GroupReference Hethods GroupReference Methods GroupReference Hethods GroupReference Invested Types 		
GetVisitorCountSql GetVisitorDSSql GetVisitorDSSql GetVisitorDSSql GroupData InverseClauses JoinTables ProcessColumn ProcessValue QueryTable SqlVisitorQuery TranslateOperatorToString Nested Types GroupReference ST ColumnName Table Methods GroupReference SoinTableReference Gess GetUnnLeft focumnRight Properties ColumnRight Table.eft ColumnRight Table.eft ColumnLeft ColumnRight Table.eft ColumnRight Table.eft ColumnRight Table.eft ColumnRight GroupReference ColumnRight Table.eft ColumnRight Table.eft ColumnRight Table.eft ColumnRight ColumnRight ColumnRight SoinTable.eft ColumnRight Stable.eft ColumnRight SoinTable.eft SoinTa	- 🙀	GetSelectIDsClause
 GetViltor/DSQI GetViltor/DSQI GetWhereClause (+ 1 overload) GroupData InverseClauses JoinTables ProcessColumn ProcessValue QueryTable SqlVisitorQuery TranslateOperatorToString Nested Types GroupReference Table Properties ColumnName Table Methods GroupReference BinTableReference GetVisitorQuery Fields columnRight Properties ColumnRight Properties ColumnRight Table.eft 		
	- - Q	GetVisitorCountSql
GroupData InverseClauses JoinTables JoinTables ProcessColumn ProcessColumn ProcessColumn SqlVisitorQuery TranslateOperatorToString Nested Types GroupReference Class ColumnName Table Methods GroupReference S ColumnRight Properties ColumnLeft S ColumnRight Properties ColumnLeft S ColumnLeft S ColumnLeft S ColumnRight Table.eft		
 JoinTables ProcessColumn ProcessColumn ProcessColumn ProcessColumn QueryTable SqlVisitorQuery TranslateOperatorToString Nested Types GroupReference Properties ColumnName Table Methods GroupReference BinTableReference Class Fields columnRight Properties ColumnRight Table.eft ColumnLeft ColumnRight Table.eft Table.eft 		GroupData
ProcessColumn ProcessColumn ProcessValue QueryTable SqlVisitarQuery TranslateOperatorToString Nested Types GroupReference Class Properties ColumnName Table Hethods GroupReference Class Interface Fields ColumnRight Properties ColumnRight Properties ColumnRight TableLeft TableLeft TableLeft		Inverse Lauses
ProcessValue QueryTable SqlVisitorQuery TranslateOperatorToString Nested Types GroupReference Class Properties ColumnName Table Hethods GroupReference Cass Fields ColumnLeft C		ProcessColumn
QueryTable SqlVisitarQuery TranslateOperatorToString Nested Types GroupReference Class Properties ColumnName Table Methods GroupReference JoinTableReference Gass Fields columnLeft ColumnRight Properties ColumnLeft ColumnRight Table.eft		
SqlVisitorQuery TranslateOperatorToString Nested Types GroupReference Class Properties ColumnName Table Methods GroupReference JoinTableReference Class JoinTableReference B Fields ColumnRight Properties ColumnRight Table.eft ColumnLeft ColumnRight Table.eft ColumnLeft ColumnRight Table.eft ColumnLeft		
Nested Types GroupReference Cass Properties ColumnName Table Methods GroupReference JoinTableReference Fields Fields Fields ColumnRight Properties ColumnRight Fields		
GroupReference Image: Class Image: ColumnName Image: ColumnName Image: ColumnName Image: ColumnName Image: ColumnRight Image: ColumnLeft Image: ColumnLeft Image: ColumnLeft Image: ColumRight Image: ColumRight		
Class □ Properties □ ColumnName □ Table □ Methods □ Methods □ GroupReference ③ Cless □ Fields □ Fields □ Properties □ ColumnRight □ Properties □ ColumnLeft □ ColumnRight □ TableLeft	⊨ Nes	ted Types
Class □ Properties □ ColumnName □ Table □ Methods □ Methods □ GroupReference ③ Cless □ Fields □ Fields □ Properties □ ColumnRight □ Properties □ ColumnLeft □ ColumnRight □ TableLeft	6	nunReference
Image: Second system Image: Second system Image: Second system JoinTableReference Image: Second system		o apreciencies
Image: Second system Image: Second system Image: Second system JoinTableReference Image: Second system		
Image: Second state of the second		
Hethods GroupReference JoinTableReference Gess Fields for columnLeft for columnRight Properties ColumnRight ColumnRight TableLeft TableLeft		
		Table
JoinTableReference		Methods
Cless		GroupReference
Cless	\geq	
Fields for the set of		
Image: State of the state	C	
 <i>w</i>² columnRight <i>Properties</i> ²⁸ ColumnLeft ²⁸ ColumnRight ²⁸ TableLeft 	Cla	
E Properties	н	
ColumnLeft ColumnRight TableLeft	н	🔗 columnLeft
 ColumnRight TableLeft 	Ш	🔗 columnLeft 🔗 columnRight
🛅 TableLeft	Ш	🔗 columnLeft 🔗 columnRight
	Ш	
T TableRight	Ш	columnLeft columnRight Properties ColumnLeft ColumnLeft ColumnLeft ColumnRight
	Ш	columnLeft columnRight Properties ColumnLeft ColumnRight TableLeft
	Ш	columnLeft columnRight Properties ColumnLeft ColumnRight TableLeft
	Ш	columnLeft columnRight Properties ColumnLeft ColumnRight TableLeft

The properties of the SqlVisitorQuery class are private and read-only.

The methods of this class are described in the <code>VisitorQuery</code> class description section.



Sitecore.Analytics.Data.DataAccess.DataAdapters.ClauseBase

This is the base class for all the classes that handle the rule clause. It defines the common methods and properties of the clause.

ClauseBase 🔊	MethodDescription		
► Methods ■ Equals ■ Equals To ■ GetClauseTables ■ GetHashCode ■ operator != ■ operator ==	bool Equals(object obj)	Is the regular C# Equal method and it determines whether or not the specified clause object is equal to that of the current clause.	
	bool EqualsTo(ClauseBase clause)	Determines whether or not the specified clause is equal to the current clause. It is a special version of the previous method that only compares ClauseBase objects.	
	<pre>AnalyticsTable[] GetClauseTables()</pre>	Returns the array of tables that are used in the clause.	
	<pre>int GetHashCode()</pre>	Returns the hash code for the current clause. This hash code is suitable for hashing algorithms and data structures in a hash table.	

The following table describes the methods of the ClauseBase class:

Sitecore.Analytics.Data.DataAccess.DataAdapters.ColumnClause

You can use the ColumnClause clause in a query that compares two columns of two tables.

For example:

```
Viritors.VisitorId = AutomationStates.VisitorId
```

This class works with two operands left and right that represent the table columns and the conditional operator.

ColumnClause	The following table describes the properties of the ColumnClause class:PropertyDescription	
 Properties ColumnNameLeft ColumnNameRight Operator TableLeft TableRight 	string ColumnNameLeft	Represents the column name in the left operand of the clause. In the previous example, ColumnNameLeft is assigned to VisitorId
Methods ColumnClause QualsTo QetClauseTables	string ColumnNameRight	Represents the column name in the right operand of the clause.
	QueryConditionOperator Operator	Represents the logical operator.
	AnalyticsTable TableLeft	Represents the table name in the left operand. In the previous example, TableLeft is assigned to Visitors
	AnalyticsTable TableRight	Represents the table name in the right operand.



The following table describes the methods of the ColumnClause class:

Method	Description
ColumnClause(AnalyticsTable tableLeft, string columnNameLeft, AnalyticsTable tableRight, string columnNameRight, QueryConditionOperator @operator)	Initializes a new instance of the ColumnClause class.
bool EqualsTo(ClauseBase obj)	Determines whether or not the clauses are the same. Note You should not use this method. It is reserved for the internal logic.
<pre>AnalyticsTable[] GetClauseTables()</pre>	Returns an array of the tables that are used in the clause.

Sitecore.Analytics.Data.DataAccess.DataAdapters.ComplexClause

A complex clause is a clause that contains two or more logical expressions that are associated with a logical operator, such as and and or.

For example:

```
select VisitorId from Visitors as v where v.ExternalUser= 'someuser' and v.Value > 5
```

This query consists of two clauses associated by the and operator. To construct the complex clause in this query, you should use the following code snippet:

```
var clauseLeft = new ValueClause(AnalyticsTable.Visitors, "ExternalUser", "someuser",
QueryConditionOperator.Equal);
var clauseRight = new ValueClause(AnalyticsTable.Visitors, "Value", 5,
QueryConditionOperator.GreaterThan);
var clause = new ComplexClause(clauseLeft, ClauseOption.And, clauseRight);
query.AddClause(clause, "somegroupname");
```

This is an example and you can also use the AddClause method in the VisitorQuery class that accepts two clauses and an operator.

ComplexClause 🔊	The following table describe	describes the properties of the ComplexClause class:	
Class → ClauseBase	Property	Description	
Properties	ClauseBase Left	Represents the left clause of the statement.	
Ieft G Option 양 Right	ClauseOption Option	Represents the logical operator that is used to aggregate the clause.	
Methods	ClauseBase Right	Represents the right clause of the statement.	
 ○ ComplexClause ○ EqualsTo ○ GetClauseTables 			



The following table describes the methods of the ComplexClause class:

Method	Description	
ComplexClause(ClauseBase left, ClauseBase right, ClauseOption option)	Takes the left and right clauses, logical operator and initializes the complex clause.	
bool EqualsTo(ClauseBase obj)	Determines if the specified clause is equal to the current complex clause.	
	Note You should not use this method. It is reserved for the internal logic.	
AnalyticsTable[] GetClauseTables()	Returns an array of the tables that are used in the complex clause.	

Sitecore.Analytics.Data.DataAccess.DataAdapters.HavingClause

You can use this clause in your query to describe the Having clause. For more information, see the article <u>Having (Transact-SQL)</u> in the Microsoft Developer Network.

The Having clause has the following format:

```
HAVING <AggregateFunction>(<ColumnName>) <QueryConditionOperator> <Value>
```

For example:

```
HAVING Count (LineTotal) > 100
```

HavingClause 🔕	The following table describes the properties of the HavingClause class:	
Class	Property	Description
Properties	string ColumnName	Represents the name of the column that is used in the Having clause.
ColumnName Function Operator Table	AggregateFunction Function	Represents the function that is used in the Having clause.
Methods EqualsTo GetClauseTables AvingClause		Note AggregateFunction currently supports Count and DistinctCount.
	QueryConditionOperator Operator	The operator that is used in the Having clause.
	AnalyticsTable Table	The table that is used in the Having clause.
	object Value	The value to compare the function value with.



The following table describes the methods of the ${\tt HavingClause}$ class:

Method	Description
bool EqualsTo(ClauseBase obj)	Determines if the clause object is equal to the current Having clause.
	Note You should not use this method. It is reserved for the internal logic.
<pre>AnalyticsTable[] GetClauseTables()</pre>	Returns an array of the tables that are used in the Having clause.
<pre>HavingClause(AnalyticsTable table, string columnName, AggregateFunction function, object value, QueryConditionOperator @operator)</pre>	Initializes a new instance of the HavingClause class.

Sitecore.Analytics.Data.DataAccess.DataAdapters.ValueClause

You should use this clause in the query that compares a table column with a value.

For example:

```
Visitors.VisitorNumber = 5
```

ValueClause	The following table describes the properties of the ValueClause cla	
Class → ClauseBase	Property	Description
Properties ColumnName	string ColumnName	The column name to compare the value with.
 Operator Table Value 	QueryConditionOperator Operator	The comparison operator.
Methods EqualsTo	AnalyticsTable Table	The table that the column belongs to.
 GetClauseTables ValueClause 	object Value	The value to compare the column with.

The following table describes the methods of the ValueClause class:

Method	Description
bool EqualsTo(ClauseBase obj)	Determines whether or not the clause object is equal to the current value clause.
	Note You should not use this method. It is reserved for the internal logic.
<pre>AnalyticsTable[] GetClauseTables()</pre>	Gets an array of the tables that are used in the value clause.



Method	Description
<pre>public ValueClause(AnalyticsTable table, string columnName, object value, QueryConditionOperator @operator)</pre>	Initializes a new instance of the ValueClause class.

Sitecore.Analytics.Data.DataAccess.DataAdapters.InverseClause

You can use the InverseClause class to negate the specified clause, you should use the.

InverseClause (€) Class → ClauseBase	The following table describes Property	the only property of the InverseClause class: Description
 Properties Clasuse Methods EqualsTo GetClauseTables InverseClause 	ClauseBase Clause	The clause object to be negated.
	The following table describes the methods of the InverseClause class:MethodDescription	
	bool EqualsTo(ClauseBase obj)	Determines whether or not the clause object is equal to the current inverse clause.
		Note You should not use this method. It is reserved for the internal logic.
	AnalyticsTable[] GetClauseTables()	Gets an array of the tables that are used in the clause.
	InverseClause(ClauseBase clause)	Initializes a new instance of the InverseClause class.

Sitecore.Analytics.Data.DataAccess.DataAdapters.InClause

You can use this clause in your query to describe the IN clause. For more information, see the article \underline{In} (<u>Transact-SQL</u>) in the Microsoft Developer Network.

InClause 🛞	The following table describes the properties of the InClause class:	
Class → ClauseBase	Property	Description
Properties ColumnName	string ColumnName	The name of the column to use as a left operand of IN clause.
Parameters Subquery	<pre>object[] Parameters</pre>	The parameters of subquery.
🚰 Table	string Subquery	The subquery that is used as a right operand of
Methods		IN clause.
 ■ EqualsTo ■ GetClauseTables 	AnalyticsTable Table	The table that the ColumName column belongs to.
InClause		



The following table describes the methods of the InClause class:

Method	Description
bool EqualsTo(ClauseBase obj)	Determines whether or not the clause object is equal to the current IN clause.
	Note You should not use this method. It is reserved for the internal logic.
AnalyticsTable[] GetClauseTables()	Gets an array of the tables that are used in the clause.
<pre>InClause(AnalyticsTable table, string columnName, string subquery, object[] parameters)</pre>	Initializes a new instance of the InClause class.

Creating a Segment Builder Rule as a Class

This section describes how to use the API of the Segment Builder to create a rule. This rule selects all the visitors who belong to a specific country.

The input parameters are:

- The name of the country.
- The string comparison operators you can create a rule to specify an operator from a list.

Note

The Segment Builder supports all of the existing types of the rules operators. These operators are defined in the Master database. To navigate to these rules in the content tree: /sitecore/system/Settings/Rules/Common/Operators and /sitecore/system/Settings/Rules/Common/String Operators

In the **Content Editor**, the creation the Segment Builder rules is the same as the creation of the regular Sitecore rules and not covered in this document.

Creating a Visitors Query

To implement a rule class, you should know the format that your SQL statement should follow. You should know which columns and tables to use.

For example, you can use the following SQL to search for certain visitors:

```
select v.VisitorId
from Visitors as v left join Visits as vi on v.VisitorId=vi.VisitorId
where vi.Country group by v.VisitorId
```

You should not worry about the select clause and the aliases because they are automatically created by the VisitorQuery class. You should only specify the tables and columns.

Sitecore® is a registered trademark. All other brand and product names are the property of their respective holders. The contents of this document are the property of Sitecore. Copyright © 2001-2013 Sitecore. All rights reserved.



Creating the Rule Class

The rule that you are creating checks on the string value of the country name and supports the choice of the comparison operator. The rule class should be derived from the

```
Sitecore.Rules.Conditions.StringOperatorCondition<T> class, where T is the rule's context:
```

```
public class CountryCondition<T> : StringOperatorCondition<T> where T :
VisitorRuleContext
{
```

The StringOperatorCondition class contains the GetQueryOperator method that returns the comparison operator of type StringConditionOperator. This comparison operator should be then converted to QueryConditionOperator.

Creating the Rule Parameter

You must represent the rule parameter by a property in the rule class.

```
public string Value
{
  get;
  set;
```

You can specify Country as the rule parameter item and add it as a clause in the rule.

```
string value = this.GetValue(ruleContext);
    var clause = new ValueClause(AnalyticsTable.Visits, "Country", value, @operator);
    ruleContext.VisitorQuery.AddClause(clause,
    ruleContext.GetUniqueConditionGroupName("CountryCondition"));
```

Creating the Rule Logic

In the CountryCondition<T> rule class, you must override the Execute method that is based on StringOperatorCondition<T>.

You must override the Execute method to implement the rule logic:

```
protected override bool Execute([NotNull] T ruleContext)
{
    Assert.ArgumentNotNull(ruleContext, "ruleContext");
}
```

Putting it All Together

Now, you already know the requirements of creating a class to represent a Segment Builder rule. This rule creates a query as an SQL statement.

The following class:

- 1. Converts the comparison operator of type StringConditionOperator to an operator of type QueryConditionOperator.
- 2. Gets the country name and pass it as a rule parameter.
- 3. Creates the new query context and the appropriate clause.



4. Add the created clause to the query.

```
public class CountryCondition<T> : StringOperatorCondition<T> where T :
VisitorRuleContext
  public string Value { get; set; }
 protected override bool Execute([NotNull] T ruleContext)
   Assert.ArgumentNotNull(ruleContext, "ruleContext");
    //converting string operator to query one.
   QueryConditionOperator @operator = this.GetQueryOperator(ruleContext);
   if (@operator == QueryConditionOperator.Unknown)
    {
     Log.Warn("Cannot evaluate visit condition. Condition operator is not defined.",
     this);
      return true;
    }
     //creating rule context with clause
   var innerContext = VisitorRuleContext.Create(ruleContext.Item) as T;
   Assert.IsNotNull(innerContext, "right context");
    //ioin the necessary tables
   innerContext.VisitorQuery.JoinTables(AnalyticsTable.Visitors, "VisitorId",
   AnalyticsTable.Visits, "VisitorId");
    //grouping data by the visitor's ID.
   innerContext.VisitorQuery.GroupData(AnalyticsTable.Visitors, "VisitorId");
    //Add the country clause.
    this.AddClause(innerContext);
   object[] parameters;
    //Retrieving the SQL from the temporary query. This is a part of the method and
    should be the same for almost any rule. The final version of the SQL should look
    like "select VisitorId from Visitors where VisitorId IN (<select SQL from rule1>)
   AND VisitorId IN (<select SQL from rule2>) ...". In this way, you ensure that you
    don't break the whole statement by our rule.
   string subquery = innerContext.VisitorQuery.GetVisitorIDsSql(out parameters);
   var inClause = new InClause(AnalyticsTable.Visitors, "VisitorId", subquery,
   parameters);
   ruleContext.VisitorQuery.AddClause(inClause,
    ruleContext.GetUniqueConditionGroupName("VisitCondition"));
    //return "true" at the end of the rule. This tells to the system that everything
    is fine and your rule should be considered.
   return true:
 protected virtual void AddClause([NotNull] T ruleContext)
   Assert.ArgumentNotNull(ruleContext, "ruleContext");
   QueryConditionOperator @operator = this.GetQueryOperator(ruleContext);
   if (@operator == QueryConditionOperator.Unknown)
      Log.Warn("Cannot evaluate country condition. Condition operator is not
     defined.", this);
     return:
   string value = this.GetValue(ruleContext);
   var clause = new ValueClause(AnalyticsTable.Visits, "Country", value, @operator);
   ruleContext.VisitorQuery.AddClause(clause,
   ruleContext.GetUniqueConditionGroupName("CountryCondition"));
  }
 protected virtual QueryConditionOperator GetQueryOperator([NotNull] T ruleContext)
  {
   Assert.ArgumentNotNull(ruleContext, "ruleContext");
   StringConditionOperator stringOprtator = this.GetOperator();
   return ruleContext.VisitorQuery.TransformOperator(stringOprtator);
 protected virtual string GetValue([NotNull] T ruleContext)
   Assert.ArgumentNotNull(ruleContext, "ruleContext");
   return this.Value ?? string.Empty;
  }
```