Sitecore CMS 6.1

Rules Engine Cookbook

Rules Engine and Conditional Rendering Tips and Techniques for Developers

Table of Contents

Chapter 1

Introduction

Sitecore administrators and developers should read this document before using the Sitecore rules engine. You can use the rules engine to manipulate insert options dynamically, to handle system events, and for conditional rendering (sometimes referred to as personalization, behavioral targeting, multivariate testing, A/B testing, or behavioral marketing).

This document contains the following chapters:

- Chapter 1 — Introduction
- Chapter 2 — The Sitecore Rules Engine
- Chapter 3 — Conditional Rendering

Chapter 2

The Sitecore Rules Engine

This chapter describes concepts supporting the Sitecore rules engine, including an overview of the rules engine, scripts, conditions, actions, and rules.

This chapter contains the following sections:

- Rules Engine Overview
- Scripts
- Conditions
- Actions
- Rules

2.1 Rules Engine Overview

The Sitecore rules engine applies logic defined in actions when specific conditions evaluate to True. The rules engine provides user interfaces to control Sitecore features including insert options and the dynamic response engine (also called conditional rendering).[1]

2.1.1 Scripts

Scripts contain logic to implement specific functions. A script consists of a script definition item that contains inline code or references the .NET class that implements that script.

The fields in the Data section of a script definition item contain:

- **Type** — The.NET class signature (`Namespace.Class, assembly`).
- **Code** — Inline code in any of the languages supported by the .NET framework.
- **References** — List of .NET assemblies to reference, separated by commas (“,”).
- **Language** — A .NET language specifier, such as `CSharp` for C#.

Important
Enter a value in the Type field, or a value in the Code, References, and Language fields. Do not enter values in all four fields.

Important
In most cases, you can use an action in place of a script. For more information about actions, see the section Actions.

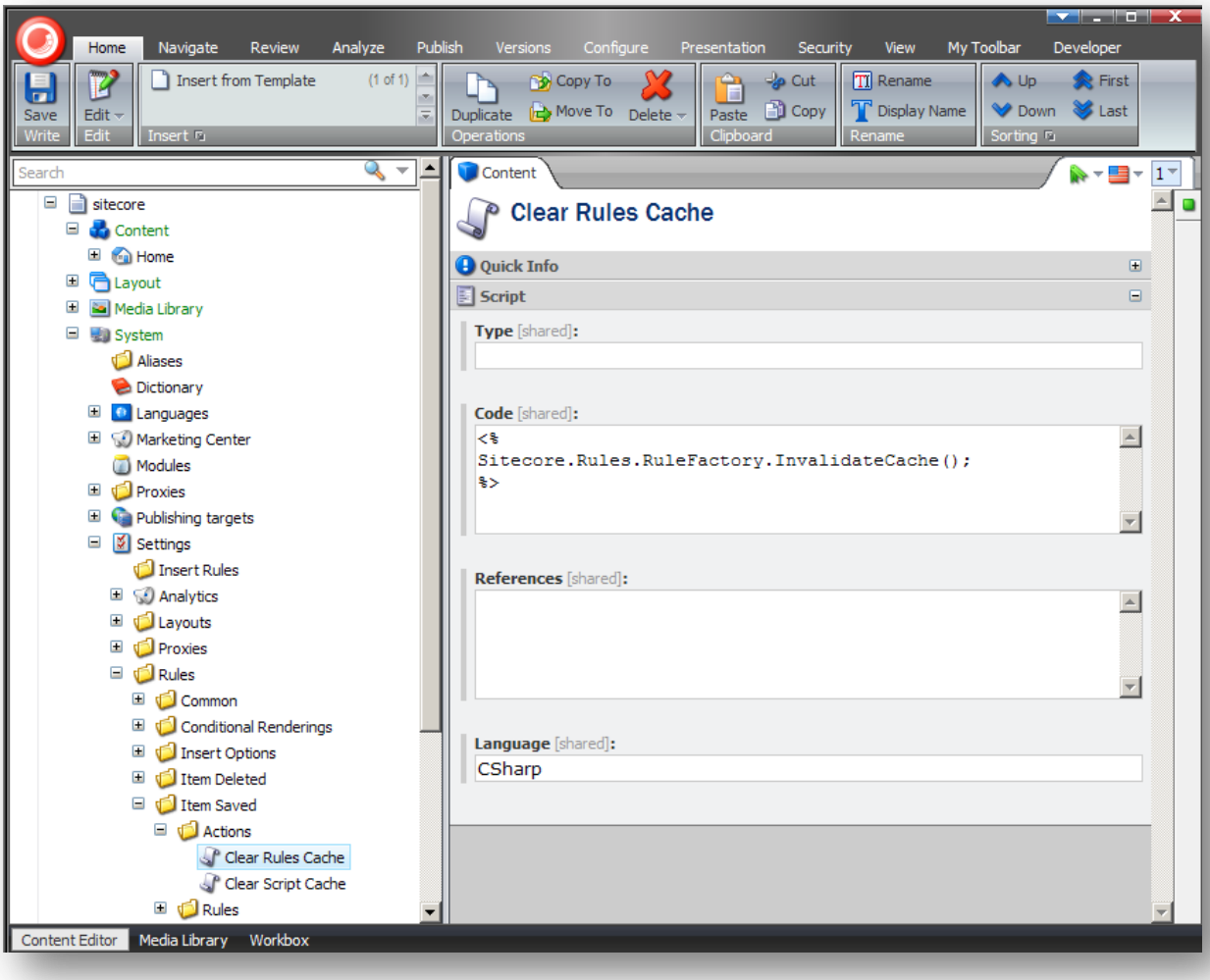
Note
Where these fields appear in other data templates, they serve the purpose described in this section.

Tip
If the value of the Code field begins with `<%` and ends it with `%>` as shown in the following image, Sitecore will wrap the code in the equivalent of the following:

```
namespace Sitecore
{
  public class DefaultClass
  {
    public void DefaultMethod()
    {
      // Contents of <% %>
    }
  }
}
```

For example, the Clear Rules Cache script clears all caches related to the rules engine.

[1] For more information about the layout engine, see the Presentation Component Reference manual at http://sdn.sitecore.net/Reference/Sitecore%206/Presentation%20Component%20Reference.aspx. For more information about insert options, see the Data Definition Reference manual at http://sdn.sitecore.net/Reference/Sitecore%206/Data%20Definition%20Reference.aspx.



Note
As demonstrated in the previous image, you can use a script as an action. For more information about actions, see the section Actions.

How to Implement a Script

To implement a script:

1. In the Content Editor, select the item under which to store the script definition item.
2. In the Content Editor, insert a script definition item using the `System/Rules/Script` data template. Name the script definition item something that clearly and concisely indicates what the script does.
3. In the script definition item, in the Data section, in the Type field, enter the .NET class signature, or enter values for the Code, References, and Language fields.

2.1.2 Conditions

Conditions contain logic to determine whether a single condition is true. For example, the When Descends From condition is true when an item is a descendant of another item.

A condition consists of a condition definition item and the .NET class that implements that condition.

Default Conditions

Sitecore provides a number of conditions that you can use:

- **When Descends From** — when the item is a descendant of a specific item.
- **When Field** — when a field value in the item equals a specific value.
- **When Persona** — when the visitor profile matches a specific persona.
- **When Query** — when the specified Sitecore query relative to the item returns one or more items.
- **When Rule** — when a given condition evaluates to True.
- **When Template** — when the item is associated with a specific data template.

Condition Text

Sitecore uses the value of the Text field in the Data section of condition definition items in user interfaces that allow the user to select conditions and enter condition parameters. Sitecore replaces specific tokens in the Text field with features that allow users to specify parameters for the condition.

Sitecore adds links around the words `if`, `when`, and `where`. If the user clicks the link, Sitecore reverses the condition, alternating between `if` and `except if`, `when` and `except when`, or `where` and `except where`.

Note
If the user reverses the condition, the rules engine automatically reverses the result of the condition logic. Code to implement individual conditions does not account for the user's selection.

Tokens in square brackets ("`[]`") enable links allowing the user to specify parameters. The brackets contain four positional parameters separated by commas ("`,`"):

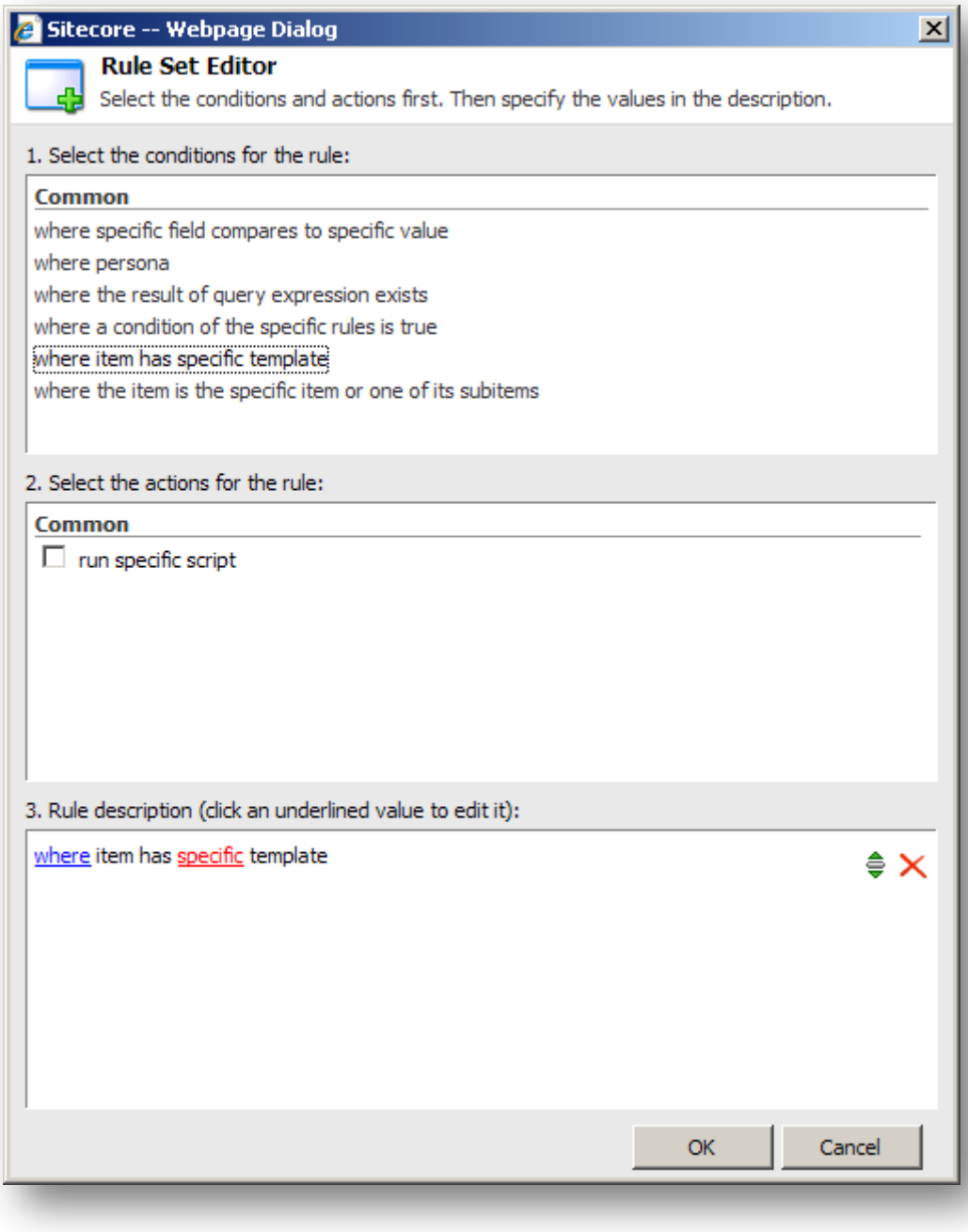
1. The name of the property that the parameter sets in the .NET class that implements the condition.
2. The name of an item beneath the `/Sitecore/System/Settings/Rules/Common/Macros` item providing predefined functionality, or an empty string. For example, to activate a user interface allowing the user to select an item from the content tree, specify the value tree.
3. URL parameters to the user interface activated by the macro specified by the previous parameter, or an empty string. For example, to specify the `/Sitecore/Content/Home` item as the root for a selection tree, enter `root=/sitecore/content/home`.
4. Text to display if the user has not specified a value for the parameter.

Tip
Investigate the default condition definition items for more information about the syntax of the Text field.

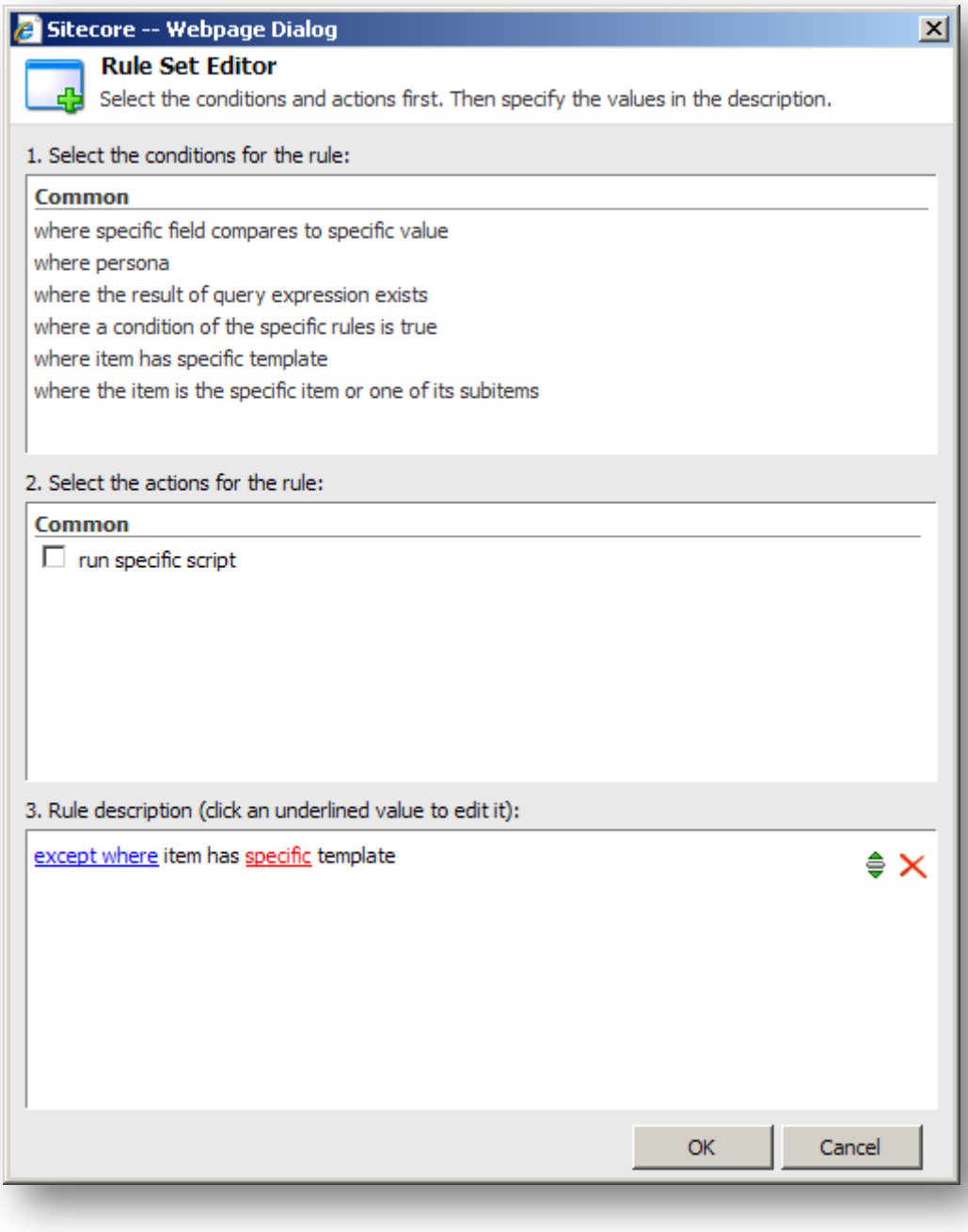
For example, the When Template condition evaluates to True when an item is associated with a specific data template. The text for this condition is as follows:

```
where item has [templateid,Tree,root=/sitecore/templates,specific] template
```

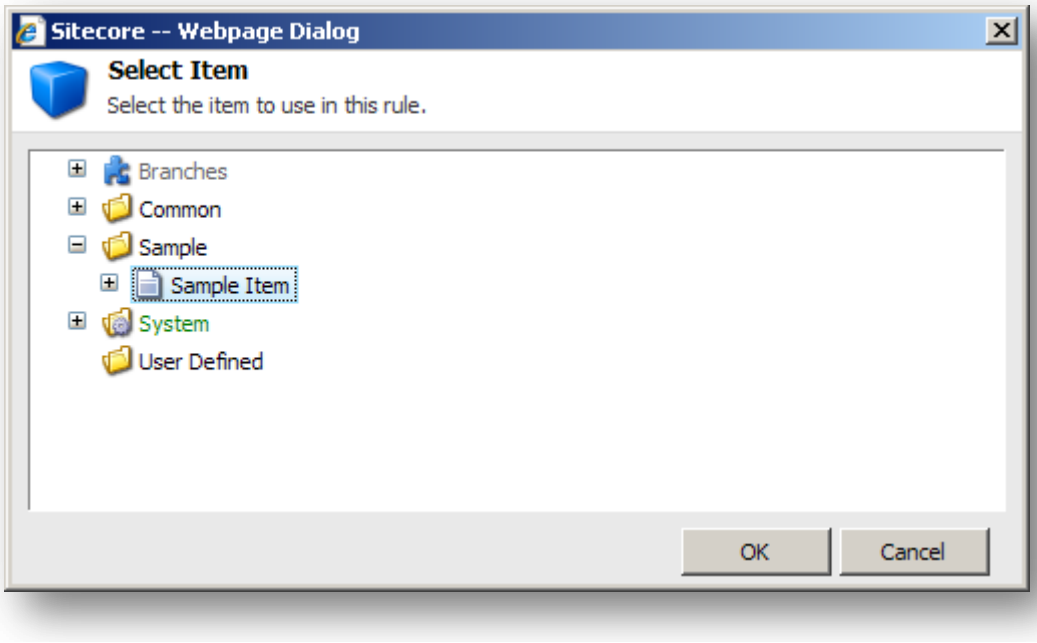
In the user interface that allows to the user to enter condition parameters, Sitecore translates the word where and the tokens in square braces ("`[]`") and renders that text as shown in the third field of the following image showing the rule definition user interface. For more information about defining condition parameters, see the section Rules.



This dialog allows the user to select a data template and optionally reverse the condition. If the user clicks the word where, then Sitecore reverses the condition.



If the user clicks the word specific, then Sitecore prompts the user to select a data template.



The user's selection in this dialog controls the `TemplateID` property of the class that implements the condition.

How to Implement a Condition

To implement a condition:

1. In the Visual Studio project, create the condition class by inheriting from `Sitecore.Rules.Conditions.OperatorCondition`, `Sitecore.Rules.Conditions.WhenCondition`, or `Sitecore.Rules.Conditions.StringOperatorCondition`.

Note
Inherit from `WhenCondition` to implement Boolean conditions. Inherit from `StringOperationCondition` to compare string values. Inherit from `OperatorCondition` to compare values of other types.

Tip
Duplicate an existing condition with similar functionality.

2. In the condition class, implement the `Execute()` method to indicate whether the condition is `true` or `false`.
3. In the Content Editor, select the item under which to store the condition definition item.

Note
Insert condition definition items specific to conditional rendering under the `/Sitecore/System/Settings/Rules/Conditional Renderings/Conditions` item. Insert any other type of condition definition item under the appropriate descendant of the `/Sitecore/System/Settings/Rules` item.

4. In the Content Editor, insert a condition definition item using the `System/Rules/Condition` data template. Name the condition definition item after the .NET class that implements the condition.

5. In the Content Editor, in condition definition item, in the Data section, in the Text field, enter text as described in the section Condition Text.
6. In the Content Editor, in the condition definition item, in the Script section, in the Type field, enter the type signature of the .NET condition class.
7. Configure the condition on one or more conditional rendering rules as described in the section Conditional Rendering Rules.

2.1.3 Actions

Actions contain logic to implement when one or more conditions are true. For example, the Add Insert Option action adds an item to the effective insert options for an item.

An action consists of an action definition item and the .NET class that implements that action.

Note
In cases that do not present any user interfaces, actions can sometimes function as scripts as described in the section Scripts, and scripts can function as actions as described in this section.

Default Actions

Sitecore provides default actions that you can use:

- **Add Insert Option** — adds an item to effective insert options.
- **Run Script** — invokes the specified script.

Action Text

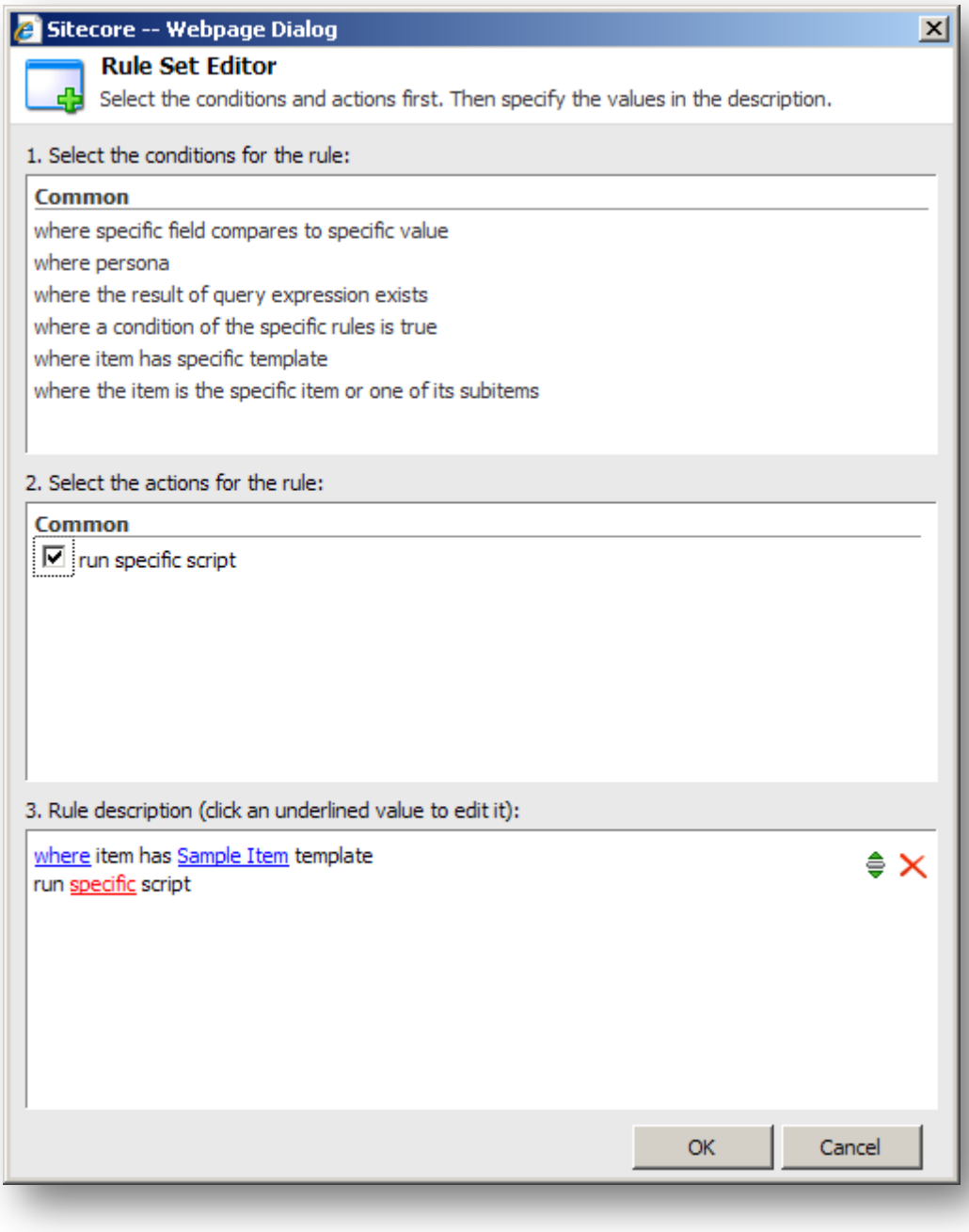
Sitecore uses the value of the Text field in the Data section of action definition items in user interfaces that allow the user to select actions and enter action parameters. For more information about the format of the Text field, see the section Condition Text.

Tip
Investigate the default action definition items for more information about the syntax of the Text field.

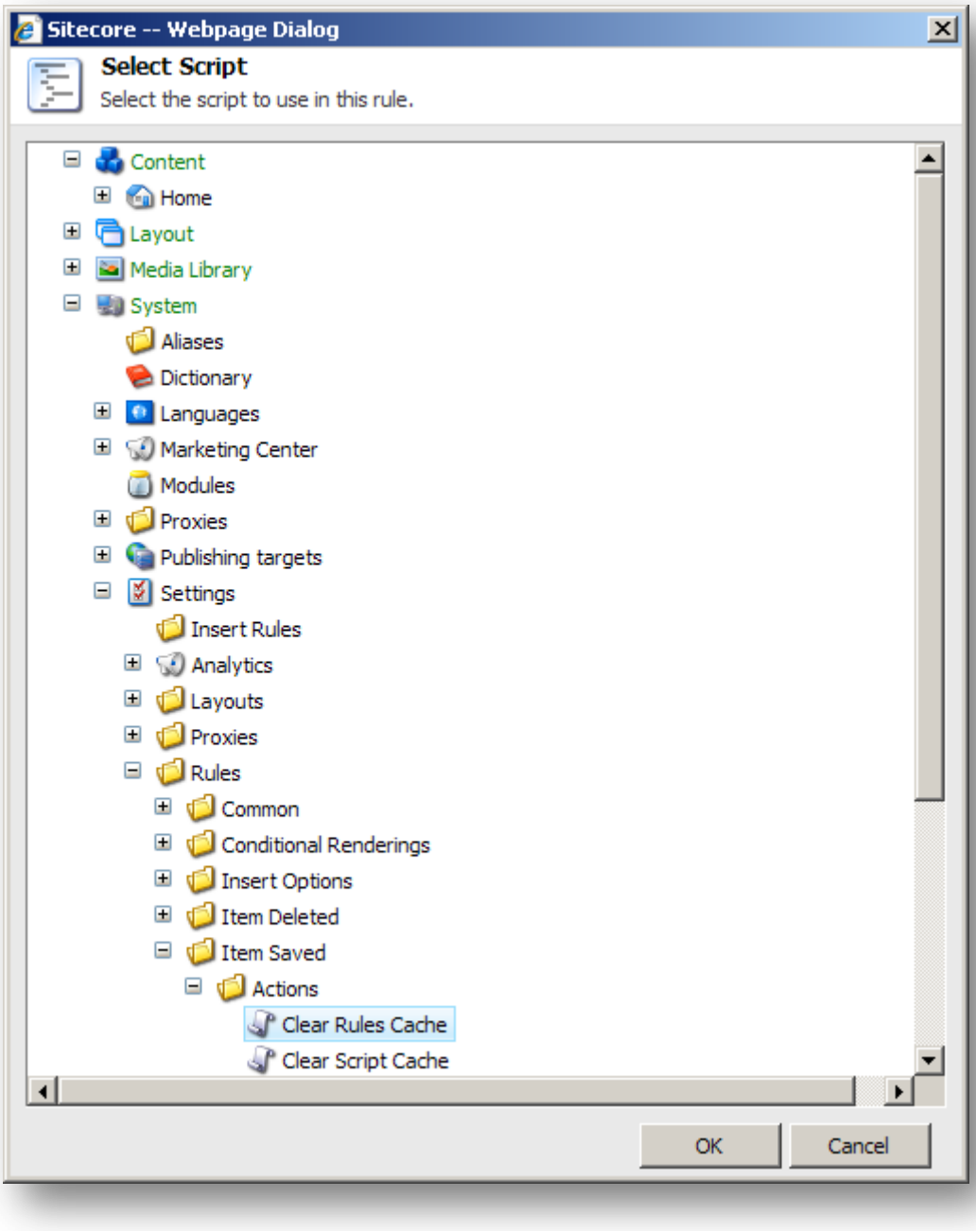
For example, the Run Script action invokes a script. The text for this action is as follows:

```
run [scriptid,Script,,specific] script
```

In the user interface that allows to the user to enter action parameters, Sitecore translates the tokens in square braces (“`[]`”) and renders that text as shown in the second line of the third field of the following image. For more information about defining rule parameters, see the section Rules.



This dialog allows the user to select a script. If the user clicks the word specific, then Sitecore prompts the user to select a script.



What the user selects in this dialog controls the `ScriptID` property of the class that implements the action.

Note
You can configure a script as an action. For more information about scripts, see the section Scripts.

How to Implement an Action

To implement an action:

1. In the Visual Studio project, create a class that inherits from the `Sitecore.Rules.Actions.RuleAction<T>` class, and implement the `Apply()` method.
2. In the Content Editor, select the item under which to store the action definition item.

Note
Insert action definition items specific to conditional rendering under the `/Sitecore/System/Settings/Rules/Conditional Renderings/Actions` item. Insert any other type of action definition item under the appropriate descendant of the `/Sitecore/System/Settings/Rules` item.

3. In the Content Editor, insert an action definition item using the `System/Rules/Action` data template. Name the action definition item after the .NET class that implements the action.
4. In the Content Editor, in the action definition item, in the Data section, in the Text field, enter text as described in the section Action Text.
5. In the Content Editor, in the action definition item, in the Script section, in the Type field, enter the signature of the .NET class that implements the action.
6. Configure one or more rule definition items to reference the action definition item as described in the section Conditional Rendering Rules.
7. Configure the conditional rendering rule(s) for one or more items as described in the section Conditional Rendering Application.

2.1.4 Rules

Rules associate one or more actions with one or more conditions, including parameter values for both actions and conditions. For example, the Conditional Renderings rule processes multivariate tests (action) if multivariate tests are enabled (condition). While it would be illogical in this case, a parameter could reverse the condition (process multivariate tests unless multivariate tests are enabled).

You can use logical operators such as `and` and `or` to only invoke actions under specific combinations of conditions.

Types of Rules

In addition to insert options, insert rules, and the `uiGetMasters` pipeline, you can implement insert rules to control effective insert options at runtime.[2] Configure insert options rules beneath the `/Sitecore/System/Settings/Rules/Insert Options/Rules` item.

Note
Sitecore uses event handlers to invoke rules. To determine when rules execute for events, investigate the corresponding event handlers.

Tip
Use insert rules when you need to select insert options logic for specific items. Use the `uiGetMasters` pipeline for insert options logic that applies to all items. Use insert options rules to allow an administrator to configure insert options using variable logic.

In addition to the `item:deleting` and `item:deleted` events and the `uiDeleteItems` pipeline, you can implement item deleted rules to configure actions to invoke after a user deletes items matching specific criteria. Configure item deleted rules beneath the `/Sitecore/System/Settings/Rules/Item Deleted/Rules` item.

In addition to the `item:saving` and `item:saved` events and the `saveUI` pipeline, you can implement item saved rules to configure actions to invoke after a user saves an item matching specific criteria. Configure item deleted rules beneath the `/Sitecore/System/Settings/Rules/Item Saved/Rules` item.

[2] For more information about insert options, see the Data Definition Reference manual at http://sdn.sitecore.net/Reference/Sitecore%206/Data%20Definition%20Reference.aspx.

In addition to the `item:versionRemoving` and and `item:versionRemoved` events and the `deleteVersionsUI` pipeline, you can implement version removed rules to configure actions to invoke after a user deletes a version of an item matching specific criteria. Configure version removed rules beneath the `/Sitecore/System/Settings/Rules/Version Removed/Rules` item.

Sitecore uses different data templates for conditional rendering, but they follow the same concepts. For more information about conditional rendering, see Chapter 3, Conditional Rendering.

How to Implement a Rule

To implement a rule:

1. In the Content Editor, select the item under which to store the rule definition item.

Note
Insert rule definitions to handle deletion as children of the `/Sitecore/System/Settings/Rules/Item Deleted/Rules` item. Insert rule definitions to handle save as children of the `/Sitecore/System/Settings/Rules/Item Saved/Rules` item. Insert rule definitions to handle version removal as children of the `/Sitecore/System/Settings/Rules/Version Removed/Rules` item. Insert rule definition items specific to conditional rendering under the `/Sitecore/System/Marketing Center/Personalization/Rules` item. Insert global conditional rendering rules under the `/Sitecore/System/Settings/Rules/Conditional Renderings/Global Rules` item. Insert any other type of rule definition item under the appropriate descendant of the `/Sitecore/System/Settings/Rules` item.

2. In the Content Editor, insert a rule definition item using the `System/Rules/Rule` data template.

Note
For conditional rendering rules, use the `System/Rules/Conditional Rendering Rule` data template. For insert options rules, use the `System/Rules/Insert Options Rule` data template. For all other rules, use the `System/Rules/Rule` data template.

3. In the Content Editor, in the rule definition item, in the Data section, in the Name field, enter the name of the rule to appear in user interfaces.
4. In the Content Editor, in the rule definition item, in the Data section, in the Rule field, select conditional rendering conditions and actions, and then enter parameters for both conditions and actions.
5. Apply conditional rendering rules that are not global as described in the section Conditional Rendering Application.

Chapter 3

Conditional Rendering

This chapter provides specific instructions to implement conditional rendering, allowing runtime manipulation of presentation components. This chapter provides a complete example of conditional rendering for Sitecore developers.

This chapter contains the following sections:

- Conditional Rendering Overview
- Conditional Rendering Example Requirements
- Conditional Rendering Implementation
- Conditional Rendering Application

3.1 Conditional Rendering Overview

The term conditional rendering refers to the ability of the Sitecore layout engine to manipulate the presentation controls used to service an HTTP request, such as including or excluding a rendering, controlling its placeholder, setting its data source and other properties, processing multivariate conditions, and other logic.

3.2 Conditional Rendering Example Requirements

This document provides examples of conditional rendering components to meet the following requirements:

- If a random number generated by the computer when a user access the page matches a specific number, a rendering indicates that the user has won.
- The lower limit for the random number is `1`.
- The CMS user must specify the upper limit for the random number.
- The CMS user must specify the number for the random number to match.

3.3 Conditional Rendering Implementation

This section describes the implementation of conditional rendering.

3.3.1 Conditional Rendering Actions

A conditional rendering action contains logic to implement a conditional rendering feature, such as excluding the rendering or setting one of its properties. A conditional rendering action is an action as described in the section Actions. Layout details reference conditional rendering rules, which in turn reference conditional rendering actions.

Default Conditional Rendering Actions

Sitecore provides several default conditional rendering actions that you can use:

- **Hide Rendering** – Prevents the layout engine from including a rendering.
- **Set Data Source** – Sets the data source of a rendering.
- **Set Parameters** – Sets parameters or properties of a rendering.
- **Set Placeholder** – Sets the placeholder to which a rendering will bind.

How to Implement a Conditional Rendering Action

To implement a conditional rendering action:

1. In the Visual Studio project, create a class that inherits from the `Sitecore.Rules.Actions.RuleAction<T>` class, and implement the `Apply()` method.
2. In the Content Editor, select the `/Sitecore/System/Settings/Rules/Conditional Renderings/Actions` item.
3. In the Content Editor, under the `/Sitecore/System/Settings/Rules/Conditional Renderings/Actions` item, insert an action definition item using the `System/Rules/Action` data template.

Tip
Name the conditional action definition item after the .NET class that implements the conditional rendering action.

4. In the conditional action definition item, in the Data section, in the Text field, enter the appropriate pattern. For more information about the Text field, see the section Action Text.
5. In the conditional action definition item, in the Script section, in the Type field, enter the signature of the .NET class that implements the conditional rendering action.
6. Configure the conditional rendering action in one or more conditional rendering rules as described in the section Conditional Rendering Rules.
7. Apply the conditional rendering rule as described in the section Conditional Rendering Application.

Example: Set Rendering Property Conditional Rendering Action

You can implement a conditional rendering action based on the following example that sets the `IsWinner` property of a rendering:

```
namespace Sitecore.Sharedsource.Rules.ConditionalRenderings
{
  using System;
  using Sitecore.Collections;

  [UsedImplicitly]
```

```
        public class SetWinner<T> : Sitecore.Rules.Actions.RuleAction<T>
          where T : Sitecore.Rules.ConditionalRenderings.ConditionalRenderingsRuleContext
        {
          public bool IsWinner
          {
            get;
            set;
          }

          public override void Apply(T ruleContext)
          {
            string key = "winner";

            if (String.IsNullOrEmpty(ruleContext.Reference.Settings.Parameters))
            {
              ruleContext.Reference.Settings.Parameters =
                key + "=" + this.IsWinner.ToString();
              return;
            }

            SafeDictionary<string> parameters = Sitecore.Web.WebUtil.ParseQueryString(
              ruleContext.Reference.Settings.Parameters);

            foreach (string check in parameters.Keys)
            {
              if (String.Compare(check, key, true) == 0)
              {
                key = check;
                break;
              }
            }

            parameters[key] = this.IsWinner.ToString();
            ruleContext.Reference.Settings.Parameters =
              Sitecore.Web.WebUtil.BuildQueryString(parameters, false);
          }
        }
      }
```

For the Text field in the Data section of the conditional rendering action definition item, allow the user to specify a value for the `IsWinner` property if the random number matches:

```
      set parameters to (true or false) [IsWinner,Text,,value]
```

The conditional rendering action class (`Sitecore.Sharedsource.Rules.ConditionalRenderings.SetWinner`) defines the `IsWinner` property. For an image showing these parameters in use, see the section Example: Random Number Conditional Rendering Rule.

Example: Conditional Rendering Action to Add a Rendering

You can implement a conditional rendering action based on the following example that adds a rendering to a placeholder:

```
      namespace Sitecore.Sharedsource.Rules.ConditionalRenderings
      {
        using System;

        public class AddRenderingAction<T> : Sitecore.Rules.Actions.RuleAction<T>
          where  T: Sitecore.Rules.ConditionalRenderings.ConditionalRenderingsRuleContext
        {
          public string Placeholder
          {
            get;
            set;
          }

          public string RenderingID
          {
            get;
            set;
          }
```

```
            public override void Apply(T ruleContext)
            {
              Sitecore.Diagnostics.Assert.IsNotNullOrEmpty(
                this.RenderingID,
                "RenderingID");
              Sitecore.Data.Items.Item rendering =
                Sitecore.Context.Database.GetItem(this.RenderingID);
              Sitecore.Diagnostics.Assert.IsNotNull(rendering, "rendering");
              Sitecore.Layouts.RenderingReference rendRef =
                new Sitecore.Layouts.RenderingReference(rendering);

              if (!String.IsNullOrEmpty(this.Placeholder)
                && String.Compare(this.Placeholder, "specific", true) != 0)
              {
                rendRef.Placeholder = this.Placeholder;
              }

              ruleContext.References.Add(rendRef);
            }
          }
        }
```

In the action definition item, enter the following text to allow the user to select a rendering and enter a placeholder key:

```
        add the [RenderingID,Tree,root=/sitecore/layout/renderings,specific] rendering
          to the [Placeholder,Text,,specific] placeholder
```

3.3.2 Conditional Rendering Conditions

A conditional rendering condition contains logic to determine whether a condition is true in order to determine whether to invoke a conditional rendering action. Layout details reference conditional rendering rules, which reference conditional rendering conditions.

Tip
You can use the conditions described in the section Default Conditions as conditional rendering conditions.

How to Implement a Conditional Rendering Condition.

To implement a conditional rendering condition, follow the instructions in the section Conditions.

Example: Random Number Conditional Rendering Condition

You can implement a conditional rendering condition based on the following example that evaluates to `true` if a random number matches the specified parameter value:

```
        namespace Sitecore.Sharedsource.Rules.Conditions
        {
          using System;

          [UsedImplicitly]
          public class RandomNumberCondition<T> :
            Sitecore.Rules.Conditions.OperatorCondition<T>
            where T : Sitecore.Rules.RuleContext
          {
            private int upperLimit;

            public int MatchNumber
            {
              get;
              set;
            }

            public int UpperLimit
            {
              get
              {
```

```
            Sitecore.Diagnostics.Assert.IsTrue(this.upperLimit > 1, "upper limit");
            return this.upperLimit;
          }

          set
          {
            Sitecore.Diagnostics.Assert.IsTrue(value > 1, "value");
            this.upperLimit = value;
          }
        }

        protected override bool Execute(T ruleContext)
        {
          Sitecore.Diagnostics.Assert.ArgumentNotNull(ruleContext, "ruleContext");
          int random = new Random(DateTime.Now.Millisecond).Next(this.UpperLimit + 1);

          switch (this.GetOperator())
          {
            case Sitecore.Rules.Conditions.ConditionOperator.Equal:
              return random == this.MatchNumber;
            case Sitecore.Rules.Conditions.ConditionOperator.GreaterThanOrEqual:
              return random >= this.MatchNumber;
            case Sitecore.Rules.Conditions.ConditionOperator.GreaterThan:
              return random > this.MatchNumber;
            case Sitecore.Rules.Conditions.ConditionOperator.LessThanOrEqual:
              return random <= this.MatchNumber;
            case Sitecore.Rules.Conditions.ConditionOperator.LessThan:
              return random < this.MatchNumber;
            case Sitecore.Rules.Conditions.ConditionOperator.NotEqual:
              return random != this.MatchNumber;
          }

          return false;
        }
      }
    }
```

To implement the conditional rendering condition, follow the instructions provided in the section How to Implement a Condition, but use the `System/Rules/Conditional Rendering Rule` data template instead of the `System/Rules/Rule` data template. In the Content Editor, in the conditional rendering definition item, in the Data section, in the Text field, enter the following text to allow the user to set the `UpperLimit` and `MatchNumber` parameters of the conditional rendering condition, and to select an operator such as the equality operator ("="):

```
when a random number between 1 and [UpperLimit,,,upper limit]
[operatorid,Operator,,compares to] [MatchNumber,,,number]
```

The base class for the conditional rendering condition (`Sitecore.Rules.Conditions.OperatorCondition<T>`) defines the `OperatorId` property. The conditional rendering condition class (`Sitecore.Sharedsource.Rules.Conditions.RandomNumberCondition`) defines the `UpperLimit` and `MatchNumber` properties. For an image showing these parameters in use, see the section Example: Random Number Conditional Rendering Rule.

Note
To meet the strict requirements outlined for this example, the developer could have hard-coded comparison logic instead of allowing the user to select an operator.

3.3.3 Conditional Rendering Rules

Sitecore invokes conditional rendering rules to control conditional rendering logic. Conditional rendering rules associate one or more conditional rendering actions with one or more conditional rendering conditions, including parameter values for both actions and conditions. Conditional rendering rules are rules as described in the section Rules, but based on the `System/Rules/Conditional Rendering Rule` data template instead of the `System/Rules/Rule` data template.

Example: Random Number Conditional Rendering Rule

To implement a conditional rendering rule to meet the example requirements:

1. Insert a conditional rendering rule definition item named Pick a Random Winner as described in the section How to Implement a Rule.
2. In the conditional rendering rule definition item, in the Data section, in the Rule field, under Select the conditions for the rule, click `when a random number between 1 and the upper limit compares to number` to add that condition to the rule.
3. In the conditional rendering rule definition item, in the Data section, in the Rule field, under Select the actions for the rule, select the `user wins` checkbox to add that action to the list of actions executed if the specified condition evaluates to True.
4. In the conditional rendering rule definition item, in the Data section, in the Rule field, under Rule description, click `upper limit`, and then enter the upper limit for the random number generator.
5. In the conditional rendering rule definition item, in the Data section, in the Rule field, under Rule description, click `compares to`, and then select the `is equal to` operator.
6. In the conditional rendering rule definition item, in the Data section, in the Rule field, under Rule Description, click `number`, and then enter the number for the random number to match.
7. Apply the conditional rendering rule as described in the section How to Apply Conditional Rendering.

3.4 Conditional Rendering Application

This section describes how to apply conditional rendering.

3.4.1 How to Apply Conditional Rendering

Configure conditional rendering in layout details of standard values for data templates.[3]

3.4.2 Global Conditional Rendering Rules

For each rendering, in addition to evaluating conditional rendering rules selected in layout details, Sitecore evaluates the global conditional rendering rules defined under the `/Sitecore/System/Settings/Rules/Conditional Renderings/Global Rules` item.

Important
Because they execute for each rendering, it is important to optimize the performance of global conditional rendering rules, especially as the number of renderings specified in layout details increases.

To define a global conditional rendering rule, implement a conditional rendering rule under the `/Sitecore/System/Settings/Rules/Conditional Renderings/Global Rules` item as described in the section How to Implement a Rule, but using the `System/Rules/Conditional Rendering Rule` data template.

[3] For more information about layout details, see the Presentation Component Reference manual at http://sdn.sitecore.net/Reference/Sitecore%206/Presentation%20Component%20Reference.aspx.