Date Dropdown Web Control 1.0 Component Specification

1. Design

The most common form of selecting dates on a web page is a date popup. Often times, however, not all dates may be selected. Some applications may only allow dates that fall on certain days of the month to be selected; some may only allow users to select Thursdays. The Date Dropdown Component is built on top of the standard DropDownList and ListBox web controls, and allows the developer to specify a set of dates for the user to select from.

1.1 Design Considerations

This design is very simple. The custom web-controls (DateDropDownList and DateListBox) are directly inherited from the DropDownList and ListBox standard ASP.NET web-controls. Therefore, full functionality from those standard list controls is present in our custom controls at zero force. The customization relates to allowing only list items with the predefined date/time format. The input and display date/time formats are configured by the properties. The initial selection of the list item(s) is supported through several properties (InitialSelection, InitialSelectionTimeStamp, and InitialSelectionDateFormat) and has several selection algorithms. The user can easily retrieve currently selected date/time items by the SelectedDate (and SelectedDates for DateListBox) property.

The main property of the DateDropDownList and DateListBox classes is the DatePattern. There are several properties in this custom property class. It also allows setting several date/time items generation rules (collection sub-property). Next, the DatePattern will generate the date/time entries for the items list of the control by using all predefined rules. The Rule class defines a custom date/time items generation rule with a flexible format. It has several properties allowing to generate both date and time data with several frequencies' types.

The add/remove methods of the DateDropDownList and DateListBox classes allow to manually manage date/time items stored (and displayed) by those list controls.

The DateDropDownList and DateListBox classes override several methods from the parent class to correctly validate bound data and the list items set manually through the ASP tags. Aslo those methods allow setting date/time items generated by DatePattern class to the list of items for the custom web-controls.

There are several helper classes in this component. The DatePatternConverter class performs serialization/de-serialization for the DatePattern property to/from string. It is needed to save/load DatePattern date to/from ViewState. The ItemValidator class contains the overloaded Validate function, which is very useful for validating of the list items of the custom web-controls to have only date/time data with the predefined format. And the InitialSelectionManager class simplifies selection of the list items of the custom web-controls by abstracting several selection routines.

1.2 Design Patterns

None.

1.3 Industry Standards

XML 1.0, ASP.NET 2.0.

1.4 Required Algorithms

1.4.1 Initial Date Selection

The initial date selection is implemented by three properties:

- InitialSelection – represents a specific date that should be initially selected or a rule that sepecifies the date that should be initially selected or a combination of several dates or rules that specify the dates that should be initially selected(';' delimited and only applicable for DateListBox);
- InitialSelectionTimeStamp – an additional date/time for "Select:ClosestNotBeforeTimeStamp" and "Select:ClosestNotAfterTimeStamp" rules of the InitialSelection property;
- InitialSelectionDateFormat – the date format description string for dates and times data placed in InitialSelection or InitialSelectionTimeStamp property.

The next values are allowed for InitialSelection property:

- "2007-07-05" – just a simple date to select from the list. The date string should be in the correct format, described by InitialSelectionDateFormat property. There may be multiple dates (for DateListBox control supporting multiple selected items). For example, "2007-07-05;2007-07-12;2007-07-19". The ";" delimiter is used.
- "2007-07-05 7:12 PM" – just a simple date/time to select from the list. The date/time string should be in the correct format, described by InitialSelectionDateFormat property. There may be multiple date/times (for DateListBox control supporting multiple selected items). For example, "2007-07-05 9:00 AM;2007-07-12 6:25 PM; 2007-07-19 2:10 AM". The ";" delimiter is used.
- "Select:ClosestToday" – defines a rule to select one date, which is closest to the current date.
- "Select:ClosestNotBeforeToday" – defines a rule to select one date, most close to the current date, but not earlier than current date.
- "Select:ClosestNotBeforeTimeStamp" – defines a rule to select one date, most close to timestamp (defined by InitialSelectionTimeStamp property), but not earlier than that timestamp.
- "Select:ClosestNotAfterToday" – defines a rule to select one date, most close to the current date, but not later than current date.
- "Select:ClosestNotAfterTimeStamp" – defines a rule to select one date, most close to timestamp (defined by InitialSelectionTimeStamp property), but not later than that timestamp.
- "Select:FirstListItem" – this rule will select one date, which is placed in the beginning of the list (ListItems) for that control. So, the topmost element of the list will be selected.
- "Select:LastListItem" – this rule will select one date, which is placed in the ending of the list (ListItems) for that control. So, the bottommost element of the list will be selected.

Please note, all selection rules in the list above can be mixed in the InitialSelection property in the same way as it was shown for the simple dates. Of course, it is only supported for the DateListBox control, which allows multiple selections. For example, the next value for InitialSelection property is valid: “2007-07-05;Select:ClosestToday;Select:LastListItem”. So, up-to three elements can be selected in the list by this expression. Note, the “;” delimiter is used.

The InitialTimeStamp property has a simple text string format, for example “2007-07-05 9:00 AM”. Its format is defined by InitialSelectionDateFormat. Please note, several timestamps can be present in the same string (with “;” delimiter). It may be useful, if several timestamp-oriented selection rules are present in the InitialSelection.

The InitialSelectionDateFormat defines the format to be used for date/time data of the InitialSelection and InitialSelectionTimeStamp properties. This is a simple text string, with the standard formatting used for DateTime Class. Please refer to “Date and Time Format Strings”MSDN article (http://msdn2.microsoft.com/en-us/library/97x6twsz(vs.80).aspx) for more information. Both standard and custom DateTime format strings should be supported. This property always contains just one value, so the same format is used for all date/time values of InitialSelection and InitialSelectionTimeStamp properties.

Please note, all the properties described above can be overridden by the explicit “Selected” property of the ListItems. So, the initial selection algorithm should first check all list elements, are they already selected or not. If no elements of the ListItems are selected, then the InitialSelectionXXX properties will be evaluated and the proper initial selection will be set. But if at least one element of the ListItems has “Selected” property set to “true”, then no initial selection will be performed.

The initial selection setting should be implemented in the overridden OnPreRender(…) method, because some data can be bound before calling to it. In the OnPreRender(…) method all the items in the web-control are set and correctly configured.

1.4.2 Date Generation Pattern Format

Both DateDropDownList and DateListBox controls share the same format of date generation patterns. Only one pattern can be assigned to the control at a time, but each pattern can contain several generation rules. The date generation pattern is described by an XML node inside the ASPX tag for the dedicated control (DateDropDownList and DateListBox). The XML node example is shown below.

```
<cc1:DatePattern InputDateFormat="yyyy-MM-dd"
                 DisplayDateFormat="MM/dd/yyyy"
                 StartDate="2007-01-01"
                 StopDate="2008-12-31">
  <cc1:Rule DateType="Day" DateValue="2007-07-05" />
  <cc1:Rule DateType="Day" DateValue="2007-07-05" TimeValue="3:00 PM" />
  <cc1:Rule DateType="DayMonth" DateValue="02-29" YearDivisor="4" />
  <cc1:Rule DateType="DayMonth" DateValue="02-29" TimeValue="6:00 PM"
  YearDivisor="4" />
  <cc1:Rule DateType="DayOfMonth" DateValue="15" />
  <cc1:Rule DateType="DayOfMonth" DateValue="15" TimeValue="9:45 AM;
  3:00 PM;6:15 PM" />
  <cc1:Rule DateType="DayOfMonth" DateValue="5;10;15;20;25" />
  <cc1:Rule DateType="DayOfMonth" DateValue="10-15" />
  <cc1:Rule DateType="DayOfMonth" DateValue="1" ReverseOrder="true" />
  <cc1:Rule DateType="DayOfWeek" DateValue="Thursday" />
  <cc1:Rule DateType="DayOfWeek" DateValue="Thursday" TimeValue="9:00 AM" />
  <cc1:Rule DateType="DayOfWeek" DateValue="Monday-Friday" />
  <cc1:Rule DateType="DayOfWeek" DateValue="Monday;Wednesday;Friday;Sunday" />
  <cc1:Rule DateType="DayOfWeek" DateValue="Thursday" Interleave="1" />
```

```
</cc1:DatePattern>
```

The simple sub-properties of the DatePattern are:

- InputDateFormat – a simple formatting string used for all dates and times in the DatePattern sub-properties;
- DisplayDateFormat – a simple formatting string used for showing the ListItems of the control. Note, the output can contain not only the dates, but time information too;
- StartDate – the lowest date/time (inclusive) used for generating dates by the given rules;
- StartDate – the highest date/time (inclusive) used for generating dates by the given rules.

The “Rule” is a collection style sub-property of the DatePattern. The description of sub-properties for “Rule” is below:

- DateType – the type of the date to be generated by the rule;
- DateValue – the date value used for generating rule;
- TimeValue – the time value used for generating rule together with the DateValue. This property can be freely combined with any rule;
- YearDivisor – allows generating dates on every year divisible by the provided value. This property is can be used for any DateType;
- ReverseOrder – defines that the DateValue should be calculated not from the start of month, but from the end of month. It can be only used with DateType=”DayOfMonth”;
- Interleave – allows to generate dates on the interleaved weeks. It can be only used with DateType=”DayOfWeek”.

The possible values for DateType sub-property are below:

- Day – defines a rule to generate any given date of the concrete year;
- DayMonth – defines a rule to generate any given date of any year;
- DayOfMonth – defines a rule to generate the date of the concrete day of the any month;
- DayOfWeek – defines a rule to generate the date on the concrete weekdays (or weekends) of the any week.

The possible values for DateValue sub-property are below:

- full date (like “2007-07-05”) – the concrete date of the given day;
- month and day (like “02-29”) – the concrete date of any year;
- day number (like “15”) – the number of the day of any month. The ranges (like “1-31”) and multiple values (like “20;25;27”) are allowed.

- weekday name (like “Thursday”) – the name of the week of any month. The ranges (like “Monday-Wednesday”) and multiple values (like “Tuesday;Thursday;Saturday”) are allowed.

The possible values for TimeValue sub-property are below:

- full time (like “3:12 PM”) – the concrete time of the concrete date. The time format is fully described by the InputDateFormat string, so it can contain seconds, or can be in 24-hour style. The multiple values (like “3:12 PM;6:45 PM”) are allowed.

The YearDivisor values are simple positive numbers greater than 0. For example, use YearDivisor=”4” for leap years.

The ReverseOrder is a boolean, so “true”/”false” values are supported.

The Interleave values are simple positive numbers (including 0). The zero value means no interleaving. It can be used for DayMonth (interleave years), DayOfMonth (interleave months), DayOfWeek (interleave weeks). For example, the value one for DayOfWeek means generating dates every other week. The value two means two weeks gap before generated dates, and so on.

1.4.3 Data Binding and Validation

The data binding process is hooked by overriding PerformDataBinding(…) method in both DateDropDownList and DateListBox classes. The actions to be implemented in this method are described below:

1. The dataSource:IEnumerable argument of this method will be sent to the validation routine.
2. The list items of the control will be sent to the validation routine. Note, the user has an ability to set list items by control tags. So, the format of manually added date/times items should be verified.
3. Call the base implementation of the PerformDataBinding(…) method.
4. Note, all methods above can throw exceptions if validation fails.

The validation routine works almost in the same way for dataSource and for ListItemCollection/string[]/IList<string>:

1. For each element in the dataSource or ListItemCollection/string[]/IList<string>:
   - 1.1. If the dataSource is validated, then check its type (string or DateTime are only allowed). If the type is DateTime, then skip to next element (to step 1), because it is always valid.
   - 1.2. Try to parse the string element to be validated by using the InputDateFormat formatting string. The result should be valid DateTime class instance. The successful parsing means that the element is valid. Please refer to chapter 1.4.5 for more information about date/time parsing.

1.4.4 Web-control Custom Simple Properties and Sub-properties Implementation

The DateDropDownList and DateListBox web-controls have several custom properties. An example implementation for some of them can be found in the provided prototype (refer to “docs\Prototype” folder). The simple properties implementation is trivial. An examples and guidelines for sub-properties implementation are on the next web-site: http://msdn2.microsoft.com/en-us/library/ms178648(VS.80).aspx . And the next information can be helpful for implementing collections properties inside sub-properties: http://msdn2.microsoft.com/en-us/library/ms178654(VS.80).aspx . The next link describes a special trick to correctly incorporate the sub-property to the both inherited control and to the MS Visual Studio designer: http://west-wind.com/WebLog/posts/5452.aspx .

1.4.5 Date and Time Formatting

There are many date/time formatting properties in this component. All of them are described by a simple text string, with the standard formatting used for DateTime Class. Please refer to “Date and Time Format Strings”MSDN article (http://msdn2.microsoft.com/en-us/library/97x6twsz(vs.80).aspx) for more information. Both standard and custom DateTime format strings should be supported. Please note, the items of the developed web-controls can display both date and time information.

Any DateTime object (say dateObj) can be converted to a string by using the formatting in the next way:

string formattedDateObj = dateObj.ToString(formattingString).

The DateTime.ParseExact(…) method (together with DateTimeFormatInfo) can be used for parsing the DateTime objects from the string.

1.4.6 DatePattern Custom Property Serialization and De-serialization

The DatePattern custom property (and all its sub-properties) will be stored in the ViewState as a simple string. The serialization to a string is a task of ConvertTo(…) method of the DatePatternConverter class. The de-serialization is implemented by the ConvertFrom(…) method of the DatePatternConverter class. The format of serialized string is below. It is based on the XML and fully corresponds to the same XML format of the configured DatePattern property (with rules) in the ASPX page. Just minor changes with namespaces were made.

```
<DatePattern InputDateFormat="yyyy-MM-dd"
             DisplayDateFormat="MM/dd/yyyy"
             StartDate="2007-01-01"
             StopDate="2008-12-31">
  <Rule DateType="Day" DateValue="2007-07-05" />
  <Rule DateType="Day" DateValue="2007-07-05" TimeValue="3:00 PM" />
  <Rule DateType="DayMonth" DateValue="02-29" YearDivisor="4" />
  <Rule DateType="DayMonth" DateValue="02-29" TimeValue="6:00 PM"
  YearDivisor="4" />
  <Rule DateType="DayOfMonth" DateValue="15" />
  <Rule DateType="DayOfMonth" DateValue="15" TimeValue="9:45 AM;
  3:00 PM;6:15 PM" />
  <Rule DateType="DayOfMonth" DateValue="5;10;15;20;25" />
  <Rule DateType="DayOfMonth" DateValue="10-15" />
  <Rule DateType="DayOfMonth" DateValue="1" ReverseOrder="true" />
  <Rule DateType="DayOfWeek" DateValue="Thursday" />
  <Rule DateType="DayOfWeek" DateValue="Thursday" TimeValue="9:00 AM" />
  <Rule DateType="DayOfWeek" DateValue="Monday-Friday" />
  <Rule DateType="DayOfWeek" DateValue="Monday;Wednesday;Friday;Sunday" />
  <Rule DateType="DayOfWeek" DateValue="Thursday" Interleave="1" />
</DatePattern>
```

Please refer to CS chapter 1.4.2 for more information about what data of the file relate to which sub-property of DatePattern custom property (and its Rules collection sub-property fields). It is suggested to not put all data and the name of the sub-property to the serialized string, if this sub-property contains an empty string value. Because empty string value means nothing to all the sub-properties, then such approach will slightly improve serialization/de-serialization performance.

The developer can use any internal .NET library (like DOM) or custom code for serialization/de-serialization. But it looks like DOM parser is too heavy and not efficient for writing/parsing such simple XML data. Possible, a custom solution will produce greater performance and will save much memory.

1.5 Component Class Overview

1.5.1 TopCoder.Web.UI.WebControl.DateDropDown Namespace

DateDropDownList:

This is a custom web-control that allows the user to select a single date/time item from a drop-down list. The class is inherited from the standard DropDownList web-control and provides a special support for list items, which can only contain date and/or time. The control supports adding dates to its items list by four different ways:

1) Pattern generation. The user can set period of generation, given date/times, and special rules for automatic generation of date/times.
2) Add method. Special methods for programmatically adding/removing dates are implemented.
3) Data binding. The user can bind external data source to this control items list. The data from data source will be parsed, validated and set to this control.
4) Standard ListItem ASP tags. The standard ASP ListItem tags can be used for manually set the items to this control list. All set data will be validated to have proper display format.

The control supports properties for input date/times parsing and display formatting. The initial (single) selection of the control item can be configured through the implemented properties, or set explicitly.

The selected item can be easily retrieved (as DateTime object) at any time by using a provided function. Note, this control fully maintain all formatting and rendering capabilities of the underlying DropDownList control.

DateListBox:

This is a custom web-control that allows the user to select a single or multiple date/time items from a list. The class is inherited from the standard ListBox web-control and provides a special support for list items, which can only contain date and/or time. The control supports adding dates to its items list by four different ways:

1) Pattern generation. The user can set period of generation, given date/times, and special rules for automatic generation of date/times.
2) Add method. Special methods for programmatically adding/removing dates are implemented.
3) Data binding. The user can bind external data source to this control items list. The data from data source will be parsed, validated and set to this control.

4) Standard ListItem ASP tags. The standard ASP ListItem tags can be used for manually set the items to this control list. All set data will be validated to have proper display format.

The control supports properties for input date/times parsing and display formatting. The initial (single/multiple) selection of the control items can be configured through the implemented properties, or set explicitly.

The selected items can be easily retrieved (as DateTime or DateTime[] object) at any time by using a provided function. Note, this control fully maintain all formatting and rendering capabilities of the underlying ListBox control.

DatePattern:

This class represents a custom DatePattern property for the web control. The abstraction of date generation pattern is implemented by this class. It contains several properties (InputDateFormat, DisplayDateFormat, StartDate, StopDate), which serves as sub-properties of this custom property class. An additional Rules property of this class is a collection of inner Rule-based items. So, the DatePattern custom property class contains several simple sub-properties and a collection sub-property. A special method (generateDates()) will produce an array of generated date items (for each rule), which can be directly reused by any list-based web-control.

Rule:

This class represents a custom Rule property for the web control. The abstraction of a rule for date generation pattern is implemented by this class. It contains several properties (DateType, DateValue, TimeValue, YearDivisor, ReverseOrder, and Interleave), which serves as sub-properties of this custom property class. So, the DatePattern custom property class contains just several simple sub-properties. A special method (generateDates(...)) will produce an array of generated date items for the rule, which can be directly reused by any list-based web-control.

DatePatternConverter:

The custom type converter for converting expandable DatePattern custom property to and from other representations. The main purpose of this class is to serialize the DatePattern object to a string object for further storing in the ViewState. And the de-serialization from string to the DatePattern is supported.

InitialSelectionManager:

The class for managing initial selection of the list-based web-control. The user should provide the values of the initial selection properties and the collection of the list items. The class provides the SelectItems(...) method, which will examine the data of initial selection properties and the list of items, and next will produce a correct initial selection.

ItemValidator:

This is a simple validator for date items. It checks whether the given items in the correct user provided format. The format is defined by a standard .NET formatting date/time string. And the overloaded "validate" method works with several input data types (like string[], IList{string}, etc.).

1.6 Component Exception Definitions

1.6.1 System Exceptions

ArgumentNullException:

This exception is used in many classes for checking null value in arguments. Each method that used this exception has clear documentation stated which explicitly mentions it. See the method documentation for exception details.

ArgumentException:

This exception is used in some classes for invalid values of numeric arguments. Each method that used this exception has clear documentation stated which explicitly mentions it. See the method documentation for exception details.

1.6.2 Custom Exceptions

DateDropDownException:

A common parent for the all custom exceptions of the component. It can help if a new exception added later. The code that catches the common parent exception will still be good. Also it wraps some external exceptions not related directly to this component.

InitialSelectionInvalidDataException:

This exception will describe bad data in the properties used for setting initial selection of the list. The initial selection of the control can be set by using InitialSelection, InitialSelectionTimeStamp, InitialSelectionDateFormat properties. If any data in such properties becomes invalid (absent required property, or exception during parsing, or incorrect numeric value, etc.), then this exception will be thrown.

DatePatternInvalidDataException:

This exception will describe bad data in the properties used for setting date generation pattern. The date generation pattern can be set by using DatePattern property. If any data in sub-properties of this property becomes invalid (absent required property, or exception during parsing, or incorrect numeric value, etc.), then this exception will be thrown.

RuleInvalidDataException:

This exception will describe bad data in the properties used for setting rule of the date generation pattern. The rule of date generation pattern can be set by using Rule property. If any data in sub-properties of this property becomes invalid (absent required property, or exception during parsing, or incorrect numeric value, etc.), then this exception will be thrown.

BindDataInvalidException:

This exception will describe bad data in the data source object to be bound to the list control. The data from external data source (for example, XML file) can be bound to this control. If any entry of bound data becomes invalid (type is not DateTime or String, or the parsing of text string produced an exception, etc.), then this exception will be thrown.

Please refer to the Class Diagram documentation on each method for detailed explanation of the custom exceptions handling.

1.7 Thread Safety

This component is not thread-safe.

The web control classes are not thread safe. These classes don't need to be thread-safe, as the page accessed by each user will create and use its own web control instances, and these objects will never be shared by multiple-threads.

The ItemValidator class is immutable and therefore thread-safe.

All the other classes are mutable and not thread-safe. But they are used only from the web control classes, which do not execute any worker threads.

There is no need to use these controls in a thread safe manner. Besides, these classes extend the predefined controls which are not thread safety. So there is no meaningful way to make them thread safe.

2. Environment Requirements

2.1 Environment

- .NET Framework 2.0 (with ASP.NET 2.0).
- C# 2.0 programming language.

2.2 TopCoder Software Components

- None.

2.3 Third Party Components

- None.

NOTE: The default location for 3rd party packages is ../lib relative to this component installation. Setting the ext_libdir property in topcoder_global.properties will overwrite this default location.

3. Installation and Configuration

3.1 Package Name

TopCoder.Web.UI.WebControl.DateDropDown – contains main classes and custom exceptions of the component.

3.2 Configuration Parameters

None.

3.3 Dependencies Configuration

For this component to work properly, the DateDropDownList and DateListBox controls must be embedded in an ASP.NET page, like all web-controls.

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.

- Follow Dependencies Configuration.
- Execute ‘Nant test’ within the directory that the distribution was extracted to.

4.2 Required steps to use the component

Embed the control on a page, change the design time attributes, and call public methods for adding date items. The embed tag for the control looks like this:

```
<%@ Register Assembly="TopCoder.Web.UI.WebControl.DateDropDown"
Namespace="TopCoder.Web.UI.WebControl.DateDropDown" TagPrefix="cc1" %>
…

<cc1:DateListBox ID="DateListBox1" runat="server"></cc1:DateListBox>
<cc1:DateDropDownList ID="DateDropDownList1" runat="server">
</cc1:DateDropDownList>
```

4.3 Demo

The demonstration of DateDropDownList and DateListBox custom web-controls is shown below. The ASPX source code is provided as an example of most likely user scenario.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication1._Default" %>
<%@ Register Assembly="TopCoder.Web.UI.WebControl.DateDropDown"
Namespace="TopCoder.Web.UI.WebControl.DateDropDown" TagPrefix="cc1" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Dropdown Web Controls Demonstration Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <%-- XML datasources definition --%>
        <asp:XmlDataSource ID="MySource" DataFile="XmlData.xml" runat="server">
        </asp:XmlDataSource>

        <asp:XmlDataSource ID="MySource2" DataFile="XmlData2.xml" runat="server">
        </asp:XmlDataSource>

        <%-- Examples of custom DateDropDownList --%>

        <%-- Set manual data to the control (with explicit initial selection) --%>
        <cc1:DateDropDownList ID="DateDropDownList1" runat="server"
        DisplayDateFormat="yyyy/MM/dd">
            <asp:ListItem>2007/07/05</asp:ListItem>
            <asp:ListItem Selected="True">2007/07/12</asp:ListItem>
            <asp:ListItem>2007/07/19</asp:ListItem>
        </cc1:DateDropDownList>

        <%-- Bind data to the control from the XML file and set
        initial selection --%>
        <cc1:DateDropDownList ID="DateDropDownList2" runat="server"
        DataSourceID="MySource2" DataMember="ListItem" DataTextField="Text"
        InputDateFormat="dd/MM/yyyy HH:mm:ss" DisplayDateFormat="yyyy/MM/dd hh:mm tt"
        InitialSelection="Select:ClosestToday">
        </cc1:DateDropDownList>

        <%-- Generate the data for the control from the Date Pattern. All available rules
        are described. --%>
        <cc1:DateDropDownList ID="DateDropDownList3" runat="server"
        DisplayDateFormat="yyyy/MM/dd"
        InitialSelection="Select:ClosestNotBeforeTimeStamp"
```

```
InitialSelectionDateFormat="MM/dd/yyyy HH:mm"
InitialSelectionTimeStamp="07/05/2005 15:00">
    <cc1:DatePattern InputDateFormat="yyyy-MM-dd"
         DisplayDateFormat="MM/dd/yyyy"
         StartDate="2007-01-01"
         StopDate="2008-12-31">
        <%-- Generate a concrete date --%>
        <cc1:Rule DateType="Day" DateValue="2007-07-05" />
        <%-- Generate a concrete date and time --%>
        <cc1:Rule DateType="Day" DateValue="2007-07-05" TimeValue="3:00 PM" />
        <%-- Generate a concrete day of the month for years divisible by 4
        without reminder --%>
        <cc1:Rule DateType="DayMonth" DateValue="02-29" YearDivisor="4" />
        <%-- Generate a concrete day and time of the month for years divisible
        by 4 without reminder --%>
        <cc1:Rule DateType="DayMonth" DateValue="02-29" TimeValue="6:00 PM"
        YearDivisor="4" />
        <%-- Generate date for 15th day of months --%>
        <cc1:Rule DateType="DayOfMonth" DateValue="15" />
        <%-- Generate date and time for 15th day of months --%>
        <cc1:Rule DateType="DayOfMonth" DateValue="15" TimeValue="9:45 AM;
        3:00 PM;6:15 PM" />
        <%-- Generate a set of days of months --%>
        <cc1:Rule DateType="DayOfMonth" DateValue="5;10;15;20;25" />
        <%-- Generate a range of days of months --%>
        <cc1:Rule DateType="DayOfMonth" DateValue="10-15" />
        <%-- Generate a last day of months --%>
        <cc1:Rule DateType="DayOfMonth" DateValue="1" ReverseOrder="true" />
        <%-- Generate dates for all Thursdays --%>
        <cc1:Rule DateType="DayOfWeek" DateValue="Thursday" />
        <%-- Generate date/times for all Thursdays --%>
        <cc1:Rule DateType="DayOfWeek" DateValue="Thursday" TimeValue="9:00 AM"
        />
        <%-- Generate dates for a range of days in the week --%>
        <cc1:Rule DateType="DayOfWeek" DateValue="Monday-Friday" />
        <%-- Generate dates for a set of days in the week --%>
        <cc1:Rule DateType="DayOfWeek" DateValue="Monday;Wednesday;Friday;Sunday"
        />
        <%-- Generate dates for all other Thursdays (with one week
        interleave) --%>
        <cc1:Rule DateType="DayOfWeek" DateValue="Thursday" Interleave="1" />
    </cc1:DatePattern>
</cc1:DateDropDownList>

<br /><br />

<%-- Examples of custom DateListBox --%>
<%-- Set manual data to the control (with explicit multiple
initial selection) --%>
<cc1:DateListBox ID="DateListBox1" runat="server" SelectionMode="Multiple"
DisplayDateFormat="yyyy/MM/dd">
    <asp:ListItem>2007/07/05</asp:ListItem>
    <asp:ListItem Selected="True">2007/07/12</asp:ListItem>
    <asp:ListItem>2007/07/19</asp:ListItem>
    <asp:ListItem Selected="True">2007/07/26</asp:ListItem>
</cc1:DateListBox>

<%-- Bind data to the control from the XML file and set
initial selection --%>
<cc1:DateListBox ID="DateListBox2" runat="server"
DataSourceID="MySource" DataMember="ListItem" DataTextField="Text"
SelectionMode="Multiple" InputDateFormat="dd/MM/yyyy HH:mm:ss"
DisplayDateFormat="yyyy/MM/dd hh:mm tt"
InitialSelection="Select:FirstItem;Select:LastItem">
</cc1:DateListBox>

<%-- Generate the data for the DateListBox control from the Date Pattern.
All available rules are described. --%>
<cc1:DateListBox ID="DateListBox3" runat="server" DisplayDateFormat="yyyy/MM/dd"
InitialSelection="Select:ClosestNotAfterTimeStamp"
InitialSelectionDateFormat="MM/dd/yyyy HH:mm"
```

```
        InitialSelectionTimeStamp="07/05/2005 15:00">
            <cc1:DatePattern InputDateFormat="yyyy-MM-dd"
                 DisplayDateFormat="MM/dd/yyyy"
                 StartDate="2007-01-01"
                 StopDate="2008-12-31">
                <%-- Generate a concrete date --%>
                <cc1:Rule DateType="Day" DateValue="2007-07-05" />
                <%-- Generate a concrete date and time --%>
                <cc1:Rule DateType="Day" DateValue="2007-07-05" TimeValue="3:00 PM" />
                <%-- Generate a concrete day of the month for years divisible by 4
                without reminder --%>
                <cc1:Rule DateType="DayMonth" DateValue="02-29" YearDivisor="4" />
                <%-- Generate a concrete day and time of the month for years divisible
                by 4 without reminder --%>
                <cc1:Rule DateType="DayMonth" DateValue="02-29" TimeValue="6:00 PM"
                YearDivisor="4" />
                <%-- Generate date for 15th day of months --%>
                <cc1:Rule DateType="DayOfMonth" DateValue="15" />
                <%-- Generate date and time for 15th day of months --%>
                <cc1:Rule DateType="DayOfMonth" DateValue="15" TimeValue="9:45 AM;
                3:00 PM;6:15 PM" />
                <%-- Generate a set of days of months --%>
                <cc1:Rule DateType="DayOfMonth" DateValue="5;10;15;20;25" />
                <%-- Generate a range of days of months --%>
                <cc1:Rule DateType="DayOfMonth" DateValue="10-15" />
                <%-- Generate a last day of months --%>
                <cc1:Rule DateType="DayOfMonth" DateValue="1" ReverseOrder="true" />
                <%-- Generate dates for all Thursdays --%>
                <cc1:Rule DateType="DayOfWeek" DateValue="Thursday" />
                <%-- Generate date/times for all Thursdays --%>
                <cc1:Rule DateType="DayOfWeek" DateValue="Thursday" TimeValue="9:00 AM"
                />
                <%-- Generate dates for a range of days in the week --%>
                <cc1:Rule DateType="DayOfWeek" DateValue="Monday-Friday" />
                <%-- Generate dates for a set of days in the week --%>
                <cc1:Rule DateType="DayOfWeek" DateValue="Monday;Wednesday;Friday;Sunday"
                />
                <%-- Generate dates for all other Thursdays (with one week
                interleave) --%>
                <cc1:Rule DateType="DayOfWeek" DateValue="Thursday" Interleave="1" />
            </cc1:DatePattern>
        </cc1:DateListBox>
    </div>
    </form>
</body>
</html>
```

Please note, all three ways of setting items to custom list controls shown above can be combined together:

- manual list items + bound data from the data source;
- manual list items + generated items by data pattern;
- bound data from the data source + generated items by data pattern;
- manual list items + bound data from the data source + generated items by data pattern.

All the possible options for setting initial selection are shown below. Please note, those options can be combined together to make multiple selection (supported only by DateListBox control if its SelectionMode property was set to Multiple value):

Initial Selection Option	**Description**

InitialSelection=”2007-07-05”
InitialSelectionDateFormat=”yyyy-MM-dd”
	Just a simple date to select from the list.

Initial Selection Option	**Description**
InitialSelection=”2007-07-05 07:12 PM” InitialSelectionDateFormat=”yyyy-MM-dd hh:mm tt”	Just a simple date/time to select from the list.
InitialSelection=“Select:ClosestToday”	Defines a rule to select one date, which is closest to the current date.
InitialSelection=“Select:ClosestNotBeforeToday”	Defines a rule to select one date, most close to the current date, but not earlier than current date.
InitialSelection=“Select:ClosestNotBeforeTimeStamp” InitialSelectionDateFormat=”yyyy/MM/dd HH:mm” InitialSelectionTimeStamp=”2007/07/12 17:45”	Defines a rule to select one date, most close to timestamp, but not earlier than that timestamp.
InitialSelection=“Select:ClosestNotAfterToday”	Defines a rule to select one date, most close to the current date, but not later than current date.
InitialSelection=“Select:ClosestNotAfterTimeStamp” InitialSelectionDateFormat=”yyyy/MM/dd HH:mm” InitialSelectionTimeStamp=”2007/07/12 17:45”	Defines a rule to select one date, most close to timestamp, but not later than that timestamp.
InitialSelection=“Select:FirstListItem”	This rule will select one date, which is placed in the beginning of the list (ListItems) for that control. So, the topmost element of the list will be selected.
InitialSelection=“Select:LastListItem”	This rule will select one date, which is placed in the ending of the list (ListItems) for that control. So, the bottommost element of the list will be selected.

The next JScript code can be used to manually work with the DateDropDownList and DateListBox custom web-controls. So, the fourth method of adding items to the custom list controls is available in addition to three previously described ways.

```
// Manage DateDropDownList custom control from the JScript code
// Add date/time items to the list
DateDropDownList1.AddDateItem( new DateTime(2007, 7, 5) );
DateDropDownList1.AddDateItem( new DateTime(2007, 7, 5, 9, 0, 0) );
DateDropDownList1.AddDateItems( new DateTime[]{ some elements } );
DateDropDownList1.AddDateItems( new DateTime(2007,7,6), new
DateTime(2007,7,12));

// Remove date/time items from the list
// The delCount will be 1 after the next operation
int delCount = DateDropDownList1.RemoveDateItem( new DateTime(2007, 7,
5) );
// The delCount will be 0 after the next operation
delCount = DateDropDownList1.RemoveDateItem( new DateTime(2007, 7,
5) );
```

```
// Remove several items
delCount = DateDropDownList1.RemoveDateItems( new DateTime[]{ some
elements } );
delCount = DateDropDownList1.RemoveDateItems( new DateTime(2007,7,6),
new DateTime(2007,7,12) );

// Get the selected item
DateTime selItem = DateDropDownList1.SelectedDate;


// Manage DateListBox custom control from the JScript code
// Add date/time items to the list
DateListBox1.AddDateItem( new DateTime(2007, 7, 5) );
DateListBox1.AddDateItem( new DateTime(2007, 7, 5, 9, 0, 0) );
DateListBox1.AddDateItems( new DateTime[]{ some elements } );
DateListBox1.AddDateItems( new DateTime(2007,7,6), new
DateTime(2007,7,12));

// Remove date/time items from the list
// The delCount will be 1 after the next operation
delCount = DateListBox1.RemoveDateItem( new DateTime(2007, 7, 5) );
// The delCount will be 0 after the next operation
delCount = DateListBox1.RemoveDateItem( new DateTime(2007, 7, 5) );
// Remove several items
delCount = DateListBox1.RemoveDateItems( new DateTime[]{ some
elements } );
delCount = DateListBox1.RemoveDateItems( new DateTime(2007,7,6), new
DateTime(2007,7,12) );

// Get the first selected item
DateTime selItem = DateListBox1.SelectedDate;

// Get the selected items
DateTime[] selItems = DateListBox1.SelectedDates;
```

5. Future Enhancements

- To add new tags to the Rule class for supporting more date generation algorithms.
- New selection rules may be added to the InitialSelectionManager.