WebScheduler.NET 2.0 White Paper

This white paper discus the new features and enhancement introduced in WebScheduler.NET 2.0

Table of Contents

Timeline View

Overview

Intersoft WebScheduler.NET® is a full-featured Scheduling and Calendaring component for ASP.NET that mixes the beauty of User Interface and rock-solid architecture, allowing you to extend the functionalities and behaviors to the way you need it in your business scenarios.

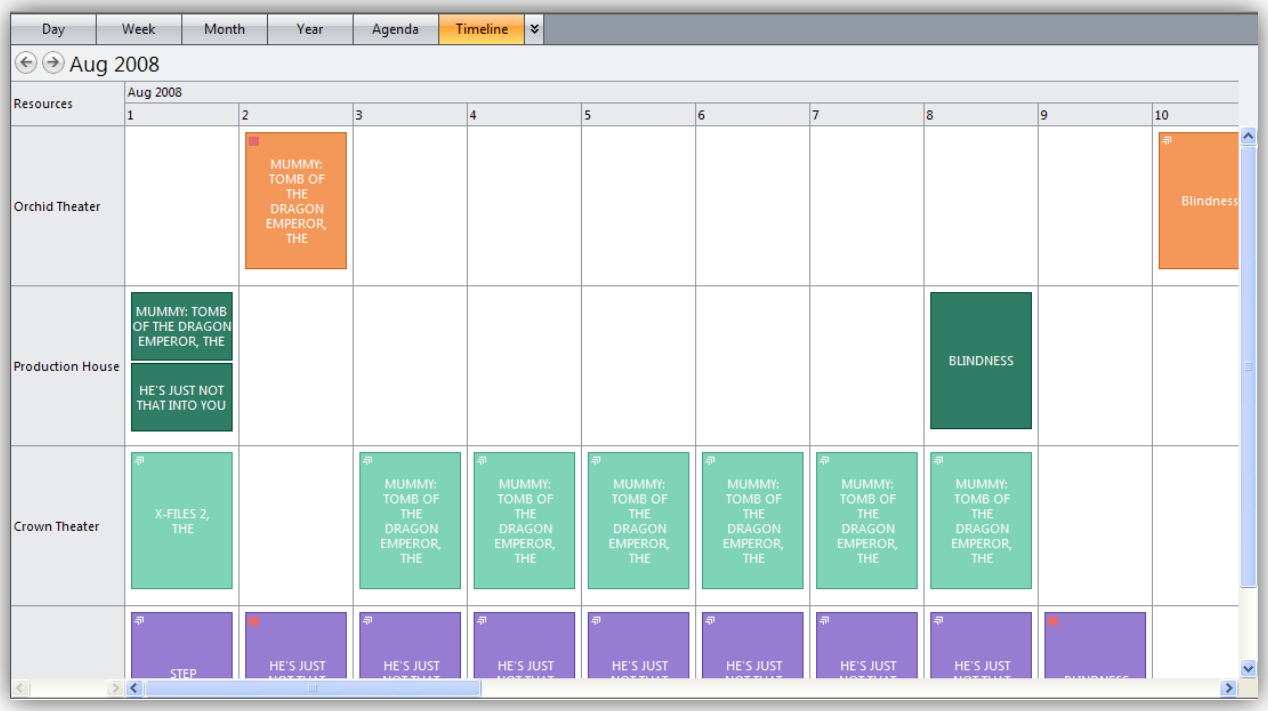
In this second generation, WebScheduler takes Scheduling functionality to the enterprise level by providing innovative Timeline View feature, which is required in most enterprise Web application such as ERP and CRM. The Timeline View feature is designed to allow users to view the event information in an unique way – where resources are listed in the rows and the time in the columns.

This mode enables users to easily look at the event information in a side-by-side manner, while easily compare it with other resources in the top-down manner.

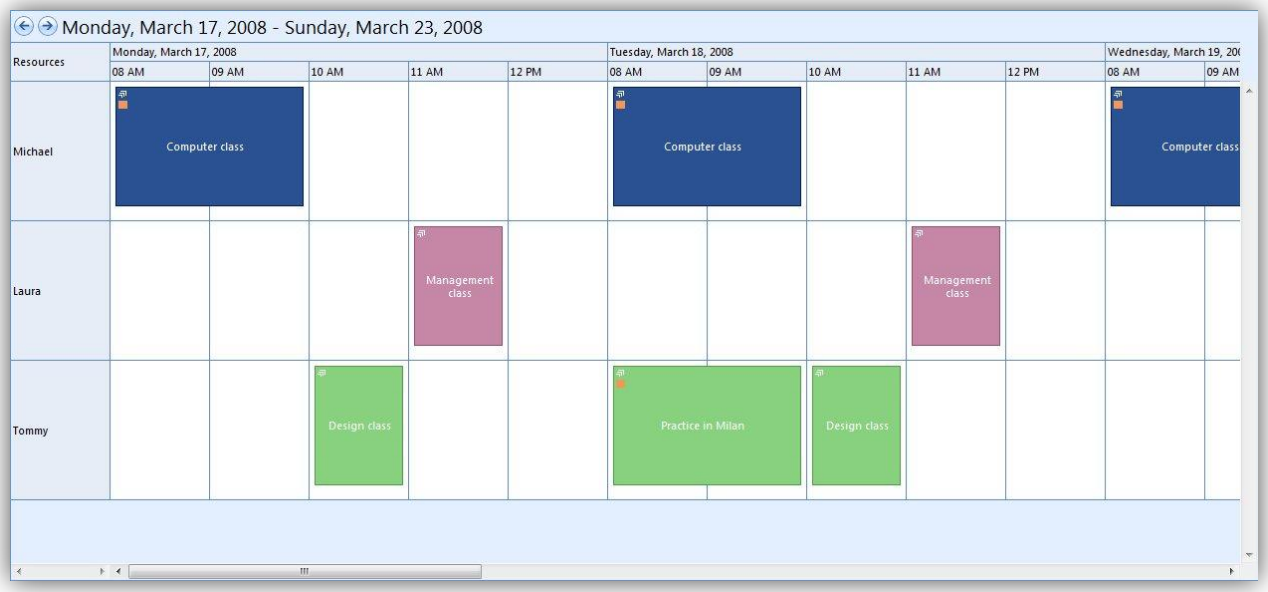
User Interface

Timeline view has 2 type of view.

1. Day view
   The purpose of day view is to give users clear and readable information within 1 month by showing day per day of selected month.



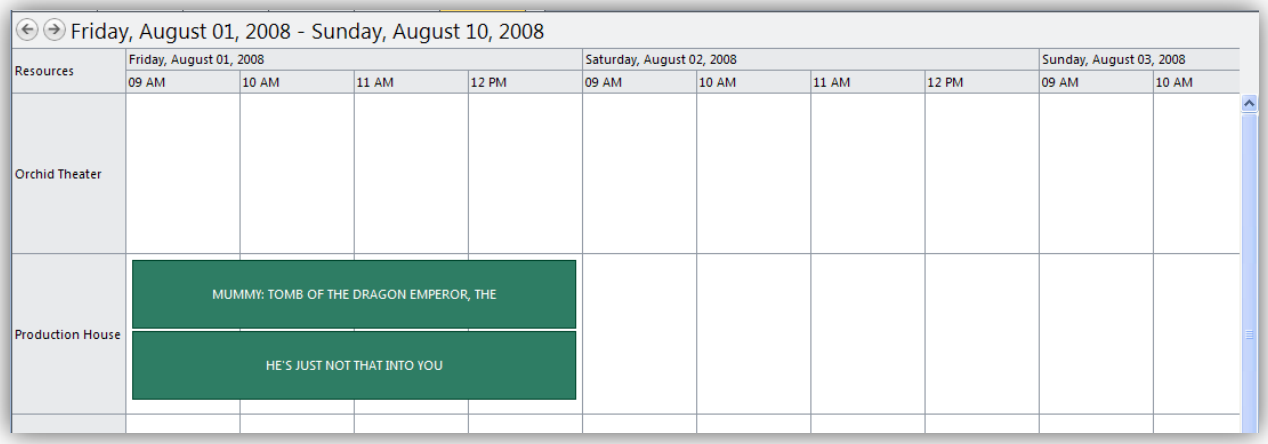
2. Time view
   This view provides user with more detailed information within 1 week by showing hours (based on *StartTime* and *EndTime* property) from day per day of the selected week.



Time view flexibility and customization based on user scenario.

By default, a day (up to 7 days) Timeline in *time view* will show from 08.00 to 17.00. Of course for the flexibility and other advanced scenarios, you can change this setting by customizing *StartTime*, *EndTime* and *NumberOfDays* property (located in *ViewSettings* >> *TimelineView*).

For example, you only need to show the scheduler activity from 09.00 to 12.00 and for 10 days. You can achieve this scenario by set *StartTime* to "t0900", *EndTime* to "t1200" and *NumberOfDays* to "10".



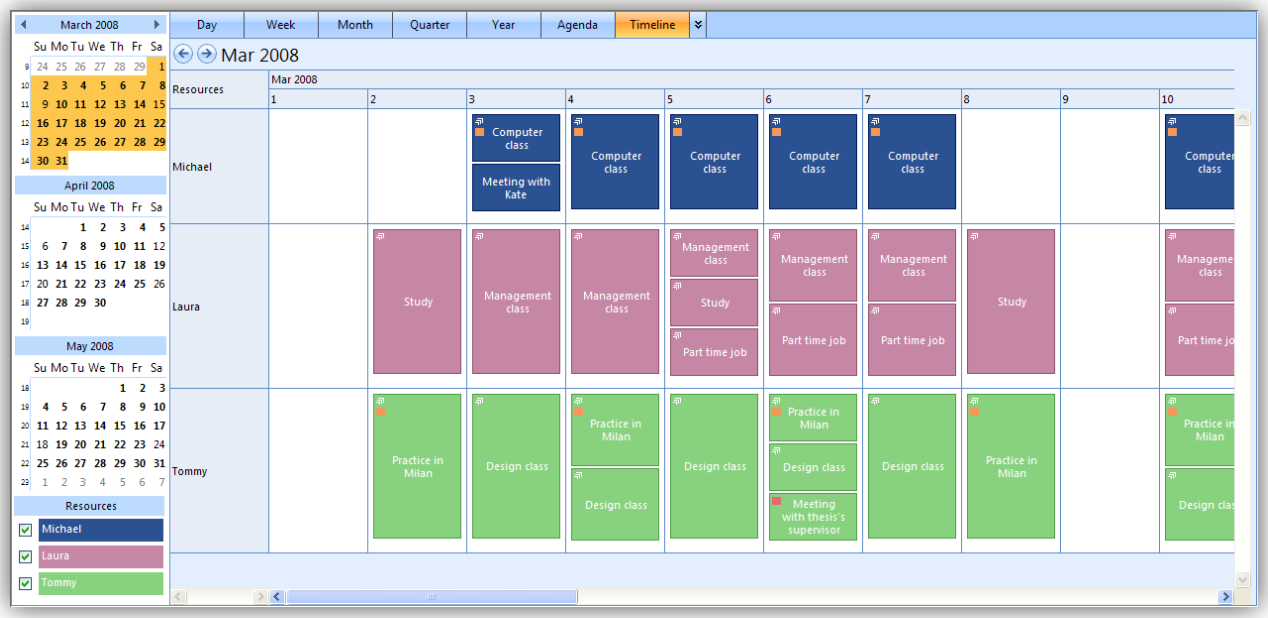
Tooltip information

Date and time information from ContentCell's will be provided with tooltip when user hovers on Timeline ContentCell.

Features

- **Flexible customization in Timeline view.**
  WebScheduler.NET® 2.0 is designed with strong customizability in mind, giving full flexibility for developers to customize the Timeline View's behaviors, look and feels, appearances and styles.

  Here is the list of properties that you can customize in order to achieve your unique Timeline view configuration: *ContentCellWidth*, *EndTime*, *MinimumEventHeight*, *NumberOfDays*, *ResourceCellWidth*, *ResourceRowHeight*, *StartTime*, and *TotalEventInCell*. You can find all of the properties in ViewSettings >> TimelineView.



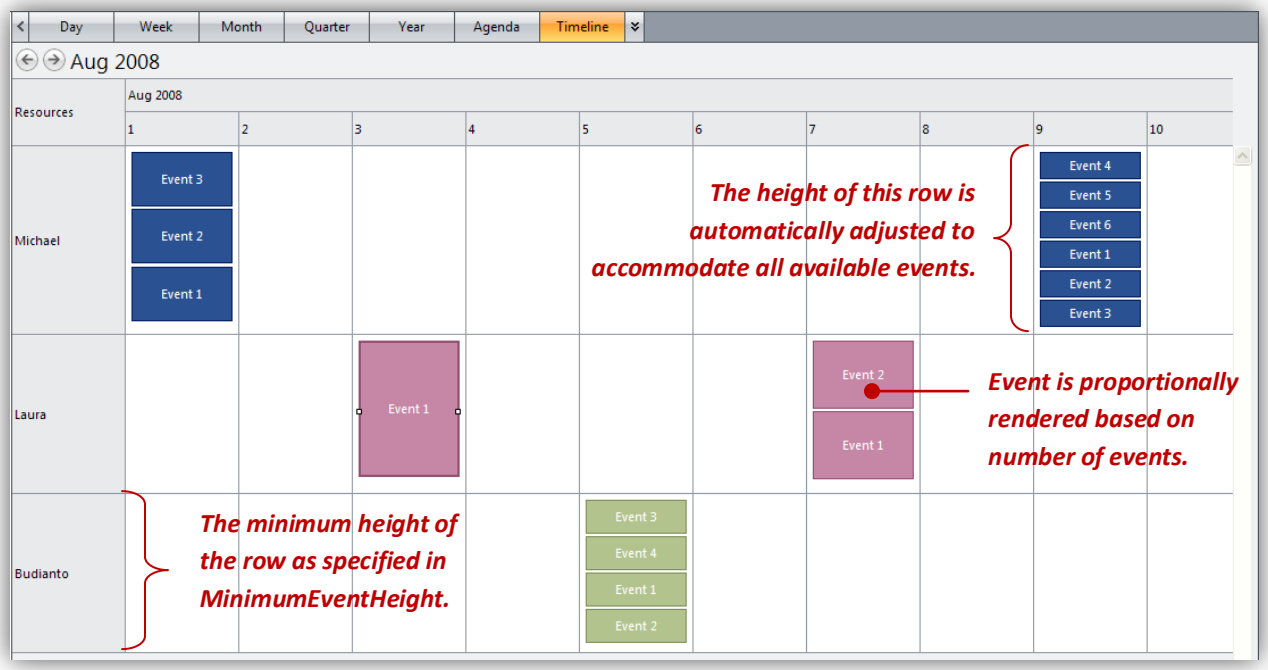
The above screen shot settings are using the following properties:
ContentCellWidth = 100;
StartTime = t0800;
EndTime = t1200;
MinimumEventHeight = 48;
ResourceCellWidth = 100;
ResourceRowWidth = 40;

- **Sophisticated Event Layout**

  A Timeline Cell has capability to show several event(s) in it. The maximum events that will be shown in the cell are based on *TotalEventInCell* property (Default value is 5). There are many scenarios in Timeline View where the number of events can be various based on the application's usage.

  One of the unique strengths in Intersoft WebScheduler 2.0 is its sophisticated management in the Event Layout, which handles the layouting automatically without any user interventions or additional coding. The Event Layout is based on *MinimumEventHeight* and *TotalEventInCell* property. Each event in the cell will have *MinimumEventHeight* as the minimum height, when the events in the cell are more than *TotalEventInCell,* the cell will then automatically adjust to accommodate the exceeding events.

  To get a big picture on the Event Layout management in WebScheduler 2.0, please take a look at the following illustration.

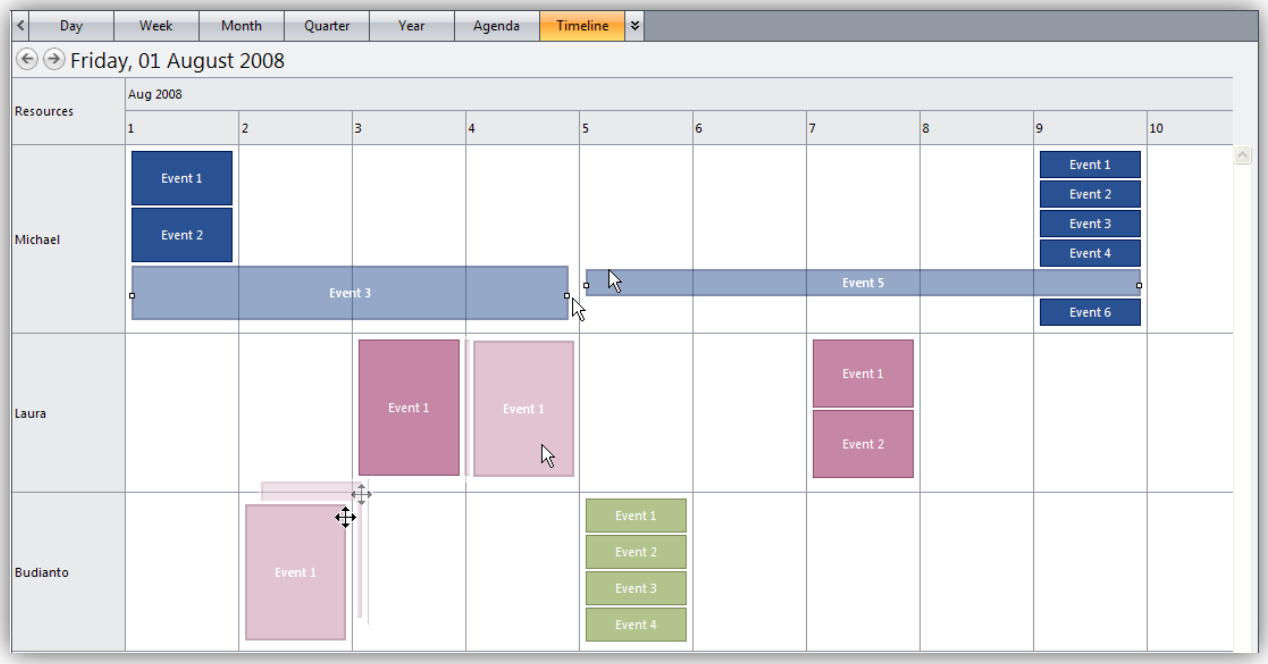


- **Moving and Resizing Event**

  As the other views in WebScheduler support moving and resizing event by using drag-drop, so does *Timeline View* as one of WebScheduler views. In order to be able to do moving and resize event, you should enable *AllowMove* and *AllowResize* property in *DataEditing*.

  You can perform the moving and resizing events in the following scenarios:

  - Moving event from 1 cell to other cell on the same resource.
  - Moving event from 1 cell to other cell on the different resource.
  - Resize event to the left on the same resource.
  - Resize event to the right on the same resource.



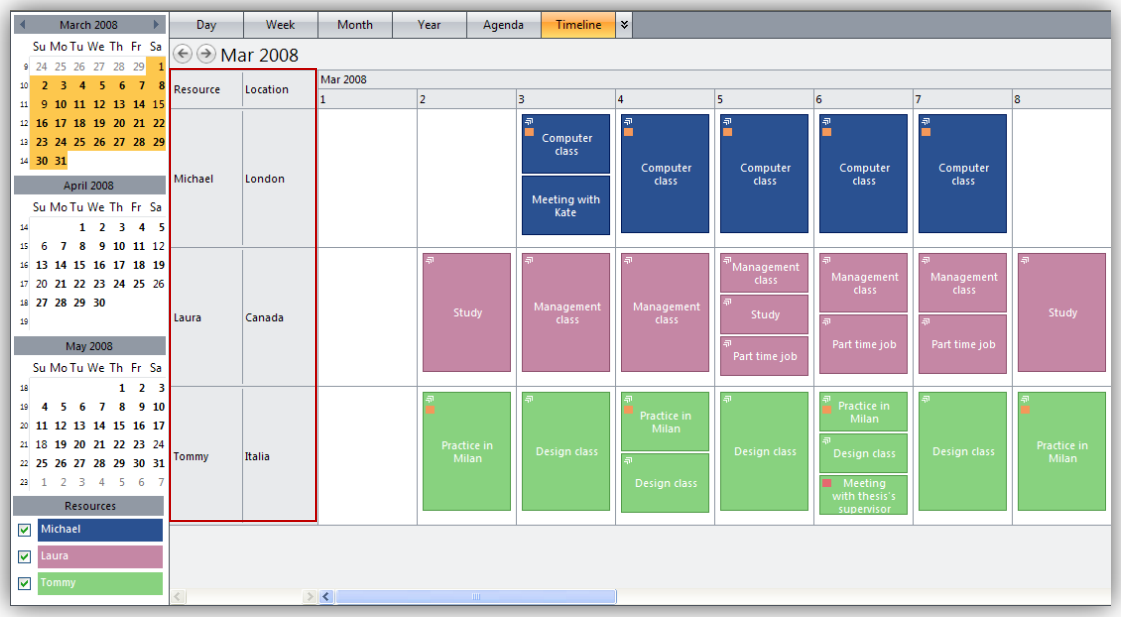
Thanks to the revolutionary RealTimeOperation™ technology in WebScheduler, the updating process in Timeline View is as fast as one tick of second, by taking advantage of the high performance and state-of-the-art rendering architecture. For more information about this feature, please read RealTimeOperation™ concept.

- **Customizing the Resources Content**

  By default, the Timeline View will list the resources at the most left side, which show each resource by its specified *ResourceName* field. However, developers often require customization on the resource content to adapt their business requirements – for instance – showing a multiple columns table in the resource list, instead of a simple *ResourceName* list.

  WebScheduler.NET™ 2.0 provides an elegant way for developers to customize the resources content. Developers can create their own template and assign it to WebScheduler via *ResourceContentTemplate* property. Simply set *ResourceContentMode* to *Custom* value and assign your own template design to achieve your own look and feel.

  The following screenshot shows WebScheduler in Timeline View mode. Notice the Resources description is using a two-column table, instead of the default simple text.



- **One-click View Switching**

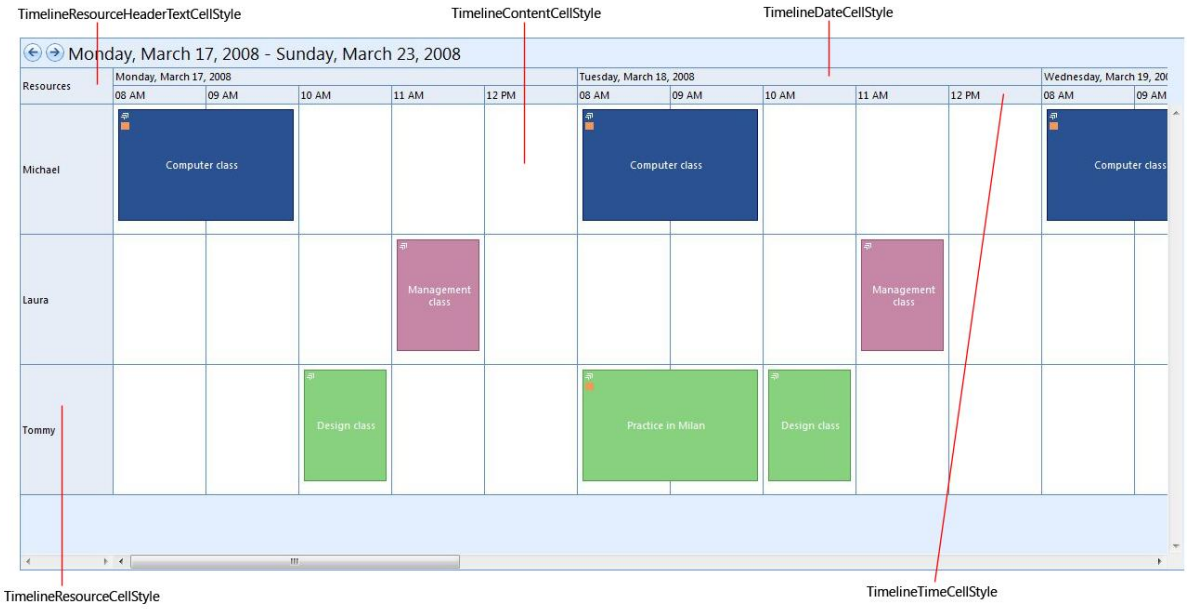
  Users often need to be able to see different view using the same data. In order to accommodate this requirement, WebScheduler implements “click to switch” feature for the Timeline. By simply clicking on the Timeline’s Date header, users can conveniently switch between *Day view* and *Time view* while the Timeline mode is active. This feature is enabled by default. You can set *EnableNavigateToOtherView* to *False* to disable this feature.

Styles

As a family member of WebUI Studio.NET®, WebScheduler is consistent in its rich and strong customizable nature. The following is the list of styles that you can customize in Timeline view.

- *TimelineContentCellStyle*.
- *TimelineDateCellStyle.*
- *TimelineResourceCellStyle*.
- *TimelineResourceHeaderTextCellStyle*.
- *TimelineTimeCellStyle.*

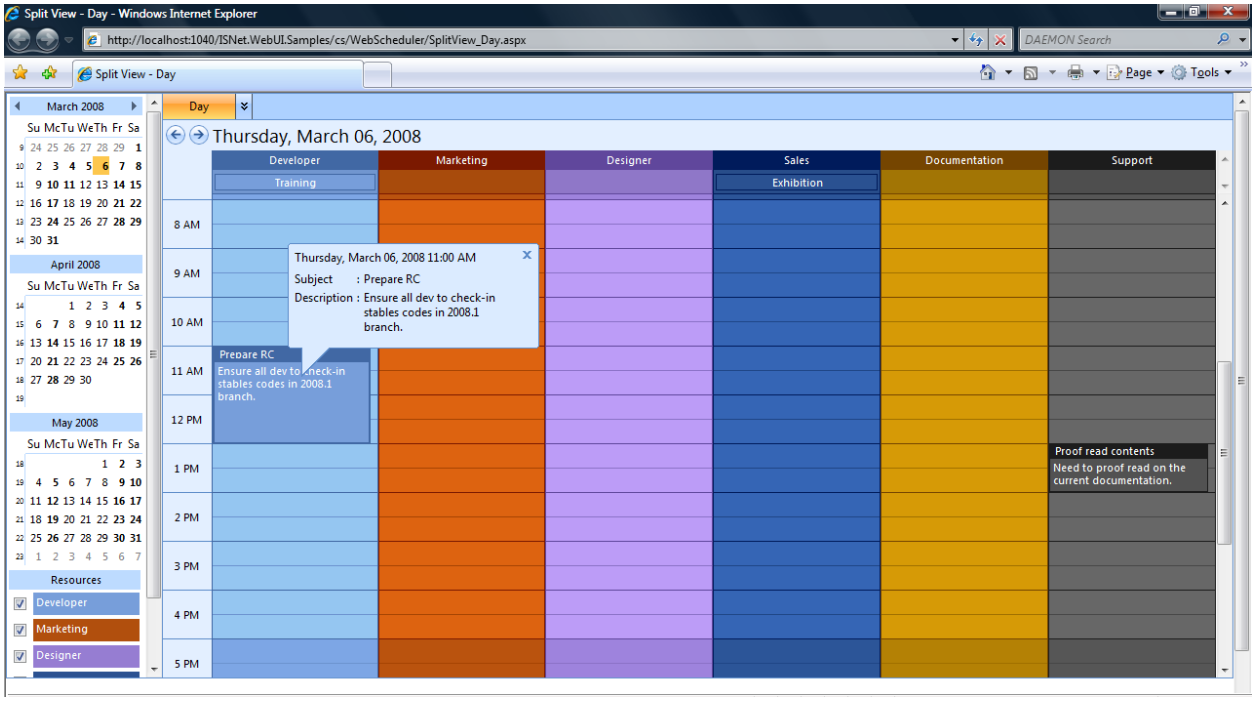
Please refer to the following illustration for a better picture on the correlation between UI elements and styles.



TrueSplitView™

Overview

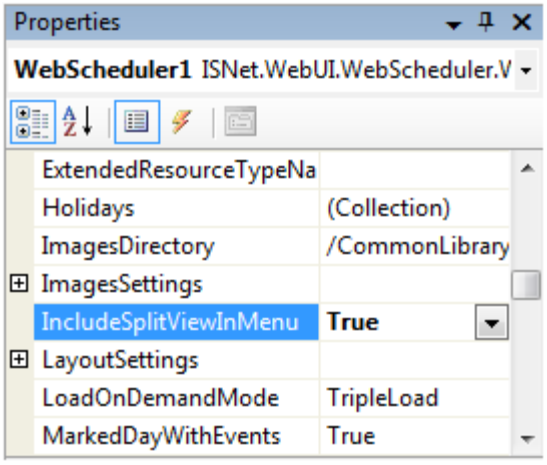
Back in the initial version of WebScheduler.NET, we introduced the innovative 6 views: Day, Week, Month, Quarter, Year, and Agenda view. End user can easily view the events based on the date range as the view name implies. And in the 2.0 version of WebScheduler.NET, we once again introduce you to the latest view, TrueSplitView™, to complement the existing 6 views.



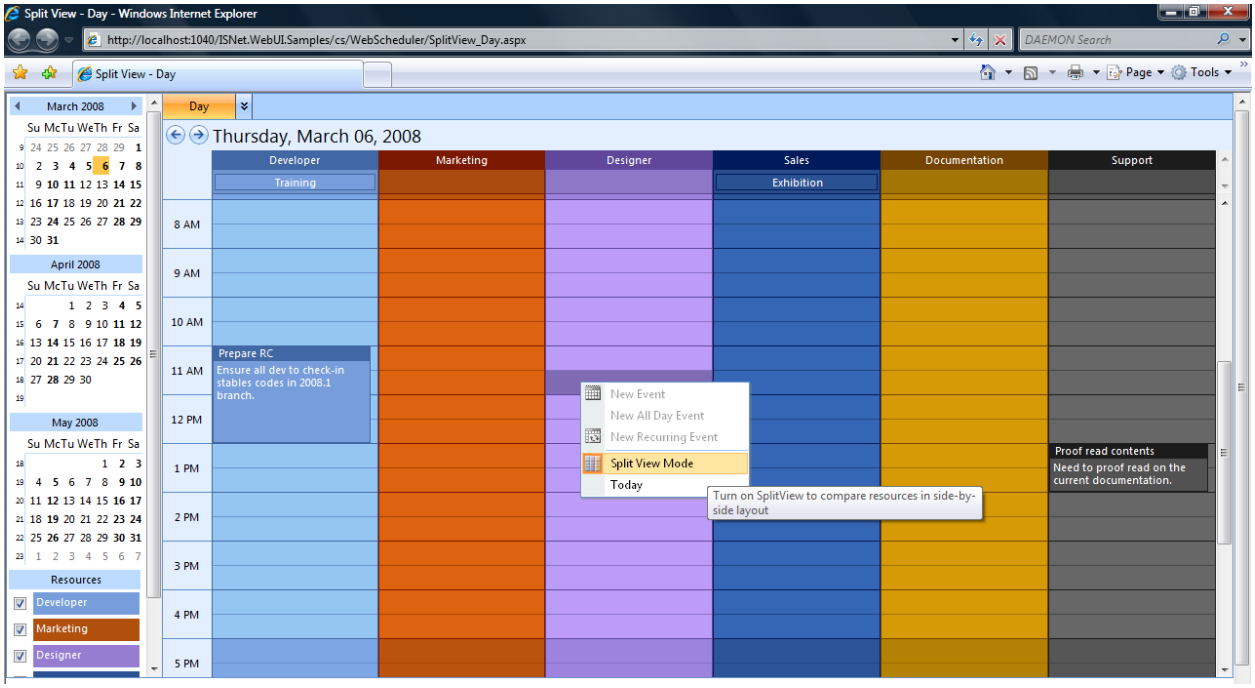
Concept

The previous behavior is that end user can only view the events grouped by date of the selected view. In TrueSplitView™, mode, the event information is grouped by resources. In the current 2.0 version, only Day View mode is supported in TrueSplitView™. As one of the key innovative features in WebScheduler.NET, we will constantly enhance this feature with more views and functionalities to support greater and more advanced enterprise scenarios.

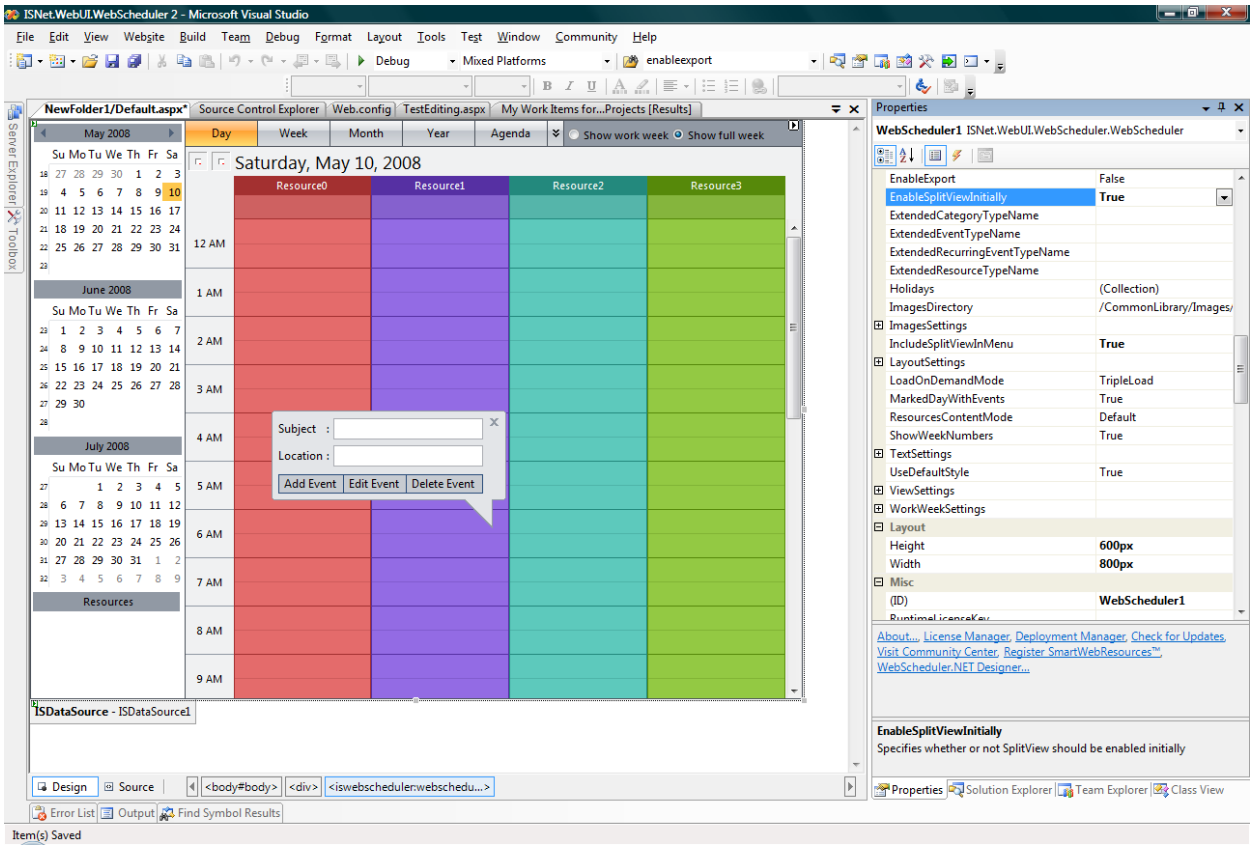
To add TrueSplitView™option in context menu and header menu, simply set the **IncludeSplitViewInMenu** property to **True**.



When this property is enabled, users can easily switch between normal view and split view using the context menu or header menu.



To enable split view on load, simply set **EnableSplitViewInitially** property to **True**. Note that when this property is enabled, TrueSplitView™, will be enabled initially.



User Interface and Styles

The content area is divided according to the total number of resources. By taking the resources color as the base color for resources column coloring, By taking resources color as the base color and diverse it as the coloring for the resources column and the element inside, WebScheduler.NET is not-only a superior performer, but also chic and beautiful in style. There are indeed many properties to control the Day Style, but when in TrueSplitView™ mode, you have full control over all styling, but not for the coloring.

Thanks to state-of-the-art rendering, when user uncheck a resource in the resources table, the unchecked resource column will be hidden and the layout will automatically adjust to the new total visible resources. And when user checks on a resource, the checked resources column will be visible and the layout will be resized automatically to hold the total visible resources.

Features

Powered by rock-solid drag-and-drop framework, you can easily drag and drop an event to the other resources columns easily without any breakings. Just drag an event, drop on the other resources column and the resource of the event will be changed automatically according to the resources of the drop column. To modify the other event information, simply use the elegant built-in editing form and modify the information as modifying an event in any time-based views. One thing that you should note is that all day event cannot be resized or dragged-and-dropped.

The innovative TrueSplitView™ works flawlessly with the existing advanced features, such as built-in Editing Form, RealTimeOperation™, navigation, and many more.

Extensibility

Overview

WebScheduler.NET provides built-in editing form and editing processes such as add, edit and delete are done automatically without additional codes or efforts. However, in the previous version of WebScheduler.NET, developers are unable to bind custom fields to the objects in WebScheduler.NET.

WebScheduler.NET™ 2.0 is designed specifically to address this issue by introducing an elegant approach and concept toward extensibility. With comprehensive extensibility support, WebScheduler.NET 2.0 takes Scheduling functionalities to the enterprise level where developers can add as much fields as they require while still be able to reuse existing assets such as built-in editing form.

This whitepaper discusses the Extensibility concept of WebScheduler 2.0 in details.

Extensibility Concept

WebScheduler is designed with full Object-Oriented (OO) approach. Some of main objects that heavily utilized in WebScheduler are *WebSchedulerEvent*, *WebSchedulerRecurringEvent*, *WebSchedulerResource*, and *WebSchedulerCategory*. For instance, *WebSchedulerEvent* represents an event that displayed in the WebScheduler view.

These primary objects are now extensible in WebScheduler 2.0 by using elegant *inheritance* (OOP) approach, allowing developers to add as many custom properties/fields as their business scenarios require. However, when using built-in editing form, Resource and Category custom fields cannot be reflected to the related controls, although the custom fields are populated in the object.

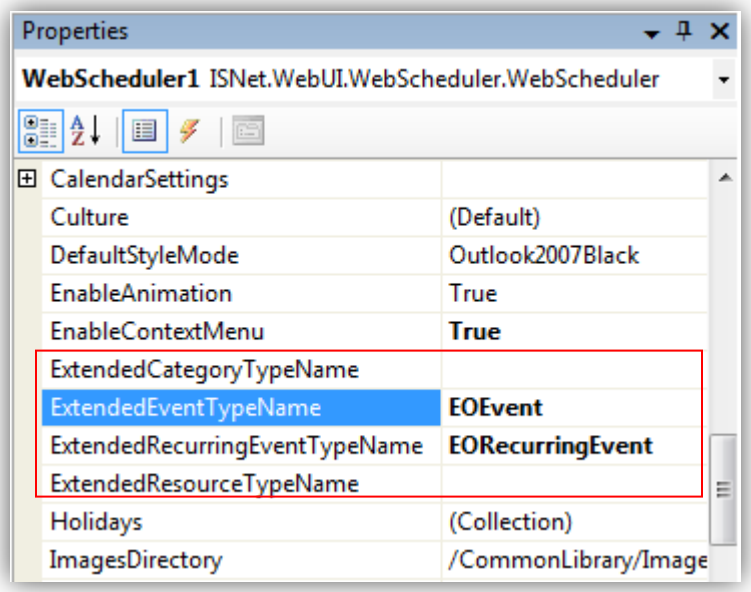
The extended class must be inherited from one of the above objects. Custom fields can then be added in the extended class. The custom properties should have *XmlSerializable* and *BinarySerializable* attributes. Note that the custom field's property has to be the same name as the custom field's name in the physical database.

The following C# sample codes demonstrate a new event class named *EOEvents*, which is derived from *WebSchedulerEvent* class.

```
public class EOEvents : WebSchedulerEvent
{
    private string _notes = "";
    private int _totalAttendees = -1;

    [XmlSerializable(), BinarySerializable()]
    public string Notes
    {
        get { return this._notes; }
        set { this._notes = value; }
    }

    [XmlSerializable(), BinarySerializable()]
    public int TotalAttendees
    {
```

```
        get { return this._totalAttendees; }
        set { this._totalAttendees = value; }
    }
}
```

The extended class must be specified as the extended type name. There are four types of extended type name, which are *ExendedEventTypeName*, *ExtendedRecurringEventTypeName*, *ExtendedResourceTypeName*, and *ExtendedCategoryTypeName*.



The data of the custom field need to be bound in the *DataBound* server side event of WebScheduler instance. The *e* argument consisted of two child arguments: Type and DataObject. The type argument indicates the type of object while the DataObject argument is the extended object (in object type). To access the extended object's properties, DataObject argument has to be casted to the related extended class.

The following C# codes show how developer can handle the databinding of custom properties in DataBound event.

```
protected void WebScheduler1_DataBound(object sender,
ISNet.WebUI.WebScheduler.WebSchedulerDataBoundDataArgs e)
    {
        switch (e.Type)
        {
            case WebSchedulerObjectType.Event:
                EOEvent evt = e.DataObject as EOEvent;

                evt.Notes = e.DataRow["Notes"].ToString();

                if (e.DataRow["TotalAttendees"] != System.DBNull.Value)
                    evt.TotalAttendees =
int.Parse(e.DataRow["TotalAttendees"].ToString());
```

```
                break;

            case WebSchedulerObjectType.RecurringEvent:
                EORecurringEvent recurringEvent = e.DataObject as
EORecurringEvent;

                recurringEvent.Notes = e.DataRow["Notes"].ToString();

                if (e.DataRow["TotalAttendees"] != System.DBNull.Value)
                    recurringEvent.TotalAttendees =
int.Parse(e.DataRow["TotalAttendees"].ToString());

                break;
        }

    }
```

In runtime, the extended class will be serialized to the client side. In order to be recognized, the extended class has to be initialized in client side.

```
<script type="text/javascript">
function EOEvent()
{
      // required codes
      WebSchedulerEvent.call(this);
      this._Type = "EOEvent";

      // custom properties start here
      this.Notes = "";
      this.TotalAttendees = 0;
}

function EORecurringEvent()
{
      // required codes
      WebSchedulerRecurringEvent.call(this);
      this._Type = "EORecurringEvent";

      // custom properties start here
      this.Notes = "";
      this.TotalAttendees = 0;
}
</script>
```

To indicate that the class is inherited from WebScheduler object, use the following code format: [WebScheduler base object].call(this);this._Type = "[extended class name]". For example: the above code describes that *EOEvent* class is inherited from *WebSchedulerEvent* class and *EORecurringEvent* class is inherited from *WebSchedulerRecurringEvent* class.

You can then modify the editing form and add controls for the extended fields. *OnEditingFormInitialize* can then be implemented to set the initial values for the extended fields.

Before saving the changes, the values of extended fields need to be saved to the OriginalObject and MasterObject (for RecurringEvent object). It's recommended to implement OnBeforeAdd and OnBeforeEdit client side events for this purpose. For example:

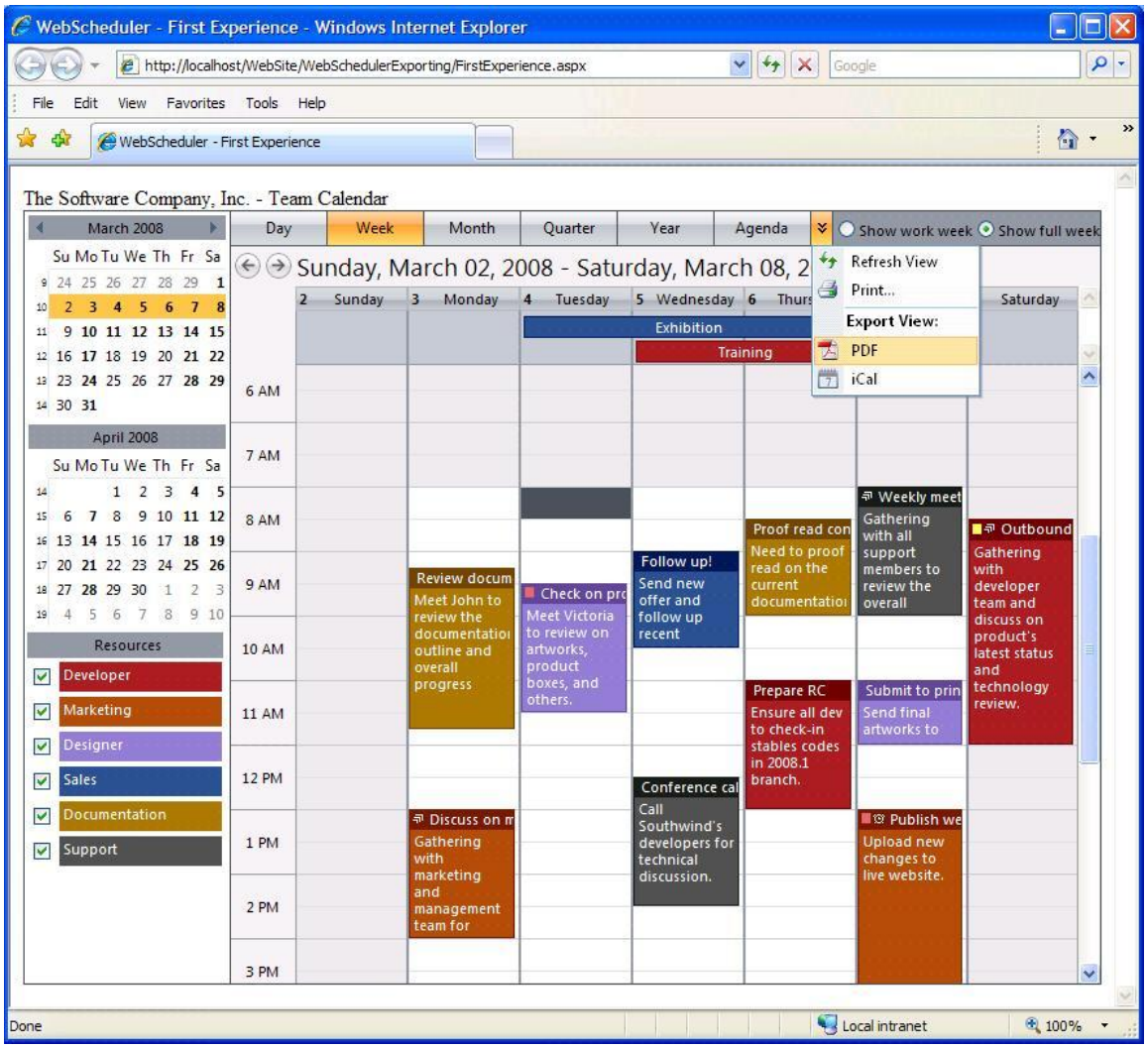
```
function WebScheduler1_OnBeforeAdd(controlId, evt)
{
	var s = ISGetObject(controlId);
	SetFields(s, evt.Parent);

	return true;
}

function WebScheduler1_OnBeforeEdit(controlId, evt)
{
	var s = ISGetObject(controlId);
	SetFields(s, evt.Parent);

	return true;
}
function SetFields(s, eventView)
{
	var editWindow = s.GetEditingFormWindow();

	var notes = "";
	var totalAttendees = 0;

	if (editWindow != null)
	{
		notes = editWindow.ISGetObject("wiNotes").GetValueData();
		totalAttendees =
editWindow.ISGetObject("wiTotalAttendees").GetValueData();
	}

	eventView.SetProperty("Notes", notes);
	eventView.SetProperty("TotalAttendees", totalAttendees);
}
```

So, now custom fields can be added to WebScheduler and the changes can be saved to physical database. This extensibility feature makes it possible to implement custom scheduling functionality with more advanced scenarios using WebScheduler.NET.

Exporting

Technology

WebScheduler exporting is an extensible feature of WebScheduler.NET 2.0. The exporting feature complements WebScheduler.NET as a very powerful web scheduling component from Intersoft Solutions. Built to provide high-end scheduler data presentation, WebScheduler exporting feature allows user to export events data which maximizes the user interface and experience for your Web applications. Like any other technologies invented by Intersoft Solutions, WebScheduler exporting has some essential features and concepts.



Picture 1. WebScheduler.NET 2.0 with exporting feature

Features

- **Exporting to PDF**

  In current web wide world, PDF has become an open standard for electronic document. It is commonly used for almost any application (documentation), on any computer system. Individuals, businesses, and government agencies around the world trust and rely on PDF to communicate their ideas and vision.

  With WebScheduler 2.0, scheduler data and appearance can be exported to PDF similar to the current active view. Undoubtedly, with exporting to PDF enable feature, WebScheduler provides richer user experience and gives more benefits for enterprise users. For instance, *sharing* and *offline viewing* of the events information are now possible with the exporting feature.

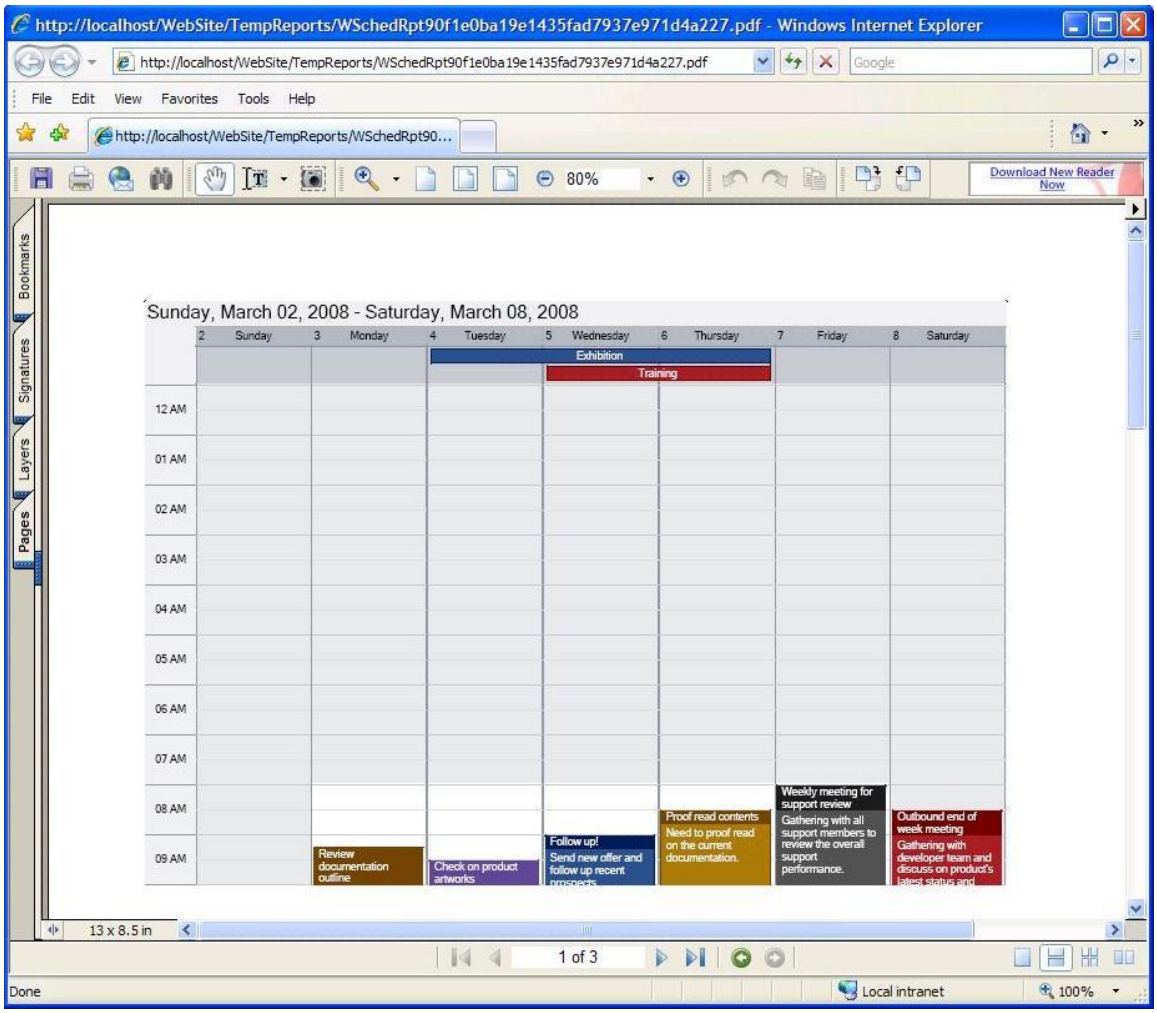
  Exporting WebScheduler data to PDF have many advantages:

- User can save and use the documented version of WebScheduler later.
- User is able to utilize all PDF functionality such as advanced document viewing, zooming, printing pages, selecting, and snapshot a part or whole of document.
- User can exchange PDF file easily for any applications and to any computer systems.

PDF exporting is not only useful to save electronic document but also for printing purposes. In previous version of WebScheduler, there is no way to save and print scheduler except from browser itself. The saved html document or mht document is available, but text showed data is truncated and the functionally is not run perfectly. The browser printed version has a very rough layout and inconsistent scheduler data, so it is unreadable and somewhat useless.

PDF exporting has slightly different GUI behavior than its web. PDF exporting flows any text so there is no truncated data. It makes sure that all data in header and content of the scheduler is shown, so user is able to read, print, mark, scratched, remark, clip, adhere, etc.

Text flowing basically is an adjusting layout so that any text will not be truncated. It is done by wrapping the text, adjust the dimension of text and parent text container according to the dimension of the printing paper. So the exported PDF file of WebScheduler will have useful, meaningful, and complete scheduler data as well as its layout.



Picture 2. WebScheduler PDF export file, with flow layout in event header

WebScheduler PDF exporting supports WebScheduler layout views, such as Day view, Week view, Month view, Quarter view, Year view and Agenda view. Timeline view and Split view are not supported yet, but some enhancement will be made and it will be supported later in the future.

Text flowing feature is usable when showing events in some scheduler views, as follows:

- Day view - flows AllDayEvent header text, flows Event header text and flows Event description text
- Week view - flows AllDayEvent header text, flows Event header text and flows Event description text
- Month view - flows AllDayEvent and Event header text

WebScheduler PDF exporting also supports any WebScheduler styles in any layout views, including SchedulerFrameStyle, SchedulerHeaderStyle, SchedulerTodayHourStyle, WorkHourStyle, CalendarHeaderStyle, etc.

Styles support makes exporting more intuitive, attractive, and easier to view. When user changed styles in WebScheduler, it will also reflect in WebScheduler exporting, hence user will have the same representation between the web and the PDF version of the WebScheduler.

- **Exporting to iCal**

  iCalendar is a standard (RFC 2445) for calendar data exchange. It is sometimes referred to as "iCal", which is also named after Apple, Inc. Calendar program (see iCal) provides one of the implementations of the standard.

  iCalendar allows users to send meeting requests and tasks to other users through email. Recipients of the iCalendar email (with supported software) are able to respond to the sender easily or counter propose another meeting date/time.

  iCalendar is implemented / supported by a large number of products. Its data is typically exchanged using traditional email, but the standard is designed to be independent to the transport protocol. For instance, iCalendar can be shared and edited by using a WebDav server, or SyncML.

  The filename extension of "ics" is to be used to specify a file containing calendaring and scheduling information consistent with this MIME content type. And ics file consists of one or more scheduler events.

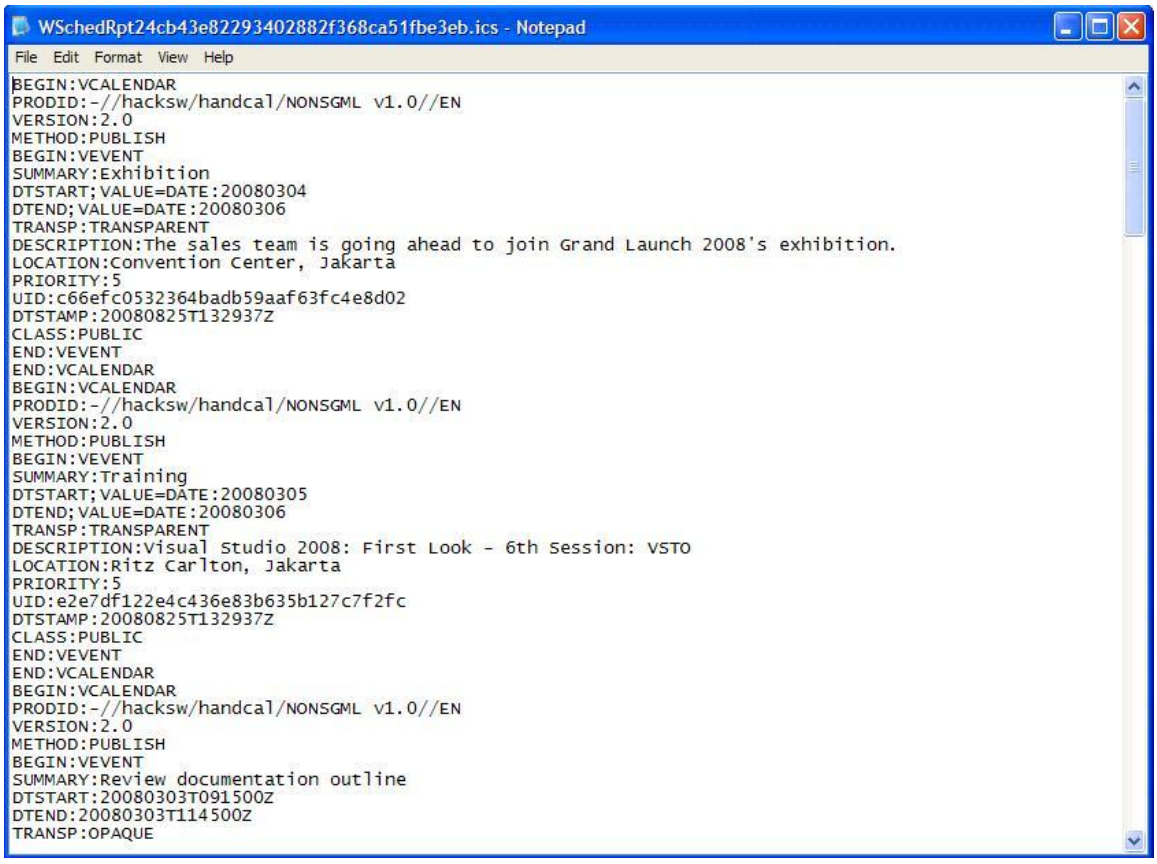
  By exporting to ics format, WebScheduler data is able to be used, edited, or customized in any of your favorite calendar mail programs that support ics scheduler data format. With exporting to ics capability, WebScheduler data is now reusable, customizable and extensible for other programs. Just open an iCal file with your favorite scheduler program and the data will be presented in an elegant way.

  WebScheduler is built to fulfill current scheduler data standard and enhance it with various custom scheduling data. Therefore, exporting to ics is seamlessly easy from WebScheduler.

  Scheduler data format, use by WebScheduler for exporting, are:

  - A "VEVENT" component – provides a grouping of component properties that describe an event to represent a scheduled amount of time on a calendar. This component represents WebScheduler event and WebScheduler all day event.
  - A “SUMMARY” component property – defines a short summary or subject for the calendar component. This property represents WebScheduler event subject.
  - A “DTSTART” component property – specifies when the calendar component begins. This property represents WebScheduler event start time.
  - A “DTEND” component property – specifies when the calendar component ends. This property represents WebScheduler event end time.

- A “DESCRIPTION” component property – provides a more complete description of the calendar component, than that provided by the "SUMMARY" property. This property represents WebScheduler event description.
- A “LOCATION” component property – defines the intended venue for the activity defined by a calendar component. This property represents WebScheduler event location.
- A “CATEGORIES” component property – defines the categories for a calendar component. This property represents WebScheduler event CategoryId.
- A “RRULE” component property – defines a rule or repeating pattern for recurring events, to-dos, or time zone definitions. This property represents WebScheduler event recurrence info. Some sub-properties of “RRULE” are also supported, like: “FREQ” property, “INTERNAL” property, “BYDAY” property, and “BYMONTHDAY” property.
- A “PRIORITY” component property – defines the relative priority for a calendar component. This property represents WebScheduler event mode (WebSchedulerImportanceMode).
- A “VALARM” component property – provides a grouping of component properties that define an alarm. This property represents WebScheduler reminder time span event.



Picture 3. WebScheduler ICal export file, opened by Notepad

- **Optimizes reporting process**

Exporting process can be cached, so whenever a report is created, it uses cached document in file instead of creating a new document again in memory. When you need to export several report data continuously, this will optimizes reporting process as it will run faster than usual mechanism.

- **Output path in server**

  Exporting always creates a physical exported file. In pdf or ics format, these files locate in server and can be downloaded later by user. By default, these exported files are placed inside a sub directory named "TempReports", below root website application path.

  The report path can be specified using **WebSchedulerReportInfo.IISReportPath** property. The default value is "*~/TempReports*". WebScheduler will translate the relative path specified in IISReportPath property to a physical path to store the report files.

- **Automatic caching and cleaning output file**

  When WebScheduler exports data, it will create a temporary physical file in the TempReports folder.

  Temporary files will gradually become large, as more files are exported. Logically, it is always good to use the data only when it is needed and dispose when it is no longer useful. Hence, an automated delete process is created to keep the folder free of space.

  By default, the temporary physical files will be disposed after 20 minutes. However, this value can be changed in **ReportInfo.DeleteOutputTime.**

- **DPI support**

  In today's computer world, presentation is one of the most important things. There are so many computer systems with their own presentation mechanism and hardware specification, for instance, Windows, Apple, Linux, UNIX, and etc.

  Standardization is needed to provide a consistent way of presenting a document in various computer systems. DPI is one mechanism to achieve this scenario. Dots per inch (DPI) are a measure of spatial printing or video dot density, in particular the number of individual dots that can be placed within the span of one linear inch (2.54 cm.) The value of DPI tends to correlate with image resolution, but it is related only indirectly.

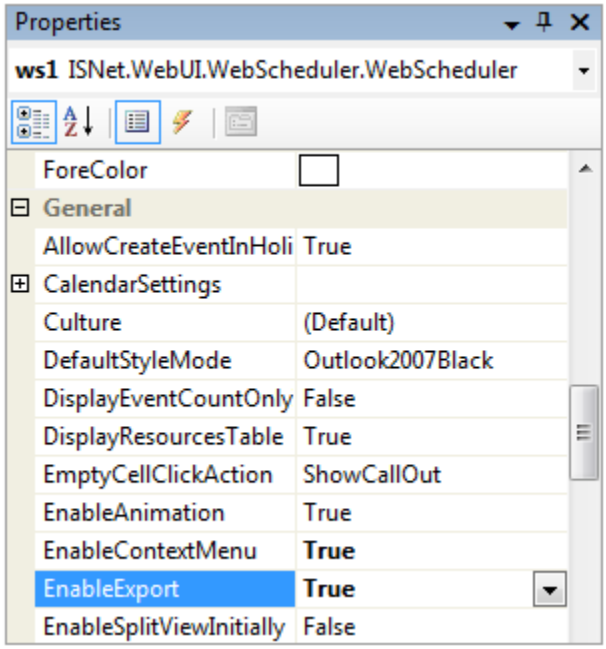
  Printers with higher DPI produce clearer and more detailed output. A printer does not necessarily have a single DPI measurement; it is dependent on print mode, which is usually influenced by driver settings. The range of DPI supported by a printer is most dependent on the print head technology it uses. A dot matrix printer, for example, applies ink via tiny rods striking on ink ribbon, and has a relatively low resolution, typically in the range of 60 to 90 DPI. An inkjet printer sprays ink through tiny nozzles, and is typically capable of 300-600 DPI. A laser printer

applies toner through a controlled electrostatic charge, and may be in the range of 600 to 1800 DPI.

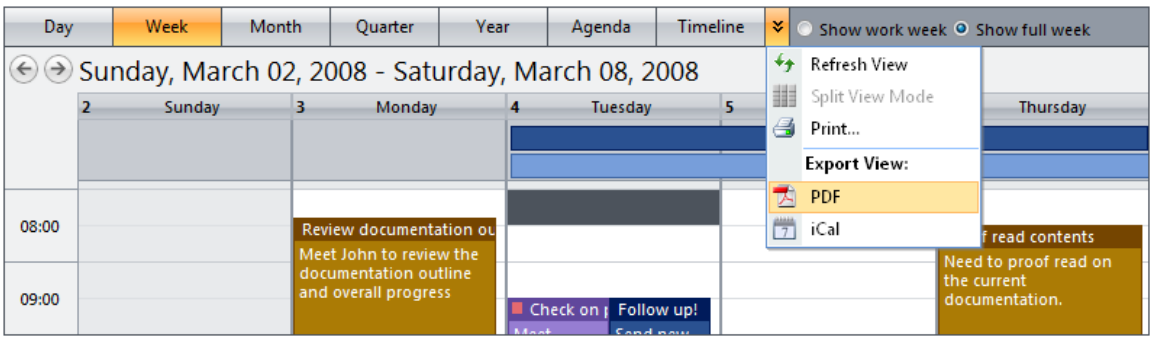
With WebScheduler exporting, you can set your own DPI setting according to your customer's printer setting. The default value of WebScheduler report is 96 dots per inch.

Enable WebScheduler Exporting Feature

To enable exporting, simply set **EnableExport** property to **True**.



When this property is enabled, end user can export the active view to PDF or iCal format using the header menu.



You can also export the scheduler view programmatically using **ExportScheduler** method.

In Server Side

Default export settings will be used when you use **ExportScheduler** method. The default settings will export the active view in PDF format and Landscape orientation. You can also specify **WebSchedulerReportInfo** object to specify custom report info that will be used in exporting.

The following code will export the active view in iCal format and Portrait page orientation, using the file name "WebSchedulerToiCal.ics". The **ExportScheduler** method will return the physical path of the exported file.

```
protected void Button1_Click(object sender, EventArgs e)
{
    WebSchedulerReportInfo info = new WebSchedulerReportInfo();
    info.FileName = "WebSchedulerToiCal.ics";
    info.PageOrientation =
  DataDynamics.ActiveReports.Document.PageOrientation.Portrait;
    info.Type = WebSchedulerReportType.ICal;
    string file = ws1.ExportScheduler(info);
}
```

In Client Side

ExportScheduler method in client side requires two parameters: export type and page orientation. The export type options are "PDF" and "iCal", while the page orientation options are "Landscape" and "Portrait". The following code will export the active view to PDF format and Landscape page orientation. The exported file will be opened in a new window.

```
function button1_click()
{
    var s = ISGetObject("WebScheduler1");
    s.ExportScheduler("PDF", "Landscape");
}
```

Deployment

Exporting assembly is required to be copied to your project's bin folder if the web application utilizes exporting feature in WebScheduler. The exporting assembly can be found in C:\Program Files\Intersoft Solutions\WebScheduler.NET 2.0\Bin. The exporting assembly is: **ISNet.ActiveReports.Exporting.dll.**

Enhancements

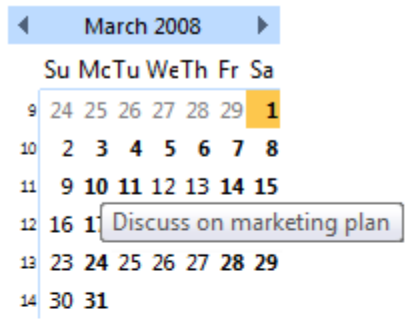
We hereby present to you the enhancements of WebScheduler.NET based on the top feature requests submitted by you. There are several properties added to cover this feature requests which will be described in the following section.

Hide Calendar ToolTip

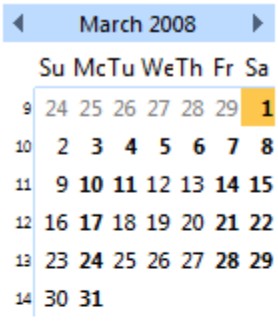
By default, tooltip is displayed when you hover a date or range of dates (when you switch to Week View) in the Calendar. The tooltip will show the event's subject if you hover on a date which holds only one

event and it will display the total numbers of event if you hover through a date which holds more than one event.

In this latest WebScheduler.NET version, you can actually hide the calendar tooltip simply by setting the **DisplayCalendarCellToolTip** property to **False**.



DisplayCalendarCellToolTip = true

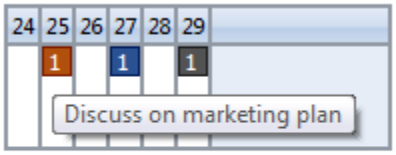


DisplayCalendarCellToolTip = false

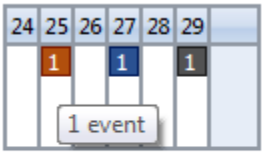
Display Event Count Only In ToolTip

In Year and Quarter view, the event is displayed based on its resource color and the tooltip displays the total number of events for the resource in the specified date. If there is only one event in the resource, the subject of the event will be displayed in the tooltip. The same thing applies to the tooltip of calendar's date.

Now, you can actually set the tooltip to display the total number of events even though there is only one event on that date. This can be achieved by setting the **DisplayEventCountOnlyInToolTip** property to **True**.



DisplayEventCountOnlyInToolTip = false



DisplayEventCountOnlyInToolTip = true

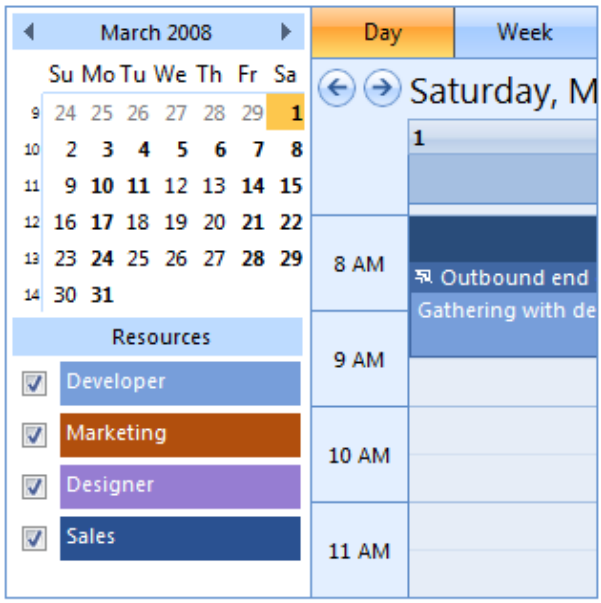
Show Editing Form When Empty Cell Is Clicked

By default when editing is enabled, WebCallOut is displayed if an empty cell is clicked. User can use WebCallOut as direct shortcut to add a new event. However, in certain scenarios where more detailed information is required, user must use the Editing Form to add a new event.

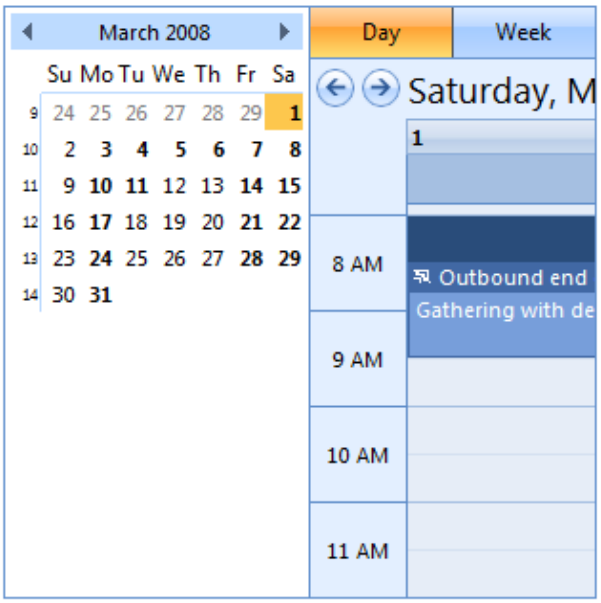
Simply set **EmptyCellClickAction** property to **ShowEditingForm** to skip the WebCallOut and display the Editing Form instead.

Hide Resources Area

Resource lists are usually displayed below Calendar. To hide the resources area, simply set **DisplayResourcesTable** to **False**. Note that you can still navigate between views and dates without any issues.



DisplayResourcesTable = true

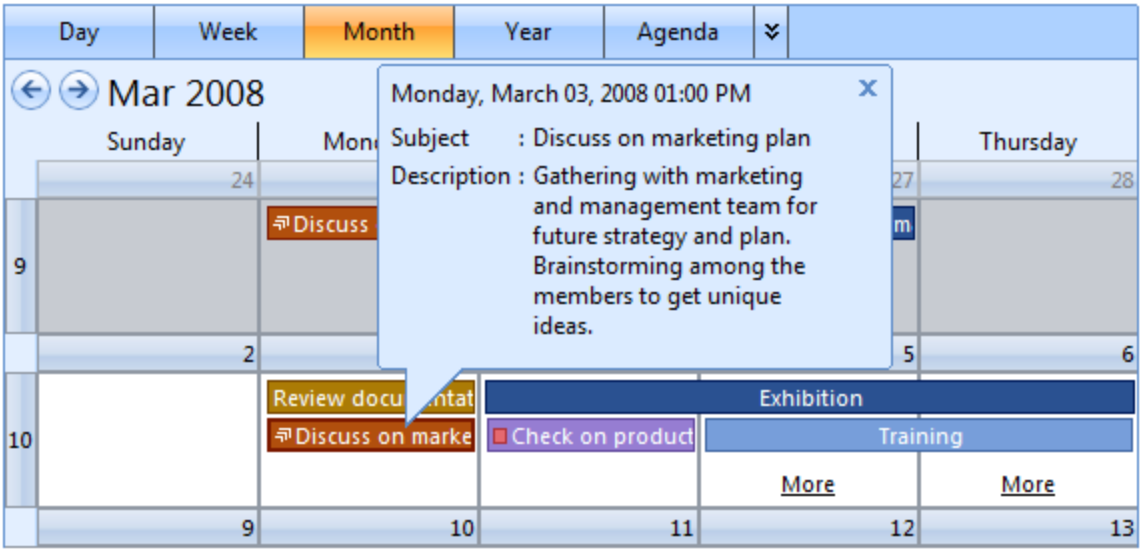


DisplayResourcesTable = false

Show Event Subject Only In Month View

In order to differ between all-day event and time-based event in Month view, the start time of the event is displayed in time-based event. In certain scenarios, this default behavior is not suitable; especially under a scenario which only requires the subject and the time must be invisible.

To achieve the above scenarios, simply set the **ViewSettings-MonthView-TimeBasedEventDisplay** to **ShowSubject**, to display only the subject in time-based event.



Customize Week Number

One of the top requested features is the ability to customize the week number elements.

The customizable Week Number elements are the one in Month view and in Calendar. Two client side events are added for this purpose, **OnAfterRenderView** and **OnAfterRenderCalendar** events.

To customize the Week Number element in Month view, **GetSchedulerWeekNumber([row index])** function can be used in OnAfterRenderView client side event. To customize the Week Number element in Calendar, **GetCalendarWeekNumber([calendar index], [row index])** function can be used in OnAfterRenderCalendar client side event.

Please browse to Customize Week Number sample for further information.