TTIWWEBPLANNER – TTIWDBWEBPLANNER
DEVELOPERS GUIDE

January 2019

Web: http://www.tmssoftware.com
Email : info@tmssoftware.com

Table of contents

TTIWWebPlanner / TTIWDBWebPlanner availability

TTIWWebPlanner and TTIWDBWebPlanner are available for IntraWeb. Versions are available as well as VCL, FMX, .NET, FNC components.

IntraWeb versions :
TTIWWebPlanner is available for
Delphi 2009,2010,XE,XE2,XE3,XE4,XE5,XE6,XE7,XE8 ,10 Seattle,10.1 Berlin, 10.2 Tokyo
C++Builder 2009,2010, XE,XE2,XE3,XE4,XE5,XE6,XE7,XE8, 10 Seattle,10.1 Berlin, 10.2 Tokyo, 10.3 Rio
TTIWDBWebPlanner is available for
Delphi 2009,2010,XE,XE2,XE3,XE4,XE5,XE6,XE7,XE8, 10 Seattle,10.1 Berlin, 10.2 Tokyo
C++Builder 2009,2010,XE,XE2,XE3,XE4,XE5,XE6,XE7,XE8, 10 Seattle,10.1 Berlin, 10.2 Tokyo, 10.3 Rio

TTIWWebPlanner, TTIWDBWebPlanner have been designed for and tested with Windows 2000, Windows XP, Windows Vista, Windows 7, Windows 8, Windows 10 and Internet Explorer 6+, Firefox 1+, Opera 7+, Chrome 5+

TTIWWebPlanner / TTIWDBWebPlanner use

The TMS TTIWWebPlanner and TTIWDBWebPlanner components are designed to be used in the most broad types of planning and scheduling type of applications. This can range from the typical single person PIM application to schedulers of activities for a group of persons, time planning for resources such as hotel rooms, car rental, university courses and so much more. As such, the TTIWWebPlanner and TTIWDBWebPlanner are very highly configurable components to suite all these various types of applications. The underlying framework of TTIWWebPlanner therefore has an open interface towards the coupling to time or resources. Standard modes include day time view, week view, month view, day period view, half-day period view, multi-month view while at the same time custom modes allow to view any type of timescale. Multi day and multi resource views can be combined as well.

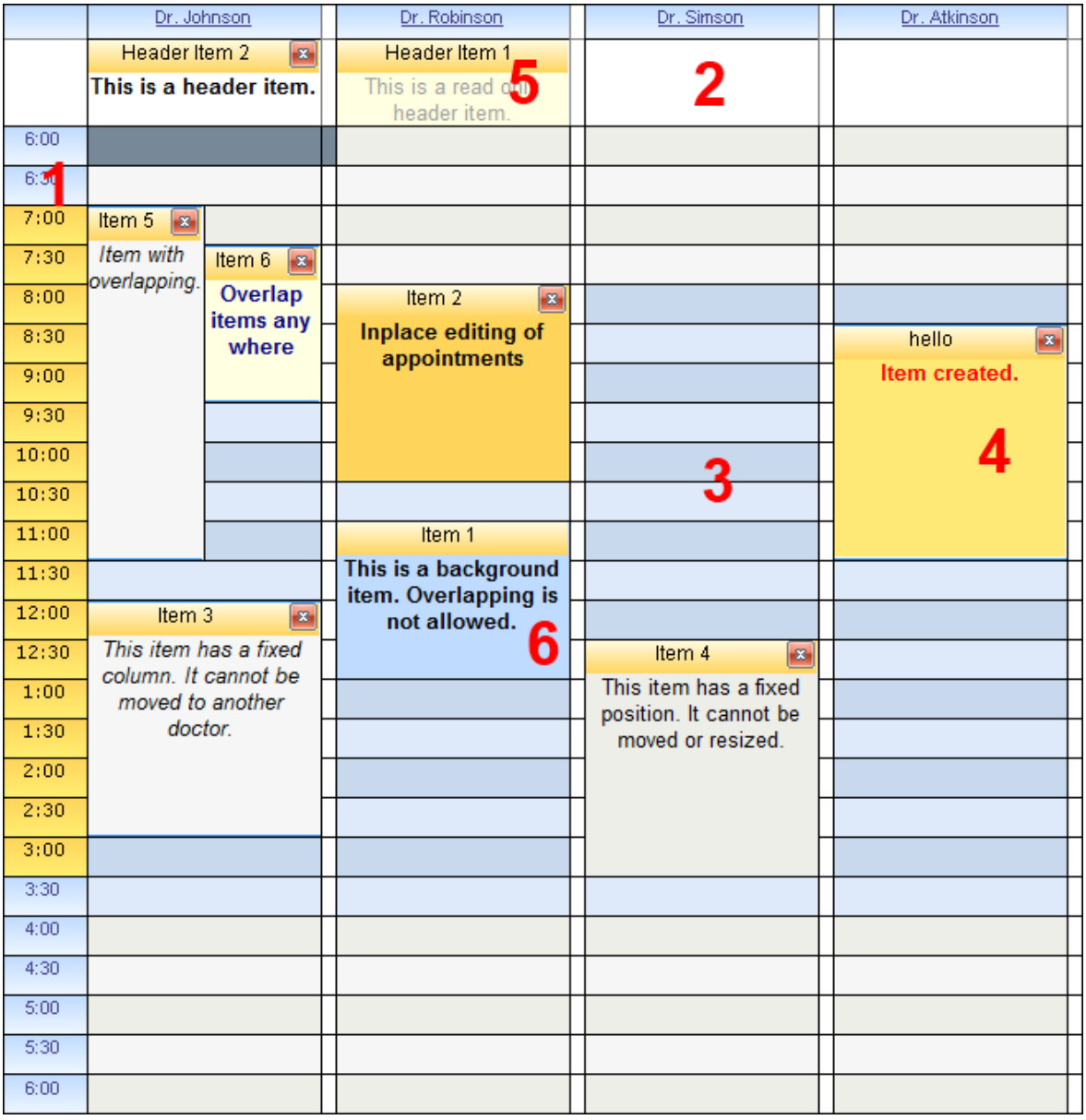
TTIWWebPlanner / TTIWDBWebPlanner organisation

Non visual organisation of TTIWWebPlanner / TTIWDBWebPlanner

The TTIWWebPlanner holds a collection of TPlannerItem objects. A TPlannerItem fully specifies how an event or appointment is displayed in the TTIWWebPlanner grid. TPlannerItem objects can be added or inserted in the TPlannerItems collection and the TTIWWebPlanner takes care of the visualisation of the TPlannerItem objects in the grid. At the same time, it takes care that TPlannerItem objects are modified when edited, moved or resized at run-time. It is thus key that only items that should be displayed in the TTIWWebPlanner are added.
In the not data-aware version TTIWWebPlanner, it is the responsibility of the programmer to save and load these TPlannerItem objects to file or a database for persistence. With TTIWDBWebPlanner, a TIWDBItemSource descendent component takes care of the streaming of TPlannerItem objects to and from the TTIWDBWebPlanner from a dataset and updating for appropriate records in the database when TPlannerItem objects are changed at run-time. Currently, TIWDBItemSource based interfaces exist for day, month, period and multi-month mode.

Visual organisation of TTIWWebPlanner / TTIWDBWebPlanner

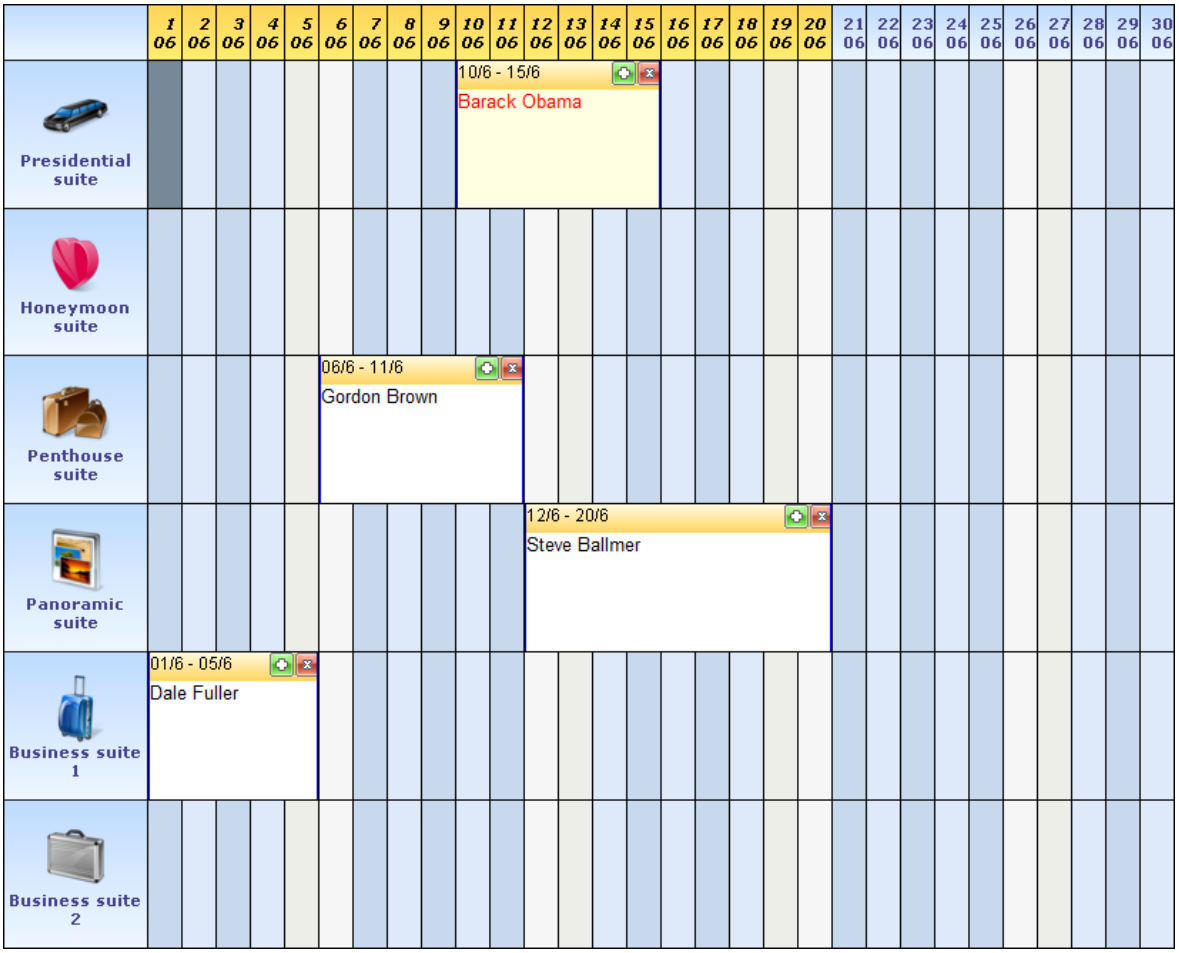
Visually the TTIWWebPlanner consists of various elements shown here:



1 : TTIWWebPlanner sidebar
Settings are controlled through the Planner.Sidebar property. The Sidebar can be visible or not. It can be at the left side, right side, left and right side, or top side of the grid.
2 : TTIWWebPlanner header
Settings are controlled through the Planner.Header property. It can be selected whether the header is visible or not.
3 : TTIWWebPlanner grid
Various settings are controlled through the Planner properties as well as Planner.Display property.
4 : TPlannerItem
Normal text TPlannerItem with caption with caption text.
5: TPlannerItem

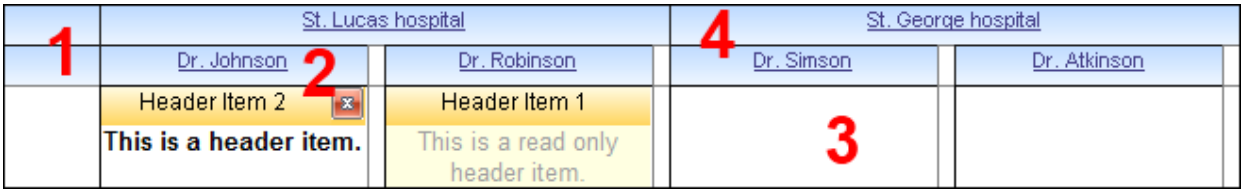
In header displayed TPlannerItem.
6 : TPlannerItem
Background TPlannerItem.

TTIWWebPlanner supports a vertical view (as shown above) as well as a horizontal view. In the horizontal view, all elements of the TTIWWebPlanner simply rotated. This means that the Sidebar here at left is displayed on top and the header and navigator buttons are displayed at the left side in horizontal mode. The position coordinates of TPlannerItem objects along vertical and horizontal axis in vertical mode become the coordinates along horizontal and vertical axis respectively in horizontal mode.
The TTIWWebPlanner component can be easily switched from vertical view to horizontal view by setting the TTIWWebPlanner.Sidebar.Position property to spTop.



Settings of TTIWWebPlanner / TTIWDBWebPlanner visual elements

TTIWWebPlanner Header



The header consists of various sections which automatically adapt to the size of the columns (vertical mode) or rows (horizontal mode) in the planner grid.

1 : First header section covering the sidebar space

2 : Normal header section text as defined through the stringlist Planner.Header.Captions

3 : Placeholder for items in the header

4 : Grouped columns with text set by the stringlist Planner.Header.GroupCaptions

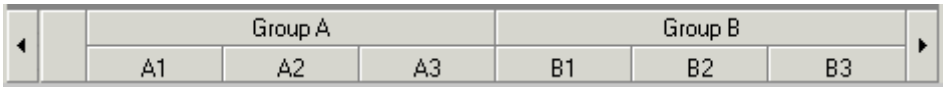
The Planner.Header has following properties :

Alignment : sets the text alignment of normal header captions
Background: sets the background color of the header
BackgroundTo: when different from clNone, shows a gradient from color Background to BackgroundTo.
CaptionHeight: sets the height of the header caption
Captions: stringlist holding header captions (note that the caption with Index zero (0) is usually occupied by the sidebar)
Font: sets the font of the normal header caption text
GradientDirection: selects the gradient direction to vertical or horizontal
GroupCaptions: when the planner has grouping, captions for the grouped sections are set through this stringlist
HeaderItemWidth: sets the width of the header items in horizontal view
LineColor: sets the line color of the divider line between header sections
Visible : sets the visibility of the header

Note that if no group captions are used (PositionGroups = 0), the height of the header is CaptionHeight. If group captions are shown, the height of the header will be CaptionHeight*2

Identical settings apply when the header is displayed on the left side of the TTIWWebPlanner when horizontal mode is chosen. Note that the meaning of a height property should in this case be interpreted as a width setting.

Programmatically setting captions :



The planner header above shows normal header captions and group captions (settings of grouping is discussed later in the documentation)

Example:

Programmatically, this can be set by filling the Planner.Header.Captions and Planner.Header.GroupCaptions stringlists in following way :

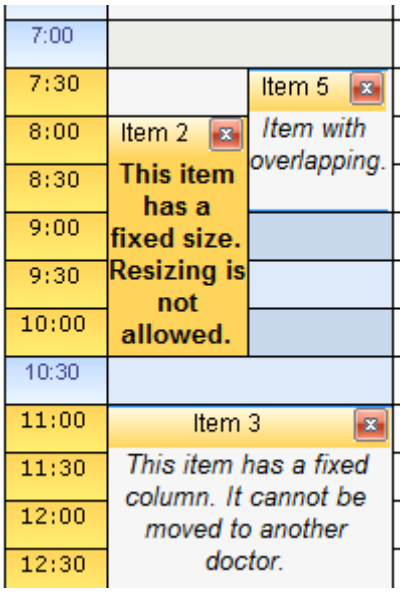
```
with Planner1.Header do
begin
  Captions.Clear;
  GroupCaptions.Clear;
  Captions.Add(''); // take first sidebar header section into account
  Captions.Add('A1');
  Captions.Add('A2');
  Captions.Add('A3');
  GroupCaptions.Add('Group A');
  Captions.Add('B1');
  Captions.Add('B2');
  Captions.Add('B3');
  GroupCaptions.Add('Group B');
end;
```

Note:
At design time it is easy to enter multiline header captions by using
 as line separator. Setting at design time in the Captions stringlist editor : 'This is line 1
 and here line2' will result in a caption with:

This is line1
and here line 2

TTIWWebPlanner Sidebar

The sidebar is the time indicating visual element in the TTIWWebPlanner component. It can sit in various positions in the TTIWWebPlanner. The height of the sidebar sections automatically adapts to the row height in the planner grid.



This is a left positioned sidebar showing the occupied time zones in a different color

TTIWWebPlanner.Sidebar has following properties to control display of the sidebar :

Alignment : sets the alignment of the sidebar text
Background: sets the background color of the sidebar
BackgroundTo: when different from clNone, shows a gradient from color Background to BackgroundTo.
DateTimeFormat : date / time format string that can be used to set the format of the text in the SideBar. Refer to the Delphi / C++Builder FormatDateTime format specifier for options.
Font : sets the font for the sidebar
GradientDirection: selects the gradient direction to vertical or horizontal
OccupiedColor : sets the color to show occupied time zones in the planner
OccupiedColorTo : when different from clNone, a gradient is shown for occupied time zones from OccupiedColor to OccupiedColorTo
OccupiedFont : sets the font for the occupied time zones
Position : sets the sidebar position relative to the grid, it can be set to the left, the right, the top, or both the left and the right of the grid. Setting the position to the top of the grid, rotates the whole grid 90 degrees, exchanging columns/rows and width/height settings.to left from the TPlanner, on top or on both left and right side.
ShowDayName : Adds the name of day in month and period planner mode to day.
ShowHint: displays hint for possible partially displayed text in the sidebar
ShowOccupied : when true, occupied time zones are displayed in the sidebar in the Occupied / OccupiedFontColor
Visible : sets the visibility of the sidebar
Width : sets the width of the sidebar (or height if the Position is at the top of the grid)

What is displayed in the sidebar depends on the mode of the TTIWWebPlanner (see TTIWWebPlanner.Mode.PlannerType). This can be the hours of the day, when mode is set to plDay, or the dates in plMonth, plPeriod, plMultiMonth, and plWeek modes.

If the TTIWWebPlanner is in day mode, then the time of the day is displayed in the sidebar. The range and unit of the timescale in the sidebar is determined by the TTIWWebPlanner.Display property (see under TTIWWebPlanner display) It consists of 3 parts :

1 : hour text
2 : minutes text
3 : optional AM/PM string

It is possible to override what is displayed in the sidebar by using the TTIWWebPlanner.OnGetSideBarLines event, which queries these 3 parts for each section of the sidebar.

If the TTIWWebPlanner.Mode.PlannerType is plWeek, then the sidebar shows day names. Those can be configured in the TTIWWebPlanner.DayNames property.

If the TTIWWebPlanner.Mode.PlannerType is plMultiMonth, then the sidebar shows day numbers within the month.

If the TTIWWebPlanner.Mode.PlannerType is plDayPeriod, plMonth, or plPeriod, plMultiMonth, plWeek, then the sidebar shows the date (range of dates is also set through the TTIWWebPlanner.Mode property). The formatting of the date displayed is according to the DateTimeFormat property, and Day names are taken from the SysUtils. ShortDayNames variable.

Example:

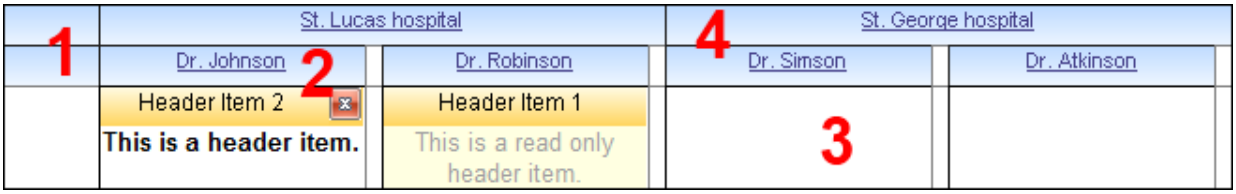
When setting TIWWebPlanner.Sidebar.DateTimeFormat is equal to : 'ddd m mmm / yyyy', the date 7/7/2002 is displayed as 'Fri 7 Jul / 2002'

TTIWWebPlanner position display control

The TTIWebPlanner introduces the concept of Positions. This can be considered as an orientation independent name for columns in vertical mode and rows in horizontal mode. So Position zero corresponds to the first column, Position 1 to the second column and so on.
In day mode, the Positions are typically used to represent TPlannerItem objects for several days or resources. In week of month mode, the Positions typically represent multiple resources. Finally, in multi-month mode the number of positions controls the number of months displayed simultaneously.

At design-time these visualisation settings can be controlled with :

PositionGroup : sets the number of positions that are displayed in a group as shown in the picture below. This view is typically used when multiple resources are displayed for multiple days or multiple days are displayed for multiple groups. Note that when PositionGroup is different from zero, the TTIWWebPlanner header starts displaying GroupCaptions.



In this sample, the PositionGroup is set to 2. That is, each group holds 2 positions, meaning also that only for every 2 position captions in the header a group caption needs to be set. The code for this example is a few pages back.

PositionProps : This is a collection holding optional display properties such as colors for each position. The use of PositionProps is discussed later.

Positions : this sets the total number of positions in the planner. In the example above, the number of positions was is 4.

TTIWWebPlanner modes

The main TTIWWebPlanner mode settings that control the relationship between displayed items and time are centralized under TTIWWebPlanner.Mode. The key property is TTIWWebPlanner.Mode.PlannerType which can currently be :

plDay : day mode
- the time axis represents hours of a day
- active & non-active cells are set through the TTIWWebPlanner.Display property
- positions can represent multiple days or multiple resources, coupling is not predefined

plWeek : week mode
- the time axis represents days for one or more weeks
- active & non-active cells are set per day. By default Saturday & Sunday are considered as non-active days but this can be overridden with the TTIWWebPlanner.InActiveDays property
- positions represent multiple resources
- first day displayed is set by TTIWWebPlanner.Mode.WeekStart. 0 is Saturday … 6 is Sunday.
- first date is first week of month set by TTIWWebPlanner.Mode.Month / TTIWWebPlanner.Mode.Year

plMonth : month mode
- the time axis represents all days of a selected month
- active & non-active cells are set per day. By default Saturday & Sunday are considered as non-active days but this can be overridden with the TTIWWebPlanner.InActiveDays property
- positions represent multiple resources
- month displayed is set by TTIWWebPlanner.Mode.Month

plDayPeriod : configurable period mode
- the time axis represents days of a selected period
- active & non-active cells are set per day. By default Saturday & Sunday are considered as non-active days but this can be overridden with the TTIWWebPlanner.InActiveDays property
- period displayed is between PeriodStartDay / PeriodStartMonth / PeriodStartYear and PeriodEndDay / PeriodEndMonth / PeriodEndYear

plHalfDayPeriod : configurable half day period mode
- the time axis represents half days of a selected period
- active & non-active cells are set per day. By default Saturday & Sunday are considered as non-active days but this can be overridden with the TTIWWebPlanner.InActiveDays property
- period displayed is between PeriodStartDay / PeriodStartMonth / PeriodStartYear and PeriodEndDay / PeriodEndMonth / PeriodEndYear

plMultiMonth : multi month mode
- time axis represents days of the month, positions represent multiple months
- active & non-active cells are set per day. By default Saturday & Sunday are considered as non-

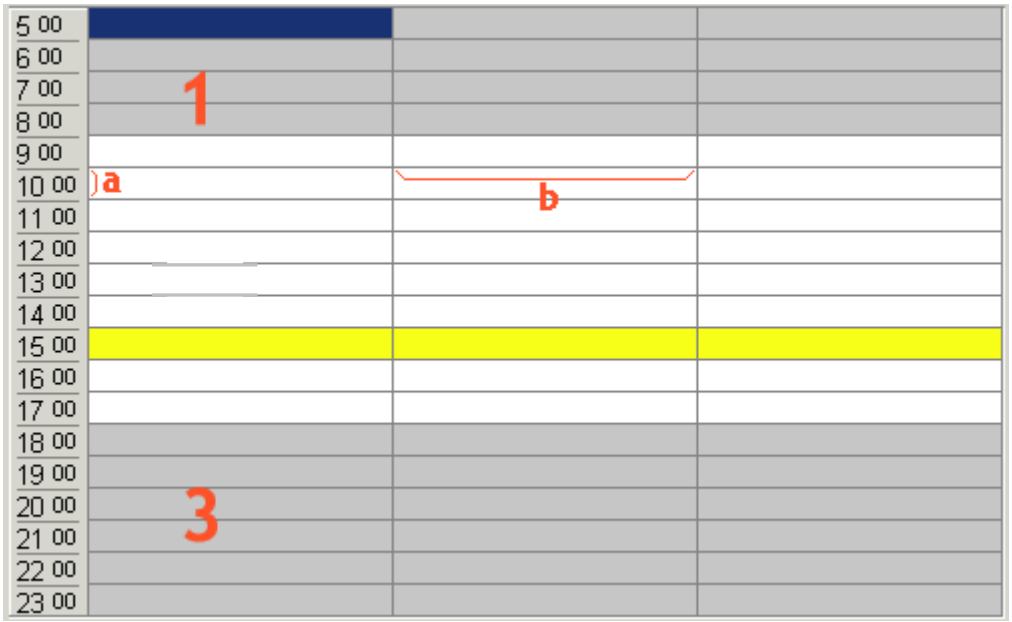
active days but this can be overridden with the TTIWWebPlanner.InActiveDays property
- first month displayed is set by TTIWWebPlanner.Mode.Month, number of months displayed is controlled by the number of positions set by TTIWWebPlanner.Positions

plTimeLine: timeline mode
- time axis represents multiple days with days divided in units defined by TTIWWebPlanner. Display.DisplayUnit
- starting day is set by TPlanner.Mode.TimeLineStart
- number of days is defined by number of units to display set by TTIWWebPlanner.Display.DisplayEnd

plCustom : custom mode
- no fixed relationship between time axis and planner grid
- no predefined active & non-active cells
- number of cells displayed is controlled by TTIWWebPlanner.Display.DisplayStart / TTIWWebPlanner.Display.DisplayEnd

plCustomList: custom mode with timelist
- the relationship between time axis and planner grid is set through the list TTIWWebPlanner.DateTimeList (this is a list of TDateTime values)
- each entry in the DateTimeList corresponds to the start time of a row (in vertical mode) or column (in horizontal mode)

TTIWWebPlanner time axis display control



Through the TTIWWebPlanner.Display property, several settings can be done to control the display of the grid along the time axis. In the above image, **1** is considered the first inactive zone and **3** is the second inactive zone. Distance **a** is the scale of the time axis and is set through the property TTIWWebPlanner.Display.DisplayScale in pixels. Distance **b** is the width (or height in horizontal mode) of a position.
In day mode, the time difference between 2 rows is set by the property TTIWWebPlanner.Display.DisplayUnit. This value is expressed in minutes and can be any value starting from 1. A complete overview of the TPlanner.Display property is here :

ActiveEnd : sets the end of the active time zone. This is expressed in number of cells from the top of the grid.
ActiveStart : sets the start of the active time zone. This is expressed in the number of cells from the top of the grid.
ColorActive : sets the color of the active time zone
ColorNonActive : sets the color of the non active time zone
DisplayEnd : sets the end of the displayed time zone. This is expressed in number of display units.
DisplayStart : sets the start of the displayed time zone. This is expressed in the number of display units.
DisplayScale : sets the height (or width) in pixels of a single time unit.
DisplayText : Sets the interval where text is displayed in the sidebar. When DisplayText is zero, this is ignored and for every position in the sidebar, the text is displayed. For example: setting DisplayText = 2, will show the text in the sidebar only on every other position.
DisplayUnit : in Day mode sets the time in minutes that corresponds to a single row (in vertical mode) or column (inhorizontal modeview) in the planner. For the other modes, it scales the displayed number of rows (or columns in horizontal view).
ScaleToFit : when the property is set to true, the DisplayScale property is automatically adapted to make sure all cells along the time axis fit in the grid without the need to display scrollbars.
CurrentPosFrom, CurrentPosTo : When different from -1, only items in positions between CurrentPosFrom, CurrentPosTo can displayed as 'current' items

Example:

The planner needs to display an active time between 9 AM and 5 PM in 15 minute intervals. The total time displayed is from 6AM to 22PM. The settings for Display are :

```
var
  RowsPerHour: integer;
begin
  with Planner1.Display do
  begin
    DisplayUnit := 15;
    RowsPerHour := (60 div DisplayUnit);
    DisplayStart := 6 * RowsPerHour;
    DisplayEnd := 22 * RowsPerhour;
    ActiveStart := (9 - 6) * RowsPerHour;
    ActiveEnd := (17 - 6) * RowsPerHour;
  end;
end;
```

Other WebPlanner properties

AlertOutsidePlanner :
Sets the text displayed in a popup window when the user moved a planneritem outside the planner.

ChangeCaption :
If subproperty EnterCaptionAfterItemCreate is true, the user will be asked to enter the caption immediately after creating an item by selection.
If subproperty ShowDialogWindow is true, the user will be asked if he wants to change the caption of an item after right click on this caption. When false, this dialog window will not be displayed and a prompt window immediately appears to enter the caption. If subproperty ShowPromptWindow is true, a prompt window will be shown after the user has right clicked on a caption of an item which CaptionType is ctText or ctTimeText. If this property is false, the user has no possibility to change the caption of an item.
Attention : when the EditType of the planner is edPopup, it is not possible to change the caption of the item on this way because the user can change the caption after clicking twice on the itemtext and there in the popup window, he can change the caption.

ClientEvents :
There are currently 4 clientevents available for the webplanner:
- 'EditStart'
will be executed when the user starts to edit an item;
- 'EditDone'
will be executed when the user has confirmed the editing of an item;
- 'DeleteClick'
will be executed when the user clicks on the delete button of an item;
- 'Select'
will be executed when the user has selected an item.

EnableSelection :
When true, the user can make a selection of cells. When false, the user isn't able to selection a range of cells. This can be used when a user doesn't have permission to add an item.

ViewOnly :
When true, the user can only view the planner. He is not able to add new items, to move items, to resize, to delete, to edit text or caption,…

SelectionMode:
When SelectionMode is SingleSelect, only one cell can be selected to add a new item. It's not possible to have an item that larger (in horizontal mode) or higher (in vertical mode) is than one cell.
When SelectionMode is MultiSelect, the user can select more than one cell at a time, so he's able to make an item over a range of cells.

The user also has the possibility to select a cell or a range of cells with the keyboard. To move between the cells, use the arrow keys. To select a range of cells, use the Shift-key and the arrow keys. To add an item at the selected location, use the Insert-key.

Template :
This can be used by the DBWebPlanner to define what will be displayed in the notes of an item. For example, your database has a field 'Location' and a field 'details'.
You can make that the notes of an item contains the location and the details field of the database :
'Location : (#Location) Details : (#Details)'

AutoInsDel:
When this property is set to true, you are able to add a new item or delete an existing item without writing any code.
If the events OnItemDeleted and OnItemCreated are assigned, they will be executed before deleting/adding the item.
When this property is false, you have to assign the events OnItemCreate and OnItemDelete to create/delete an item.

Hints :
You can define the following hints:

- HintCancelItem
  a user hovers the cancel button of an item in editing mode
- HintConfirmDialog
  a user has selected a range of cells and a popup window appears to ask if the user want to add an item at the selected range.
- HintDeleteItem
  a user hovers the delete button of an item
- HintEditItem
  will be displayed when hovering the text of an item if the item is editable
  (the hint will not appear when the item is a background item or read-only,…)
- HintUpdateItem
  a users hovers the button to update the changes that were made of the item in editing mode

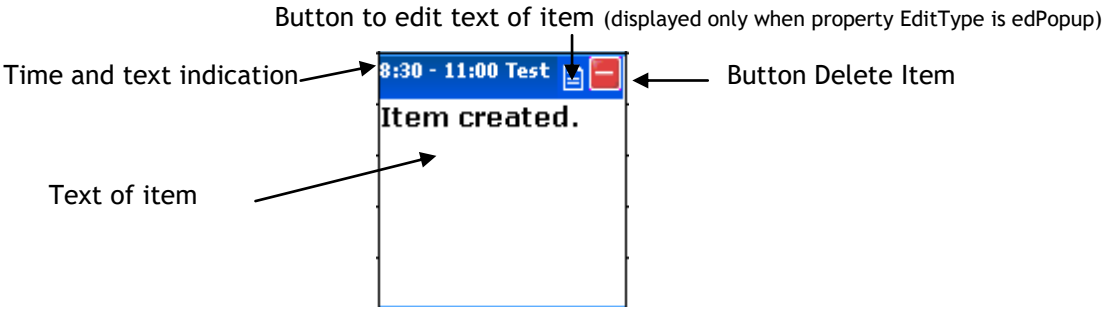
Another subproperty is the CaptionHint. This property defines what will be displayed in the hint that appears when the user is hovering the caption of an item.
When chCaption, only the caption will be displayed in the hint. When chCaptionNotes, the caption and the notes are displayed. When chNone, there will be no hint when the user hovers the caption of an item.

Inside the PlannerItem

The TPlannerItem is the central object that stores all information for display of events and appointments in the TTIWWebPlanner. TPlannerItems are inserted into the TPlanner.Items collection and TTIWWebPlanner takes care of showing these at the correct place in the TTIWWebPlanner. The TTIWWebPlanner has a property DefaultItem through which default properties for newly created items are defined. When calling TTIWWebPlanner.CreateItem: TPlannerItem or TTIWWebPlanner.CreateItemAtSelection: TPlannerItem, all initial property settings are taken from the default item. This allows to define a standard look for the TPlannerItem and only set some specific properties that are unique for the created TPlannerItem.

First a brief overview is given of the TPlannerItem properties and then the methods that are available in the TPlannerItems collection to add, find, move items and more ...

The image below is an example of the look of an item:



TPlannerItem position control

The most important properties of the TPlannerItem are the properties that control the position of the TPlannerItem within the planner grid. As the base framework of the TTIWWebPlanner is time independent, the most simple properties with which the TPlannerItem position can be set are :

TPlannerItem.ItemBegin: Integer;
TPlannerItem.ItemEnd: Integer;
TPlannerItem.ItemPos: Integer;

The ItemBegin / ItemEnd properties set the cell index of start of the item and end of the item along the time axis, ie. either the row indexes when the TTIWWebPlanner is in vertical mode (sidebar on left) or the column indexes in horizontal mode (sidebar on top) The ItemPos property then sets the position index. This is the or column index in vertical mode or and the row index in horizontal mode. Note that ItemBegin, ItemEnd and ItemPos are independent of the TTIWWebPlanner.Display.DisplayUnit.

When a TTIWebPlanner mode is choosen such as plDay, plMonth, or plPeriod, the position of the TPlannerItem can be set in a more convenient way through public properties :

TPlannerItem.ItemStartTime: TDateTime;
TPlannerItem.ItemEndTime: TDateTime;

Setting or getting the position through these properties take the TTIWWebPlanner mode and Display settings into account to convert a TDateTime to a row and column position of the item in the planner grid.

Example:

In day mode, a TPlannerItem can be created from 8 AM to 9 AM in following way :

```
with TIWWebPlanner1.CreateItem do
begin
  ItemStartTime := EncodeTime(8,0,0,0); // 08:00 AM
  ItemEndTime := EncodeTime(9,0,0,0);   // 09:00 AM
end;
```

Suppose the TTIWWebPlanner.Display property settings have the TTIWWebPlanner configured to display a time axis in units of 30 minutes starting from 0h, the equivalent code for using TTIWWebPlanner.ItemBegin and TPlannerItem.ItemEnd would be :

```
with TIWWebPlanner1.CreateItem do
begin
  ItemBegin := 8 * (60 div TIWWebPlanner1.Display.DisplayUnit);
  ItemEnd := 9 * (60 div TIWWebPlanner1.Display.DisplayUnit);
end;
```

When setting item positions through ItemBegin / ItemEnd, the time of the item is of course always set or returned on boundaries of currently selected TPlanner.Display.DisplayUnit. When changing from one DisplayUnit to another DisplayUnit, it is preferred that the full time precision is kept when switching from a small DisplayUnit to a large DisplayUnit and back. This is where the properties TPlannerItem.ItemBeginPrecis and TPlannerItem.ItemEndPrecis can be used. These properties set the TPlannerItem position by minutes starting from 0h in day mode. Creating the same item from 8 AM to 9 AM through these properties becomes :

```
const
  MinutesPerHour = 60;

with TIWWebPlanner1.CreateItem do
begin
  ItemBeginPrecis := 8 * MinutesPerHour;
  ItemEndPrecis := 9 * MinutesPerHour;
  ItemPos := 3;
end;
```

Thus, like using ItemBegin and ItemEnd, the setting of the position through ItemBeginPrecis and ItemEndPrecis is again independent of the choosen DisplayUnit. When creating a TPlannerItem this way and switching the display unit, ItemBeginPrecis / ItemEndPrecis and ItemStartTime and ItemEndTime will always keep the full time resolution.

A similar approach can be used for a TTIWWebPlanner in month or period mode. The TPlannerItem position can be set by :

```
with Planner1.CreateItem do
begin
  ItemStartTime := EncodeDate(2002,7,15);
  ItemEndTime := EncodeDate(2002,7,16);
end;
```

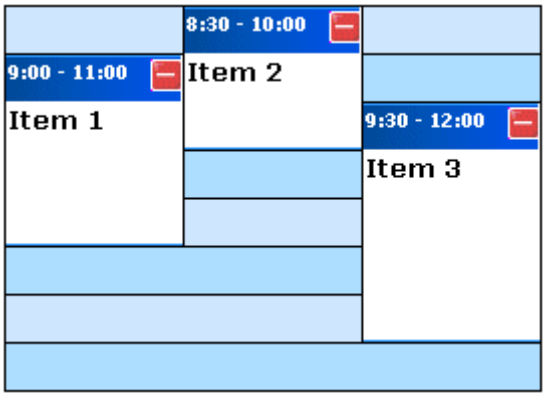
The TPlannerItem.ItemPos: Integer property is simpler and just sets the column (vertical mode) or row (horizontal mode)

Overlapping TPlannerItems

Within a given cell by default TPlannerItems can overlap. Creating 2 items at the same time will show these as overlapped in a cell.

Overlapping can be turned off in TTIWWebPlanner by the Overlap property or for each TPlannerItem separately with the TPlannerItem.AllowOverlap property.
It will not prevent though that programmatically items are created in positions that already have items. It is the responsibility of the programmer to ensure that no items are available in the position where the TPlannerItem is created. This can be checked with the function TPlanner.Items.HasItem and an appropriate message can be shown to the user if it is not allowed to create overlapping items.

If overlapping items are allowed, multiple items are displayed in a single cell. The properties TPlannerItem.Conflicts: Integer returns the number of conflicting or overlapping items that exist for this item and TPlannerItem.ConflictPos: Integer returns in which position the overlapping item is displayed in the cell.



In the sample above, each item will return 3 for the readonly property TPlannerItem.Conflicts while Item 1 will return 0 for TPlannerItem.ConflictPos, Item 2 will return 1 for TPlannerItem.ConflictPos and Item 3 will return 2 for TPlannerItem.ConflictPos.

Using TPlannerItems in 3 dimensions

TTIWWebPlanner has support to use different layers. This means items can be assigned to a layer in the TTIWWebPlanner and the TTIWWebPlanner can be instructed to show all, a single or a combination of different layers. Therefore, each TPlannerItem has a Layer property. The TTIWWebPlanner component itself also has a Layer property.

Layer selection is done in a binary way. Layer numbers are 1,2,4,8,16 … (i.e. all powers of 2). If a TPlannerItem.Layer is 2, it will be displayed in the TTIWWebPlanner if TTIWWebPlanner.Layer is 2. If TTIWWebPlanner.Layer is 0, this means all layers are displayed. To display items from 2 layers in the TTIWWebPlanner, this can be done by a logical OR of the layer numbers in the TTIWWebPlanner.Layer property.

Example :

3 TPlannerItems are in the TTIWWebPlanner, TPlannerItem A has a Layer property 1, and TPlannerItem B has a Layer property 2 and TPlannerItem C has a Layer property 4.

If TTIWWebPlanner.Layer is 0, items A,B,C are displayed in the TTIWWebPlanner. If TTIWWebPlanner.Layer is 1, only item A will be displayed in the TTIWWebPlanner. If TTIWWebPlanner.Layer is 2, only item B is displayed in the TTIWWebPlanner. When items from both layer 1 and layer 2 need to shown in the TTIWWebPlanner, this can be done by setting the TTIWWebPlanner.Layer property to 3 (= 1 OR 2)

Controlling moving and sizing of TPlannerItems at runtime

With default properties, all items shown in the planner grid can be sized and moved at runtime without limitations. It might be desirable to limit what can be done with a TPlannerItem at runtime. This is done through following properties :

FixedPos : Boolean : When true, prevents that items are moved in the TPlanner
FixedSize : Boolean : When true, prevents that items are sized in the TPlanner
FixedPosition : Boolean : When true, items can only be moved within the same position (position being a column in vertical mode or row in horizontal mode)
Background : Boolean : When true, the item cannot be selected, moved or sized

Further finer control is possible through the events :

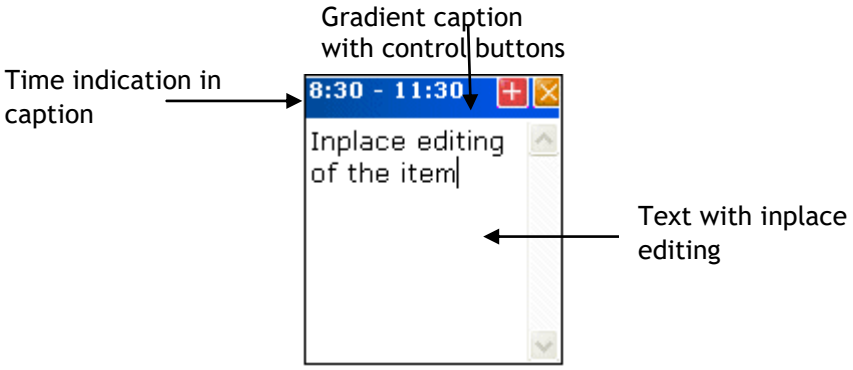
OnItemMove : triggered after the item has been moved
OnItemSize : triggered after the item has been sized

Editing items

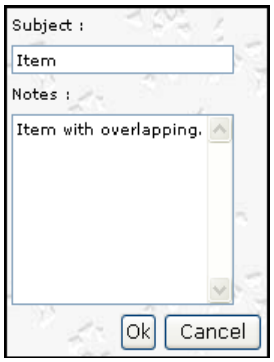
The TPlannerItem.ReadOnly property controls whether the item can be edited at runtime or not. If the ReadOnly property is false, you can edit an item by clicking twice on an item. The first time you click an item, the item will be shown in a different background and font color, to give you a visual view of the selected item. When you click again this item, you can edit the item. It is possible to insert html-formatted text when you edit the text of an item.

There are two types to edit an item : by inplace editing or by a popup window. This can be set by the property EditType.

When you use the InplaceEditor, you will see something similar to the image below.
The text of the item becomes editable and after you changed the text of the item, you can update the changes or you can cancel the changes. This can be done using the buttons in the caption.



When you use the PopupEditor, a popup window appears after clicking twice on the item text or after clicking on the button on the left side of the delete button in the caption of the item. When you use the popup editor, you are able to edit the text of the item and the subject of the item. You confirm the changes with the button Ok, cancel the changes with the button Cancel.



Other properties that control the PlannerItem appearance

Additional properties that control the appearance of a TPlannerItem in the grid are :

Alignment : sets the alignment of text in the TPlannerItem
Attachement : string that can point to an attachement. If Attachement is a non-empty string, this is indicated by a clip in the TPlannerItem caption *(not yet implemented)*
Background: when true, the item is a non moveable, non sizeable background item
CaptionAlign : sets the alignment of the TPlannerItem caption text
CaptionBkg : sets the background color of the TPlannerItem caption. This property has no effect when UniformBkg is true, as the TPlannerItem will always have a full uniform color
CaptionBkgTo : when different from clNone and UniformBkg is false, the caption is drawn with a gradient from CaptionBkg color to CaptionBkgTo color
CaptionFont : sets the font for the TPlannerItem caption
CaptionText : sets the text for the caption. This text is only displayed when CaptinTupe is either ctText or ctTimeText
CaptionType : can be
- ctNone : no caption is displayed
- ctText : CaptionText is displayed
- ctTime : time of item is displayed
- ctTimeText : time and CaptionText are displayed at the same time

Color : sets the background color of the TPlannerItem
Font : sets the font of the TPlannerItem
Hint: sets the hint for the item
InHeader : when true, the item is displayed inside the TPlanner header
SelectColor : sets the color of the item when it is selected
SelectFontColor : sets the font color of the item when it is selected
ShowSelection : when true, selected items are displayed with SelectColor and SelectFontColor
Text : the stringlist through which the TPlannerItem text is set
TrackColor : sets the color of the trackbar. The trackbar is the small colored bar with which the item can be dragged and moved inside the planner grid.
UniformBkg : when true, CaptionBkg color has no effect
URL : string that can point to an URL. If URL is a non-empty string a link icon is displayed in the TPlannerItem caption *(not yet implemented)*
Visible : sets the visibility state of the TPlannerItem

Using a custom TPlannerItem class

For maintaining custom data with each planner item you can assign any TObject descendent class to the PlannerItem's public Object property. However there is a more convenient way to create a descendent class from TTIWWebPlanner that has a TPlannerItem with new custom properties which can be used at design time and at run time to hold any additional values with each planner item. The code involved comes down to :

1. Write your descendent class of TPlannerItem and add the additional properties

2. Write your descendent class of the TPlannerItems collection and override the GetItemClass method to instruct the collection to create collection items from your descendent TPlannerItem class.

3. Write your descendent class of TTIWWebPlanner and override the protected CreateItems method to let the planner use your descendent TPlannerItems collection.

Following code where a new property MyProperty was added to the TPlannerItem, makes this clear :

Example:

```
unit MyWebPlanner;

interface

uses
  IWWebPlanner;

type
  TMyPlannerItem = class(TPlannerItem)
  private
    FMyProperty: string;
  published
    property MyProperty: string read FMyProperty write FMyProperty;
  end;

  TMyPlannerItems = class(TPlannerItems)
  public
    function GetItemClass: TCollectionItemClass; override;
  end;

  TMyWebPlanner = class(TTIWWebPlanner)
  private
    { Private declarations }
  protected
    { Protected declarations }
    function CreateItems: TPlannerItems; override;
  public
    { Public declarations }
  published
    { Published declarations }
  end;

procedure Register;
```

```
implementation

procedure Register;
begin
  RegisterComponents('TMS', [TMyWebPlanner]);
end;

{ TMyPlannerItems }

function TMyPlannerItems.GetItemClass: TCollectionItemClass;
begin
   Result := TMyPlannerItem;
end;

{ TMyPlanner }

function TMyPlanner.CreateItems: TPlannerItems;
begin
  Result := TMyPlannerItems.Create(Self);
end;

end.
```

Note: the same method can be used for creating custom TPlannerItem classes with the TTIWDBWebPlanner. In some events, such a TIWDBItemSource.OnFieldsToItem or TIWDBItemSource.OnItemToFields, the extra properties of the custom TPlannerItem class can be accessed by casting the Item parameter to the custom TPlannerItem class.

The TPlannerItems collection

The collection that holds all TPlannerItem objects features a whole array of methods and functions that allow manipulating the items inside the planner.

Important note:
When making changes that affect a lot of items in the planner or when adding or removing a lot of items, performance will vastly improve when enclosing the operations with

```
TPlanner.Items.BeginUpdate;
// do update, adding, removing of items here
TPlanner.Items.EndUpdate;
```

Searching items in the planner

function **HasItem**(ItemBegin, ItemEnd, ItemPos: Integer): Boolean;

Returns if the planner cell has items at a given position

function **FindFirst**(ItemBegin, ItemEnd, ItemPos: Integer): TPlannerItem;

Returns the first item at a given cell

function **FindNext**(ItemBegin, ItemEnd, ItemPos: Integer): TPlannerItem;

Returns the next item at a given cell

function **FindText**(StartItem:TPlannerItem;s: string; Param: TFindTextParams):TPlannerItem;

Finds text in all TPlannerItem Object's Text property in the planner grid

function **HeaderFirst**(ItemPos: Integer): TPlannerItem;

Returns the first item in a planner header at position ItemPos

function **HeaderNext**(ItemPos: Integer): TPlannerItem;

Returns the first item in a planner header at position ItemPos

function **ItemsAtPosition**(Pos: Integer): Integer;

Returns the number of items in a given position

function **ItemsAtIndex**(Idx: Integer): Integer;

Returns the number of items at a given index along the time axis

Example:

This code fragment searches all items in the TTIWWebPlanner for the word 'Meeting' :

```
var
  plIt: TPlannerItem;
begin
  plIt := nil;
```

```
  repeat
    plIt :=
      TIWWebPlanner1.Items.FindText(plIt,'Meeting',
        [fnAutoGoto,fnText]);
    if Assigned(plIt) then
      WebApplication.ShowMessage('Found appointment');
  until plIt = nil;
  WebApplication.ShowMessage('No more items found');
end;
```

Item selection in the planner

Item selection is by default single selection. Selecting an item automatically unselects the previously selected item.

function **SelectNext**: TPlannerItem;

Selects the next item in the list

function **SelectPrev**: TPlannerItem;

Selects the previous item in the list

procedure **UnSelect**;

Unselects the currently selected item

procedure **UnSelectAll**;

Unselects all items in the planner

procedure **Select**(Item: TPlannerItem);

Selects the TPlannerItem programmatically

property **Selected**: TPlannerItem;

Returns the currently selected item.

Moving, sizing & removing items

procedure **MoveAll**(DeltaPos, DeltaBegin: Integer);

Moves all items with DeltaPos for Position and DeltaBegin for ItemBegin

procedure **MoveSelected**(DeltaPos, DeltaBegin: Integer);

Moves selected items only with DeltaPos for Position and DeltaBegin for ItemBegin

procedure **SizeAll**(DeltaStart, DeltaEnd: Integer);

Sizes all items with DeltaStart for ItemBegin and DeltaEnd for ItemEnd

procedure **SizeSelected**(DeltaStart, DeltaEnd: Integer);

Sizes selected items only with DeltaStart for ItemBegin and DeltaEnd for ItemEnd

procedure **ClearPosition**(Position: Integer);

Removes all items in a given position

procedure **ClearLayer**(Layer: Integer);

Removes all items in a given layer

procedure **ClearAll**;

Removes all items

Events of the TTIWWebPlanner / TTIWDBWebPlanner

OnFailedInsert : occurs when user adds an item to a location which doesn't allow this item. An example of a situation like this that on this location, there is a background item and a background item doesn't allow overlapping.
OnFailedMove : occurs when user has moved an item to an invalid location, for example a location that already contains an item but this item doesn't allow any overlapping.
OnFailedSize : occurs when user has resized an item to an invalid location, for example a location that contains a background item.
OnItemCaptionChange : occurs when user has changed the caption of an item by right clicking on the caption of this item.
OnItemChange : occurs when the user has move, resized or edited an item.
OnItemClick : occurs when user has clicked an item.
OnItemCreate : occurs when user has created an item.
OnItemDelete : occurs when user deleted an item.
OnItemEdit : occurs when user edited an item.
OnItemMove : occurs when user has moved an item.
OnItemSize : occurs when user has resized an item.
OnPlannerGetSideBarLines : occurs when the sidebar is rendering.
OnPlanTimeToStrings : occurs when the time values of the sidebar are casted to strings. So you can manipulate the representation of the time values of the sidebar.
OnItemCreated : occurs when an item has been created.
OnItemDeleted : occurs when an item has been deleted.

Async Events of the TTIWWebPlanner / TTIWDBWebPlanner

OnAsyncEditDone : occurs when user updates an item that is in edit mode. Use the allow parameter to prevent that the item is updated.
OnAsyncEditStart : occurs when user sets an item in edit mode. Use the allow parameter to prevent that the item is set in edit mode.
OnAsyncHeaderClick : occurs when user clicks a caption in the header.
OnAsyncItemClick : occurs when a user has selected an item.
OnAsyncItemDelete : occurs when user deleted an item. Use the allow parameter to prevent that the item is deleted.
OnAsyncItemMove : occurs when user has moved an item. Use the allow parameter to prevent that the item is moved.
OnAsyncItemSize : occurs when user has resized an item. Use the allow parameter to prevent that the item is resized.

Saving and loading items in non data-aware TTIWWebPlanner

Several methods exist to save and load TPlannerItem objects to a stream or to a file. Using these methods, all properties of the TPlannerItem objects are saved on a stream or file. The format of the TPlannerItem's on the stream or in the file is both a binary format.

The methods can be summarized as :

Save and load all TPlanner items to a stream or file :
procedure **SaveToStream**(Stream: TStream);
procedure **LoadFromStream**(Stream: TStream);
procedure **SaveToFile**(FileName: string);
procedure **LoadFromFile**(FileName: string);

Save and load only items in a given position to a stream or file :
procedure **SavePositionToStream**(Stream: TStream; Position: Integer);
procedure **LoadPositionFromStream**(Stream: TStream; Position: Integer);
procedure **SavePositionToFile**(FileName: string; Position: Integer);
procedure **LoadPositionFromFile**(FileName: string; Position: Integer);

Save and load only items of a given layer to a stream or file :
procedure **SaveLayerToStream**(Stream: TStream; Layer: Integer);
procedure **LoadLayerFromStream**(Stream: TStream; Layer: Integer);
procedure **SaveLayerToFile**(FileName: string; Layer: Integer);
procedure **LoadLayerFromFile**(FileName: string; Layer: Integer);

Add items from from a stream or file to the existing TPlannerItems
procedure **InsertFromFile**(FileName: string);
procedure **InsertFromStream**(Stream: TStream);

Example:

If there are 2 TTIWWebPlanner components are on a form, the method below makes a copy of all TPlannerItem objects from Planner1 to Planner2 by a memory stream :

```
procedure TForm1.Save(Sender: TObject);
var
  ms: TMemoryStream;
begin
  ms := TMemoryStream.Create;
  Planner1.SaveToStream(ms);
  ms.Position := 0;
  Planner2.LoadFromStream(ms);
  ms.Free;
end;
```

Additional TTIWWebPlanner / TTIWDBWebPlanner methods and properties

Cell selection

Several functions exist to get the selected cells in the planner grid or to convert date & time to cell coordinates and vice versa.

The selected cells in the planner grid can be retrieved with :

SelPosition: Integer; retrieves the index of the position where the selected cell is
SelItemBegin: Integer; retrieves the index along the time axis where the cell selection starts
SelItemEnd: Integer; retrieves the index along the time axis where the cell selection ends

In a vertical oriented grid, the SelPosition thus indicates the column index of the selected cell and SelItemBegin, SelItemEnd indicate the row indexes where the selection starts and ends.

Conversion of absolute time to cell index along the time axis depends of the mode of the TPlanner and can be done with :

procedure **CellToAbsTime**(x: Integer;var dtStart,dtEnd: TDateTime);

and vice versa :

function **AbsTimeToCell**(DateTime: TDateTime): Integer;

Position handling

As explained earlier, the number of positions is either the number of columns (in vertical mode, Sidebar is on left or right) or the number of rows (in horizontal mode, Sidebar is on top) The number of positions can be set with TTIWWebPlanner.Positions. When TTIWWebPlanner.PositionWidth is zero, positions are scaled to fit in the TTIWWebPlanner either vertically and horizontally. If TTIWWebPlanner.PositionWidth is different from zero, this width is applied for all positions. The TTIWWebPlanner provides 3 additional methods to move, delete and insert positions :

procedure **MovePosition**(FromPos, ToPos: Integer);

Moves a position from one position FromPos to a new position ToPos.

procedure **DeletePosition**(Position: Integer);

Deletes a position from the planner.

procedure **InsertPosition**(Position: Integer);

Inserts a new position in the planner at Position index in the planner.

Position properties

As the TTIWWebPlanner.Display settings normally apply to the full planner and thus all positions, the PositionProps offer the capability to override these global settings per position. The PositionProps property is a collection of TPositionProp objects that control Active / Non Active colors, selection color and ActiveStart / ActiveEnd per position. When inserting TPositionProp objects in the PositionProps collection, these control properties of consecutive

positions in the planner, ie. the first PositionProp object controls position 0, the second position 1 and so further. The PositionProp only has effect when its **Use** property is set true. The full list of PositionProp properties is:

ActiveStart: Integer; sets the start of the active time zone
ActiveEnd: Integer; sets the end of the active time zone
ColorActive: TColor; sets the color of the active time zone
ColorNonActive: TColor; sets the color of the non active time zone
MinSelection: Integer; if different from zero, sets the minimum selectable cell
MaxSelection: Integer; if different from zero, sets the maximum selectable cell
ColorNoSelect: TColor; sets the color of the non selectable cells in the position. This has effect only when MinSelection / MaxSelection are set *(not implemented yet)*
Use: Boolean; when true, the PositonProp settings are used, otherwise global Display settings apply.

Example:

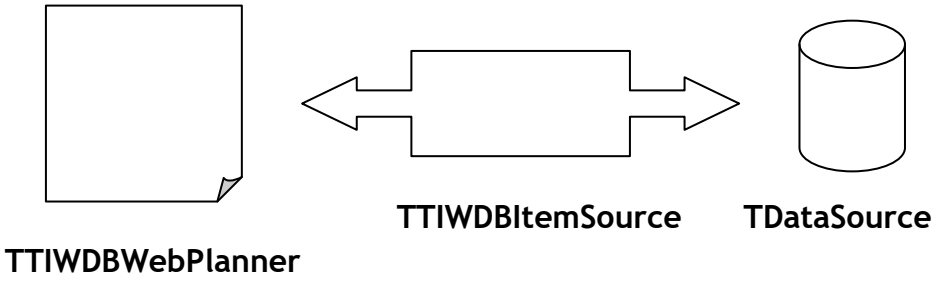
Suppose the TTIWWebPlanner is in day mode with 5 positions and each position represents a day of the week. On Monday and Friday office hours are from 9AM to 17PM while on other days the office hours are from 8AM to 17PM. To indicate the office hours as active hours in each position, following code is used :

```
with TIWWebPlanner1.PositionProps.Add do
begin
  ActiveStart := Planner1.AbsTimeToCell(EncodeTime(9,0,0,0));
  ActiveEnd := Planner1.AbsTimeToCell(EncodeTime(17,0,0,0));
  ColorActive := clYellow;
end;

with TIWWebPlanner1.PositionProps.Add do
begin
  ActiveStart := Planner1.AbsTimeToCell(EncodeTime(8,0,0,0));
  ActiveEnd := Planner1.AbsTimeToCell(EncodeTime(17,0,0,0));
end;

with TIWWebPlanner1.PositionProps.Add do
begin
  ActiveStart := Planner1.AbsTimeToCell(EncodeTime(8,0,0,0));
  ActiveEnd := Planner1.AbsTimeToCell(EncodeTime(17,0,0,0));
end;

with TIWWebPlanner1.PositionProps.Add do
begin
  ActiveStart := Planner1.AbsTimeToCell(EncodeTime(8,0,0,0));
  ActiveEnd := Planner1.AbsTimeToCell(EncodeTime(17,0,0,0));
end;

with TIWWebPlanner1.PositionProps.Add do
begin
  ActiveStart := Planner1.AbsTimeToCell(EncodeTime(9,0,0,0));
  ActiveEnd := Planner1.AbsTimeToCell(EncodeTime(17,0,0,0));
  ColorActive := clYellow;
end;
```

TTIWDBWebPlanner architecture

TTIWDBWebPlanner provides a seamless connection to a dataset for a codeless solution to store events, appointments, allocations to a database. By using standard Borland database connectivity technology, the TTIWDBWebPlanner component can successfully connect to all well known databases supported in Borland Delphi, C++Builder and Kylix. TTIWDBWebPlanner has been used and tested for using BDE dBase & Paradox, ADO MS Access and SQL server, DBIsam, Apollo, Flashfiler, Interbase, MySQL, TurboDB …

As TTIWDBWebPlanner is a multi-purpose scheduling viewer and user interface handling component, the database connectivity is abstracted in the TIWDBItemSource that handles all communication with a TDataSource as in the diagram below

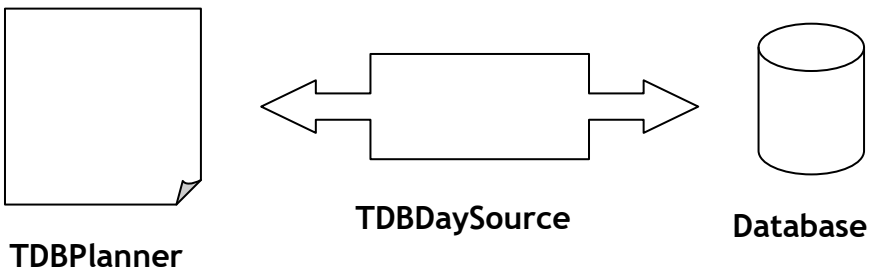


The TIWDBItemSource component is an abstract class for communication handling between TTIWDBWebPlanner and the database. Descendent classes of TTIWDBItemSource handle specific TTIWDBWebPlanner scheduler setups for a given database of appointment, events, … records. TTIWDBWebPlanner comes standard with following descendent components of TIWDBItemSource :

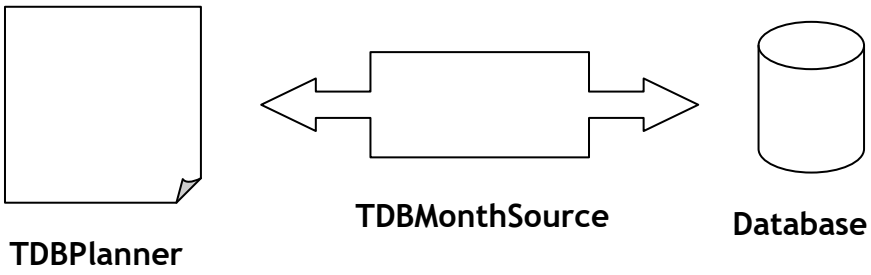
TTIWDBDaySource : day mode connection between database and TTIWDBWebPlanner
TTIWDBWeekSource : week mode connection between database and TTIWDBWebPlanner
TTIWDBMonthSource : month mode connection between database and TTIWDBWebPlanner
TTIWDBPeriodSource : period mode connection between database and TTIWDBWebPlanner
TTIWDBMultiMonthSource : multi month mode connection between database and TTIWDBWebPlanner

To display appointments or events in a database in different ways, this can be achieved by replacing the TDBItemSource with a new appropriate interface.

Case 1 : day mode view :



Case 2 : month mode view :



In case 2, the TDBDaySource is simply replaced by a TDBMonthSource. Programmatically, a TDBDaySource and TDBMonthSource can be on the form or datamodule and upon need connected to the TTIWDBWebPlanner with :

```
case Mode of
monthmode: DBPlanner1.ItemSource := DBMonthSource1;
daymode: DBPlanner1.ItemSource := DBDaySource1;
end;
```

As such, it is relatively easy to write custom TDBItemSource descendents to accommodate special needs for mapping between the database and the TTIWDBWebPlanner

Database requirements

The minimum requirement is that 3 fields exist in the database :

- **Start time** : full date/time field (can be a 20 char field for databases not supporting datetime fields)

- **End time** : full date/time field (can be a 20 char field for databases not supporting datetime fields)

- **Key** : 40 char unique item key field (or other key field type when the ItemSource.OnCreateKey event is used) When no OnCreateKey event handler is assigned the unique key to identify an appointment is a GUID. If the OnCreateKey event handler is used any other scheme that guarantees a unique key creation can be used such as auto increment fields.

Other fields are optional. TDBItemSource has built-in support for mapping following fields to a TPlannerItem

- Notes field : memo field or character field holding the planner item text

- Subject field : character field holding the planner item caption text

- Resource field : numeric field holding the planner item position in dmMultiResource mode

Any other fields that link to planner item properties can be added with field type of choice. Use the **DBItemSource.OnFieldsToItem** and **DBItemSource.OnItemToFields** event for setting field values to Item properties and vice versa.

Example:

Following code maps additional database fields COLOR to the TPlannerItem.Color, IMAGE field to the TPlannerItem.ImageID and CAPTION field to captiontype of the TPlannerItem :

```
procedure TForm1.DBDaySource1FieldsToItem(Sender: TObject; Fields:
TFields;
  Item: TPlannerItem);
begin
   Item.Color := TColor(Fields.FieldByName('COLOR').AsInteger);
   Item.CaptionBkg := Item.Color;
   Item.ImageID := Fields.FieldByName('IMAGE').AsInteger;
   if Fields.FieldByName('CAPTION').AsBoolean then
     Item.CaptionType := ctTime
   else
     Item.CaptionType := ctNone;
end;
```

```
procedure TForm1.DBDaySource1ItemToFields(Sender: TObject; Fields:
TFields;
  Item: TPlannerItem);
begin
  Fields.FieldByName('COLOR').AsInteger := Integer(Item.Color);
  Fields.FieldByName('CAPTION').AsBoolean := Item.CaptionType =
ctTime;
  Fields.FieldByName('IMAGE').AsInteger := Item.ImageID;
end;
```

Note: in some cases, it might be necessary to store the start and end times of an event in a different way. This could be a starttime field and a duration field. In this case, the events OnFieldsToTime and OnTimeToFields can be used. The definition for OnFieldsToTime is:

```
procedure TForm1.DBDaySource1FieldsToTime(Sender: TObject; Fields: TFields;
  var dtS, dtE: TDateTime);
```

With this event, the fields required can be read and the start and end time can be set in the dtS and dtE parameters.

The inverse operation is done with:

```
procedure TForm1.DBDaySource1TimeToFields(Sender: TObject; Fields: TFields;
  dtS, dtE: TDateTime);
```

Setting up TTIWDBWebPlanner, TDBItemSource and the database

Drop a database table or query component with datasource on the form. Add a TDBItemSource component on the form and assign a datasource component to your used database.

Set the applicable fields to the TDBItemSource fields properties.

It is required that the KeyField, StartTimeField, EndTimeField properties are defined.

The TDBItemSource can be
TDBDaySource component for a day mode planner
TDBWeekSource component for a month mode planner
TDBMonthSource component for a month mode planner
TDBPeriodSource component for a day period mode planner
TDBHalfDayPeriodSource component for a half-day period mode planner
TDBMultiMonthSource component for a month mode planner
TDBTimeLineSource component for a timeline mode planner

There is only a single connection between the TTIWDBWebPlanner and the TDBItemSource. Each TTIWDBWebPlanner must have as such its unique TDBItemSource. A datasource component can have multiple TDBItemSource components connected though.

Example :

An Access database is used with following fields :

KEYFIELD : 40 char field
STARTTIME : datetime field
ENDTIME : datetime field
SUBJECT : 40 char field
NOTES : memo field
COLOR : long integer field
IMAGE : long integer field
CAPTION : boolean field

On the form, a TADOTable with connection to the Access database is dropped, a TDataSource component and a TDBDaySource. The TDBDaySource component is assigned to the TTIWDBWebPlanner.ItemSource property. The DataSource component is connected to the TDBDaySource.DataSource property :

Next, the fields that are defined in the TDBDaySource component properties :

Active	True
AutoHeaderUpdate	False
⊞DataSource	DataSource1
DateFormat	mm/dd/yyyy
Day	28/02/2001
DayIncrement	1
EndTimeField	ENDTIME
KeyField	KEYFIELD
Mode	dmMultiDay
Name	DBDaySource1
NotesField	NOTES
NumberOfDays	7
NumberOfResources	1
ReadOnly	False
ResourceField	
StartTimeField	STARTTIME
SubjectField	SUBJECT
Tag	0

When making the ADO table active, the database is searched for all items that should be displayed in the TTIWDBWebPlanner for the date(s) selected by TDBDaySource and these are displayed appropriately in the TTIWDBWebPlanner.

Important note :

As the TDBItemSource is the interface that has the knowledge of how to map and display the items in the TTIWDBWebPlanner, it is the TDBItemSource that sets TTIWDBWebPlanner properties automatically for selecting modes and display. As such, in the above example, connecting a TDBDaySource to the TTIWDBWebPlanner.ItemSource property will automatically put the TTIWDBWebPlanner.Mode.PlannerType into plDay. Depending on what each TDBItemSource component visualises, other TTIWDBWebPlanner properties might be automatically set by the TDBItemSource component.

Performance guidelines

The TDBItemSource will search all records of the dataset for items that should be displayed in the TTIWDBWebPlanner. As such, it is important that the number of records that must be searched through is as small as limited to minimum number possible. This can be achieved by using a TQuery with a SQL statement that returns only a minimum set of records to search through or by applying a filter. This filter or query can be set at any time, but a good place would be to do this as well as just before the TDBItemSource searches through the records, using the TDBItemSource events OnSetFilter and OnChangeQuery.. The TDBItemSource will therefore trigger the events OnSetFilter and OnChangeQuery to allow updating filter or query before it needs to do a new search.

Example :

When using a TDBDaySource for a single day and a single resource in a database with appointments for multiple resources and multiple days, it is recommend to specify a query or set a filter to return only a dataset of records for this single day and single resources in order to avoid useless needless searches of the TDBDaySource through records of appointments for other days and other resources.

This is a good design method for databases applications in general: always try to minimize the amount of data that you load from the database.

TTIWDBWebPlanner and database synchronisation

Records can be added either directly in the database as well as via the TTIWDBWebPlanner component. When the TTIWDBWebPlanner is connected to a database via a TDBItemSource component, following interactions happen :

TTIWDBWebPlanner.CreateItem and TTIWDBWebPlanner.CreateItemAtSelection :

A new record is created in the database.

TTIWDBWebPlanner.FreeItem(APlannerItem: TPlannerItem);

The record associated in the database with APlannerItem is deleted.

Whenever planner item properties are changed, the property changes are reflected in the database by calling the PlannerItem's Update method.

When records from the database that are displayed in the TTIWDBWebPlanner are changed through other DB-aware controls , the field changes are immediately updated in the TTIWDBWebPlanner by changes in the planner item properties. When a record is deleted or inserted from the database through another DB-aware component or programmatically, the TTIWDBWebPlanner must be synchronized. This synchronization is achieved by calling the TDBItemSource SynchDBItems method in the dataset AfterDelete or AfterInsert event.

Example :

This creates a one hour appointment now in a TTIWDBWebPlanner with TTIWDBDaySource connected to a database :

```
TIWDBDaySource1.Day := Now;
with TIWDBWebPlanner1.CreateItem do
begin
  ItemStartTime := Now;
  ItemEndTime := ItemStartTime + EncodeTime(1,0,0,0);
  CaptionText := 'Auto created';
  Text.Text := 'This item is auto created';
  Update;
end;
```

The standard TDBItemSource components

TTIWDBWebPlanner comes default with ready to use TDBItemSource components for several modes of the TPlanner :

TTIWDBDaySource component for a day mode planner
TTIWDBWeekSource component for a month mode planner
TTIWDBMonthSource component for a month mode planner
TTIWDBPeriodSource component for a day period mode planner
TTIWDBMultiMonthSource component for a month mode planner
TTIWDBTimeLineSource component for a timeline mode planner
TTIWDBDisjunctDaySource component for disjunct day mode planner
TTIWDBActiveDaySource component for active-day only planner

These different TTIWDBItemSource components have properties that are specific for each mode which is discussed here.

TTIWDBDaySource

The TTIWDBDaySource is the interface for handling a database connection in 4 different TTIWDBWebPlanner modes. The main setting for these capabilities is the TTIWDBDaySource.Mode property with can be:

dmMultiday : Single or multiple days for a single resource
dmMultiResource : Single or multiple resources for a single day
dmMultiDayRes : Multiple days for multiple resources
dmMultiResDay : Multiple resources for multiple days

The first 2 modes are the most simple modes:

dmMultiDay
The number of days shown in the TTIWDBWebPlanner is set with the TTIWDBDaySource.NumberOfDays property. Multiple days will be displayed in multiple positions of the TTIWDBWebPlanner. As such, this NumberOfDays property sets the TTIWDBWebPlanner.Positions property. The first day displayed is set with the property TTIWDBDaySource.Day. The date difference between 2 positions is set by the TTIWDBDaySource.DayIncrement property. Setting this property to 7 for example can show only a given day of the week for multiple weeks in the TTIWDBWebPlanner.
This TTIWDBDaySource component can also automatically fill the TTIWDBWebPlanner Header captions with the dates displayed. This is done when its property AutoHeaderUpdate is true. The format of the displayed dates is set with the TTIWDBDaySource.DateFormat property.

dmMultiResource
In this mode, multiple resources (identified by the RESOURCEFIELD) can be displayed in different positions in the TTIWDBWebPlanner. The date for which the resources are displayed is set by the TTIWDBDaySource.Day property. The number of resources displayed (and thus also the number of positions that will be in the TTIWDBWebPlanner) is set by TTIWDBDaySource.NumberOfResources. The normal mapping between the resource field in the database and the positions in the TTIWDBWebPlanner is that an integer value of 0 in the resource field maps to position 0 in the TTIWDBWebPlanner. However, the resource identification might be different in particular cases, when used for example as employee or room numbers. Therefore, the TTIWDBDaySource component has 2 events that allows the mapping of the resource field value to a position in the TTIWDBWebPlanner and vice versa. The 2 events are:

OnResourceToPosition(Sender: TObject; Field: TField; var Position: Integer; var Accept: Boolean);

This event is triggered for each appointment or event that can be shown in the TTIWDBWebPlanner. The field values can used to determine the appropriate position for this record and through the Accept parameter, setting it to false can disable loading the particular appointment or event in the TTIWDBWebPlanner. The OnResourceToPosition event will be triggered when the TTIWDBWebPlanner is loading items from the database.

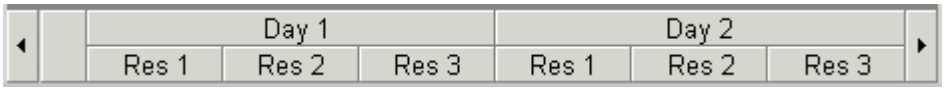
OnPositionToResource(Sender: TObject; Field: TField; Position: Integer);

This event allows to set a field value according to the position in the Fields collection of the record. This event is triggered whenever a TPlannerItem is updated or moved in the TTIWDBWebPlanner and needs to be written back to the database.

Finally, as with the dmMultiDay mode, the TTIWDBDaySource component can automatically update the TTIWDBWebPlanner header captions with the resource names displayed. An event TTIWDBDaySource.OnGetResourceName is used to query this resource name for each position. Each time the TTIWDBWebPlanner reloads items from the database it will query the resource names again to update its header captions.

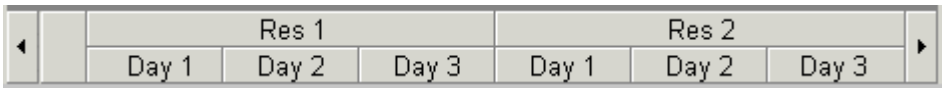
dmMultiDayRes

Displays multiple days for multiple resources. This combines both multi day and multiresource modes. The comments for previous are thus applicable for the dmMultiDayRes mode. The difference is that now the TTIWDBWebPlanner positions and header are setup to show multiple days and multiple resources. In the dmMultiDayRes mode, the number of groups created is equal to the TTIWDBDaySource.NumberOfDays property while each day is divided in TTIWDBDaySource.NumberOfResources positions. The image below makes this clear :



dmMultiResDay

This is the inverse setup of the dmMultiDayRes mode and organises the TTIWDBWebPlanner as in the image below :



TTIWDBMonthSource

The TTIWDBMonthSource shows appointments and events from the database in for a month in the TTIWDBWebPlanner. Mapping of the fields to the TPlannerItem object is identical as in the TTIWDBDaySource component. If multiple resources are used, these can be mapped on multiple positions in the TTIWDBWebPlanner. Custom mapping between resource fields and position index is also possible with the events TTIWDBMonthSource.OnResourceToPosition and TTIWDBMonthSource.OnPositionToResource.

The month that is displayed in the TTIWDBWebPlanner is selected by the properties TTIWDBWebPlanner.Month and TTIWDBWebPlanner.Year properties.

TTIWDBPeriodSource

The TTIWDBPeriodSource is similar to the TTIWDBMonthSource but allows viewing any period between the dates set by its properties TTIWDBPeriodSource.StartDate and TTIWDBPeriodSource.EndDate properties.

TTIWDBHalfDayPeriodSource

The TTIWDBHalfDayPeriodSource is similar to the TTIWDBPeriodSource but allows viewing any period between the dates set by its properties TTIWDBHalfDayPeriodSource.StartDate and TTIWDBHalfDayPeriodSource.EndDate properties with a halfday resolution.

TTIWDBMultiMonthSource

This database interface component puts the TTIWDBWebPlanner into multimonth mode. This means that multiple months are displayed in the TTIWDBWebPlanner in multiple positions. Mapping of the database fields to the TPlannerItem properties is identical as with the TTIWDBDaySource. The TDBMultiMonthSource component does not support a multiresource view though as the positions are used to show multiple months. A mapping to the ResourceField is thus ignored.
The months displayed are selected with these properties :

TTIWDBMultiMonthSource.StartMonth : first month to display in TTIWDBWebPlanner position 0
TTIWDBMultiMonthSource.NumberOfMonths : number of months to display and thus number of positions shown in the TTIWDBWebPlanner
TTIWDBMultiMonthSource.Year : year of the first month displayed.

As some items in multimonth mode can be displayed in 2 or more positions when the start and end date of the appointment or event are in different months, the TTIWDBMultiMonthSource displays part of the appointment in one month while the other part in another month. For such events, the TTIWDBMultiMonthSource creates fixed size and fixed position items. Repositioning of such events should thus happen by editing the start and / or end date through a popup editor. Events that have start and end date in the same month can be sized or moved in the TTIWDBWebPlanner but can of course be edited with a popup editor as well.

TTIWDBTimeLineSource

The TTIWDBTimeLineSource is a database interface for a timeline mode planner. In a timeline mode planner, the days between StartDate and EndDate are displayed with a time resolution set by the property TTIWWebPlanner.Display.DisplayUnit.

TDBDisjunctDaySource

The TDBDisjunctDaySource component displays different selected days only. With such component, it is possible to display each Friday for four weeks for example. This is done by putting the days to view in the TDBDisjunctDaySource.Dates collection. To update the days to be viewed in the planner, set the TDBDisjunctDaySource.Active property to false, fill the Dates collection with dates that should be displayed in the planner and set the TDBDisjunctDaySource.Active to true again.

TDBActiveDaySource

The TDBActiveDaySource is very similar to the TDBDaySource. The difference with the TDBDaySource is that this TDBActiveDaySource only displays the active days in the planner. If Sunday and Saturday are marked as inactive days, the TDBActiveDaySource will skip these days and only display Monday through Friday. If the Day property of TDBActiveDaySource is set to a day that is indicated as an inactive day, it will show only the first active day as first day in the planner.

Using the TDBItemSource.ResourceMap

If the field values that identify resources in the database do not match the indexes of the positions where resources are displayed, the events OnResourceToPosition and OnPositionToResource can be used. Using the TDBItemSource.ResourceMap makes it easier by avoiding to have to use these events and implement there in code the mapping between the resource indexes and the position indexes. The ResourceMap is a collection of TResourceMapItem objects. This object has following properties:

```
property ResourceIndex: Integer;
property PositionIndex: Integer;
property DisplayName: string;
```

The ResourceIndex is the value of the resource in the database, the PositionIndex is the value of the position where this resource should be displayed. The DisplayName is an optional property with which the display name of the resource can be set.

Example:

Suppose that three employees have tasks scheduled in the planner. Employee John has an internal company employee number assigned of 515. Employee Bill has number 264 and employee Richard 177. In the database, these employee numbers are used in each record to identify to who the task is assigned. In the planner, we want to display these employees in the sequence: Bill, Richard and John. The TDBItemSource.ResourceMap is configured with:

```
with DBItemSource.ResourceMap.Add do
begin
  ResourceIndex := 264;
  PositionIndex := 0;
  DisplayName := 'Bill';
end;

with DBItemSource.ResourceMap.Add do
begin
  ResourceIndex := 177;
  PositionIndex := 1;
  DisplayName := 'Richard';
end;

with DBItemSource.ResourceMap.Add do
begin
  ResourceIndex := 515;
  PositionIndex := 2;
  DisplayName := 'John';
end;
```

Writing custom database interface components

This can be done by creating a descendent class from TTIWDBItemSource and overriding a set of methods that perform the interfacing between database and TTIWDBWebPlanner and vice versa. These methods are listed here :

```
procedure SynchDBItems; override;
procedure ReadDBItems; override;
procedure WriteDBItem; override;
procedure ReadDBItem; override;
procedure AddDBItem; override;
procedure GotoDBItem; override;
procedure DeleteDBItem(APlanner: TPlanner); override;
procedure ItemChanged(DBKey:string); override;
procedure Next; override;
procedure Prev; override;
```

The TTIWDBItemSource makes a reference to the connected TTIWDBWebPlanner available through the property TTIWDBItemSource.Planner.

For single item methods such as ReadDBItem, WriteDBItem, GotoDBItem, AddDBItem, DeleteDBItem and ItemChanged, the TTIWDBWebPlanner always puts a reference to the TPlannerItem in the property TTIWDBWebPlanner.Items.DBItem: TPlannerItem.

Thus, when writing a method for ReadDBItem, set all properties read from DB fields to the TPlannerItem provided by TTIWDBItemSource.Planner.Items.DBItem. The same is applies for WriteDBItem which should write the property settings of the TPlannerItem in TDBItemSource.Planner.Items.DBItem to the current database record. In the methods ReadDBItems and SynchDBItems, the full TTIWDBItemSource.Planner.Items collection can be manipulated.