

Combining kbmMW and kbmWABD

for
kbmWABD v. 2.44+
and
kbmMW v. 1.00+

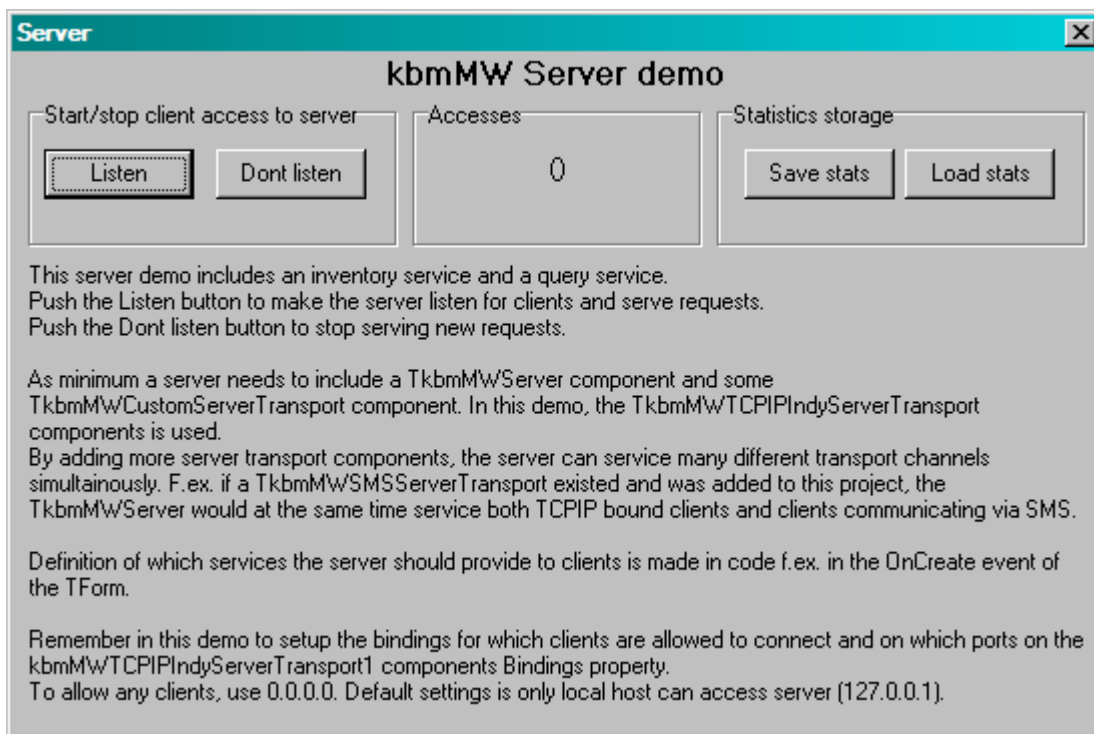
The combination of kbmWABD and kbmMW gives a very powerful web application setup with advanced database handling and separation of business logic from the web server itself. A 4-tier solution.

It is really pretty easy to use kbmMW with kbmWABD. If you have already used kbmWABD with databases, you will already have seen some of the techniques needed to combine it with kbmMW.

The purpose of this whitepaper is to create a relatively simple web application as a front end to the standard kbmMW demo server (in this whitepaper we use the BDE server, but any other compatible demo server can be used).

Preparing for building the web application

First we start the demo application server to have something to work against and make sure to click on Listen to activate the server.

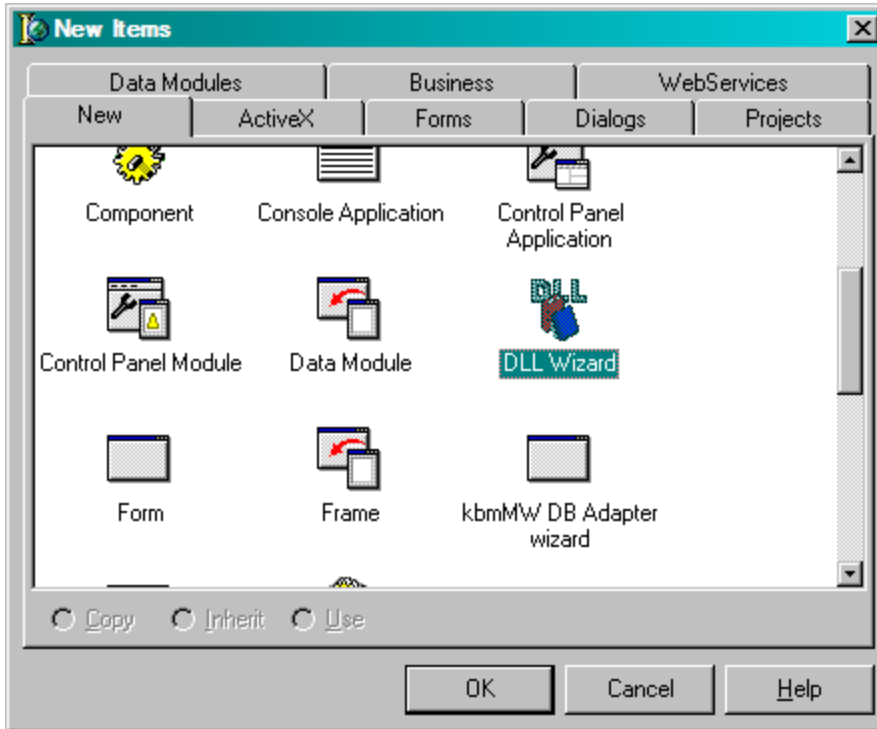


This makes the application server ready for clients to contact it.

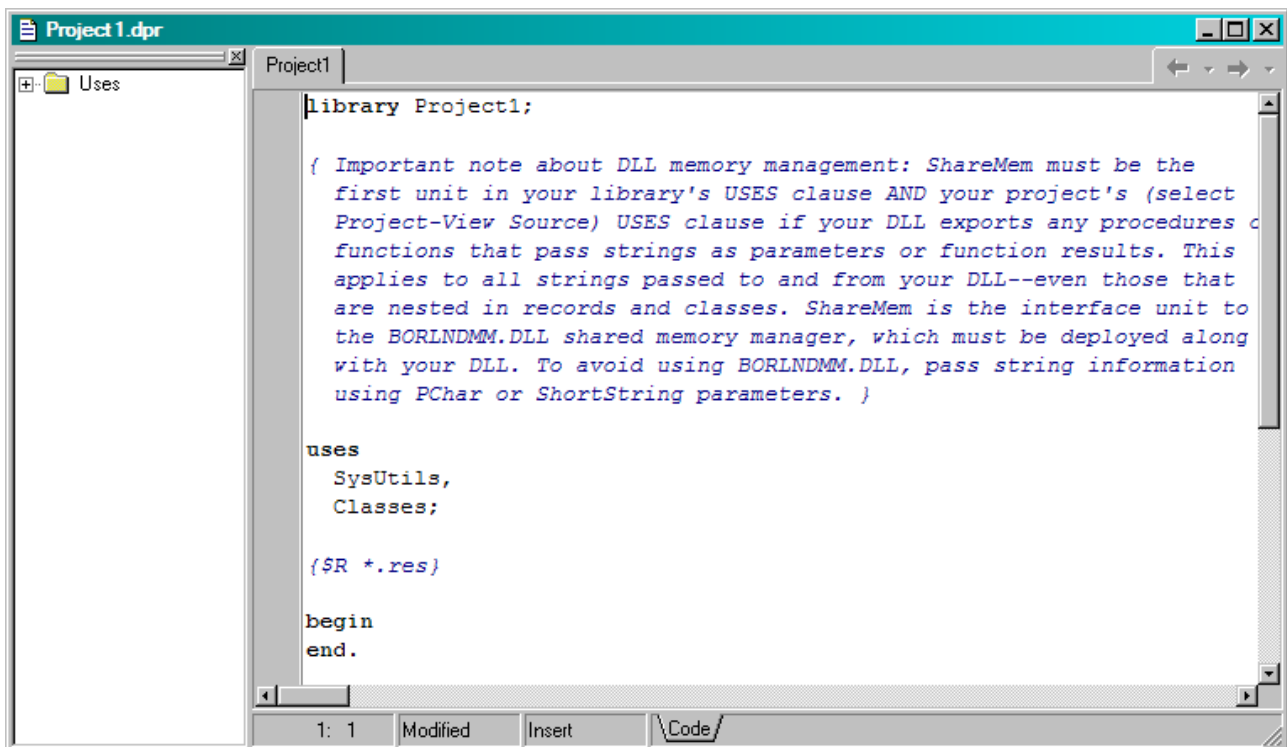
Building a web application the RAD way

Now we must start a new web application using kbmWABD.

In Delphi (version 6 used for this whitepaper) create a new DLL

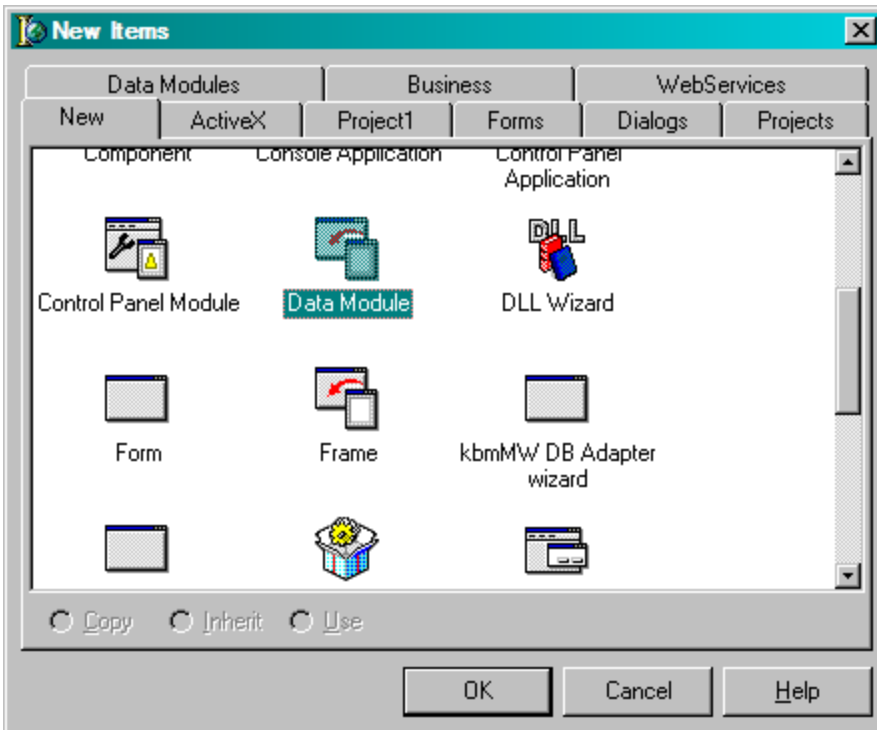


This gives us an empty DLL skeleton.



Then we add two new TDatamodules, one which in the kbmWABD terminology is called the session manager module, and the other which is called the session module. The session module is the one producing the web pages for a client browser. The session manager module is the one controlling the lifetime of the clients connected and lots of other administrative stuff. Read more about it in the document [kbmWABD_GettingStarted.pdf](http://www.components4developers.com/kbmWABD_GettingStarted.pdf) document which can be downloaded at www.components4developers.com from the kbmWABD products page.

Thus select File->New->Datamodule or File->New->Other and select Data Module:

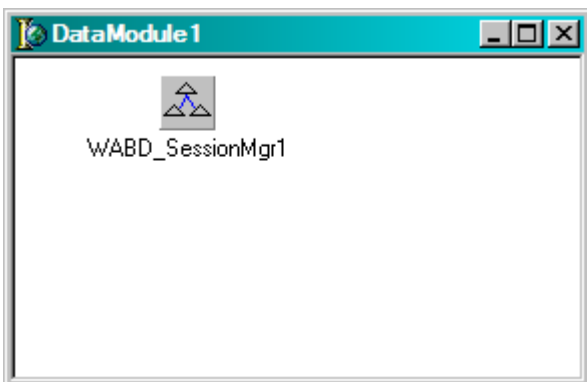


This created an empty datamodule (TDatamodule1) which will serve as our session manager module.

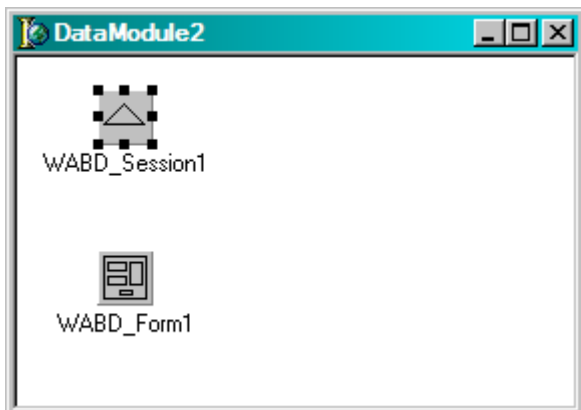
Then add another datamodule the same way. This new one (TDatamodule2) will serve as our session module.

Now your DLL project contains two datamodules.

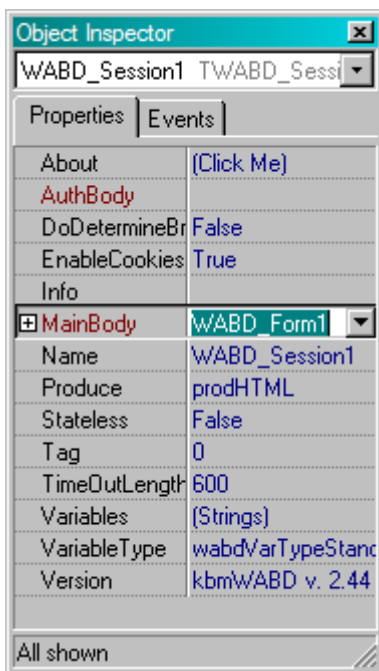
On TDatamodule1 add a TWABD_SessionMgr component:



On TDataModule2 add a TWABD_Session and a TWABD_Form component:



Set the MainBody property of the WABD_Session1 component to WABD_Form1.



Now back to TDataModule1 (the session manager module) where we have to fill in some code in the OnCreateSession and OnDestroySession events of the TWABD_SessionMgr. Before doing that we need to add Datamodule2 to the uses section by File->Use Unit and select Unit2.

Make the OnCreateSession event look like the following:

```
procedure TDataModule1.WABD_SessionMgr1CreateSession(
  var NewSession: TWABD_Session; Request:TWABD_CustomRequest);
begin
  with TDataModule2.Create(nil) do NewSession := WABD_Session1;
end;
```

Make the OnDestroySession event look like the following:

```
procedure TDataModule1.WABD_SessionMgr1DestroySession(
    Session: TWABD_Session);
begin
    Session.Owner.Free;
end;
```

Finally we need to add a little code to the initialization section of Unit1.pas (TDataModule1) - right before the final end. statement:

```
Initialization
    DataModule1 := TDataModule1.Create(nil);
```

These operations should make your complete unit1.pas look like this:

```
unit Unit1;

interface

uses
    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
    WABD_Objects, WABD_Request;

type
    TDataModule1 = class(TDataModule)
        WABD_SessionMgr1: TWABD_SessionMgr;
        procedure WABD_SessionMgr1CreateSession(var NewSession: TWABD_Session;
            Request: TWABD_CustomRequest);
        procedure WABD_SessionMgr1DestroySession(Session: TWABD_Session);
    private
        { Private declarations }
    public
        { Public declarations }
    end;

var
    DataModule1: TDataModule1;

implementation

uses Unit2;

{$R *.DFM}

procedure TDataModule1.WABD_SessionMgr1CreateSession(
    var NewSession: TWABD_Session; Request: TWABD_CustomRequest);
begin
    with TDataModule2.Create(nil) do NewSession := WABD_Session1;
end;

procedure TDataModule1.WABD_SessionMgr1DestroySession(
    Session: TWABD_Session);
begin
    Session.Owner.Free;
end;

initialization
    DataModule1 := TDataModule1.Create(nil);

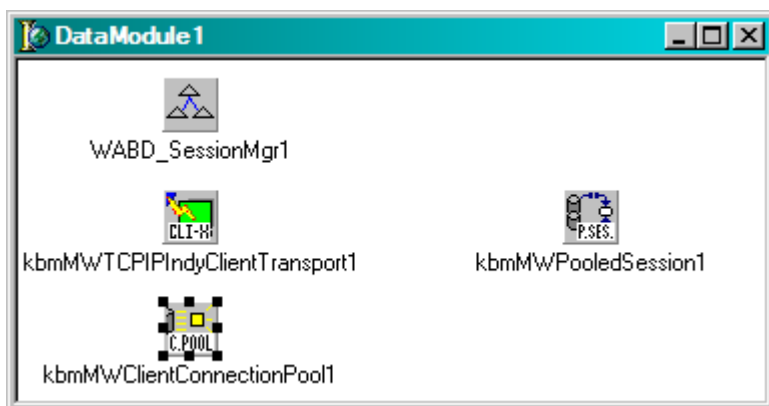
end.
```

This is just about the same setup you would get by using basic.zip as the base for your web application. This is a good point to save your web application.

Getting kbmMW into the picture

Now we need to add a little kbmMW functionality to it. We would like to display the result from a query on a web browser and issue a call to the inventory service to list the different services available for the clients to use.

First add a TkbmMWClientTCPIPIndyTransport, a TkbmMWClientConnectionPool and a TkbmMWPooledSession to the Session manager module:



Connect the kbmMWClientConnectionPool1.Transport property to the kbmMWTCPIPIndyClientTransport1 component.

Connect the kbmMWPooledSession1.ConnectionPool property to the kbmMWClientConnectionPool1 component.

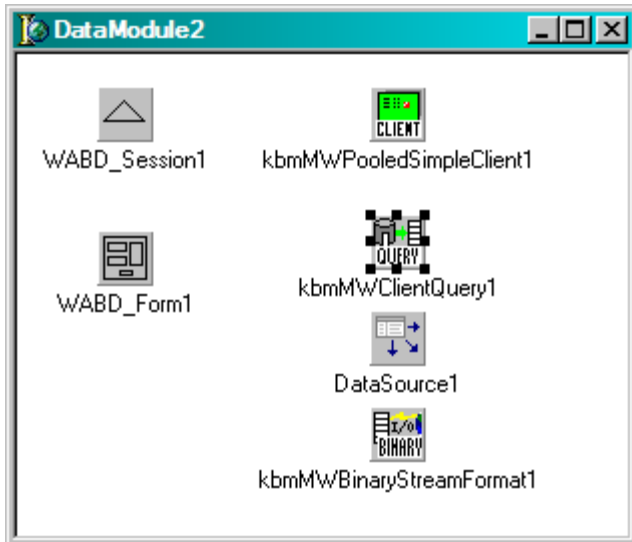
Set the SessionName of the kbmMWPooledSession1 component to something. Eg. DEMO

Set the Host and Port properties of the kbmMWTCPIPIndyClientTransport1 to the IP address and port number of the kbmMW based server. If its running on the same machine as you are developing on, simply write 127.0.0.1 for Host and 3000 for Port.

These steps have now prepared the web application for connection to the application server.

Preparing the session module for kbmMW

Before designing the web pages, we need to prepare the session module by adding a few kbmMW components. Add a TkbmMW Pooled Simple Client, a TkbmMW Client Query, a TkbmMW Binary Stream Format and a TDataSource to the Session module (TDataModule2):



In File->Use Unit select Unit1. This gives access to TDataModule1's components from TDataModule2.

The pooled simple client will be used for getting the inventory from the application server.

The clientquery and binary stream format are going to be used to get the results of a query from the application server.

Set the kbmMWClientQuery1.TransportStreamFormat property to point on kbmMWBinaryStreamFormat1.

Set the kbmMWClientQuery1.SessionName to 'DEMO' (the same name as the TkbmMW Pooled Session.SessionName).

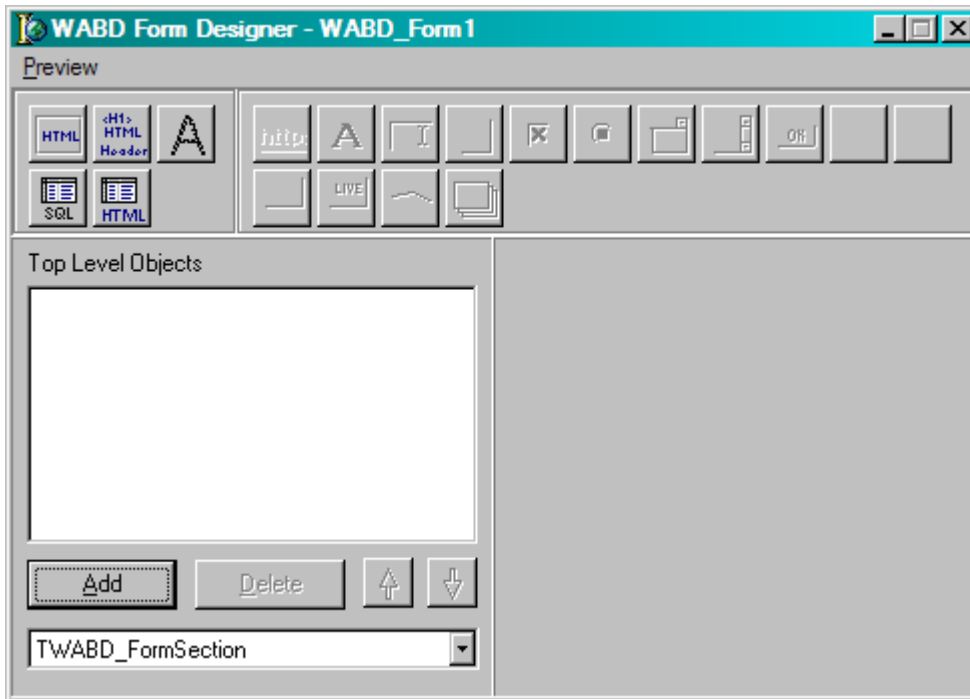
Set the kbmMW Pooled Simple Client1.ConnectionPool to point on the kbmMWClientConnectionPool1 on TDataModule1.

Set DataSource1.Dataset to point on the kbmMWClientQuery1 component.

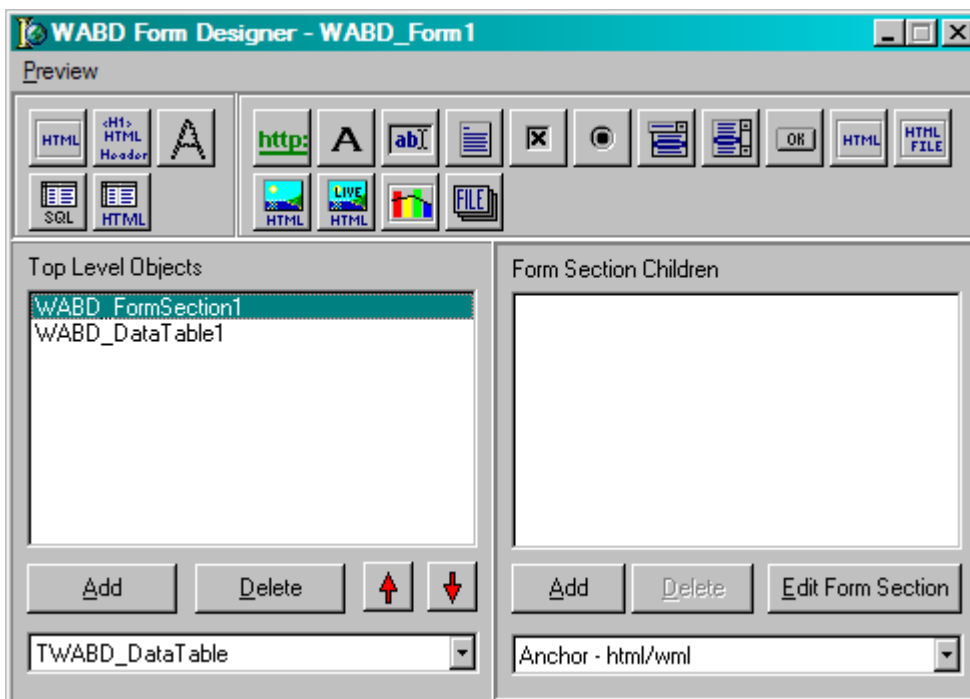
Designing the page layout

A very easy way to design a dynamic web page in kbmWABD in to add display elements to the TWABD_Form. This is done via the form editor.

Double click the TWABD_Form on TDatamodule2:

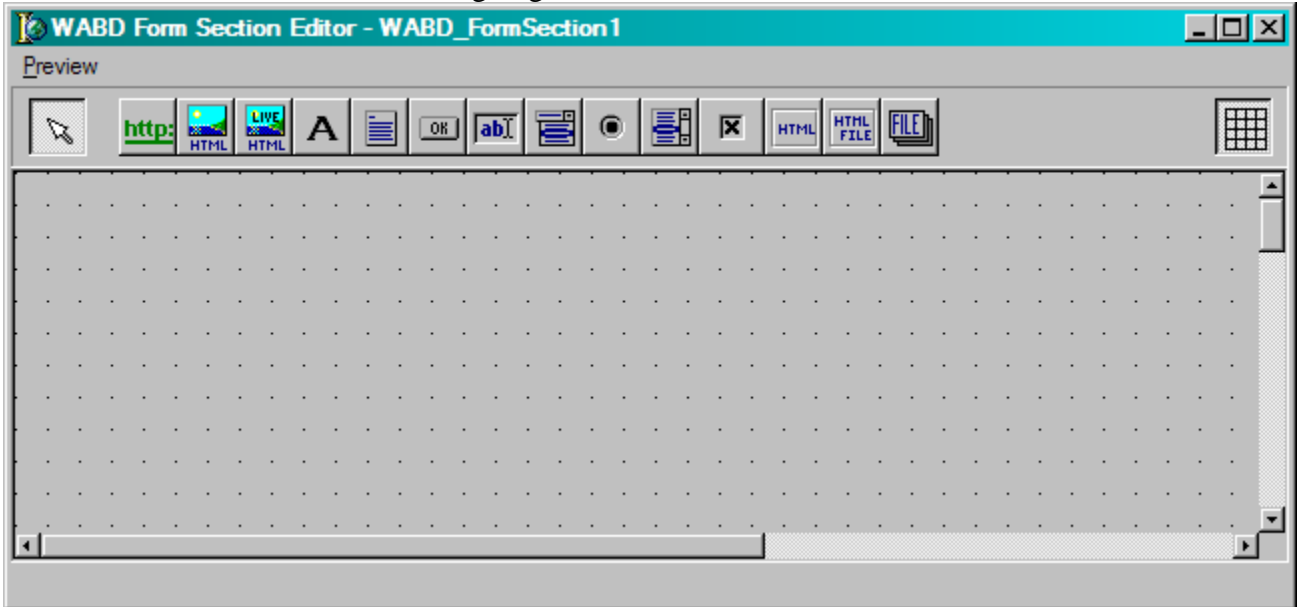


Add a TWABD_FormSection, and a TWABD_DataTable (select them in the combobox and click 'Add'). Select WABD_DataTable1 and set its visible property to false. Then in the Top Level Objects list select the WABD_FormSection you just added:

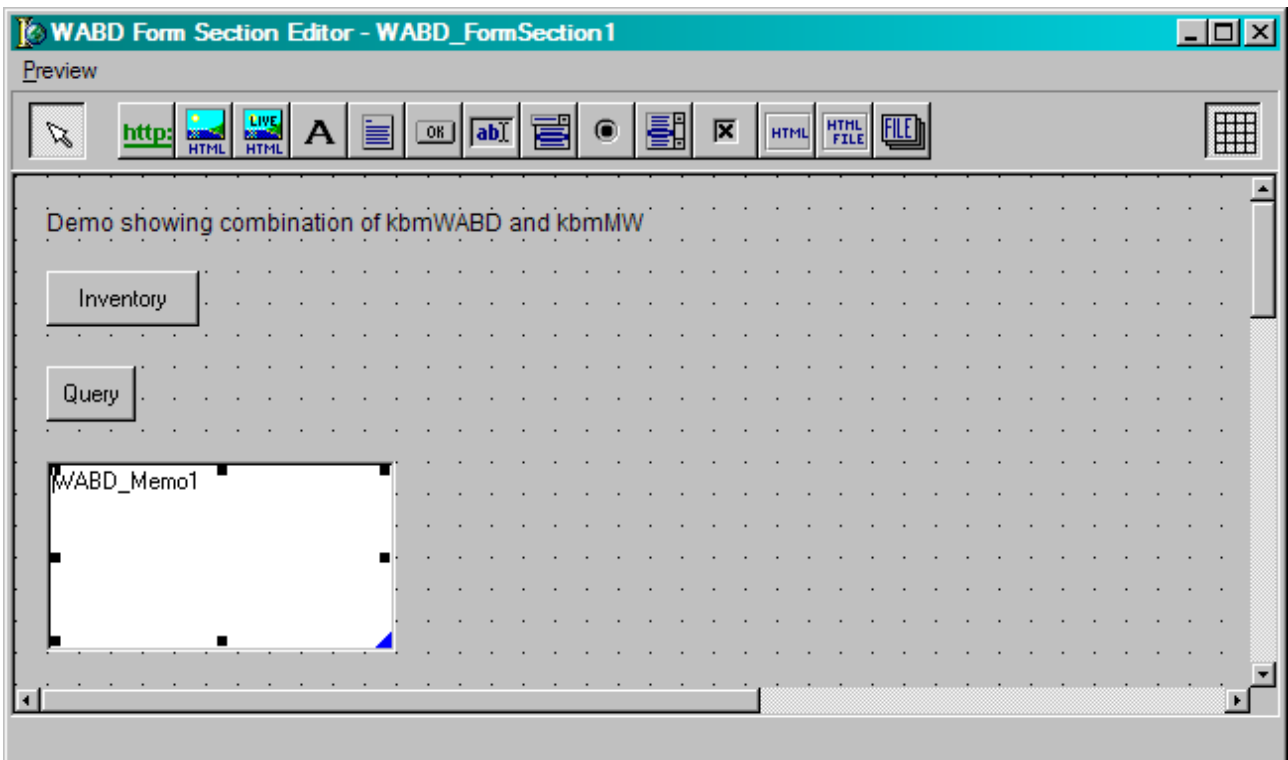


The formsection is a container for other form elements like text boxes, labels, buttons etc. The datatable is a higher level component which resembles a TDBGrid, but only for the web.

Click Edit Form Section to start designing the contents of the formsection container.

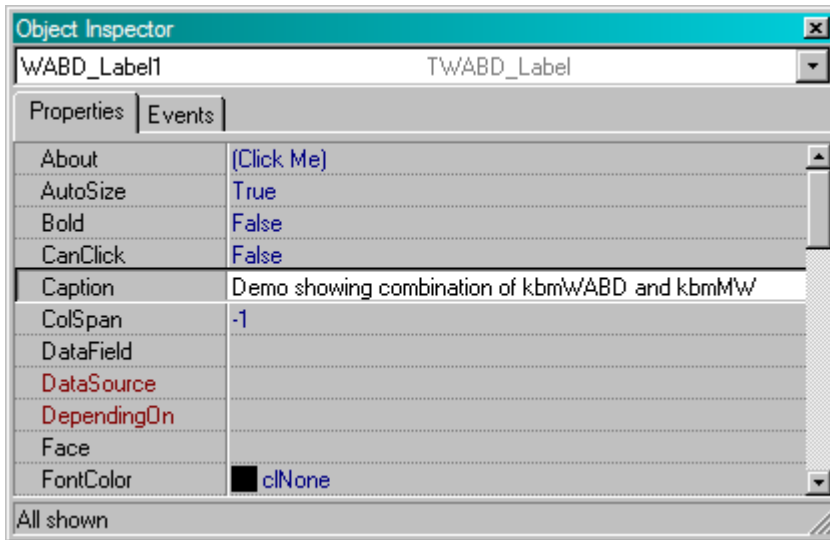


Now we add a TWABD_Label, two TWABD_Button's and a TWABD_Memo:

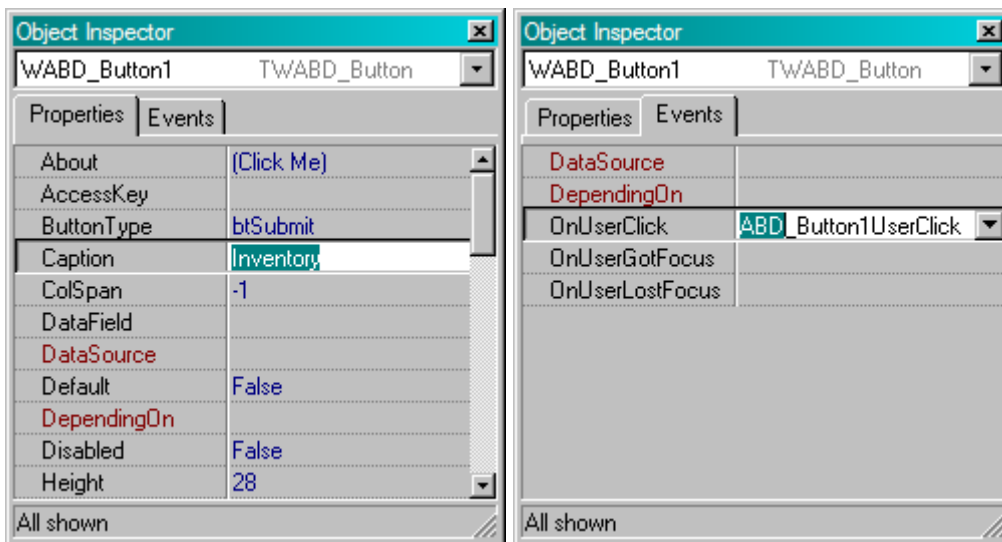


The components can be pulled to the wanted placement, and resized by dragging the little blue cornermark on them when they are selected.

Select the WABD_Label1 to change its caption:



Select WABD_Button1 to change its caption and write some code in its OnUserClick event:

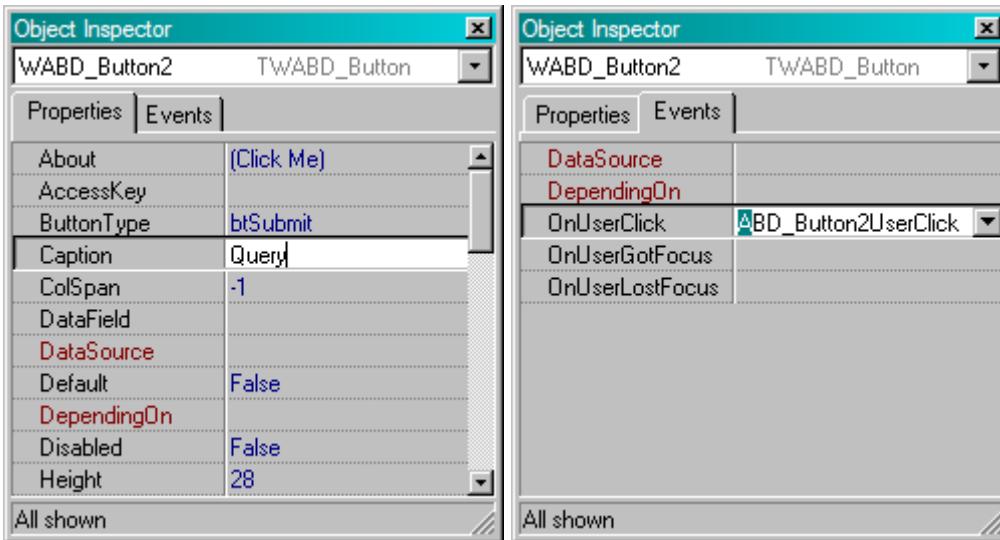


The code in the OnUserClick event is executed the moment the client clicks the Inventory button on the web page. Write the following code in the OnUserClick event:

```
procedure TDataModule2.WABD_Button1UserClick(Sender: TObject);
var
    v:variant;
begin
    v:=kbmMWpooledSimpleClient1.Request('KBMMW_INVENTORY','','',[]);
    WABD_Memo1.Lines.CommaText:=v;
end;
```

This will contact the application server and ask for a list of services. The list will be returned in a comma text format directly usable for presentation by the WABD_Memo component.

Next is to give TWABD_Button2 another caption and write some event code for it:



Make the event handler look like this:

```
procedure TDataModule2.WABD_Button2UserClick(Sender: TObject);
begin
    kbmMWClientQuery1.Query.Text := '@ALL_EVENTS';
    kbmMWClientQuery1.Open;
    WABD_DataTable1.Visible := true;
end;
```

What it does is to request the result of a predefined (named) query on the application server.

The query could contain a complete SQL statement instead if you would like that as long the application server accepts client side SQL.

What is missing now is to display the result of the query. That's why we have added a TWABD_DataTable.

Select the TWABD_DataTable from the TWABD_Form's formeditor and set its DataSource property to point on the DataSource1 component.

Now save it all, and compile it.

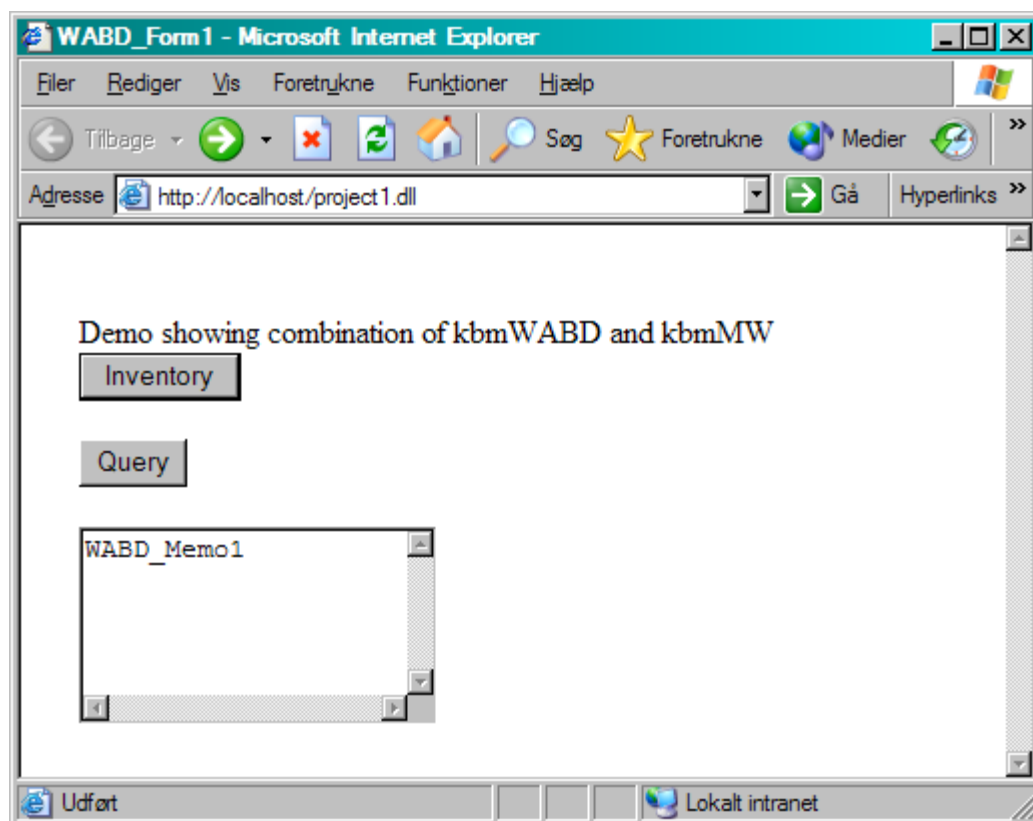
To run it, either use Internet Information Server 4 or newer from Microsoft, or use wwwserver which can be downloaded for free from www.components4developers.com

Running the creation

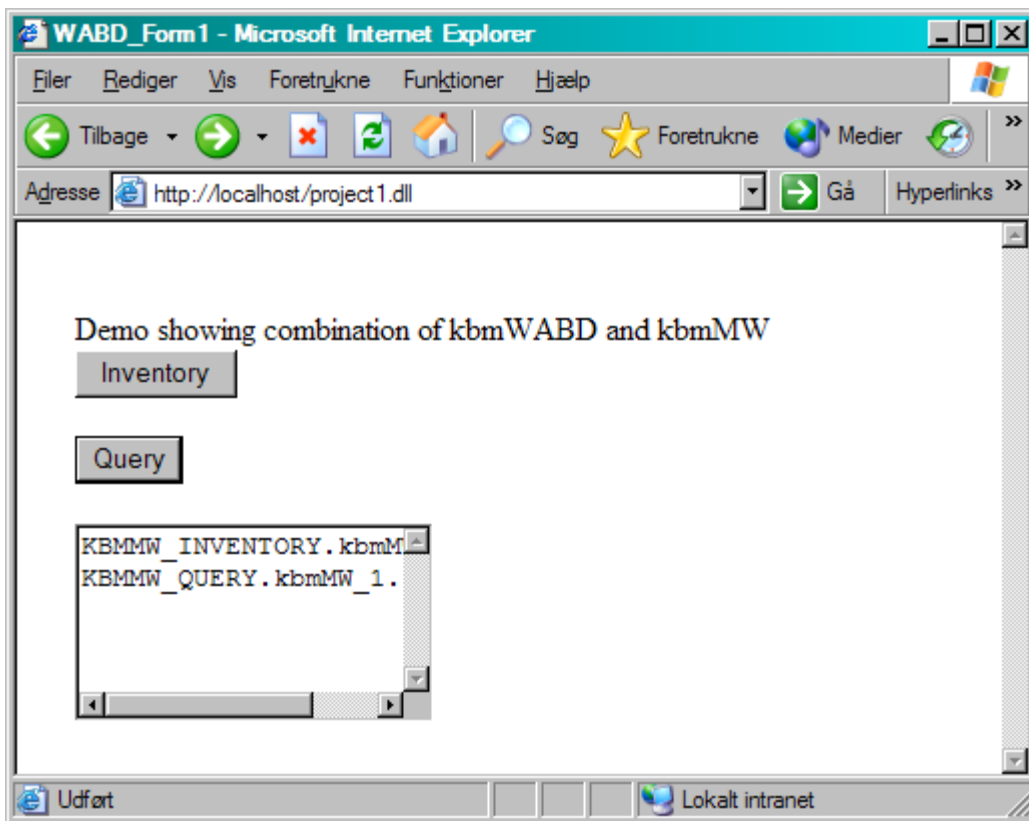
After starting the webserver, access the ISAPI dll. Eg:

<http://localhost/project1.dll>

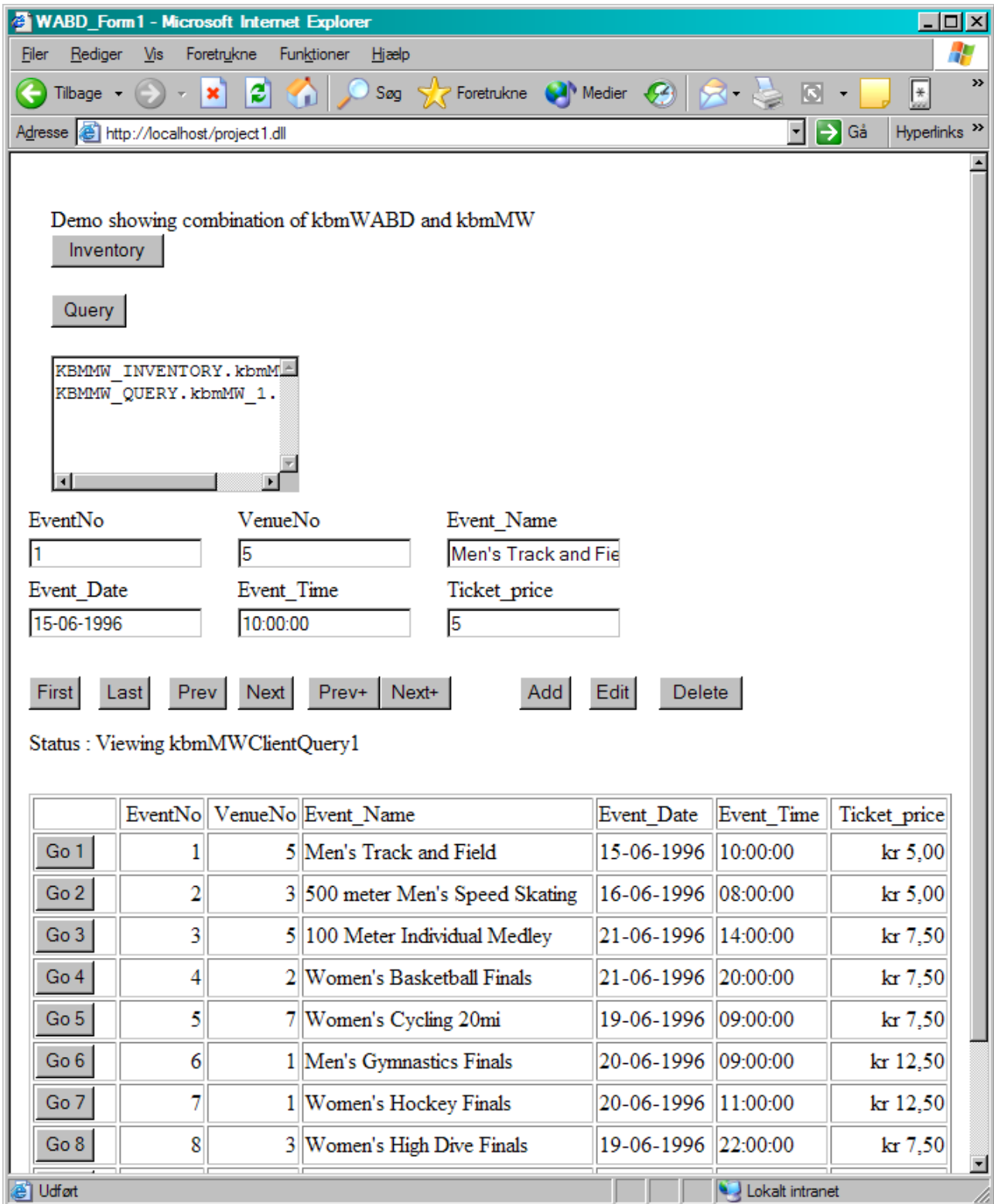
You will get a screen similar to this:



Click the Inventory button, and the web application will make a call to the application server and show the services defined as public services on the server in the WABD_Memo1 text area:



Now click on the Query button to have the query executed on the application server:



Demo showing combination of kbmWABD and kbmMW

Inventory

Query

```
KBMMW_INVENTORY.kbmMW
KBMMW_QUERY.kbmMW_1.
```

EventNo: 1 VenueNo: 5 Event_Name: Men's Track and Field

Event_Date: 15-06-1996 Event_Time: 10:00:00 Ticket_price: 5

First Last Prev Next Prev+ Next+ Add Edit Delete

Status : Viewing kbmMWClientQuery1

	EventNo	VenueNo	Event_Name	Event_Date	Event_Time	Ticket_price
Go 1	1	5	Men's Track and Field	15-06-1996	10:00:00	kr 5,00
Go 2	2	3	500 meter Men's Speed Skating	16-06-1996	08:00:00	kr 5,00
Go 3	3	5	100 Meter Individual Medley	21-06-1996	14:00:00	kr 7,50
Go 4	4	2	Women's Basketball Finals	21-06-1996	20:00:00	kr 7,50
Go 5	5	7	Women's Cycling 20mi	19-06-1996	09:00:00	kr 7,50
Go 6	6	1	Men's Gymnastics Finals	20-06-1996	09:00:00	kr 12,50
Go 7	7	1	Women's Hockey Finals	20-06-1996	11:00:00	kr 12,50
Go 8	8	3	Women's High Dive Finals	19-06-1996	22:00:00	kr 7,50



That concludes the whitepaper of how to combine kbmMW and kbmWABD.

Best regards

Kim Madsen

kbm@components4developers.com