**Technical Document**

**Travel Studio V2 Optimisation**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 1.0 | 17 Apr. 14 | Ravi Kerkar | First draft |
| 1.2 | 10 Jun 14 | Ravi Kerkar | Added more information |
|  |  |  |  |

Authorisation

|  |  |  |
| --- | --- | --- |
| Position | Date | Signature |
|  |  |  |

Table of Contents

[Table of Contents 2](#_Toc390174121)

[1 Using User Controls on page 3](#_Toc390174122)

[1.1 New Pages Designing with user controls.(Very Important) 3](#_Toc390174123)

[1.2 User Control Basic when using on Page. 3](#_Toc390174124)

[1.3 User Control Loading Types 3](#_Toc390174125)

[1.3.1 Static control loading (TSV2 Existing way of loading User controls) 3](#_Toc390174126)

[1.3.2 Dynamic Control Loading 4](#_Toc390174127)

[1.4 User Control Data and Design Loading in TSV2 (Mandatory step when creating User Controls) 6](#_Toc390174128)

[1.4.1 User Control Page\_Load Event 6](#_Toc390174129)

[1.4.2 User Control Page\_Init Event 6](#_Toc390174130)

[1.4.3 New Coding standard Defined 6](#_Toc390174131)

[1.5 Using Dynamic Loading in TSV2 (use wherever Needed) 8](#_Toc390174132)

[1.5.1 Dynamic Control Loading example 8](#_Toc390174133)

[1.6 Existing pages with performance issues because of static loading 10](#_Toc390174134)

[1.6.1 System wide Setting Page(Changed, need Proper Testing) 10](#_Toc390174135)

[1.6.2 Itinerary tab in Booking (Changed, Need proper Testing) 10](#_Toc390174136)

[1.6.3 Other pages that requires changes(Need to do) 11](#_Toc390174137)

[1.6.4 Some Challenges we faced when making the static user control reference to Dynamic 11](#_Toc390174138)

[2 Update panel Refreshing 11](#_Toc390174139)

[2.1 Refreshing Example (Service Requirement Control) 11](#_Toc390174140)

[3 Geotree on Demand changes 12](#_Toc390174141)

[3.1 Existing code: 12](#_Toc390174142)

[3.2 Changes : 12](#_Toc390174143)

[3.3 Area Impacted: 12](#_Toc390174144)

[4 SQL queries Optimisation 12](#_Toc390174145)

[5 Data caching when using multiple User Controls on page. 12](#_Toc390174146)

# Using User Controls on page

## New Pages Designing with user controls.(Very Important)

Design new pages such that we make the minimum use of user controls.

Eg if there are many tabs requires on page then design page using separate pages for each tab. Ie when we click on tab it is loaded with new page and this page can have user controls placed on it.

Avoid page designs where we have only one page and each tab is user control. This will result in many user controls on page.

Create user control only if it is required. User control is normally created when it is used on multiple pages. If it is used only on one page then avoid making it a user control.

## User Control Basic when using on Page.

* Whenever page is loaded with user control page size increases because of user control HTM and view state. This takes more processing time when page is post back and also takes time on network transfer of this page.
* if user control is loaded only when it is needed and unloaded when not needed can save lot of processing and network transfer time.
* If user control is statically loaded then make sure user control Visible property is set to false when not needed. This saves the page rendering.
* When using the dynamic loading defined below, we load control when needed and remove control when not required. This make the page size smaller and it loads faster.

## User Control Loading Types

The whole project is using User Control extensively but almost everywhere we are using static control. Static control that loads with the page, initialize and do the bit of work, which then loads their child controls and so on.

Many of these pages, Itinerary tab in booking particularly are very slow during load time because Static control creation and loading time.

### Static control loading (TSV2 Existing way of loading User controls)

We take one Small Example and see how static controls are loaded.

Eg

PageXYZ.aspx

Page register User control “UCA.ascx”

This control “UCA.ascx” internally register user controls UCAA.ascx

This controls “UCAA.ascx” again internally register UCAAA.ascx

PageXYZ🡪UCA.ascx🡪 UCAA.ascx 🡪 UCAAA.ascx

When page PageXYZ is loaded it loads the Page\_load and page\_Init events of all user controls. So in above case 4 Page\_load events and 4 Page\_init events will get executed.

All Controls code exists in Page\_load event, mostly code that requires only one time execution. Eg Loading of default settings, multilingual text for all text on page, multilingual JS messages, CSS , controls attributes adding etc..

All these things are executed when page is loaded first time. Because it is executed In “not IsPostback” check. This makes the first page load very slow. Also all other post backs will be slow as lot of data is loaded on first page load. And this is maintained in the view state of page.

**This design is ok if we need to load and display this control always on page load. But if we want to load and display this control on click of some control on the page then this design is not good because we simple load control on first page load where it is not needed to load.**

Below shows some events that occur when we post back any control by clicking on button on user control. This is very important to understand because based on this we will know how to code properly controls

|  |
| --- |
| aspx.Page\_PreInit |
| ascx-ascx.Page\_init (user control within user control case) |
| ascx..Page\_init |
| aspx.Page\_Init |
| aspx.Page\_Load |
| ascx.Page\_Load |
| ascx-ascx.Page\_Load (user control within user control case) |
| aspx.Button\_click |
| aspx.PreRender |
| ascx.PreRender |
| ascx.ascx.PreRender (user control within user control case) |

### Dynamic Control Loading

Dynamic control is not registered on page so this control does not exist on page when page is loaded first time. Backend also does not create any instance of this control and so no code of this control is executed. Basically page does not know the existence of this control. This control is added dynamically on click of any other control on page. We need to make sure we load this control on every post back because page will not handle this automatically. Dynamic controls are not handled by the normal page life cycle so we need to code all these.

When click on button to dispaly the control. Following event occurs

aspx.Page\_PreInit

aspx.Page\_Init

aspx.Page\_Load

aspx.Button\_click (Event to load and display the user control.)

ascx-ascx.Page\_init (user control within user control case)

ascx..Page\_init

ascx.Page\_Load

ascx-ascx.Page\_Load (user control within user control case)

Page event order is changed. First page event are executed and then in page button click event user control is loaded and later user control events are executed. Dynamic case control Page\_load is getting executed only when we load the control.

IsPostback flag. This is set to true by the first page load event and later for all page post back this remains true. User control inherit this flag from page so all dynamically loaded user controls will always find this as true. Even for the first time dynamically loaded control this will be true. This is the reason we cannot use this flag in dynamic loaded controls to identify whether the control loaded first time or it is post back after first load.

## User Control Data and Design Loading in TSV2 (Mandatory step when creating User Controls)

This is mandatory when we are working on user control. It shows how we can avoid adding code in Page\_load and page\_init events. Also defines new coding standards for doing this.

### User Control Page\_Load Event

Whether we load the control dynamically or statically we should always try to avoid coding in controls page\_load and Page\_init events. Always create new public method to load the control which can be called from the controls parent page.

if controls displaying is required on page load then we call the control public method to load control in page page\_load event using !IsPostback check.

If control is displayed on click of some button then load the control dynamically and then call this load method.

If user controls contains many user controls registered on it and if we want to show the child user control when we show the parent user control then we call the child user control loaddata method in parent user control Load method. Finally when this page is executing the parent user control loaddata method and child user controls loaddata method will be executed.

### User Control Page\_Init Event

Same steps should be followed here when coding this event.

### New Coding standard Defined

Here we have defined how to avoid writing code in page\_load and page\_init event using new coding standards as given below.

#### Base User Control

Created base class with some mandatory methods to override.

namespace ODLControls

{

public abstract class BaseUserControl : System.Web.UI.UserControl

{

abstract public void LoadData();

abstract public void UserControl\_Load();

public bool isInitialized //Default false

{

get { return ViewState["isInitialized"] == null ? false :

Convert.ToBoolean(ViewState["isInitialized"]); }

set { ViewState["isInitialized"] = value; }

}

protected void initControl() //can add more code in future if needed

{

if (!isInitialized)

{

UserControl\_Load();

isInitialized = true;

}

}

}

}

All user control created in TSV2 should be inherited from above BaseUserControl. This will make mandatory to override 2 Methods.

**Method 1: UserControl\_Load():** this method will be executed only once in control life cycle. So this should have code which is required to be executed only once. Eg page designs, multilingual, default loaded data, Css, etc. User is expected to just write code in this method and no call should be there for this method. Base class will ensure to execute this only once.

**Method 2: LoadData():** this is control loading method. This method can take input which can be set as page properties by the caller page. Caller page can call this method based on different condition. Eg user selects some fields on search page and click on search button. This parameter will be set as properties to control and then call the LoadData() method and this will load control with search result. So in this case “UserControl\_Load” will be executed first time when control is loaded and for all other search button click LoadData() method will be executed. User can change criteria and click on search button any number of times.

isInitialized flag : this is control level flag. This will be false by default and control will make it true when it is loaded first time. This flag is used to execute method UserControl\_Load() only once.

#### User Control Coding using base classes

First Inherit control from base controls

public partial class User\_Controls\_Address : ODLControls.BaseUserControl

Then Comment the Page\_Load code and put in UserControl\_Load() method. When we put code here we won’t use IsPostback flag.

protected void Page\_Load(object sender, EventArgs e)

{

//Code here can go in UserControl\_Load() if it si require only once to execute.

//but if it request every postback to execute then we leave it here

//some case we can put this code in LoadData() control method.

if (!IsPostBack)

{

This code will go in UserControl\_Load() method

}

}

public override void LoadData()

{

this.initControl(); //this line is mandatory

//Code to load control

}

public override void UserControl\_Load()

{

//code from page load !ispostback check

}

## Using Dynamic Loading in TSV2 (use wherever Needed)

If we have some page which is always displaying control place on this page then we can load it statically.

If controls are loaded on clicking of some button or some tab or some other control then we use the dynamic loading.

This is optional Step but user needs to follow this whenever there are many user controls used on a page.

### Loading user control dynamically on page

Following are the steps that we could perform to make the control load Dynamically on page. Below shows UserControlROE as an example.

Before we start, we need to copy the new functions to ODLPage class. These functions will help to load/unload the objects.

1. Remove the @Register markup defined for the control and replace it with @Reference, for example, remove the following line
   1. <%@ Register Src="~/User Controls/UserCtlROE.ascx" TagName="UserCtlROE" TagPrefix="UserCtlROE" %>
2. Add the @Reference tag, for example
   1. [%@Reference Control="~/User Controls/UserCtlROE.ascx" %](mailto:%25@Reference%20%20Control=%22~/User%20Controls/UserCtlROE.ascx%22%20%20%20%25)
3. Remove the Control declaration as we don’t need it anymore and it will be loaded dynamically, note the ID that was there.

<asp:UpdatePanel ID="UpdPnlMain\_ROE" runat="server" UpdateMode="Conditional">

<ContentTemplate>

<asp:Panel ID="PnlROE" runat="server">

<%--<UserCtlROE:UserCtlROE ID="UserCtlROE" runat="server" OnCloseROE="UserCtlROE\_Close" calledfrom="Booking" Visible="false" />

--%>

</asp:Panel>

</ContentTemplate>

</asp:UpdatePanel>

1. Make sure we have a Place Holder object (Panel Or PlaceHolder) defined, at the place where we are removing the User Control. See PnlROE that I have defined. This is where we will add our user control dynamically.
2. In the Code behind, create a Class Level variable, of the type that we just added.
   1. User\_Controls\_UserCtlROE UserCtlROE;
   2. Its better to use the name it had as Static Control, for example, UserCtlROE. This will make sure that we don’t have to make changes everywhere as those will be using this variable.
3. Now, find that event, a button click for example, when you want to display the control. In this case ROE\_OnClick was already setup. Call the new function addDynamicControl. For example,

UserCtlROE = (User\_Controls\_UserCtlROE)addDynamicControl("~/User Controls/UserCtlROE.ascx", "roeControl", "PnlROE");

The parameters are, Control Path, ControlID, and Parent ID.

This function will load the control and add it to the Parent control. Make sure the controlID is unique.

Now, in this function, you can treat UserCtlROE normal control, so LoadData, visible=true and everything else will be exactly as it was written there.

Since the Dynamic Controls have to reloaded everytime postbacks are issued, I have coded a reloading function in the ODLPage. However, the class level variables are never stored, so do the code in Page\_Load to set the variable to the control, if it was already loaded. For example,

if (IsPostBack)

{

UserCtlROE = FindDynamicControl ("roeControl") as User\_Controls\_UserCtlROE;

//update memeber variable for the dynamic control

if (UserCtlROE!= null)

UserCtlROE.CloseROE += new EventHandler(UserCtlROE\_Close);

Here pass the sameID that we given at the time of AddDynamicControl function.

ALo note FIndDynamicControl() can be also called from on OnPreLoad event. Some cases controls are accessed in this event so it is important to load controls before they are used.

protected override void OnPreLoad(EventArgs e)

{

base.OnPreLoad(e);

usrctlBookingSetting = (User\_Controls\_BookingSetting)FindDynamicControl("usrctlBookingSetting");

if (hdnSelectedSettingID.Value == "2")

usrctlBookingSetting.CreateBookingSettingControls();

}

1. Set any Event handler again.

Now, when a user closes the control, we need to unload it. Make sure we have an event that will be called at the close of the dialogue. Its very important to unload the control, otherwise the function “addDynamicControl” will raise error when someone click the ROE button again.Call RemoveDynamicControl

protected void UserCtlROE\_Close(object sender, EventArgs e)

{

string href = "";

UserCtlROE.Visible = false;

removeDynamicControl(UserCtlROE.ID, myLC.Parent.ID);

mpe\_ROE.Hide();

if (UserCtlROE.BookingChanged) //check if anything changed then refresh;

{

href="~/Pages/Booking/Itinerary.aspx?BookingRefNo="+Request.QueryString["BookingRefNo"].ToString();

Response.Redirect(href);

}

}

Please note that I am calling removeDynamicControl with .ID, though its not required and you could simply pass the ID using quotes like “roeControl”.

Also, I modified this function and ROEControl to give me BookingChanged parameter. Based on this I am refreshing the page, if required.

Otherwise it’s the same function, apart from removeDynamicContol call.

### Loading Dynamically user control on user control.

This is not yet implemented. We found this case recently.

User control is registered on page and it is displayed on page load. This user control internally registers many user controls and these are displayed on click of some option on main user control. Since the main user control is loaded on page on page load, all other ocntrols also are loaded on page load.

In case also we need to apply the same logic of dunamic loading user control on user control. For these we need to do few changes.

Prashant from TSV2 will be working on this. Document will be updated with more details once he is done with it.

## Existing pages with performance issues because of static loading

We have listed down some pages where we face performance issues because of user control loaded statically. These pages are with many user controls and all loaded statically.

### System wide Setting Page(Changed, need Proper Testing)

There are many controls on this pages and all these are loaded statically. These controls are displayed only when user click in left panel some icon.

Issues on this page

1. Takes time to load the page because all user controls are loaded on first page load. also all page loads lot of code exist which was executed on first page load.
2. After the page is post back then also it is taking time because first page load already loaded data on this control which makes the page heavy.
3. When we click on any icon on this page it loads user control. This user control loading also takes time because page size is already heavy and it is executing all page load events on any post back on page.

We fixed this issues

1. Removing all page\_load code and using the new way of coding using base user control. This shows some improvement.
2. Later we applied the dynamic loading this showed good improvement. It laods the page faster and also all postback are faster. On click of any icon new control loaded are also faster. It showed around 50 to 60% improvement in all loading on this page.

### Itinerary tab in Booking (Changed, Need proper Testing)

When this page is loaded around 90 user controls Page\_load events are getting executed. This is very big number because page should not have more then 15/20 user controls executed when it is loaded.

Page analysis we found that 25 controls are registered directly on this page. And these 25 controls are again having child controls registered and child controls registering more controls. So total of all tehse comes to around 90 Controls.

Main challenge here is most of these controls are used on other pages, other then itinerary page. So if we change something then we need to handle around 150 pages.

Our initial test showed that improvement will be good if we use dynamic so we did dynamic implementation here.

We removed page load code of all controls and used new design using base user control. This showed improvement.

Later we did dynamic loading of control only for controls placed on the itinerary and it showed good improvement, around 50/60% difference. Added control only when user click on icons on top and removed this control when user click on close option.

### Some Challenges we faced when making the static user control reference to Dynamic

1. Some cases controls properties are assigned before it is required to display. ie tis controls are displayed on click of some option but properties of this control are loaded in first page load by calling some API method.

We want to make this control dynamic because it is required only if user clicks some option. We cannot make it dynamic directly because it is access to assigned default data on page load of page.

In such case we will have to keep the data it is assigned in view state and remove the control property assignment code. Then load this control dynamically and assign this view state data to its property. Require to keep data in view state to avoid API method call again.

Note some cases data is already available in view state and also it is assigned to control perty. In this case we use the existing view state.

1. Dynamic loading we found issue when same control used twice on page. Eg Control A uses control C as child control. And Control B also uses Control C as child control. Control A and Control B are registered on page. In this case we found that javascripts used on control C does not work properly. It gives wrong result. When control C is displayed from control B and we click on some option which execute javascript (with value passed as parameter) then it execute the control C on A Javascript with default parameter values.

This issue is not there when it is use with static reference.

It took us some time to find this reason. We found only one case with such reference. Solution is not found for this because it is very rare case. Solution may require some extra coding and it was not required in this case because this control was not causing any performance issue.

This note will be useful in future if such cases are found then we know the reason and time will not be wasted to debug and find the reason.

# Page/user control with many dropdowns

If we have many dropdown placed on page and each dropdown is loaded with some data from database then it is advisable to use odl controls which loads data on demand. Ie on click of dropdown. this dropdown will show the selected value when page loads. Only this value is fetched and loaded in this dropdown on page load.

All dropdowns are not loaded on page load so page load will be faster. Also network data transfer will be faster because page size will be smaller.

# Update panel Refreshing

Some of these places are not using the Update Panel correctly, or refreshing everytime user clicks on the tab. For example, Service Requirement Control, which appears when user clicks on Change Detail button. When you click on Allocation Tab, it loads the data, not only first time but everytime. We need to make sure we do not refresh the data but it’s already been uploaded.

## Refreshing Example (Service Requirement Control)

In this control, whenever you click on any tab, the screen seems to be refreshed, takes about 1-2 seconds though but reflects badly as it appears to be taking longer time and refreshing for no reason. This was simply achieved by tracking if the tab is already loaded. For this, I used viewstate to track it.

Example code for allocation tab

protected void TabChange(object sender, EventArgs e)

{

.

.

.

.

.

.

case "Allocations":

//Allocations

tabLoaded=!String.IsNullOrEmpty((String)ViewState["tabLoadedTabNumber" + "1"]);

SetCSSToActiveTab(tabAllocations.ID);

if (!tabLoaded)

{

usrctlServiceRequirementAllocations.AllocationsBookedServiceId = GetSetBookedService.BookedServiceID;

usrctlServiceRequirementAllocations.AllocationsBookingRefNumber = BookingReferenceNumber;

usrctlServiceRequirementAllocations.BindAllocations();

updplnAllocations.DataBind();

hdnServiceAllocationsOverAllocated.Value = usrctlServiceRequirementAllocations.IsOverAllocated;

BtnSave.OnClientClick = "return CheckAccessAllocations('" + hdnServiceAllocationsOverAllocated.Value + "' );";

}

ViewState["tabLoadedTabNumber"+"1"]="yes";

mvMultiView.ActiveViewIndex = 1;

break;

Obviously, we need to set the view state to null, if we want to refresh the tab due to any change.

## System wide setting example

System wide setting we have left panel which is fix. And on click of this left panel we load the right panel.

Whenever we click on left panel it was refreshing both panels left and right panel. Because of this both panel code was getting executed on each click.

Removing the left panel refresh code showed the performance improvement on each click. This also fixed the scrolling issue. It maintained the scroll position. Before it was not maintaining the scroll position and always takes user to first item.

Refresh panel only when it is needed.

# Geotree on Demand changes

## Existing code:

full geotree was loaded when user click on Geotree icon.

## Changes :

This is now changed to load First level regions when user click on geotree icon.

And when user click on geotree + sign it will load its first level childs. Same for all nodes.

## Area Impacted:

Geotree Maintenance, All UIs where tree view control is used.

# SQL queries Optimisation

Changes some areas in TS V2 to avoid multiple queries to DB and make only one query.

Service maintenance when we go to add new option that time one query was fired to get all exsiting service option/extras/facilities and later looping on this in code and fire query for each option to get more details. This multiple queries changed to single query by sending all optionids in one query.

# Data caching when using multiple User Controls on page.

TSV2 we found some cases where multiple user controls call the same API method with same parameter to get Data from database. Some cases data returned by this call is huge.

Eg. GetBookingInfo called from Itinerary page. When we open this page we find around 5 calls to this SP. All returning the same information. This is because the same API is called from different controls on this page and all these controls are loaded on click of Itinerary tab.

If call is executed from different controls then we can use caching using http context. We cache the first call response in this http context and other user controls can use this cached data. Caching can be done at the proxy object level without changing the existing call flow.

Http context maintain the data for the page post back life cycle. Ie when any post back is done http context instance is created and it is destroyed when the page post back is completed. So each post back will creates new instance of Http context and it will be lost when postback is complete.

Some cases we need to take care of Parameters passed to the request. if parameters passed to this control is different from different user control then we can avoid this call. If call is heavy and take time to load from DB then we can think of caching multiple calls request and response. And returned response after keeping check on parameter.

Use this logic only if it makes good difference in loading time.