

# RoadMap BSP Element – Step by Step Example



## Applies to:

BSP (Business Server Pages), ABAP Workbench – SE80. For more information, visit the [ABAP homepage](#).

## Summary

As we know from Web Dynpro ABAP [1], the BSP [2] technology has also a UI element (named RoadMap) to show the steps in a wizard. In Web Dynpro ABAP, this UI element has an action property to navigate from a step to another step by mouse-click. In BSP, the RoadMap UI element has no such action, being able to navigate just with the back and forward buttons.

In this article, we are going to show not only the way we can create and use a RoadMap UI element, but also the way we can create an action for this BSP element. To create the action, we will use the BSP element FINDANDREPLACE and the BSP element HTMLBEVENT (from the BSP extension technology).

For this purpose, we are going to create a step by step BSP application.

**Authors:** Dr. Cristea Ana Daniela, Memmer Manfred, Thilo Klopfer

**Company:** CELLENT AG

**Created on:** 19<sup>th</sup> February 2011

## Author Bio



Dr. Cristea Ana Daniela has worked as an Assistant Professor at the Polytechnic University of Timișoara (Romania) for 4 years. The subject of her doctoral dissertation was “Contribution to creating and developing a new SAP authorization concept based on RFID”. She is the author and coauthor of 4 books (Web Dynpro ABAP for Practitioners, Computer Programming, Computer Utilization, Interface and Peripheral) and many articles. Currently, she is working as a SAP consultant. She was named in the “Who’s Who in Science and Engineering 2011 / 2012”.



Memmer Manfred is a SAP Consultant and has more the 18 years IT experience, therefrom more than 11 years with SAP. He has knowledge in APAP, Web Dynpro ABAP and BSP programming, XML interfaces, creating Adobe forms with the Adobe Live Cycle Designer, developing of functions in the BRF plus (Business Rule Framework plus), developing and customizing of SAP SRM 5.0-7.0 for customers and so on.

## Table of Contents

Roadmap BSP Element – Introduction .....	3
Creating the Step by Step Application .....	4
Creating the View – Roadmap.htm .....	4
Creating the Controller Class - zcl_cont .....	7
Creating the Page Fragments .....	10
Running the Application .....	11
Conclusions .....	11
Related Content .....	12
Disclaimer and Liability Notice .....	13

## Roadmap BSP Element – Introduction

This BSP element is part of the phtmlb extension. SAP offers many BSP elements divided in extensions (e.g. phtmlb, bsp, htmlb, xhtmlb, etc.). In each BSP extension, we have many UI elements that can be used in the standard mode or can be extended to create our own BSP elements.

To see the BSP extensions offered by SAP, we can use the Object Navigator (transaction SE80) – Tag Browser (Fig.1).

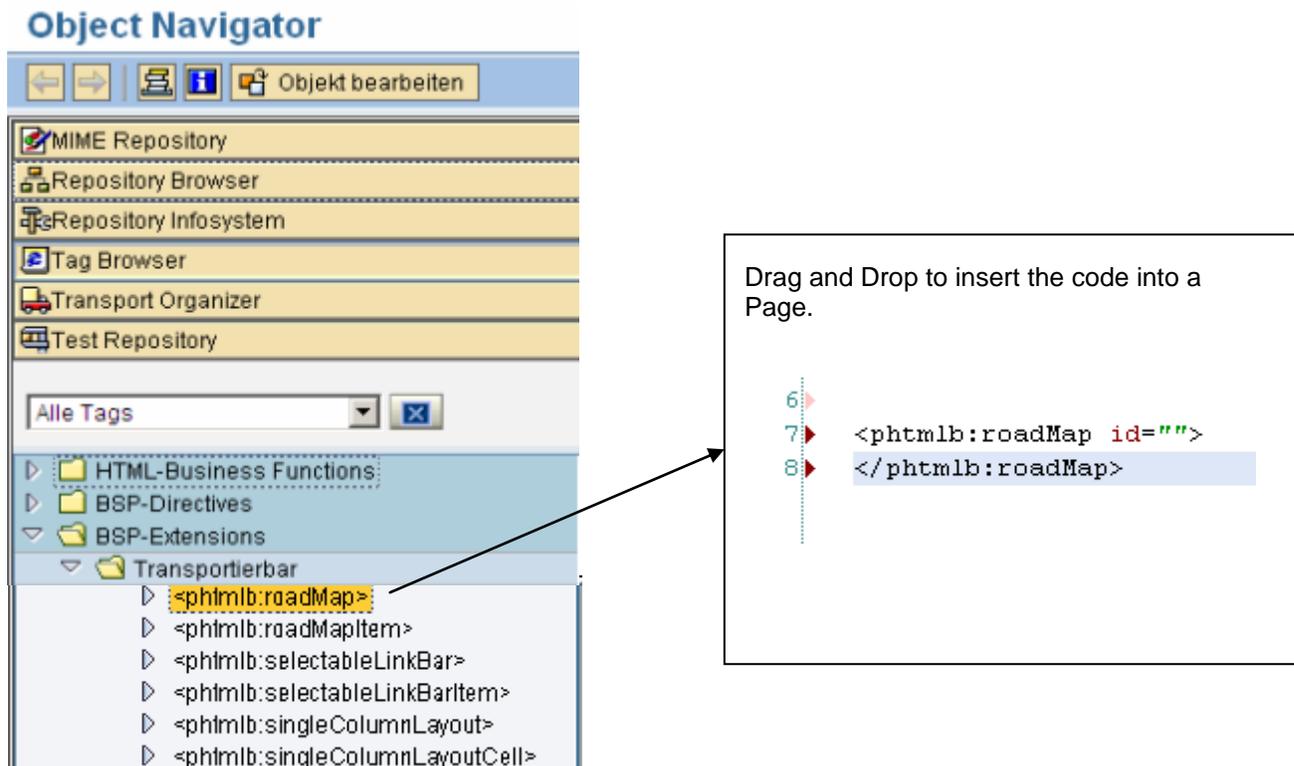


Fig.1: TagBrowser - <phtmlb:roadMap>

To insert this UI element into a view, we can use drag and drop. Each RoadMap Element can have a number of items – steps (<phtmlb:roadMap>).

Each BSP element is implemented through an ABAP class and has some attributes. By double clicking on the element name, we navigate forward to the BSP element implementation (Fig. 2).

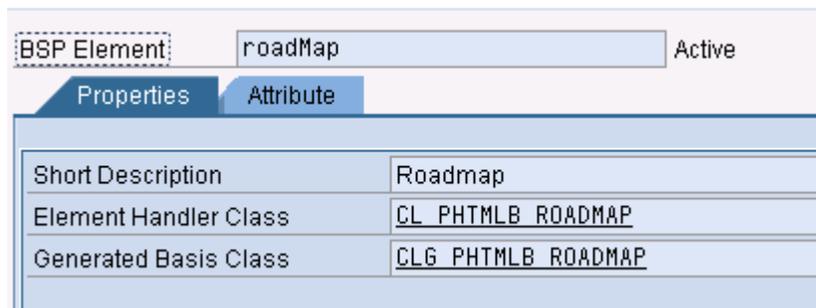


Fig.2: The BSP element implementation

To find more information about the UI element properties, we can use the Documentation Button.



## Creating the Step by Step Application

As we have mentioned above, we are going to create a step by step BSP application that has the structure presented in Fig. 3. We will create an application named “ztest\_roadmap” with a controller named “controller.do”, a view named “roadmap.htm” and 3 page fragments. Into the view “roadmap.htm”, we will create the main screen with the RoadMap UI element with three items. The three page fragments will be displayed for the end user every time he/she clicks one the RoadMap item (inclusive the RoadMap step description). For the “controller.do”, we will create an ABAP class named “zcl\_cont” that inherits from the class CL\_BSP\_CONTROLLER2.

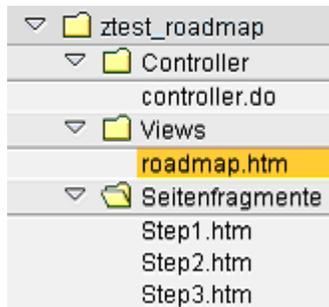


Fig. 3: The BSP application structure

### Creating the View – Roadmap.htm

This View has the properties presented in Fig. 4. As we can see, into the view we can specify the controller class we want to use. Into the MVC (Model View Controller) pattern, the controller is used (principal) to create the interface between the model and the view; and it is used to handle the navigation. Because our little application needs no model class, we will use only a controller class. In the BSP technology, a controller is created from a BSP controller and a controller class.

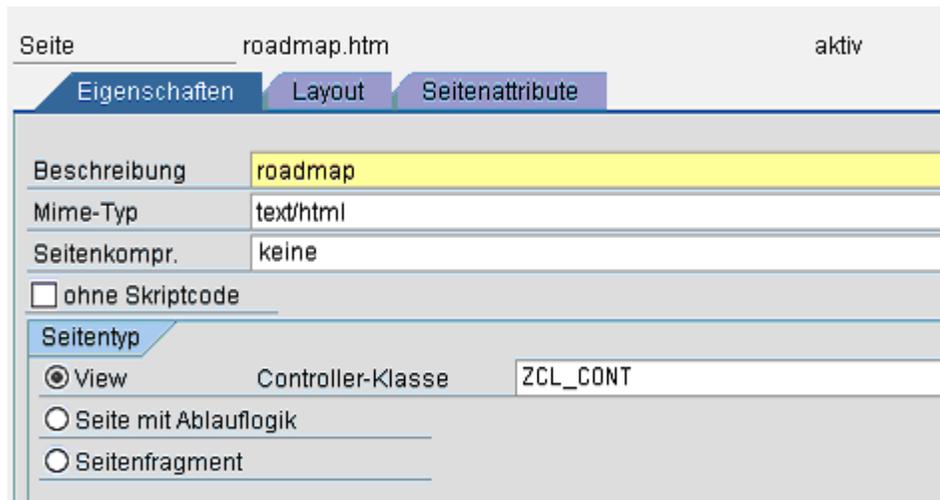


Fig. 4: View properties

Into the view, we implement the code presented in Listing 1.

```
<%@page language="abap" %>
<%@extension name="htmlb" prefix="htmlb" %>
<%@extension name="phtmlb" prefix="phtmlb" %>
<%@ extension name="bsp" prefix="bsp" %>
<% DATA: l_controller type string,
          l_find1 type string,
          l_find2 type string,
          l_find3 type string,
          l_replace1 type string,
```

```

        l_replace2 type string,
        l_replace3 type string.
    l_controller = controller->get_id( 'roadMap' ).
    CONCATENATE 'id="' l_controller '-itm-0"' INTO l_find1.
    CONCATENATE 'id="' l_controller '-itm-1"' INTO l_find2.
    CONCATENATE 'id="' l_controller '-itm-2"' INTO l_find3.
    CONCATENATE l_find1
`style="white-space:nowrap;cursor:pointer;" onclick="return DoMove('STEP1');"`
        into l_replace1.
    CONCATENATE l_find2
`style="white-space:nowrap;cursor:pointer;" onclick="return DoMove('STEP2');"`
        into l_replace2.
    CONCATENATE l_find3
`style="white-space:nowrap;cursor:pointer;" onclick="return DoMove('STEP3');"`
        into l_replace3.%)
<htmlb:content design="design2003" >
  <htmlb:page title = "roadmap" >
  <htmlb:form>
  <bsp:htmlbEvent id          = "MoveStep"
                  name        = "DoMove"
                  onClick      = "MoveEvent"
                  event_defined = "MoveStep"
                  p1           = "EventID" />
    <bsp:findAndReplace find1='<%=l_find1%>' replace1='<%=l_replace1%>'
                      find2='<%=l_find2%>' replace2='<%=l_replace2%>'
                      find3='<%=l_find3%>' replace3='<%=l_replace3%>'>
      <phtmlb:roadMap id    = "roadMap"
                    items = "<%=controller->roadmap_items %>" >
        </phtmlb:roadMap>
    </bsp:findAndReplace>
  </htmlb:form>
  <% if controller->show_step = 'STEP1'. %>
    <%@ include file = "Step1.htm" %>
  <% elseif controller->show_step = 'STEP2'. %>
    <%@ include file = "Step2.htm" %>
  <% elseif controller->show_step = 'STEP3'. %>
    <%@ include file = "Step3.htm" %>
  <% endif. %>
  </htmlb:page>
</htmlb:content>

```

Listing 1. View roadMap.htm

We will dynamically set the steps of the RoadMap UI element, by using the controller attribute (roadmap\_items):

```
<phtmlb:roadMap id = "roadMap"
            items = "<%=controller->roadmap_items %>" >
```

If we run a BSP application and check the source code, we can see the generated html code (Fig. 5).

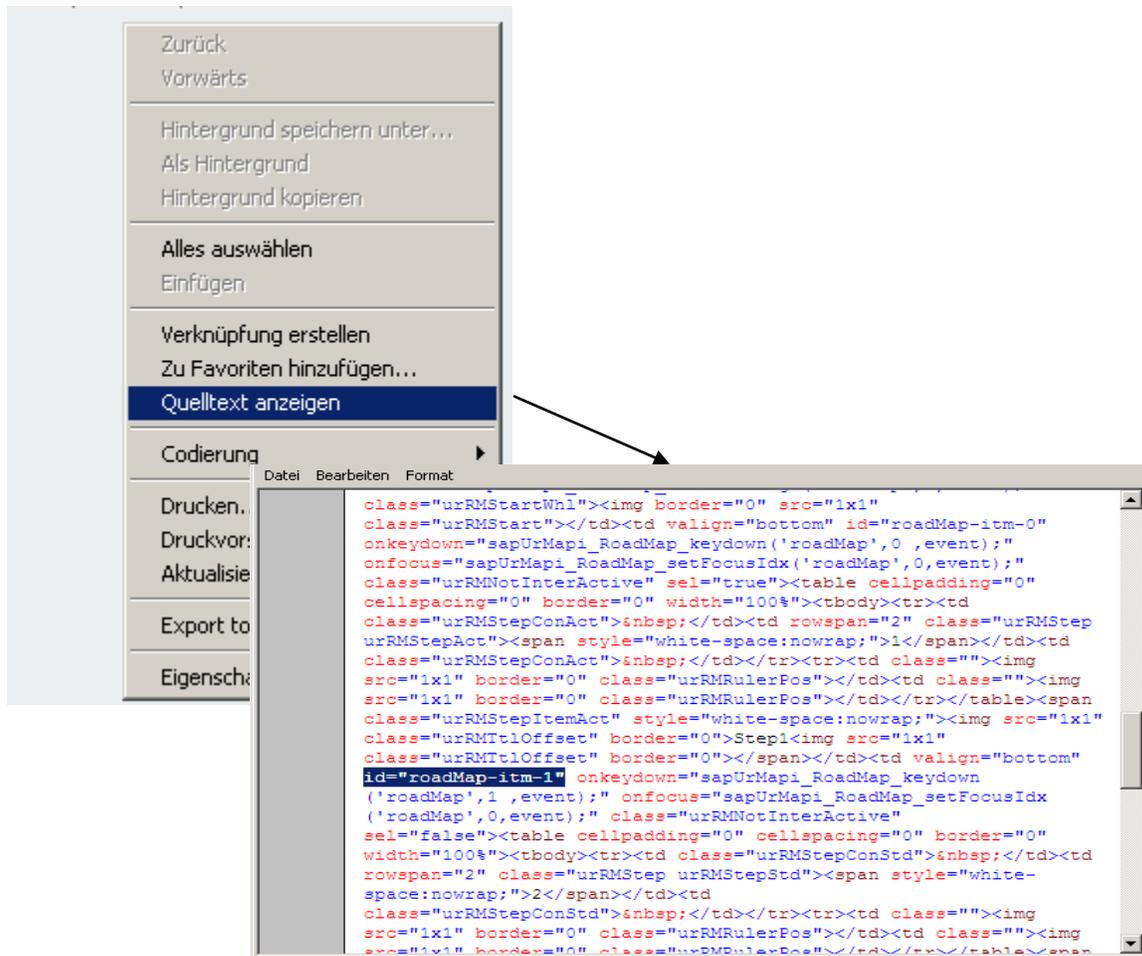


Fig. 5 Check the source code

To create the actions able to react when the end user clicks on a RoadMap step, we will use the BSP element FINDANDREPLACE. In this way, we are going to replace:

```
id="roadMap-itm-0"
id="roadMap-itm-1"
id="roadMap-itm-2"
with:
id="roadMap-itm-0" style="white-space:nowrap;cursor:pointer;"
onclick="return DoMove('STEP1');"
id="roadMap-itm-1" style="white-space:nowrap;cursor:pointer;"
onclick="return DoMove('STEP2');"
id="roadMap-itm-2" style="white-space:nowrap;cursor:pointer;"
onclick="return DoMove('STEP3');"
```

In this way, we can:

- display the hand-like cursor when the user crosses a RoadMap step (inclusive the RoadMap step description);
- react to the onclick event.

To be able to manage these actions from our controller class, we will use the HTMLBEVENT BSP element.

At the end, we use the controller attribute "show\_step" to display one of the three page fragments.

Not to forget the extension directives for every BSP extensions we used:

```
<%@extension name="htmlb" prefix="htmlb" %>
<%@extension name="phtmlb" prefix="phtmlb" %>
<%@extension name="bsp" prefix="bsp" %>
```

We also need the directive "<%@page language="abap" %>", because we have used the ABAP statements.

PS

In case we want to react only when the user click the RoadMap number ( 1, 2 or 3 ) we can replace:

```
<span style="white-space:nowrap;">1</span>
<span style="white-space:nowrap;">2</span>
<span style="white-space:nowrap;">3</span>
```

with:

```
`<span style="white-space:nowrap;cursor:pointer;" onclick="return DoMove('STEP1');">1</span>`
`<span style="white-space:nowrap;cursor:pointer;" onclick="return DoMove('STEP2');">2</span>`
`<span style="white-space:nowrap;cursor:pointer;" onclick="return DoMove('STEP3');">3</span>`
```

### Creating the Controller Class - zcl\_cont

The controller class is a normal ABAP class (created with the SE24 transaction) that inherits from the global class CL\_BSP\_CONTROLLER2. In this class, we create the two attributes we have already used into the view, and an extra attribute required to internally manage the RoadMap steps. The definition of these attributes is presented in Fig. 6.

ROADMAP_ITEMS	Instance	Public	<input type="checkbox"/>	Type	PHTMLB_ROADMAPITEMS
ROADMAP_STEP	Instance	Public	<input type="checkbox"/>	Type	PHTMLB_ROADMAPITEM-ITEMINDEX
SHOW_STEP	Instance	Public	<input type="checkbox"/>	Type	STRING

Fig. 6: Controller attributes definition

Follow we will redefine some of the inherited methods. As we know, the Hook Methods from the Web Dynpro ABAP / Java are called into a specific sequence according to the phase model. In the BSP applications we have also such kind of methods into the controller class (inherited from the CL\_BSP\_CONTROLLER2 super class).

For example, if in Web Dynpro [3-4] we have the wdDoInit Hook Method used as a constructor (to initialize the variable), in the controller class we can redefine the method DO\_INIT to make the same thing.

We will use this method to dynamically create the RoadMap items, to set their properties and to activate the first wizard step.

*METHOD do\_init.*

```
Me->roadmap_step = 1.
```

```
Road_map( ).
```

*ENDMETHOD.*

The road\_map method is presented in Listing 2.

*METHOD road\_map.*

*DATA: la\_items TYPE phtmlb\_roadmapitem.*

*IF me->roadmap\_items IS INITIAL.*

*“ Step 1*

*la\_items-itemindex = 1.*

*IF roadmap\_step = 1.*

*La\_items-stepdesign = ‘SELECTED’.*

*Show\_step = ‘STEP1’.*

*ELSE.*

*La\_items-stepdesign = ‘DEFAULT’.*

*ENDIF.*

*La\_items-isinteractive = abap\_true.*

*La\_items-stepname = ‘1’.*

*La\_items-stepdescription = ‘Step1’.*

*APPEND la\_items TO roadmap\_items.*

*“ Step 2*

*la\_items-itemindex = 2.*

*IF roadmap\_step = 2.*

*La\_items-stepdesign = ‘SELECTED’.*

*Show\_step = ‘STEP2’.*

*ELSE.*

*La\_items-stepdesign = ‘DEFAULT’.*

*ENDIF.*

*La\_items-isinteractive = abap\_true.*

*La\_items-stepname = ‘2’.*

*La\_items-stepdescription = ‘Step2’.*

*APPEND la\_items TO roadmap\_items.*

*“ Step 3*

*la\_items-itemindex = 3.*

*IF roadmap\_step = 3.*

*La\_items-stepdesign = ‘SELECTED’.*

*Show\_step = ‘STEP3’.*

*ELSE.*

*La\_items-stepdesign = ‘DEFAULT’.*

*ENDIF.*

*La\_items-isinteractive = abap\_true.*

*La\_items-stepname = ‘3’.*

```

La_items-stepdescription = 'Step3'.
APPEND la_items TO roadmap_items.
ELSE.
  la_items-stepdesign = 'DEFAULT'.
  MODIFY roadmap_items FROM la_items TRANSPORTING stepdesign WHERE stepdesign =
'SELECTED'.
  READ TABLE roadmap_items INTO la_items WITH KEY itemindex = roadmap_step.
  IF sy-subrc = 0.
    La_items-stepdesign = 'SELECTED'.
    MODIFY roadmap_items FROM la_items INDEX sy-tabix TRANSPORTING stepdesign.
  IF roadmap_step = 1.
    Show_step = 'STEP1'.
  ELSEIF roadmap_step = 2.
    Show_step = 'STEP2'.
  ELSEIF roadmap_step = 3.
    Show_step = 'STEP3'.
  ENDIF.
ENDIF.
ENDIF.
ENDMETHOD.

```

Listing 2 The road\_map method

Then, we will redefine the do\_request method.

```

METHOD do_request.
  Dispatch_input( ).
  IF is_navigation_requested( ) IS NOT INITIAL.
    RETURN.
  ENDIF.
  DATA lv_view TYPE REF TO if_bsp_page.
  Lv_view = create_view( view_name = 'roadmap.htm').
  call_view( lv_view ).
ENDMETHOD.

```

At the end, we redefine the method do\_handle\_event required to manage the navigation (Listing 3). By using the roadmap\_step attribute, we set active the item on which the end user clicks.

```

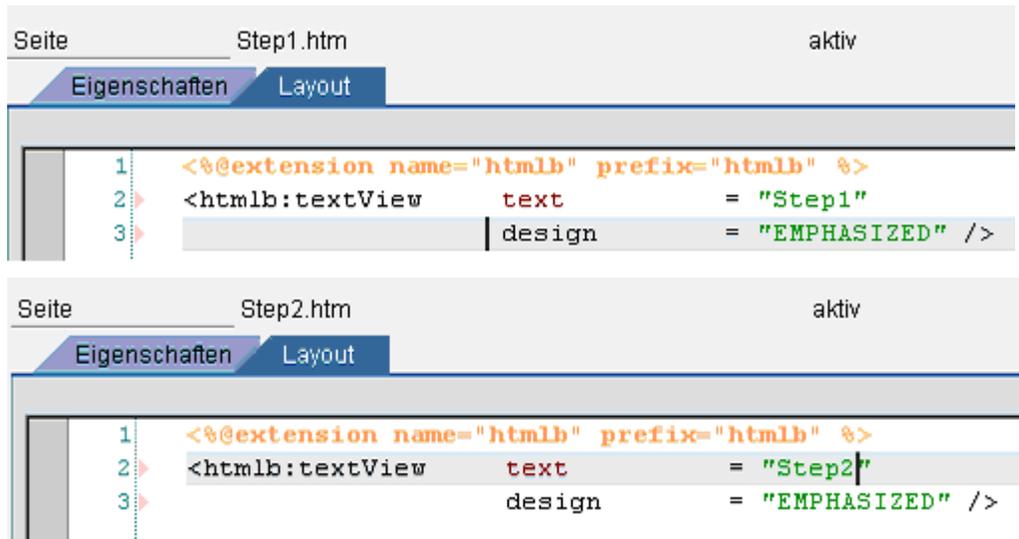
METHOD do_handle_event.
  DATA: l_move_event TYPE REF TO cl_bsp_htmlb_event,
        l_event TYPE string.
  IF event = 'movestep' AND htmlb_event_ex IS BOUND.
    l_move_event ?= htmlb_event_ex.
    l_event = l_move_event->p1.
  ENDIF.
  IF l_event EQ 'STEP1'.
    Roadmap_step = 1.
    Road_map( ).
  ELSEIF l_event EQ 'STEP2'.
    Roadmap_step = 2.
    Road_map( ).
  ELSEIF l_event EQ 'STEP3'.
    Roadmap_step = 3.
    Road_map( ).
  ENDIF.
ENDMETHOD.

```

Listing 3 The do\_handle\_event method

### Creating the Page Fragments

We create three page fragments. When the user clicks on the first step in wizard we display the Step1.htm. When the user presses the second step in the wizard we display the Step2.htm, and for the third step we display the Step3.htm. The structure of these page fragments is presented in Fig. 7.



```

Seite      Step3.htm      aktiv
Eigenschaften  Layout
1 > <%@extension name="htmlb" prefix="htmlb" %>
2 > <htmlb:textView      text      = "Step3"
3 >                       design    = "EMPHASIZED" />

```

Fig. 7: The structure of the page fragments

## Running the Application

In Fig 8: we can see the result.

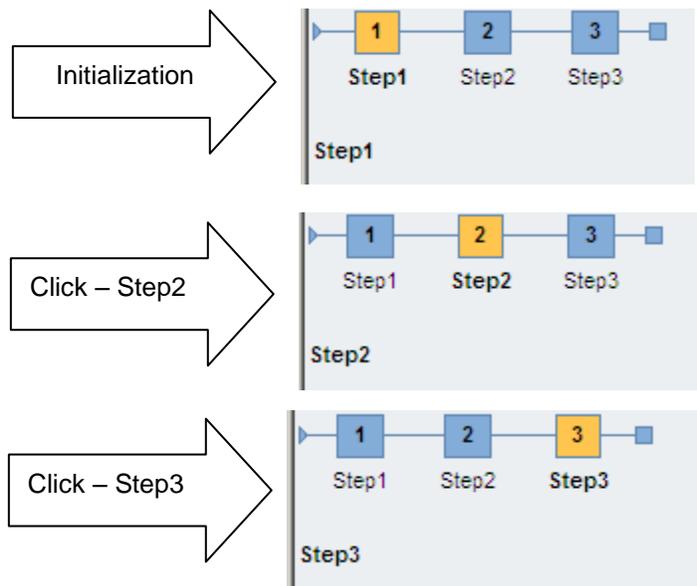


Fig. 8: Running the application

## Conclusions

In our article, we shown not only the way how to create and use a Road Map UI element. We also show the way to create an action for this BSP element.

Of course, it would be better to create an own BSP Element with this changes.

We have also to mention, that an replace in the HTML Sourcecode is depend on how SAPgenerates HTML code in the next version. So it's not an good idea to replace the HTMLSourcecode. Maybe in some situations (to react to an action) it makes sense.

To navigate from a Road Map step to another Road Map step, without defining an action for these UI Elements, we can use two extra buttons "Back" and "Next". In this case we can use the methods that we have presented without replacing the HTML Sourcecode. Inthis case we have to increase the Road Map step every time the user pressed the "Next"- Button and decrease the Road Map step every time the user pressed the "Back"-Button.

## Related Content

- [1] Ulli Gellert, Cristea Ana Daniela, [Web Dynpro ABAP for Practitioners](#), Springer 2010, ISBN: 978-3-642-11384-0
- [2] Brian McKellar, Thomas Jung, [Advanced BSP Programming](#), SAP Press 2006, ISBN: 1-59229-049-3
- [3] Cristea Ana Daniela, [Messages Internationalization with Web Dynpro ABAP](#)
- [4] Cristea Ana Daniela, Adela Berdie, Mihaela Osaci, Daniel Chirtoc, [The Advantages of Using Mind Map for Learning Web Dynpro](#), Computer Applications in Engineering Education, Volume 19
- [5] [Messages Internationalization with Web Dynpro ABAP](#)
- [6] <http://onlinelibrary.wiley.com/doi/10.1002/cae.20285/abstract>

For more information, visit the [ABAP homepage](#).

## Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.