ITSE333A ABAP

Lesson 7: ABAP Objects and BSP

Anthony D. Aquino

Agenda

1. ABAP Objects
2. Business Server Pages
   - Maintenance and execution
   - Usage of BAPIs in BSPs
   - HTMLB
   - Model View Controller

Introduction ABAP Objects

- Object-oriented enhancement of ABAP
- Stepwise conversion of existing ABAP source code into ABAP Objects
- Major tools:
  - ABAP Editor
  - Class Builder

Principles of Object Orientation I

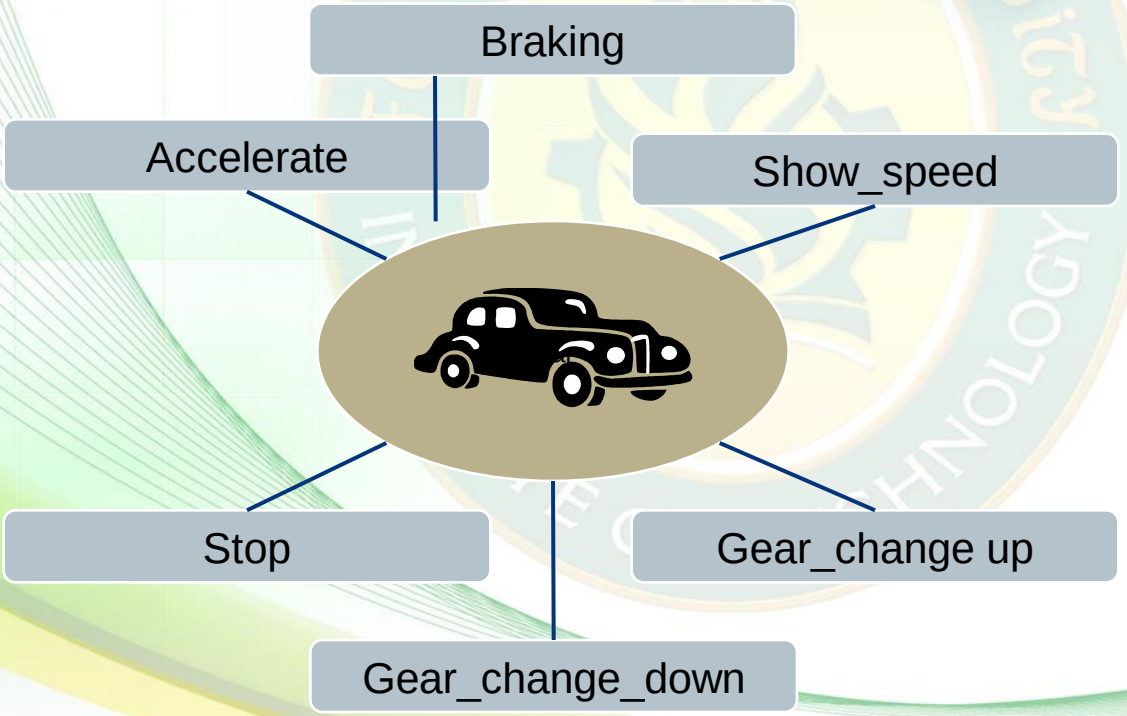
- Classes are the definition of an object including attributes and methods of this object
- Objects are instances of classes
  - Example:
    - Class: Car
    - Instance: My car with the license number xxx

Use of classes and methods

Procedurale approach

- Car_accelerate
- Car_braking
- Car_show_speed
- Car_stop
- Car_Gear_change_up
- Car_Gear_change_down

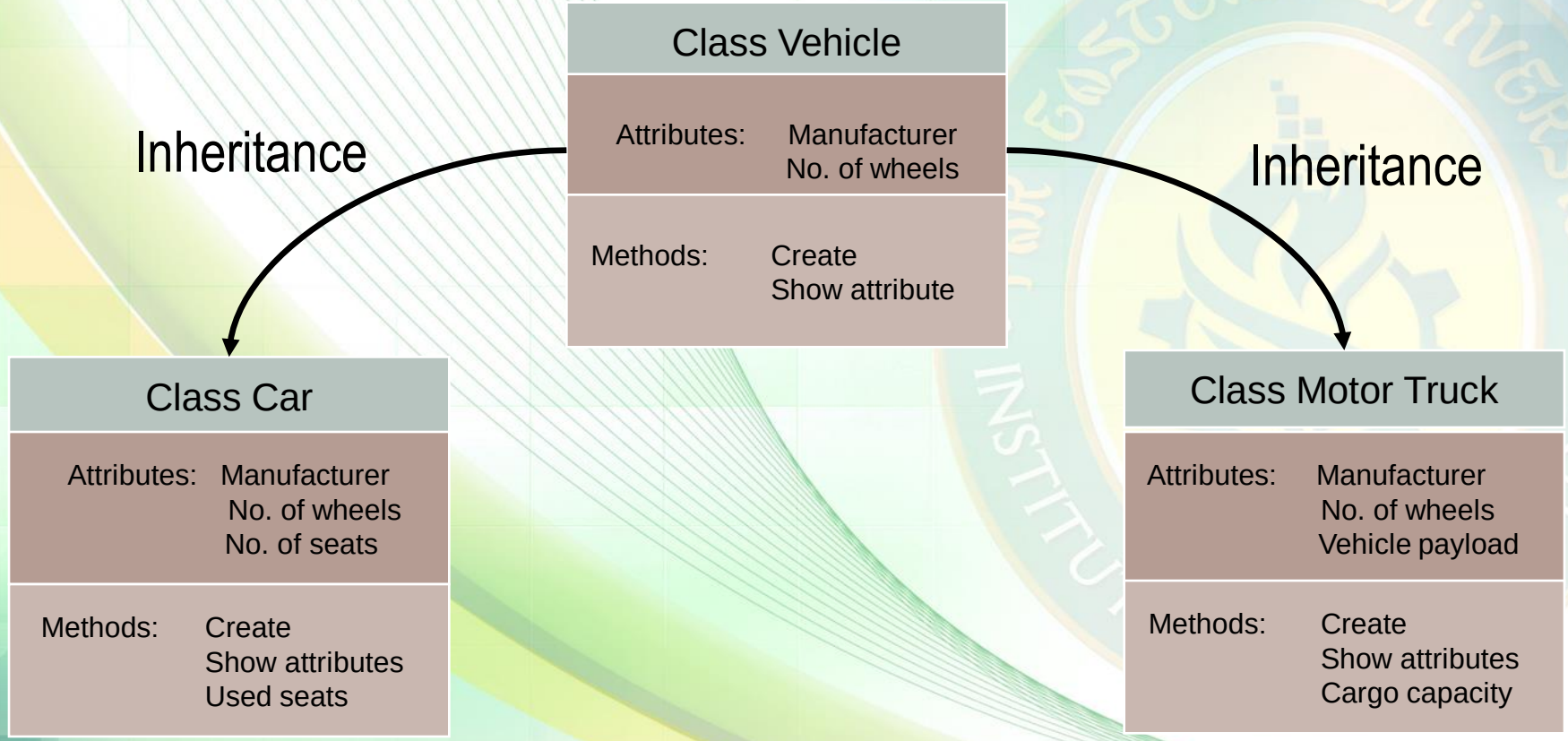
Object-oriented approach



Principles of Object Orientation II

- Encapsulation: The implementation of a class is invisible outside the class. Interaction takes place only by a defined interface.
- Polymorphism: Identically-named methods used for different classes respond according to an appropriate class-specific behavior.
- Inheritance: A new class can inherit attributes and methods from an existing class that can be extended by additional, own attributes and methods.

Example for Object Orientation



Soure: Own illustration

Classification of methods and attributes

- Private ←→ public
  - Private methods/attributes can be called/accessed only from the class‘ own methods (e.g. method „braking“ shouldn‘t be called by everyone)
  - Public methods/attributes can always be called/accessed (e.g. what colour has your car?)
- Instance ←→ static
  - Instance methods/attributes exist for each instance of a class (e.g. method „get colour“, as each instance has it‘s own colour)
  - Static methods/attributes exist only once for all instances (e.g. listing of all instances of a class)

Declaration of a class

- Classes can be created using ABAP Editor or Class Builder
- Separation of definition and implementation of a class
- Definition includes the declaration of all components of a class (e.g. methods, attributes..)
- Implementation includes the coding for all methods of a class

Definition of a class

```
CLASS <Class name> DEFINITION.
   PUBLIC SECTION.
        METHODS:
           <Method1>
                IMPORTING  <Import Parameter1> TYPE <Data type>
                           <Import Parameter2> TYPE <Data type>
        CLASS-METHODS:
           <Method2>
        DATA:              <Variable1> TYPE <Data type>
        CLASS-DATA:        <Variable2> TYPE <Data type>
   PRIVATE SECTION.
        METHODS: …
        CLASS-METHODS: …
        DATA: …
        CLASS-DATA: …
ENDCLASS.
```

Implementation of a class

```
CLASS <Class name>
IMPLEMENTATION.
   …
   METHOD <Method name>.
        …ABAP-Code…
   ENDMETHOD.
   …
ENDCLASS.
```

Calling methods and accessing attributes

- Instance method:

  [CALL METHOD] <Instance name>
  ><Method>(<Import parameter> = Value).

- Class method:

  [CALL METHOD] <Class name>
  =><Method>(<Import parameter> = Value).

- Instance attributes:

  <Instance name>-><Attribute>.

- Static attributes:

  <Class name>=><Attribute>.

Constructor

- Explicitly or implicitly defined method „constructor“ used for creating new instances
- Automatically called by „CREATE OBJECT“
- In case of missing implementation simply a new instance is created
- In case of explicit implementation within the PUBLIC SECTION additional steps can be executed when creating new instances (e.g. setting of default values)

Example constructor

```
PUBLIC SECTION
  METHODS: constructor IMPORTING
                    im_name type string
                    im_planetype type string.

CREATE OBJECT r_plane EXPORTING
   im_name = ‚Munich‘
   im_planetype=‚747‘.
```

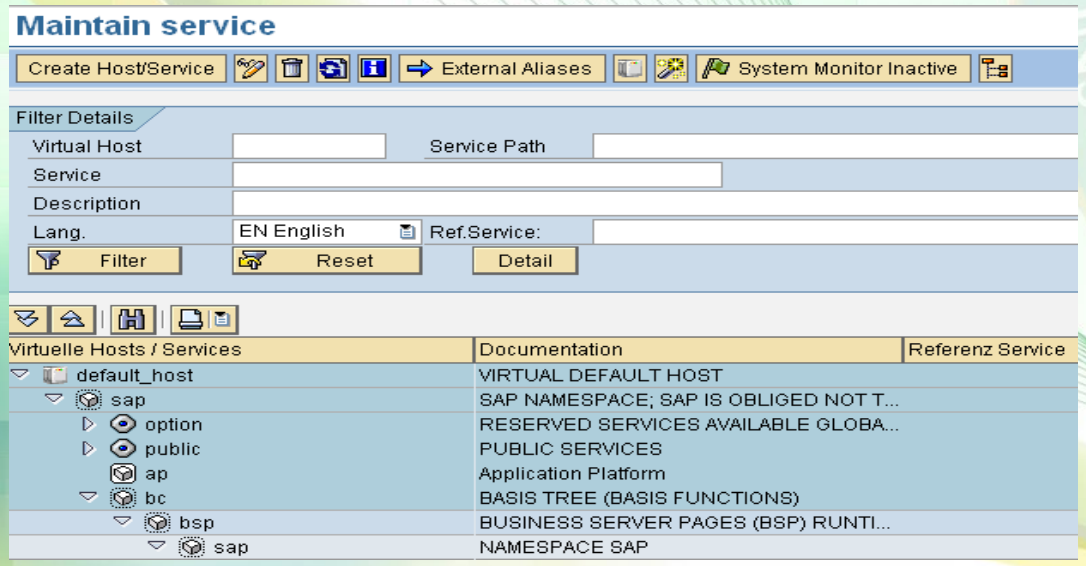
Business Server Pages (BSP)

- Web extension for ABAP
- Enables the use of ABAP and server-side JavaScript within HTML pages created and hosted on a SAP System
- Can be used for the realization of extensive portal solutions
- Availability of BSPs starting with SAP Web Application Server 6.20

Maintenance and execution of BSP‘s

- Maintenance by transaction SICF
- Execution: http://<host>:<ABAP Port>/sap/bc/bsp/sap/ <Program name>\<Page name>.htm

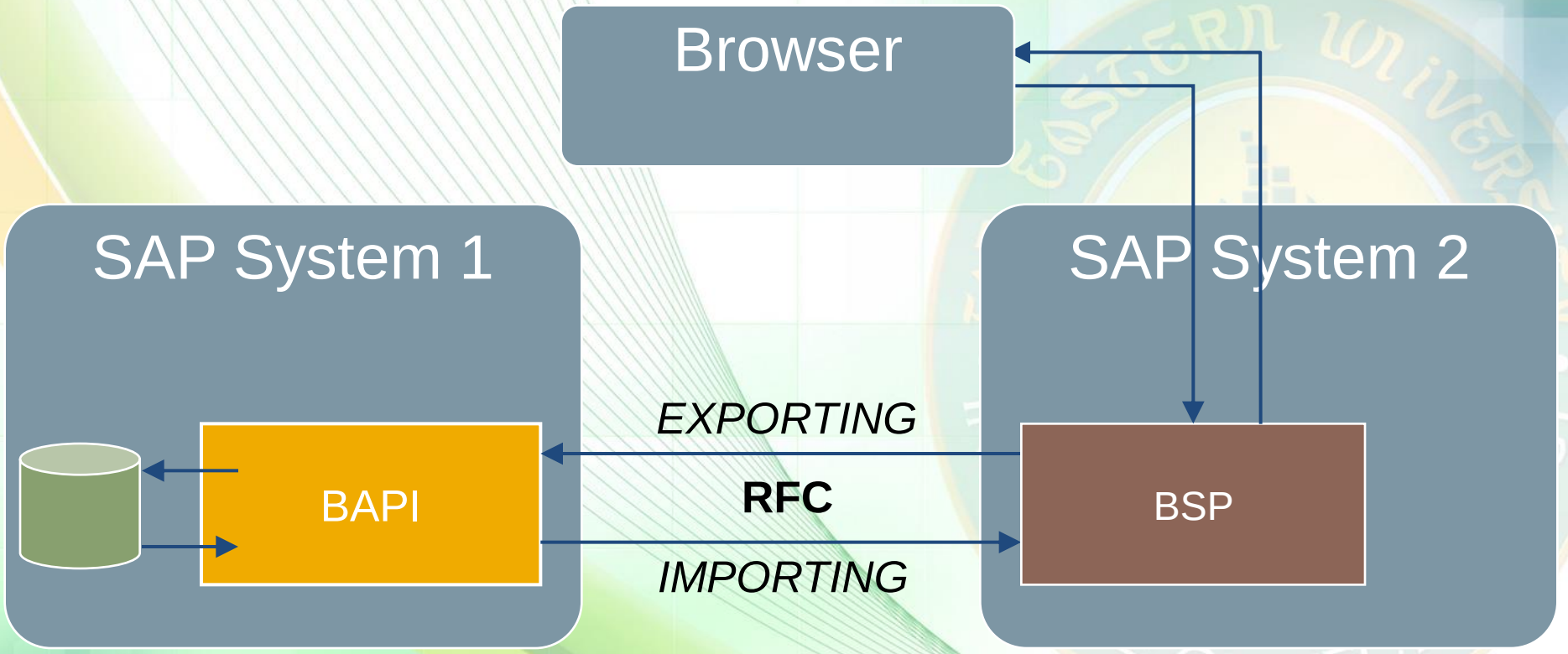
  E.g.: http://g51as1.informatik.tu-muenchen.de: 8051/sap/bc/bsp/sap/<Programname>\<Page name>.htm



Usage of BAPIs in BSPs I

- Business Application Programming Interface = BAPI
  - BAPI represent an interface to the outside world
  - SAP systems offer a wide range of different BAPIs
  - BAPI calls are executed using RFC
  - Transaction „BAPI“ shows all available BAPIs of the current SAP system
  - Search for BAPIs using BAPI Browser, cross-system access possible
  - Use of BAPIs with help of information available in the BAPI Browser

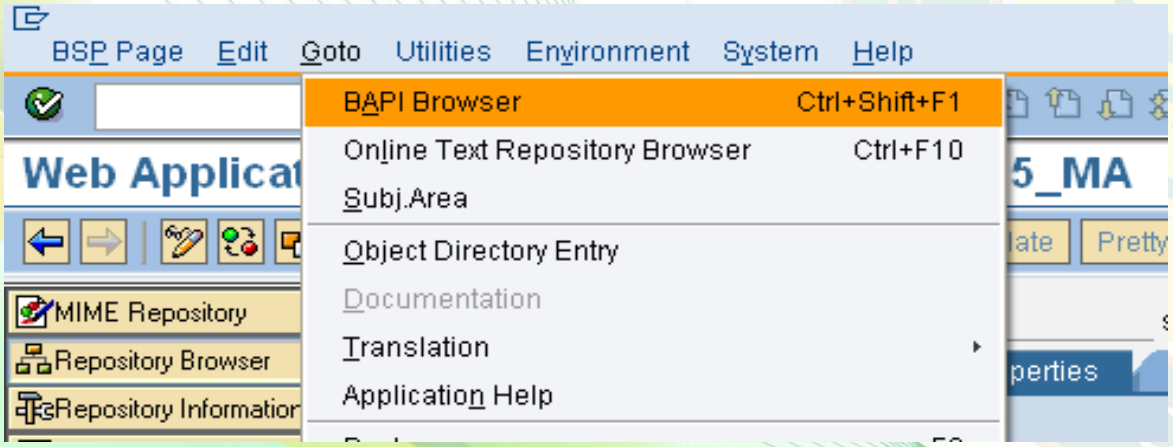
Usage of BAPIs in BSPs II



- This example: Call of an external BAPI
- Other possible examples: Call of an internal BAPI

Using BAPI Browser I

- Call BAPI Browser in transaction SE80
  - Menue item ‚Goto' → ‚BAPI Browser'

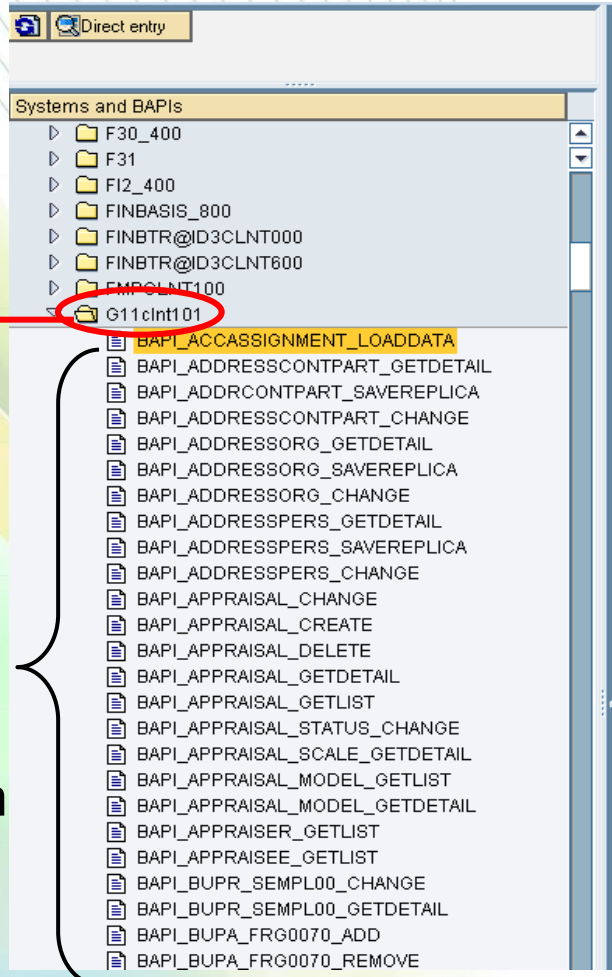


Using BAPI Browser II

- BAPI Browser components:

RFC-Connection

Available BAPIs for RFC-Connection



Interface definition

```
types: begin of BBP_STD_ACC
  CLIENT(000003)                  type   C      ,
  LOG_SYS(000010)                 type   C      ,
  ACC_CAT(000005)                 type   C      ,
  VALUE(000064)                   type   C      ,
  VALID_FROM                      type   D      ,
  VALID_TO                        type   D      ,
       end of   BBP_STD_ACC                     .
types: begin of BAPIRET2
  TYPE(000001)                    type   C      ,
  ID(000020)                      type   C      ,
  NUMBER(000003)                  type   N      ,
  MESSAGE(000220)                 type   C      ,
  LOG_NO(000020)                  type   C      ,
  LOG_MSG_NO(000006)              type   N      ,
  MESSAGE_V1(000050)              type   C      ,
  MESSAGE_V2(000050)              type   C      ,
  MESSAGE_V3(000050)              type   C      ,
  MESSAGE_V4(000050)              type   C      ,
  PARAMETER(000032)               type   C      ,
  ROW                             type   I      ,
  FIELD(000030)                   type   C      ,
  SYSTEM(000010)                  type   C      ,
       end of   BAPIRET2                        .
types: RETURN                     type table of
         BAPIRET2                               .

types: ACCOUNTASSIGNMENT          type table of
         BBP_STD_ACC                            .
```

Call example

```
*Required Data Definition
data ACCOUNTASSIGNMENT                type
           ACCOUNTASSIGNMENT                    .
data RETURN                           type
           RETURN                               .
* Call Example
CALL FUNCTION 'BAPI_ACCASSIGNMENT_LOADDATA'
         DESTINATION 'G11clnt101'
   TABLES
     ACCOUNTASSIGNMENT              = ACCOUNTASSIGNMENT
*    RETURN                         = RETURN
```

Usage of BAPIs in BSPs III

Example scenario:

- Call BAPI „BAPI_USER_GET_DETAIL“
  - Show details of a certain user
  - BAPI offers broad functionality for query of user data
  - Use of BAPI in order to show the „last changed on“ date

Example: Call BAPI in BSP – Step 1

Step 1: Definition of variables

- Temporary variables

```
data username type c.
data tmp_islocked type BAPISLOCKD.
data tmp_moddat type BAPIMODDAT.
```

- Interface variables

```
data ISLOCKED type BAPISLOCKD.
data LASTMODIFIED type BAPIMODDAT.
```

Example: Call BAPI in BSP – Step 2

Step 2: Definition CALL FUNCTION

```
CALL FUNCTION 'BAPI_USER_GET_DETAIL'
  DESTINATION 'G11clnt101,
EXPORTING …..
IMPORTING …...
```

- CALL FUNCTION refers to the BAPI name
- DESTINATION refers to the RFC-connection
- EXPORTING refers to input parameters of a BAPI
- IMPORTING refers to output parameters of a BAPI

Example: Call BAPI in BSP – Step 3

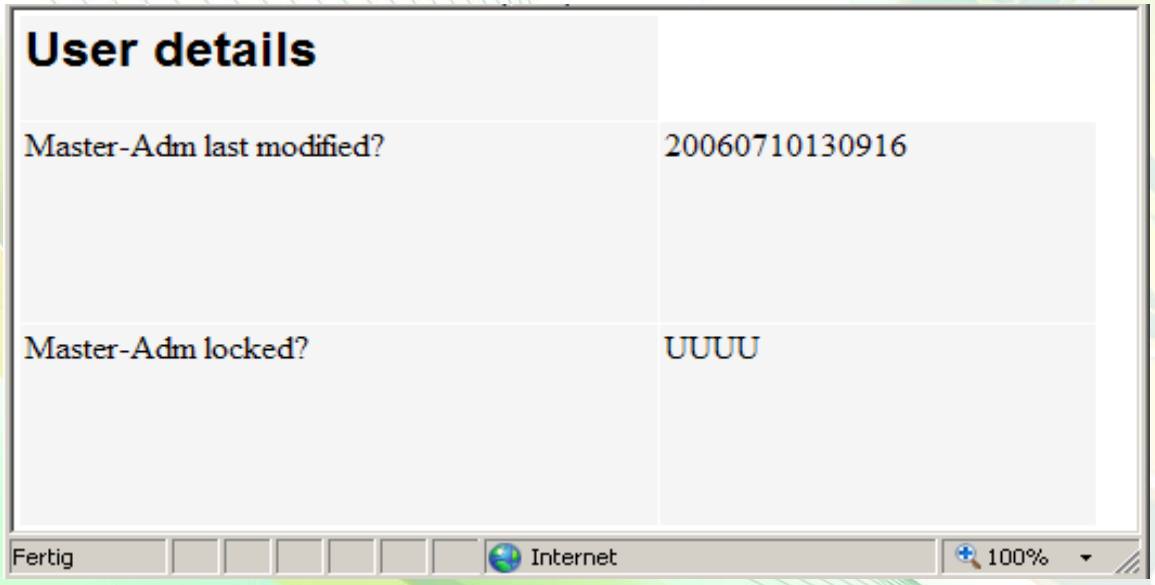
Step 3: Complete call within a BSP:

```
CALL FUNCTION 'BAPI_USER_GET_DETAIL'
 DESTINATION 'G51'
 EXPORTING
    username                = 'master-adm'
 IMPORTING
    lastmodified            = tmp_moddat.
```

- Query of the „Last changed on“ date (→lastmodified) of user „master-adm“
- Output is assigned to variable tmp_moddat

Example: Call BAPI in BSP – Result

- Result:



Example: Call BAPI in BSP

- Complete source code:

```
<%@page language="abap" %>
<%@extension name="htmlb" prefix="htmlb" %>
<htmlb:content>
  <htmlb:page title="Flugresultate" >
    <table width="100%" height="100%" cellspacing="1" cellpadding="2" border="0">
       <tr class="sapTbvCellStd"> <td bgcolor="Whitesmoke" valign="top">
           <h2> User details </h2>
        <%
           data: username type c.
           data ISLOCKED type BAPISLOCKD.
           data lastmodified type bapimoddat.
           data tmp_islocked type bapislockd.
           data tmp_moddat type bapimoddat.
           CALL FUNCTION 'BAPI_USER_GET_DETAIL'
           DESTINATION ,G51'
           EXPORTING
           USERNAME                        = 'master-adm'
           IMPORTING
           islocked                          = tmp_islocked
           lastmodified                      = tmp_moddat.
           %>
    <tr>
        <td bgcolor="Whitesmoke" valign="top"> Master-Adm last modified? </td>
        <td bgcolor="Whitesmoke" valign="top"><%= tmp_moddat %></td>
    </tr>
    <tr>
        <td bgcolor="Whitesmoke" valign="top"> Master-Adm locked? </td>
        <td bgcolor="Whitesmoke" valign="top"><%= tmp_islocked %></td>
    </tr>
    </table>
  </htmlb:page>
</htmlb:content>
```

HTMLB

- Extension of HTML by SAP
- HTML-Business
  - HTML-Business for Java
  - HTML-Business for ABAP
- HTMLB is used e.g. for Enterprise Portal

Comparison HTML / HTMLB

HTML → HTMLB

```
<%@page language="abap" %>

<html>
<head>
   <title>My BSP</title>
<% data: vari type i. %>
</head>

<body>
<%--This is a comment --%>
<% vari = 5 %>
In this coding following value is
   assigend to variable „vari":
<%= vari %>
<br>
That's it.

</body>
</html>
```

```
<%@page language="abap" %>
<%@extension name="htmlb" prefix="htmlb" %>

<htmlb:content design="design2003" >
  <htmlb:page title=„My BSP" >
    <%  data: vari type i. %>



<%--This is a comment --%>
    <% vari = 5. %>
In this coding following value is  assigend
   to variable „vari":
<%= vari. %>
<br>
    That's it using HTMLB.

</htmlb:page>
</htmlb:content>
```

HTMLB Statements

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

- Naming of extension; in this case „htmlb“

```
<htmlb:content design="design2003" >
```

- Begin of content-tags with explicit design definition
- Standard designs: design2003, design2002 and classic
- Design2003 can be used only for MS IE Version 5.5 or higher

```
  <htmlb:page title="Meine BSP" >
```

- Begin of page-tags with definition of title for BSP
- Explicit <head> tag is not required any more
- Explicit <body> tag is not required any more

```
</htmlb:page>
</htmlb:content>
```

- Closing tags for page and content

HTMLB – Text fields

- Text fields:

  <htmlb:textView text = „Hi this is a textView“ design =„Emphasized“ />

  Hi, this is a textView

- Text fields include different attributes:
  - Text: The displayed text
  - Design: Different design types
    - Emphasized
    - Header 1-3 - Headlines
    - Reference – *italic Reference*
    - Standard
    - etc.

HTMLB – Forms, Input fields

Forms

- <htmlb:form id = "myFormId" method = "post" encodingType = "multipart/form-data" >
  Equivalent of *form* in HTML

Input Fields

- <htmlb:inputField id="IP1" />
- Input field:

What about an input field:

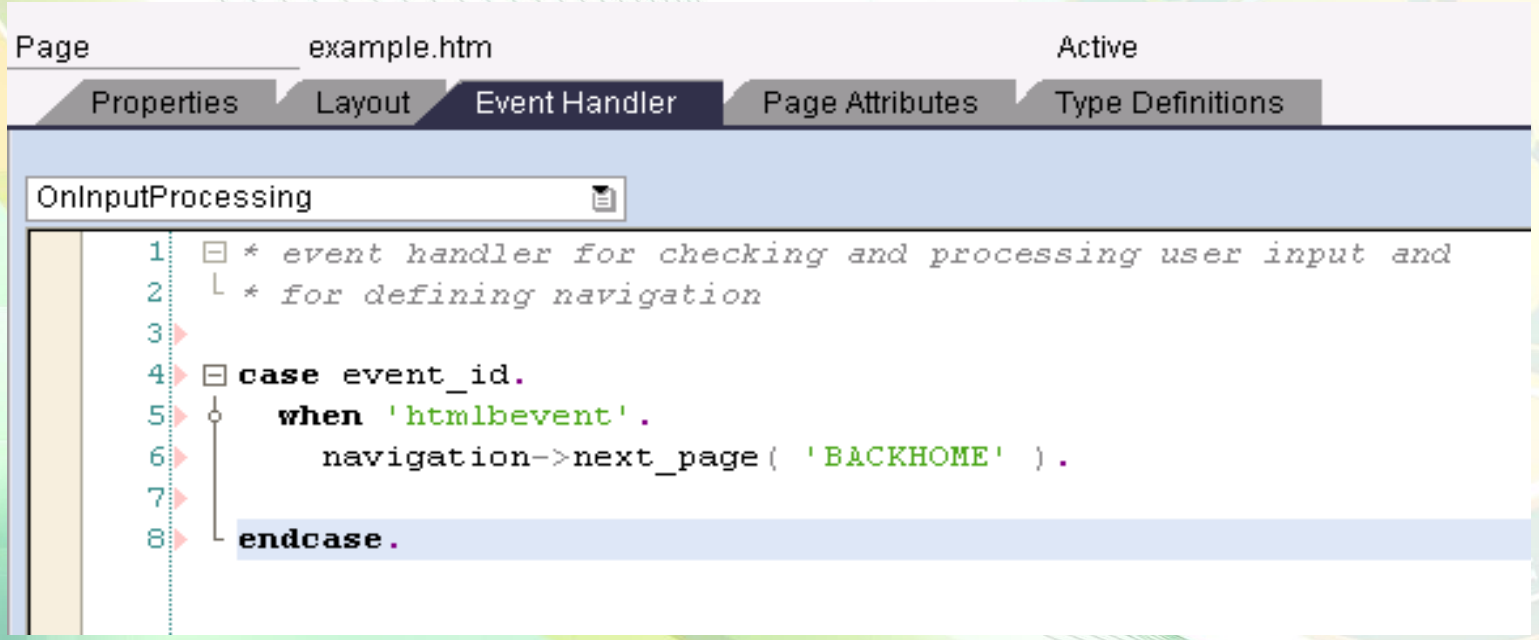
HTMLB – Buttons, Events

Buttons

- <htmlb:button id = "SaveChanges" onClick = "SubmitChange" text = "Submit Button" onClientClick = "onInputProcessing(htmlbevent);" />
- Button for sending forms
- Important: onClientClick raises event, in this case: onInputProcessing(htmlbevent)
- Event is called onInputProcessing
- Event_id is called: htmlbevent

HTMLB - Eventhandler

- Events are defined in the Eventhandler!



HTMLB – RadioButtons I

Radiobuttons

```
<htmlb:radioButtonGroup id = "myID" columnCount = "2" >
        <htmlb:radioButton id = "RBGenderFemale" text = "female"
   tooltip = "I am female" disabled = "false" />
        <htmlb:radioButton id = "RBGenderMale" text = "male"
   tooltip  = "I am male (and this button is disabled)" disabled =
   "true" />
</htmlb:radioButtonGroup>
```

- RadioButtons always belong to a RadioButtonGroup
- RadioButtonGroup with attributes
  - Id: Unique ID
  - columnCount: Number of columns
  - currentItem: Defines the active item

HTMLB – RadioButtons II

```
<htmlb:radioButtonGroup id = "myID" columnCount = "2" >
        <htmlb:radioButton id = "RBGenderFemale" text = "female"
   tooltip = "I am female" disabled = "false" />
        <htmlb:radioButton id = "RBGenderMale" text = "male"
   tooltip  = "I am male (and this button is disabled)" disabled =
   "true" />
</htmlb:radioButtonGroup>
```

- RadioButton with attributes
  - Id: Unique ID
  - Text: Text of RadioButtons
  - Tooltip: Text that is displayed as tooltip
  - Disabled: RadioButton cannot be selected

HTMLB - Layout

- RadioButtons
- Tree with treeNodes
- TabStrips
- TableView

Let's try some radio buttons:
female male

Some cool stuff: treeView
e-enviroment
Desk
Network
LAN
WAN
Infrared

Some other very cool stuff: tabstrips
Tab 1 Tab 2 Tab 4
Body of Tab 4

TableView example 1

	MDT	ID	Nr	Lnd	Abflugstadt	FLH	Lnd	Ankunftstadt	FLH	Flugdauer	Abflug	Ankunft	Entf	In	Charter	Ankunft n Tag(e) später
	902	AA	0026	DE	FRANKFURT	FRA	US	NEW YORK	JFK	7:20	08:30	09:50	3.851,0000	MI		0
	902	AA	0064	US	SAN FRANCISCO	SFO	US	NEW YORK	JFK	5:21	09:00	17:21	2.572,0000	MI		0
	902	AZ	0555	IT	ROME	FCO	DE	FRANKFURT	FRA	2:05	19:00	21:05	845,0000	MI		0
	902	AZ	0788	IT	ROME	FCO	JP	TOKYO	TYO	12:55	12:00	08:55	6.130,0000	MI		1
	902	AZ	0789	JP	TOKYO	TYO	IT	ROME	FCO	15:40	11:45	19:25	6.130,0000	MI		0

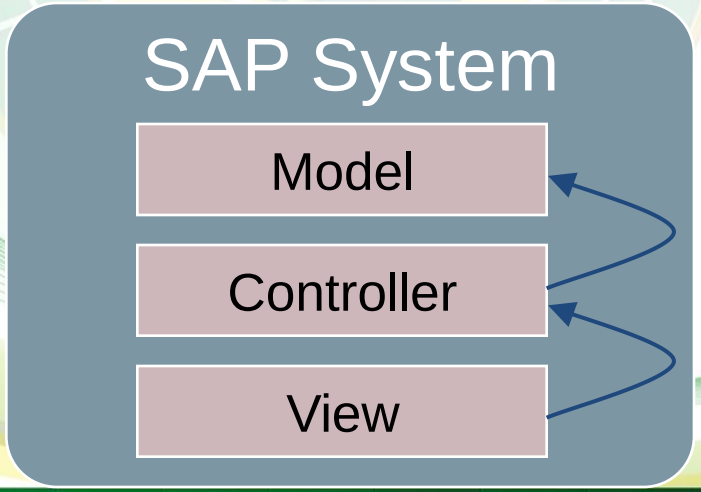
Zeile 2 von 46

Model View Controller

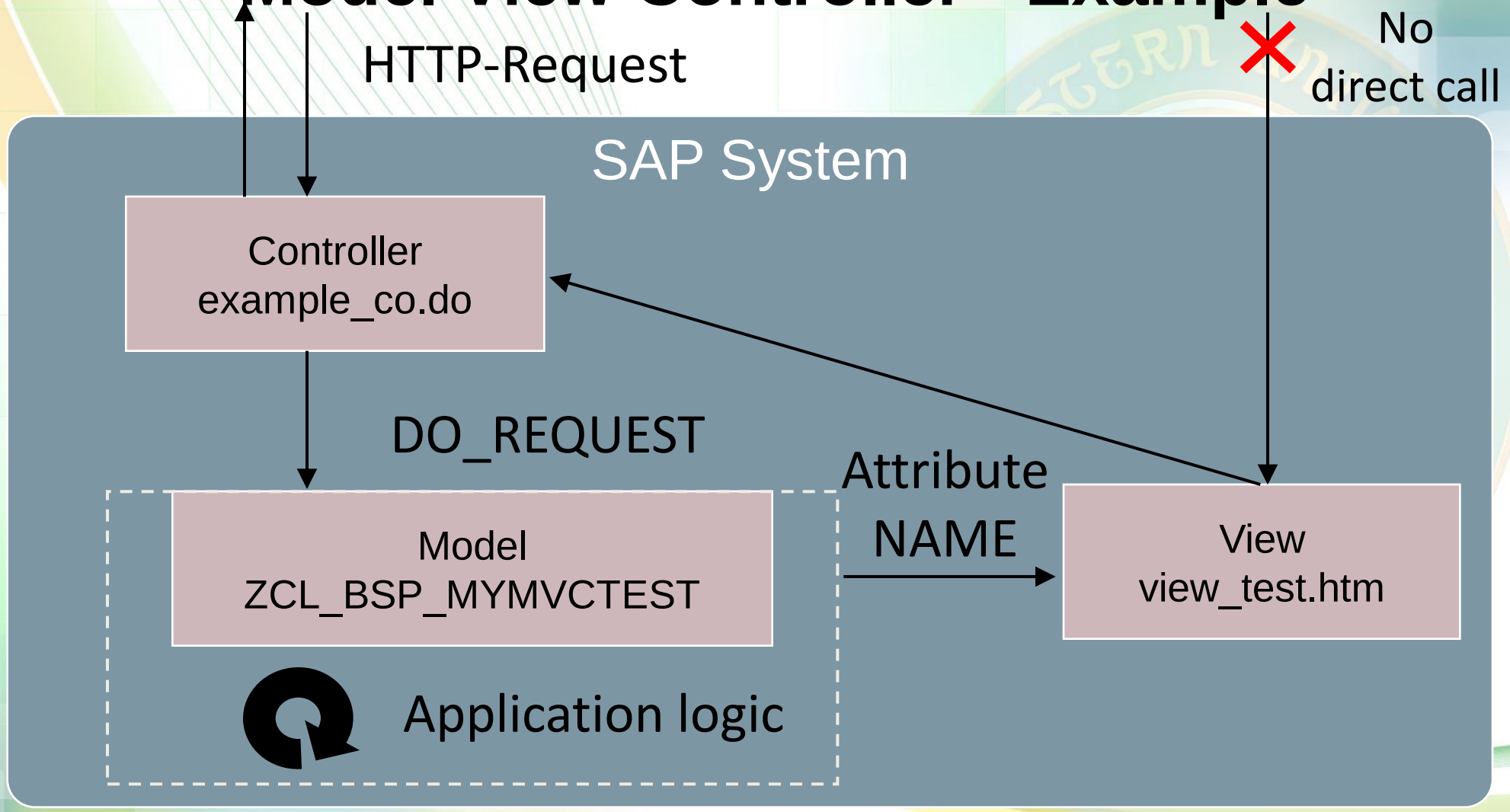
- Up-to-now: Presentation logic and application logic both are included in a BSP
- Model View Controller: Separation between logical layers

Model View Controller

- Model
  - Consists of a class derived from CL_BSP_MODEL
  - Processing takes place in background
- Controller
  - Handles Requests and forwards to Model
- View
  - Calls Controller
  - Only responsible for the visualization of data



Model View Controller - Example



Model View Controller - Example

1. Create Controller example_co.do
2. Derive own class ZCL_BSP_MYMVCTEST from CL_BSP_CONTROLLER2
3. Redefine method DO_REQUEST
4. Define view view_test.htm
5. Define attribute NAME
6. Call Controller

Model View Controller - Example