Uploading and Downloading Files in Web Dynpro Java

Applies to:

Web Dynpro for Java UI Development, SAP NetWeaver 2004s

Summary

In this tutorial you learn how to download and upload files within Web Dynpro applications running on SAP NetWeaver 04s by utilizing the new dictionary type *Resource* and its related Web Dynpro APIs `IWDResource` and `WDResourceFactory`. In contrast to the dynamic type modification of a binary context attribute in SAP NetWeaver 04, the new dictionary simple type *Resource* yields a fully declarative, zero coding data transport of download and upload resources between Web Dynpro client and controller context on server side.

Note

This tutorial covers the basic principles of *Uploading and Downloading Files in Web Dynpro Java in SAP NetWeaver 2004s.* You can read more advanced technical details in the related SDN article Uploading and Downloading Files in Web Dynpro Tables. Both articles are based on the same Web Dynpro sample application.

Author: Bertram Ganz

Company: SAP AG

Created on: 21 February 2007

Author Bio

After his studies in mathematics, physics and computer science Bertram Ganz finished his teacher training at a German grammar school stressing technical sciences. He has been a member of the Web Dynpro Java Runtime development team (SAP NetWeaver ESI Foundation UI) since 2002. The main focus of his work is on knowledge transfer, rollout, and documentation. Bertram regularly publishes articles on Web Dynpro in the context of the SAP NetWeaver Application Server. Bertram is co-author of the books “Maximizing Web Dynpro for Java” and “Java Programming with the Web Application Server”.

Table of Contents

Introduction

The Web Dynpro UI Element Library provides two special UI elements (*FileDownload* and *FileUpload*) with which you can download files from the Web Dynpro runtime environment or upload them there. This is performed using declarative *data binding*. Here, Web Dynpro runtime automatically transports different types of MIME files between the client-side user interface or UI element and the server-side controller context.

In contrast to the semi-declarative approach in SAP NetWeaver 04, which was based on invoking the `IWDModifiableBinaryType`-API in the controller code, the new dictionary simple type *Resource* within SAP NetWeaver 04s yields a fully-declarative data transport of MIME resources between Web Dynpro client and controller context on server side.

This tutorial will show you how to upload and download MIME files using the UI elements *FileUpload* and *FileDownload*.

What's New in SAP NetWeaver 04s

In SAP NetWeaver 04s file download and upload was significantly enhanced and simplified with the following new features:

- **New Dictionary Type *Resource***: The new Java dictionary type *Resource* allows to bind FileUpload and FileDownload UI elements to context attribtues storing MIME files named resources. In SAP NetWeaver 04 the primitive type binary must be used.
- **New Java Type *IWDResource***: The new Web Dynpro interface `IWDResource` is the Java type counterpart of the dictionary type `Resource`. It allows to store the file content (binary resource data) and the file metadata (MIME type, resource name) in one object. Consequently the resource metadata (MIME type, file name) must no longer be stored in the context attribute info (`IWDAttributeInfo`) using a modifiable binary type and may therefore differ among multiple resources or node elements stored in the same context node.
- **New Web Dynpro Factory *WDResourceFactory***: Resource objects of type `IWDResource` can easily be created with the new Web Dynpro factory class `WDResourceFactory`. This factory class significantly simplifies the implementation of file download scenarios where statically deployed or dynamically created resources must be stored in the context.
- **Zero Coding File Upload**: With the new Java dictionary type *Resource* it is no longer needed to implement the type modification of a binary context attribute by invoking the `IWDModifiableBinaryType` API in the controller code. Consequently file upload can be realized in a purely declarative, zero coding approach. You just have to bind a *FileUpload* UI element to a context attribute of type *Resource.* The Web Dynpro Runtime then automatically transports the uploaded file to the context attribute as an object of type `IWDResource`.
- **New File Download Behaviors**: The behavior of the *FileDownload* UI Element can now be defined with the new Java dictionary type *FileDownloadBehavior*. Its enumeration specifys three different file download behaviors: open resource in-place without opening a dialog window, save resource in local file system (open dialog) and open resource depending on the MIME type of the downloaded file (open dialog).
- **Downloading Files in Tables On-Demand**: The new on-demand streaming technique allows to download the resource content on-demand when the user actually requests it on client side. Especially when using the *FileDownload* UI element as table cell editor this new technique yields a heavily reduced context memory consumption on server side.
  Read the related SDN article Uploading and Downloading Files in Web Dynpro Tables to get more information on the additional technical details.

Using the Virus Scan Interface

To enhance your system's virus protection when working with files or documents processed by your Web Dynpro applications, you can add external virus scanners to your SAP system using the Virus Scan Interface.

To connect the *FileUpload* service contained in the Web Dynpro runtime environment to a virus scanner, you need to activate the predefined virus scan profile `webdynpro_FileUpload`. This profile must be activated/deactivated by the SAP J2EE Engine administrator. When delivered, profile `webdynpro_FileUpload` is switched off. The Virus Scan Interface cannot be used for the Web Dynpro *FileDownload* service.

Note that the example application presented in this tutorial does not use the Virus Scan Interface. For more details, call SAP Netweaver Help and refer to the section Setting up Virus Scan Providers [Extern].

Preview

The screenshots displayed below show the three views in the tutorial application.

When the application is launched, the *WelcomeView* appears, where you can navigate to the scenarios *file upload* and *file download:*

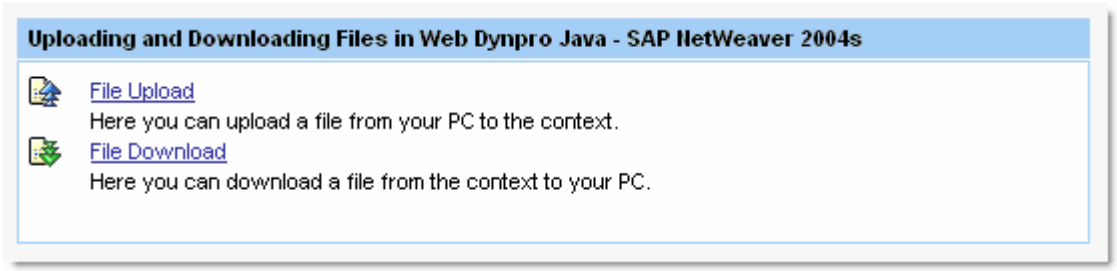


Figure 1: The WelcomeView

In the *FileUploadView*, you can upload a file from your computer to the server-side controller context. Once a MIME object has been uploaded, the system displays details of the uploaded file, such as file name, file ending and file size.

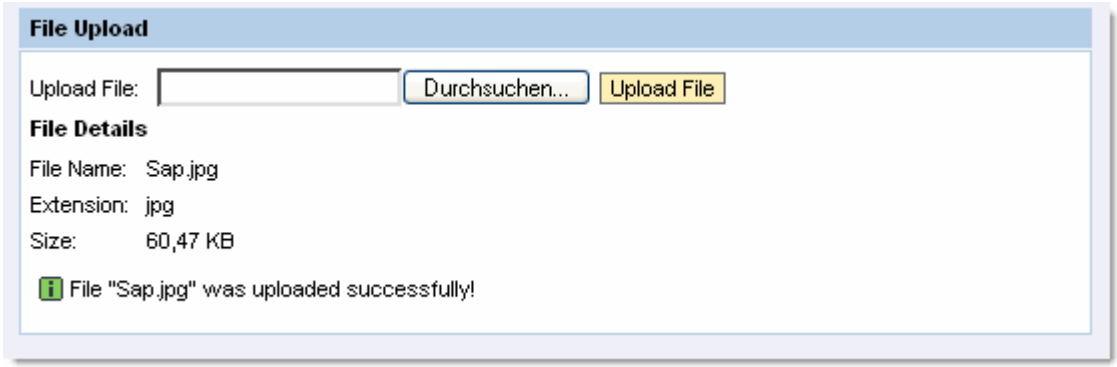


Figure 2: The FileUploadView

In the *FileDownloadView*, you can download an image file, which is deployed in the example project, from the controller context and display it on the user interface.

The file download behavior can be defined by clicking one of three radio button. Figure 3 shows that the file download behavior `ALLOWSAVE` opens a dialog window to save the downloaded resource on the local file system. With the behavior `OPENINPLACE` the resource is instantly displayed in another browser window.

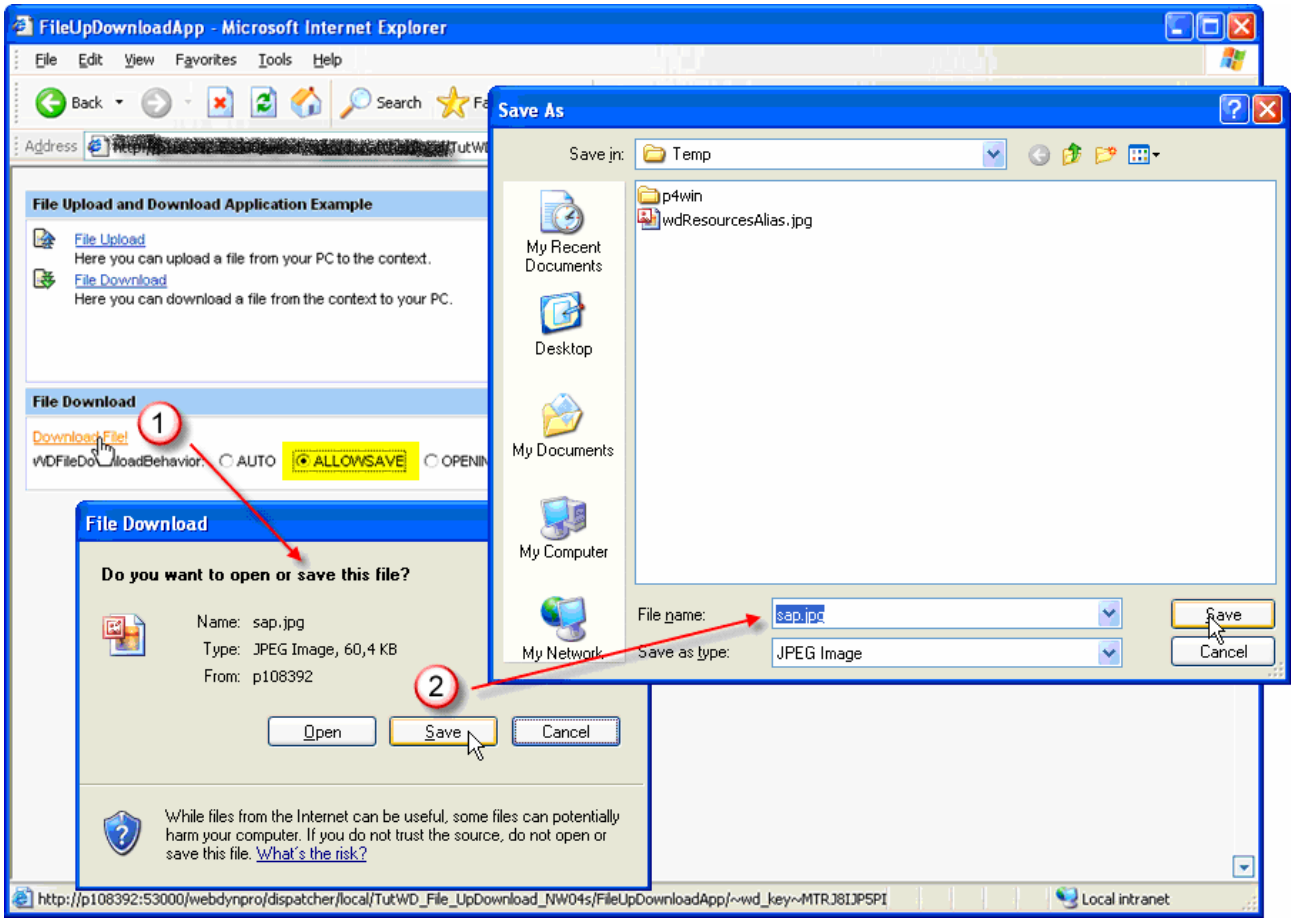


Figure 3: Downloading a file with the file download behavior ALLOW_SAVE

Objectives

By the end of this tutorial, you will be able to:

- Use the UI element *FileUpload*.
- Use the UI element *FileDownload* with different file download behaviors
- Assign the dictionary simple type *Resource* to context attributes so that they can store MIME files.
- Create objects of type `IWDResource` from an image resource deployed with the project by invoking the `WDResourceFactory` API.

Prerequisites

Systems, Installations, and Authorizations

- SAP NetWeaver Developer Studio is installed on your PC.
- You have access to the SAP J2EE Engine.

Knowledge

- Java programming language
- Development of Web Dynpro applications

Importing the Project Template

The SAP Developer Network (SDN) http://sdn.sap.com provides the following Web Dynpro projects for this tutorial:

- The project template *TutWD_FileUpDownload_NW04s_Init* (the starting point for this tutorial)
- The completed Web Dynpro project *TutWD_FileUpDownload_NW04s*

The projects can be downloaded as zip files under *SDN Home* → *Developer Areas* → *Web Application Server* → *Web Dynpro* → *How-To Guides* → *Web Dynpro Sample Applications and Tutorials* .

Prerequisites

- You have access to the SAP Developer Network (http://sdn.sap.com)
- You have installed the SAP NetWeaver Developer Studio.

Importing the Project Template into the SAP NetWeaver Developer Studio

1. Call the SAP Developer Network at http://sdn.sap.com and log on with your user ID and password. If you do not have a user ID yet, you need to register before you can proceed.
2. Choose *Home* → *Developer Areas* → *Web Application Server* → *Web Dynpro* → *How-To Guides* → *Web Dynpro Sample Applications and Tutorials* and then choose the section UI Elements.
3. Navigate to the example application (38) - *Web Dynpro FileUpload and FileDownload*.
4. Download the zip file `TutWD_FileUpDownload_NW04s_Init.zip` containing project template *TutWD_FileUpDownload_NW04s_Init* and save the file to any directory on your local hard disk or directly in the SAP NetWeaver Developer Studio work area.
5. Extract the content of the zip file `TutWD_FileUpDownload_NW04s_Init.zip` in the work area of the SAP NetWeaver Developer Studio or in any directory on your local hard disk.
6. Start the SAP NetWeaver Developer Studio.
7. Import the Web Dynpro project *TutWD_FileUpDownload_NW04s_Init:*
8. In the menu, choose *File* → *Import*…
9. In the next window, choose *Existing Project into Workspace* and click *Next* to confirm.
10. Choose *Browse*, open the folder in which you saved the project *TutWD_ FileUpDownload_NW04s _Init*, and select this project.
11. Confirm by choosing *Finish*
12. The Web Dynpro Explorer now displays the Web Dynpro project *TutWD_FileUpDownload_NW04s _Init*, allowing you to edit it and work through the tutorial.

Tabular Project Structure

Once you have imported the project template into the SAP NetWeaver Developer Studio, you will see the following structure in the Web Dynpro Explorer.

Web Dynpro Project: *TutWD_FileUpDownload_NW04s_Init*
Web Dynpro Application: *FileUpDownloadApp*
Web Dynpro Component: *FileUpDownloadComp*
View: *WelcomeView* In this View, you can choose whether to upload or download a file.
View: *FileUploadView* In this View, you can upload an existing file from your computer. V*iew-Layout:* contains predefined UI elements *Actions:* ToWelcomeView, UploadFile
View *FileDownloadView* In this View, you can download an image file contained in the example application archive. When the View Controller is initialized, this file is stored in the View Controller context. V*iew-Layout:* contains predefined UI elements *Actions:* ToWelcomeView
Window: *FileUpDownloadComp* Contains the three Views described above together with the definition of the View Composition (navigation links between inbound and outbound plugs).
src → *mimes* → *Components* → *com.sap.tc.wd.tut.fileupdownload.comp.FileUpDownloadComp* This directory contains image file `Sap.jpg` that is required for the file download.

Using the UI Elements FileUpload and FileDownload

Once you have successfully imported the initial project *TutWD_FileUpDownload_NW04s*, you can start developing the *file upload* and *file download* functionality.

The further procedure is divided thematically into two separate sections that can be worked through independently of one another:

- File Upload
- File Download

Both sections comprise information on how to define controller context and view layout, how to implement the controllers and how to run the sample application.

Uploading Files

Defining the Context and View Layout

In this step, you define a properly typed context attribute in the *FileUploadView* context and then bind a *FileUpload* UI element to it.

Declaring a Context Attribute of Type Resource

To save a MIME file in the controller context after it has been uploaded from the user interface to the server, you first need to define a context attribute of type *com.sap.ide.webdynpro.uielementdefinitons.**Resource***. The *Resource* type is a special dictionary simple type for MIME resources (figure 4, point 1). At runtime the controller context attribute stores the MIME resource in an object of type `com.sap.tc.webdynpro.services.sal.datatransport.api.`**`IWDResource`** (figure 4, point 2).

The dictionary simple type *Resource* and the `IWDResource`-API was newly introduced in SAP NetWeaver 2004s. The dictionary type *Resource* yields a fully-declarative data transport of MIME resources between Web Dynpro client and controller context as the `IWDModifiableBinaryType`-API must no longer be invoked in your custom controller coding. Furthermore this new simple type makes it possible to store the metadata of a MIME resource on attribute-level but not on attribute info level. Consequently a multiple context node can store MIME resource of different types (Adobe PDF document, GIF image, Microsoft Word Document etc.) in its node elements' context attributes of type *Resource*.

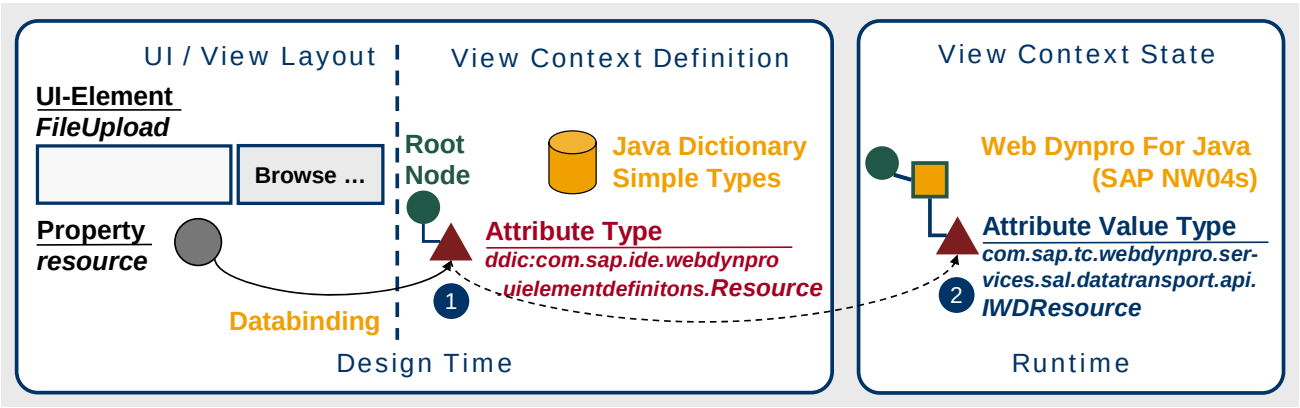
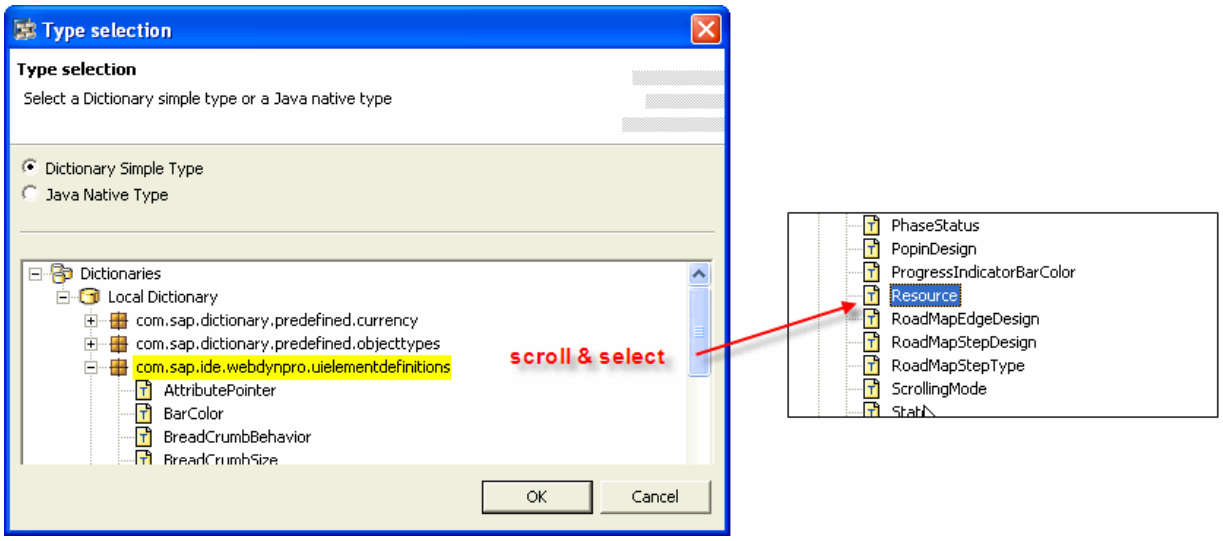


Figure 4: Data binding between FileUpload UI element and context

1. Switch to the *FileUploadView* (*TutWD_FileUpDownload_NW04s_Init* → *Web Dynpro* → *Web Dynpro Components* → *FileUpDownloadComp* → *Views* → *FileUploadView*)
2. Choose the *Context* tab page and insert a *value attribute* with the name **`FileResource`**.

3. Switch to the perspective view *Properties* and press the button ... in column *Value* of property *type* to define the attribute's data type.
4. In the Type selection window select the *Dictionary Simple Type* *com.sap.ide.webdynpro.uielementdefinitons.**Resource*** in the Java dictionary.



Binding a FileUpload UI Element in the View Layout

After defining the context attribute *FileUpload* of type *Resource*, you can bind a new *FileUpload* UI element to it. To trigger uploading of a selected file on the user interface, a *Button* UI element is inserted next to it.

5. Switch to the *Layout* tab page.
6. In the perspective view *Outline*, add a UI element of type *FileUpload* with the name **`FileUpload1`** as a child of the *FileUpload* group. Move this element to below the *Label* UI element *FileUploadLabel*.
7. Under the *FileUpload* UI element, add a *Button* UI element with the name **`UploadButton`**.
8. Define individual UI element properties according to the following table.

UI Element Property	Value
FileUpload1 of Type *FileUpload*	
Element properties – resource	*FileResource* (data binding to context attribute)
Element properties – tooltip	**`Select a file from your file system`**
UploadButton of Type Button	
Element properties – tooltip	**`Upload selected file`**
Event – onAction	*UploadFile*

Result

In this step, you have defined data binding between a *FileUpload* UI element and a context attribute of type *Resource* (see figure 4). To start uploading a file on the user interface, you have also added a new *Button* UI element and bound its *onAction* event to the *UploadFile* action.

Implementing the FileUploadView Controller

Prerequisites

- You have defined a *com.sap.ide.webdynpro.uielementdefinitons.Resource*-type value attribute in the *FileUploadView* context.
- You have bound the property *resource* of the *FileUpload* UI element to the above context attribute.

Implementing the Action Event Handler onActionUploadFile()

After the *UploadFile* action has been triggered on the user interface, the chosen file is transported to the context attribute *FileResource* of type `IWDResource`. All its metadata can be accessed by invoking the `IWDResource`-API (see figure 5).

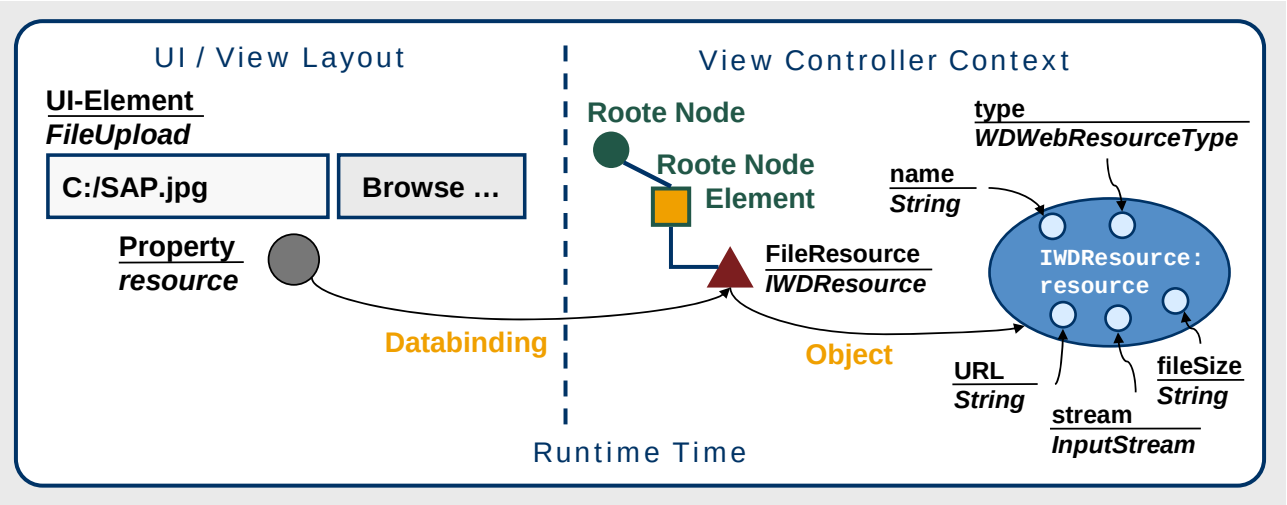


Figure 5: Data binding between FileUpload UI element and context at runtime

1. Switch to the *Implementation* tab page of the view *FileUploadView* and insert the following program code in method `onActionUploadFile()`:

onActionUploadFile() - FileUploadView.java

```
//@@begin javadoc:onActionUploadFile(ServerEvent)
/** Declared validating event handler. */
//@@end
public void onActionUploadFile(
  com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent wdEvent) {
  //@@begin onActionUploadFile(ServerEvent)
  IPrivateFileUploadView.IContextElement element =
    wdContext.currentContextElement();

  //  if a file in the FileUpload field exists
  if (element.getFileResource() != null) {
    IWDResource resource = element.getFileResource();
    // get the size of the uploaded file
    element.setFileSize(this.getFileSize(resource));
    // get the extension of the uploaded file
    element.setFileExtension(
      resource.getResourceType().getFileExtension());
    // set context attribute 'fileName' .
    element.setFileName(resource.getResourceName());
    // set the details visibility attribute
    element.setDetailsVisibility(WDVisibility.VISIBLE);
```

```
    // report success message
    wdComponentAPI.getMessageManager().reportMessage(
      IMessageFileUpDownloadComp.SF_UPLOAD,
      new Object[] { resource.getResourceName()},
      false);
  } else { //if no file in the FileUpload field exists
    // set the 'DetailsVisibility' attribute, hide details
    element.setDetailsVisibility(WDVisibility.NONE);
    // report error message
    IWDMessageManager msgMgr = wdComponentAPI.getMessageManager();
    msgMgr.reportContextAttributeMessage(
      element,
      wdContext.getNodeInfo().getAttribute(
        IPrivateFileUploadView.IContextElement.FILE_RESOURCE),
      IMessageFileUpDownloadComp.NO_FILE,
      new Object[] { "" },
      true);
  }

  // clear the 'FileResource' context value attribute
  element.setFileResource(null);
  //@@end
}
```

2. Add the required import rows using the key combination STRG+SHIFT+O.
3. In the Web Dynpro tools, save the current project metadata by choosing *Save All Metadata*.

The size of the retrieved MIME resource is calculated within the private method `getFileSize()` which is added to the last user coding area between the lines `//@begin` and `//@end`:

getFileSize() – FileUploadView.java

```
//@@begin others
/** Read resource and calculate file size.
 *  @return the file size in Bytes, KB or MB as String */
private String getFileSize(IWDResource resource) {
InputStream stream = null;
    DecimalFormat myFormatter = new DecimalFormat("###.##");
    double size = 0;
    String unit = "";
    try {
      stream = resource.read(false);
      size = stream.available();
      if (size < 1024) {
        unit = " Bytes";
      } else if (size < 1048576) {
        size = size / 1024;
        unit = " KB";
      } else if (size < 1073741824) {
        size = size / 1024 / 1024;
        unit = " MB";
      }
    } catch (IOException e) {
      wdComponentAPI.getMessageManager().reportException(
        e.getLocalizedMessage(), true);
    } finally {
      if (stream != null) {
        try {
          stream.close();
        } catch (IOException e) {
          wdComponentAPI.getMessageManager().reportException(
```



```
            e.getLocalizedMessage(), true);
        }
      }
    }
    return myFormatter.format(size) + unit;
}
//@@end
```

Running the Application - File Upload

Now that you have reached this stage, you can launch the developed example application in the Web Browser as described below, and then carry out a download.

Prerequisites

- You have made sure that the SAP J2EE Engine has been launched.

Procedure

 *Save all Metadata*

1. Save the current status of the metadata for your project.

 *Rebuild Project*

2. In the Web Dynpro Explorer, open the context menu for the project node *TutWD_FileUpDownload_NW04s_Init* and choose Rebuild Project.

 *Deploy new archive and run*

3. Open the context menu for application object  *FileUpDownloadApp* and choose *Deploy new archive and run*.

Result

Test your application by clicking on the *File Upload* link. By choosing the *Browse* button, you can now select a file to upload from your local file system. To start uploading, choose *Upload File*. Once the file has been uploaded, the system displays details of the uploaded file and a success message.

Downloading Files

File Download – Defining the Context and View Layout

In this step, you will define data binding between a *FileDownload* UI element and a *Resource*-type context attribute.

Declaring a Context Attribute of Type Resource

When the file is downloaded, a MIME file saved in the context is downloaded by the client. To do this, first define a new context attribute of dictionary simple type ***Resource*** in the view context.

1. Switch to the *FileDownloadView* (*TutWD_FileUpDownload_NW04s_Init* → *Web Dynpro* → *Web Dynpro Components* → *FileUpDownloadComp* → *Views* → *FileDownloadView*)
2. Choose the *Context* tab page and add a value attribute with the name *FileResource*.
3. Switch to the perspective view *Properties* and press the button ... in column *Value* of property *type* to define the attribute's datatype.
4. In the type selection window select the *Dictionary Simple Type* *com.sap.ide.webdynpro.uielementdefinitons.**Resource*** in the Java dictionary (see section 4, step 4 of this tutorial).

Binding a FileDownload UI Element in the View Layout

After defining the *Resource*-type context attribute *FileResource*, you can bind a new *FileDownload* UI element to it. This UI element is displayed as a link on the user interface. When this link is selected, the MIME file saved in the context attribute is automatically transported to the user interface. With the transfer of the associated MIME type, the connected software on the client side can be started in order to view the MIME file.

5. Choose the *Layout* tab page.
6. In the perspective view *Outline*, add an UI element of type *FileDownload* with the name **`FileDownload1`** as a child of the *FileDownload* group. Move this element in front of the *Button* UI element *BackButton*.
7. Define individual UI element properties according to the following table.

Properties	**Value**
UI Element *FileDownload1* of Type *FileDownload*	
Element properties – data	*FileResource (Data binding to context attribute)*
Element properties – layoutdata	*MatrixHeadData*
Element properties – text	**`Download File!`**

Web Dynpro defines *_blank* as the default value for the property *target* in the UI elements *FileDownload* and *LinkToURL.* Although the *target values* *_top*, *_parent*, and *_self* can be used, you are advised not to do so, as this will cause the Web Dynpro application to crash. Whereas *target* value *_blank* always causes a new target window to open, entering a *target* value like "**`SameWindow`**" causes a downloaded file to always open in the same target window (not the window of the application) if the UI element *FileDownload* is clicked more than once.

In the case of UI element LinkToURL, an exit plug with URL parameters should be used instead of the *target* value *_self*.

Note that no events are bound to an action when using UI element *FileDownload*. Nevertheless the downloadable resource can be retrieved from the server on-demand, this means after it is requested by the user. This on-demand streaming technique is elaborately described in the related SDN article Uploading and Downloading Files in Web Dynpro Tables.

Result

In this step, you have defined data binding between a *FileIDownload* UI element and a *Resource*-type context attribute.

Implementing the FileDownloadView Controller

In this section the controller implementation required for file downloads is restricted to method `wdDoInit()`, which is called when the view controller is created. Unlike file upload, no action event handling takes place after the download process has been triggered by the user. The MIME file selected for downloading is therefore stored in the context at the controller initialization stage.

When using the *FileDownload* UI element as a *table cell editor* you should apply the on-demand stream technique which is described in a separate SDN article Uploading and Downloading Files in Web Dynpro Tables. With this approach you must not initially store all resources in the context before the user actually requests them on client side.

Prerequisites

- You have defined a *Resource*-type value attribute in the *FileDownloadView* context.
- You have bound the property *resource* of the *FileDownload* UI element to the above context attribute.

Implementing the Hook Method wdDoInit()

To allow you to download a file in the example application, the template project *TutWD_FileUpDownload_NW04s _Init* contains the image file `Sap.jpg`. This image file is stored in the project directory *src* → *mimes* → *Components* → *com.sap.tc.wd.tut.fileupdownload.comp.FileUpDownloadComp* and is deployed in the project archive on the SAP J2EE Server.

With the Web Dynpro service class `WDWebResource` you can easily access this image file in the view controller and save it as an object of type `IWDResource` in the context attribute *FileResource* (with dictionary simple type *Resource*).

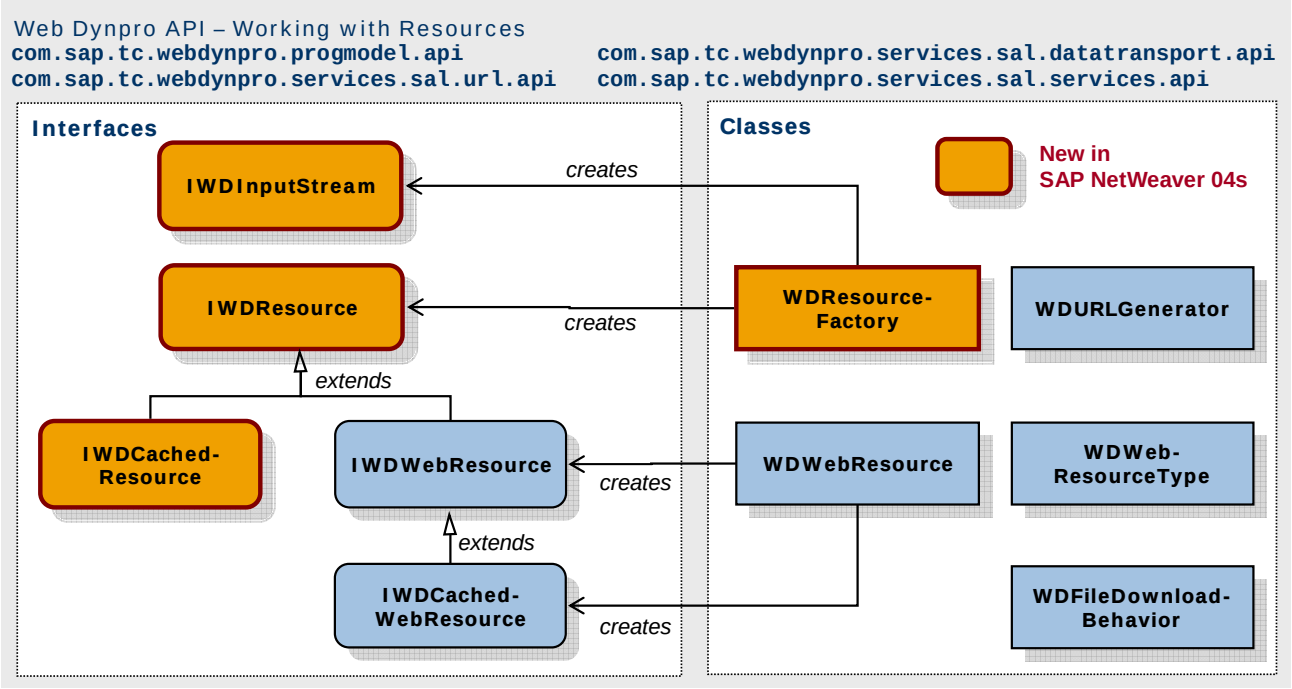


Figure 6: Web Dynpro API – working with resources

Like you can see in the above UML diagram (figure 6) the `WDWebResource` service class returns objects of type `IWDWebResource` which extends the `IWDResource`-API.

1. Switch to the *Implementation* tab page of the view *FileDownloadView* and insert the following program code in method `wdDoInit()`:

wdDoInit() – FileDownloadView.java

```
//@@begin javadoc:wdDoInit()
/** Hook method called to initialize controller. */
//@@end
public void wdDoInit()
{
  //@@begin wdDoInit()
  try {
    // ===== WORKAROUND CODE ============================

    // The image file 'sap.jpg' is deployed with the Web Dynpro project
    // (under src/mimes/Components...). The resource path (URL) for this mime
    // objects can be accessed using the WDURLGenerator service.
    String resourcePath =
      WDURLGenerator.getResourcePath(
        wdComponentAPI.getDeployableObjectPart(), FileDownloadView.FILE_NAME);
    // retrieve resource object for given resource path and create
    // a new object of type IWDResource by invoking the
    // WDResourceFactory API.
    IWDResource resource =
      WDResourceFactory.createResource(
        new FileInputStream(new File(resourcePath)),
        FileDownloadView.FILE_NAME,
        FileDownloadView.FILE_EXT,
        true);
    //  true: flush file input stream so that it gets closed immediately

    // ===== SIMPLE CODE only running in NW 04s Stack 12 ========
    // Based on a bug this simple coding cannot be implemented. The bug
```

```
    // will be fixed within NW04s SP Stack 12.
    /*IWDResource resource =
      WDWebResource.getWebResource(
        wdComponentAPI.getDeployableObjectPart(),
        FileDownloadView.FILE_EXT,
        FileDownloadView.FILE_NAME);*/

    // store resource object in context attribute 'FileResource'
    wdContext.currentContextElement().setFileResource(resource);
  } catch (WDAliasResolvingException e) {
    wdComponentAPI.getMessageManager().reportException(
      e.getLocalizedMessage(), true);
  } catch (FileNotFoundException e) {
    wdComponentAPI.getMessageManager().reportMessage(
      IMessageFileUpDownloadComp.FNF,
        new Object[] {
          wdContext.currentContextElement().getFileResource().getResourceName()},
        true);
  }
  // initialize context attribute 'FileDownloadBehavior'
  wdContext.currentContextElement().setFileDownloadBehavior(
    WDFileDownloadBehaviour.AUTO);
  //@@end
}
```

The name of the image file `sap.jpg` and its MIME type are stored in the two controller class constants `FILE_NAME` and `FILE_EXT`:

```
//@@begin others
// store image file name and file extension in member constants
private static final String FILE_NAME = "sap.jpg";
private static final WDWebResourceType FILE_EXT =
  WDWebResourceType.JPG_IMAGE;
//@@end
```

2. Add the required import rows using the key combination STRG+SHIFT+O.
3. In the Web Dynpro tools, save the current project metadata by choosing *Save All Metadata*.

Based on a bug in the Web Dynpro Java Runtime you cannot easily retrieve the deployed web resource `sap.jpg` by invoking the `WDWebResource`-API: `WDWebResource.getWebResource()`. With this more simple approach a deployed web resource can be retrieved without creating a file input stream for a resource path. The bug will be fixed within NW04s SP Stack 12.

Running the Application - File Download

Now that you have reached this stage, you can launch the completed example application in the Web Browser as described below, and then download image file `Sap.jpg` from the project.

Procedure

Save all Metadata

1. Save the current status of the metadata for your project.

Deploy new archive and run

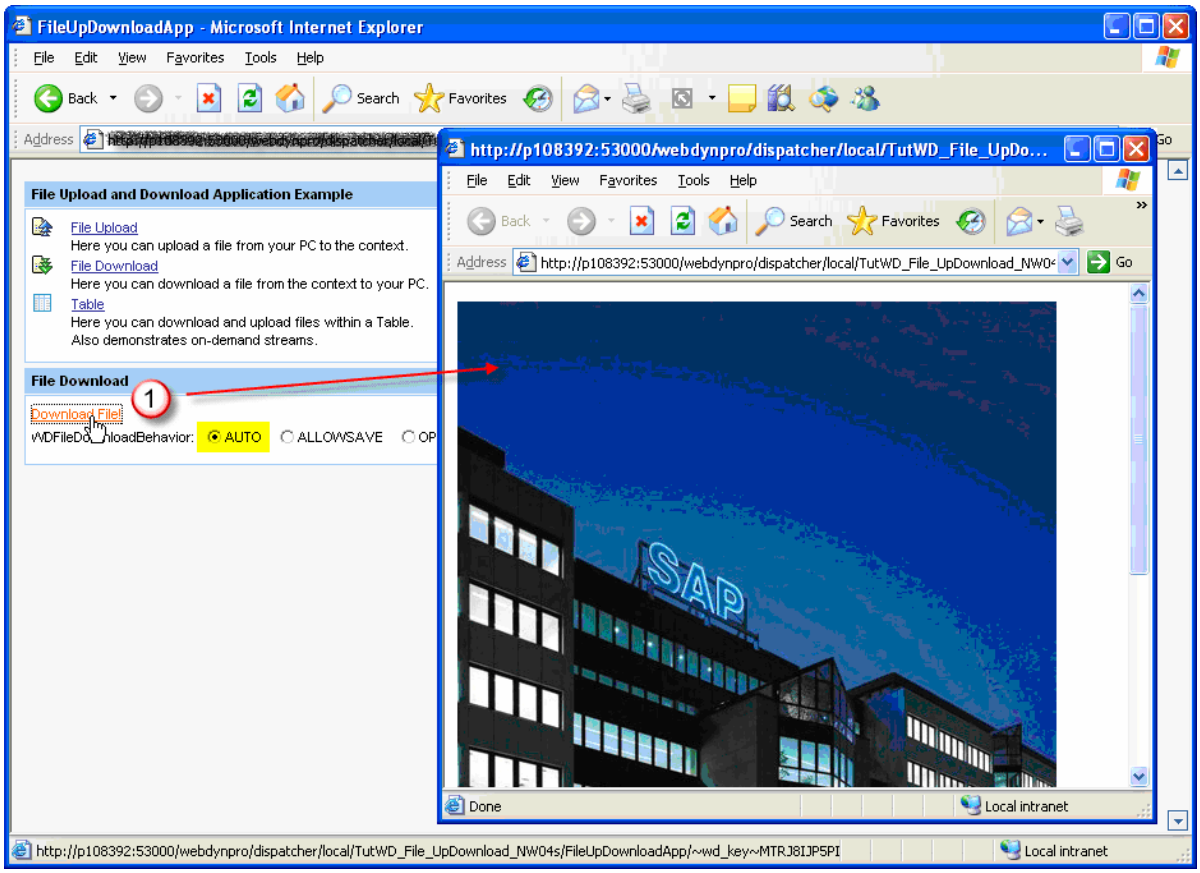
2. Open the application context menu: *Web Dynpro* → *Applications* → *FileUpDownloadApp* and choose *Deploy new archive and run*.

Result

Test your application. To do this, navigate to the File Download view by choosing the *File Download* link. This displays a download link that allows you to download the image file from the example application.

In addition you can dynamically set the *behavior* property of the FileDownload-UI-element. The enumeration type `WDFileDownloadBehaviour` determines how the downloaded file is represented on the client. This property can have three different values:

- **`AUTO`**: The behaviour is predefined and depends on the mime type of the downloaded file.
- **`ALLOW_SAVE`**: An open/save dialog asks the user.
- **`OPEN_INPLACE`**: The file will be opened in place in the web page with the browser-embedded application program.



Graphic 4: Downloading an image file with the file download behavior AUTO

Related Content

SDN Article: Uploading and Downloading Files in Web Dynpro Tables – SAP NetWeaver 04s
SAP Online Help: File Upload and File Download

Copyright