DNNrocket File Download

# Introduction

Download of files from the server can work with a simple link or JavaScript like below.

<script>

function actiondownload() {

window.location ="*<URL>*)";

}

</script>

But sometimes you need more control or security, in which can you can use the simplisity download.

Downloading of files from the server can be achieved by using the “simplisity\_filedownload” class event.

The setup server side and client side must match to make this work.

# **Important**

**To make Simplisity file downloads work, you MUST implement the link within a html element with the “simplisity\_panel” class.** Obviously, you must also link to simplisity JS and activate simplisity by the JS call to “StartUp” or a “single panel”.

**$(document).simplisityStartUp(*string* apiurl)**

*\*On the public view of a website you will need to implement simplisity on the download page.*

# Client Side

We need to create a “a” link on the client page, with the required classes and attributes.

**Example:**

<a class="w3-button simplisity\_filedownload"   
s-cmd="rocketapptheme\_export"   
s-fields='{"appthemefolder":"@appTheme.AppThemeFolder"}'>Download</a>

The most important part of the link is the “simplisity\_filedownload” class. This class tells simplicity that it should be used as a download link.

The “s-cmd” is the server-side command that will be processed to create the data file and then returns the information required to pass the file back to the client browser.

In this example, with have the “s-fields” attribute used to pass data to the server-side command, this data will be different depending of what the server-side code needs to finish processing.

## s-fields

Additional s-field values can be added to the html elment to ensure correct processing. But this are optional, because defaults/configured values are taken by Simplsity.

If you want a different interface and/or system provider to deal with the download you can add the “Interfacekey” and “systemprovider” fields.

You can add s-fields to the html element, each s-field will be added to the calling url as parameters. The length of the url string if the limited of s-fields that can be past.

# Server-Side

Because the s-fields are added to the url as parameters you can therefore access those values on server-side code by using the “genxml/urlparams/\*\*\*” xpath from the paramInfo SimplsityInfo class.

The download in DNN must be made from “api.ashx” in DNNrocketAPI. This is so we match the .Net Framework http context.

There are 2 methods of returning a file.

1 – The file exists on the server

2 – The data for the file is generated in memory in real time and downloaded without being on the server filesystem. (**ONLY small files [< 10MB], but may work with larger files if the server is powerful enough**)

## Method 1 – Existing file on file system

If the return dictionary from the interface API contains an item with a key of “filenamepath” and “downloadname” then a file download will be processed.

**Example**

private static Dictionary<string, string> ExportAppTheme()

{

var appThemeFolder = \_paramInfo.GetXmlProperty("genxml/urlparams/appthemefolder");

var appVersionFolder = \_paramInfo.GetXmlProperty("genxml/urlparams/appversionfolder");

var appTheme = new AppTheme(\_appThemeDataList.SelectedSystemKey, appThemeFolder, appVersionFolder, \_editLang, true);

appTheme.Export();

var rtnDic = new Dictionary<string, string>();

rtnDic.Add("filenamepath", appTheme.ExportFileMapPath);

rtnDic.Add("downloadname", appTheme.AppThemeFolder + ".xml");

return rtnDic;

}

In the above example with are loading AppTheme data. Which AppTheme to load is passed to the server by using the s-fields. We then use the “genxml/urlparams/\*\*\*\*” xpath to get the data we require to load the AppTheme.

The “Export” method of the AppTheme class outputs an export file MapPath to specific place on the server which is then sent to the api.ashx provider call as the dictionary “filenamepath” value.

## Method 2 – Return runtime string

This method is more secure because it does not keep the file on the server. The file must be generated each time, so it is slower, but if you need secure data downloaded, use this method.

If the return dictionary from the interface API contains an item with a key of “downloadfiledata” then a file download will be processed.

**Example**

private static Dictionary<string, string> ExportAppTheme()

{

var appThemeFolder = \_paramInfo.GetXmlProperty("genxml/urlparams/appthemefolder");

var appVersionFolder = \_paramInfo.GetXmlProperty("genxml/urlparams/appversionfolder");

var appTheme = new AppTheme(\_appThemeDataList.SelectedSystemKey, appThemeFolder, appVersionFolder, \_editLang, true);

var rtnDic = new Dictionary<string, string>();

rtnDic.Add("downloadfiledata", appTheme.Export());

rtnDic.Add("downloadname", appTheme.AppThemeFolder + ".xml");

return rtnDic;

}

In the above example with are loading AppTheme data. Which AppTheme to load is passed to the server by using the s-fields. We then use the “genxml/urlparams/\*\*\*\*” xpath to get the data we require to load the AppTheme.

The “Export” method of the AppTheme class returns the export data which is then sent to the api.ashx provider call as the dictionary “downloadfiledata” value.