Command Tracking

# Introduction

DNN Rocket “can” implement command tracking. A command tracking system remembers the last command (or required reload command) that was issued by the user.

The tracking system can then “re-run” the last tracked command when the page is reloaded. The reason is simple: we want to refresh the page and not return to the entry point of the page, we want to return to what we were working on.

# Data Storage

DNN Rocket uses the database to store the command data on a per user basis, usually the tracking is only implemented for admin pages.

Database Table: **DNNrocketTemp**

EntityTypeCode: **USERSTORAGE**

*But keep in mind, large systems with many users may create a large amount of data for tracking.*

The tacking data is controlled server side, using the **UserStorage** class.

This class is used to store temporary user data on the database, for persistence across sessions.

private UserStorage \_userStorage;

\_userStorage = new UserStorage();

The UserStorage class keeps a copy of the command, rocketinterface and paramInfo data and uses this to re-run the command.

## Example

public string InitCmd(string paramCmd, SimplisityInfo systemInfo, SimplisityInfo interfaceInfo, SimplisityInfo postInfo, SimplisityInfo paramInfo, string langRequired = "")

{

\_systemData = new SystemData(systemInfo);

\_rocketInterface = new DNNrocketInterface(interfaceInfo);

\_userStorage = new UserStorage();

\_postInfo = postInfo;

\_editLang = DNNrocketUtils.GetEditCulture();

\_paramInfo = paramInfo;

if (!CheckSecurity(paramCmd))

{

paramCmd = "securityerror";

}

else

{

if (\_paramInfo.GetXmlPropertyBool("genxml/hidden/reload"))

{

var menucmd = \_userStorage.GetCommand(\_systemData.SystemKey);

if (menucmd != "")

{

paramCmd = menucmd;

\_paramInfo = \_userStorage.GetParamInfo(\_systemData.SystemKey);

var interfacekey = \_userStorage.GetInterfaceKey(\_systemData.SystemKey);

\_rocketInterface = new DNNrocketInterface(systemInfo, interfacekey);

}

}

else

{

if (\_paramInfo.GetXmlPropertyBool("genxml/hidden/track")) \_userStorage.Track(\_systemData.SystemKey, paramCmd, \_paramInfo, \_rocketInterface.InterfaceKey);

}

}

return paramCmd;

}

As shown in the code, the tracking, in this case, is activated by passing a parameter of “track”, if this is set to true the command action is tracked.

## Example

<div class="w3-col m2 rocket-buttonbar w3-button w3-khaki simplisity\_click" s-cmd="clientimagesusers\_userlist" s-fields='{"track":"true"}'>@ButtonText(ButtonTypes.back)</div>

# Forced Tracking

Tracking can also be activated when you want to remember a server-side action. For example, when you delete a record you DO NOT want to remember that command, it could easily cause an error. So, therefore you can force the tracking command to be what you need.

## Example

\_userStorage.TrackClear(\_systemData.SystemKey);

\_userStorage.Track(\_systemData.SystemKey, "clientimagesusers\_userlist", new SimplisityInfo(), "clientimagesusers");

On delete action the list will be returned, but the return command can be tracked, so on refresh you get the list again.