



FirstSpirit™

Unlock Your Content

FirstSpirit™ DynamicPersonalization FirstSpirit™ Version 5.2

Version	3.1
Status	RELEASED
Date	2018-03-13
Department	FS-Core
Copyright	2015 e-Spirit AG
File name	DPER_EN_FirstSpirit_DynamicPersonalization

e-Spirit AG
Stockholmer Allee 24
44269 Dortmund | Germany

T +49 231 . 477 77-0
F +49 231 . 477 77-499

info@e-spirit.com
www.e-spirit.com

e-Spirit

Table of contents

1	Introduction.....	5
1.1	Overview of the functions.....	5
1.2	Topics covered in this document.....	7
1.3	Terms and concepts.....	7
1.3.1	Single sign-on concept.....	7
1.3.2	FirstSpirit PermissionService.....	8
1.3.3	Login package.....	10
1.3.4	Login module.....	12
1.3.5	Authentication module.....	12
1.3.6	Group module.....	12
1.3.7	Attribute module.....	13
2	Installation and Configuration.....	14
2.1	Installing the FirstSpirit DynamicPersonalization module.....	14
2.2	Installing the web application in the project.....	16
2.3	Configuring the web application.....	18
2.3.1	Configuring the FIRSTpersonalisation-MappingFilter.....	20
2.4	Configuring a login package.....	23
2.5	Further configuration options (web.xml).....	23
2.5.1	Filter: NtlmPreAuthenticationFilter.....	24
2.6	Further libraries.....	24
3	Modules.....	26
3.1	Login modules.....	26



3.1.1	"Kerberos Login" login module	27
3.1.2	"NTLM Login" login module	29
3.1.3	"Anonymous NTLM Login" login module	31
3.1.4	"Request Parameter Login" login module	32
3.1.5	"Request Header Login" login module	32
3.1.6	"Portlet Login" login module	33
3.1.7	"SAP Portlet Login" login module	33
3.1.8	"FS SSO" login module	33
3.2	Authentication modules	34
3.2.1	"JDBC" authentication module	35
3.2.2	"LDAP" authentication module	37
3.2.3	"ORMapper" authentication module	39
3.2.4	"PropertyFile" authentication module	40
3.2.5	"Permission Service" authentication module	41
3.3	Group modules	42
3.3.1	"JDBC" group module	43
3.3.2	"LDAP Group" group module	44
3.3.3	"LDAP Group Iterate" group module	45
3.3.4	"ORMapper" group module	47
3.3.5	"Portlet" group module	48
3.3.6	"SAP Portlet" group module	48
3.3.7	"FS SSO Groups" group module	48
3.3.8	"Permission Service" group module	48
3.3.9	"ExternalSAP" group module	49
3.3.10	"ExternalSAP73" group module	49
3.4	Attribute modules	51
3.4.1	"LDAP" attribute module	52



3.4.2	"ORMapper" attribute module.....	54
3.4.3	"ExternalSAP" attribute module	54
3.4.4	"ExternalSAP73" attribute module.....	55
4	Servlets	56
4.1	"Login Servlet"	56
4.2	"Logout Servlet"	58
5	Tags.....	59
5.1	Prefix	60
5.2	<fsp:loginRequired>	60
5.3	<fsp:authorize>	61
5.4	<fsp:userInfo>.....	62
5.5	<fsp:userGroups>.....	63
5.6	<fsp:userAttributes>	63
5.7	<fsp:isAuthorized>	65
5.8	<fsp:isNotAuthorized>.....	68
5.9	<fsp:logout>	68
6	Functional Enhancement.....	69
6.1	Session variables	69
6.2	Interface "User"	69
6.2.1	"getLogin()" method	70
6.2.2	"getGroups()" method.....	71
6.2.3	"addGroup(String)" method	71
6.2.4	"setGroups(List<String>)" method.....	71
6.2.5	"isInGroup(String)" method.....	72



6.2.6	"isInGroups(String)" method	72
6.2.7	"getAttributes()" method	72
6.2.8	"getAttribute(String)" method	73
6.2.9	"addAttributes(Map<String, String>)" method.....	73
6.2.10	"setAttribute(String, String)" method.....	73
6.2.11	"clearAttributes()" method	74
7	Configuring with JOSSO (Java Open Single Sign-On)	75
7.1	Configuring personalization.....	75
7.2	Configuring the FirstSpirit project.....	75
7.2.1	Via structure variables or project settings	75
7.2.2	Configuring the personalized pages (Page Store)	76
8	Legal notices	78



1 Introduction

The FirstSpirit™ DynamicPersonalization module can be used to personalize the display of FirstSpirit content. To this end, the module provides different options for logging in, authentication and reading out user-specific information (group membership and other user-specific attributes, e.g. eMail, phone number, etc.), which can be combined with each other in any way required. The user information (e.g. login name and group membership) can originate from the FirstSpirit environment (FirstSpirit™ SSO) or from an external system (including NTML, LDAP, etc.).

Display of the personalized content can be configured via a special Java tag library (the FirstSpirit Personalization tags). The content for specific users or groups is displayed or hidden depending on the tag choice and configuration.



This document is provided for information purposes only. e-Spirit may change the contents hereof without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. e-Spirit specifically disclaims any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. The technologies, functionality, services, and processes described herein are subject to change without notice.

1.1 Overview of the functions

The FirstSpirit™ DynamicPersonalization module supports the following aspects:

Login and authentication via external systems: The FirstSpirit™ DynamicPersonalization module supports login and authentication via external systems. To this end, FirstSpirit™ DynamicPersonalization provides different group, login and authentication modules which enable the FirstSpirit user to log in via external systems such as LDAP, Kerberos, NTLM or from a portal environment. The authentication is usually made in the form of a cookie, a ticket or an HTML form. The login information there is used to try to authenticate the user at the FirstSpirit server.

Login and authentication via external or internal data sources: Apart from authentication via an external system (see above), the authentication information can also be provided by a link to an external or internal data source (a FirstSpirit project). Here login information is



deposited in a database via the FirstSpirit Content Store and is used as the basis for authenticating the user (in the case of a link to an internal data source). If the link is to an external data source the login information can be maintained, e.g. via a JSP page. (For further information on linking data sources, see "FirstSpirit Documentation for Administrators", Chapter 4.9).

SSO: The determined login information can be further used within the FirstSpirit environments (see Chap. 1.3.1 page 7). For authentication via FirstSpirit™ DynamicPersonalization, SSO means that the login information of the current user is copied from FirstSpirit or from an external system (e.g. LDAP) and takes the place of login and password in FirstSpirit. This way SiteArchitect or ContentCreator can be started from any environment and the logged in user is authenticated on the FirstSpirit server without renewed login dialog.

Using attributes and group assignments from external systems: Apart from pure authentication of a user via an external system (e.g. LDAP), you can also copy information deposited for a user, e.g. attributes and group assignments. To this end, FirstSpirit™ DynamicPersonalization provides group and attribute modules (see Chapter 1.3.5 f.).

Link with the FirstSpirit PermissionService: FirstSpirit provides a module for the issue and evaluation of user permissions for the generated content. User permissions are assigned for the "visitor" of the generated site and are therefore always linked to the personalisation system used (unlike the editorial permissions). If FirstSpirit™ DynamicPersonalization is used as the personalization system (which is not necessarily so), a very close relationship can be established (see Chapter 1.3.2.3 page 9). (For further information on configuration see "FirstSpirit Documentation for Administrators" Chap. 13)

Personalized access to pages: Personalized access to specific navigation areas or pages can be realised using FirstSpirit™ DynamicPersonalization. The personalized content can be directly configured for a specific user or to a specific group through the user's membership. In this way, for example, it is possible to realise personalized access of a user to customer areas or employee areas.

Personalized access for individual content areas: Apart from personalized access to whole navigation area or pages, individual content within a page can also be personalized. For example, a link within a page ("Open page for editing in SiteArchitect") can be displayed for certain users and groups and can be hidden for other users and groups.



1.2 Topics covered in this document

Chapter 2: Describes the configuration and installation of the FirstSpirit DynamicPersonalization module (from page 14).

Chapter 3: Introduces the different modules for logging in, authentication, group membership of the users and access to user attributes and explains configuration of the modules using parameters (from page 26).

Chapter 4: Describes how to use the “Login” and the “Logout” Servlets (from page 56).

Chapter 5: Describes the tags from FirstSpirit DynamicPersonalization Taglib and explains the configuration and way they function by way of several examples (from page 59).

Chapter 6: Handles possibilities for specifically adapting the functionality of FirstSpirit DynamicPersonalization to the requirements of a project (from page 69).

Chapter 7: A special case for use of FirstSpirit DynamicPersonalization is configuration with JOSSO (Java Open Single Sign-On)¹. The chapter explains all the necessary installation and configuration steps (from page 75).

1.3 Terms and concepts

1.3.1 Single sign-on concept

The term Single Sign-On (SSO) stands for the option of authentication for a number of applications with a single login instead of having to perform this login procedure for each individual application.

For use of the FirstSpirit DynamicPersonalization module, SSO means that the login information of the current user is copied from FirstSpirit or from an external system (e.g. LDAP) and takes the place of login and password in FirstSpirit. This way SiteArchitect or ContentCreator can be started from any environment and the logged in user is authenticated on the FirstSpirit server without renewed login dialog.

¹ <http://www.josso.org/>



This is realised technically with the help of a cookie (e.g. FirstSpirit SSO) or a ticket (e.g. SAP portal). This is provided by FirstSpirit and contains all the necessary information such as the user name, login time and expiry date. The FirstSpirit server can accept and verify this cookie or ticket using a special login configuration.

1.3.2 FirstSpirit PermissionService

In FirstSpirit, user permissions (assignment of permissions for the generated content) are issued and checked via the FirstSpirit Permission module. The module is included in the standard scope of supply of FirstSpirit and is available directly after installation.

The module consists of two components:

- Editor (permission definition component) – (see Chapter 1.3.2.1 page 8)
- Service (PermissionService) – (see Chapter 1.3.2.2 page 8)

For details of use of FirstSpirit DynamicPersonalization with the permissions definition component see Chapter 1.3.2.3 page 9.

1.3.2.1 Permissions Definition component (CMS_INPUT_PERMISSION)

The permissions definition component can be used to extend content in SiteArchitect or ContentCreator to the definition of user permissions (for the generated content). The component is usually used as an input component in the Metadata tab. The component is assigned parameters (project-specific in the metadata form) with a unique group document and works with the appropriate service which loads and makes available the group definitions from the server (see Chapter 1.3.2.2).

For details of using the permissions definition component see Manual "FirstSpirit SiteArchitect".

1.3.2.2 Permission Service

The Permission Service is a server component which can be addressed via the permissions definition component. The Permission Service is a special service of the FirstSpirit server whose task is to manage group and user configurations. As a system service the Permission Service can be activated via FirstSpirit ServerMonitoring (FirstSpirit / Control / Services) or via ServeManager (Server Properties / Modules /System Module).

For further information on starting and stopping system services see "FirstSpirit Documentation for Administrators".



1.3.2.3 Using FirstSpirit DynamicPersonalization with the permission definition component

The FirstSpirit DynamicPersonalization module consists of a Java Tag library which can be used to personalize JSP pages, i.e. display of the page can be either completely or partly prevented (see Chapter 5 page 59). Whether an (sub-)area of a page is visible or not depends on whether the logged in user has adequate authorisation (permissions) or not. This is decided by comparing the specified permission for the page with the user's actual permissions.

In this case the specified permission (who may see a document or part of a document?) is defined using the permission definition component. The evaluation, i.e. comparison of the actual configuration (resulting from the user's login context) with the specified configuration, is performed via FirstSpirit DynamicPersonalization. There is a very close (logical) relationship between the groups used in the permission definition component and the groups to which the personalization group module relates.

Example: If a user belongs to a group which does not appear in the rights definition component, it may be that it is not possible to set any permissions for them.

This relationship between the group model in the permission definition component and in the personalization module can be established in different ways:

1. External source: Generation of the group definition file for the permission definition component and the personalization group module evaluates the same external data sources (e.g.: LDAP server or AD server). In this case data ownership and control is completely external.
2. Internal source: The permission definition component generates a group definition file and the personalization module uses this data or refers to this file. The file can be generated from the permission definition component by querying a LDAP server. The difference is that the personalization module itself does not query the LDAP server, but the database generated by the permission definition component is used instead. Groups can also be defined directly using FirstSpirit, for example, via the Content Store.

While in the first case the personalization group module is configured in "JDBC" or "LDAP" modes (and therefore the runtime environment requires a permanent link with the data source), in the second case the group module is configured in "GroupService" and then uses the same XML files as the permission definition component. Therefore, permanent access to an external resource is not required in the second case. However, it must be noted that with the deployment configuration the group configuration files are also deployed (published) in the live system.



1.3.3 Login package

FirstSpirit DynamicPersonalization is divided into four different types of module:

- Login modules for determining login information
(see Chapter 1.3.4 page 12).
- Authentication modules for evaluating the determined login information
(see Chapter 1.3.5 page 12)
- Group modules for reading out a user's group information (see Chapter 1.3.6 page 12)
- Attribute modules for reading out user attributes
(see Chapter 1.3.7 page 13)

"Login Package" is the term used to describe the combination of a login module, an authentication module and optionally a group module and/or an attribute module. A login package can therefore also only be formed from one login and one authentication module even if group information and attributes are not required. Depending on the login module used, the authentication module can also be dispensed with in several cases. For example, evaluation of the login data is not required with the NTLM login module as this takes place when the login data is read out.



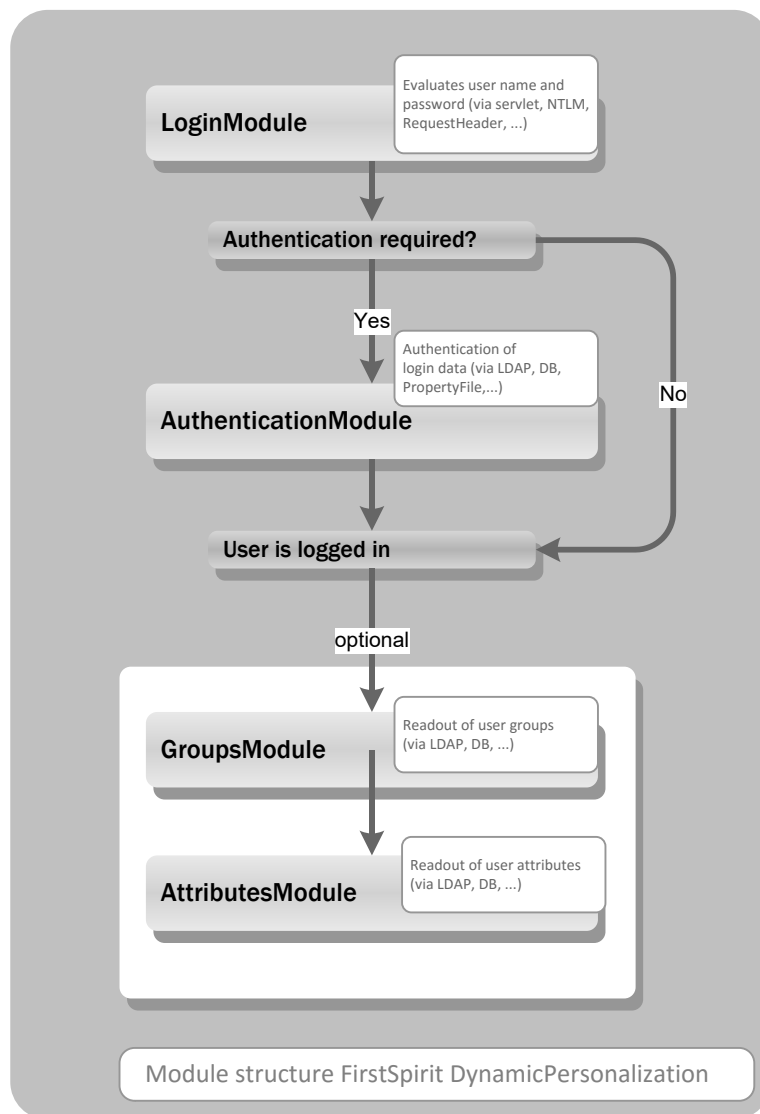


Figure 1-1: Modular structure of FirstSpirit DynamicPersonalization

During authentication an attempt is first made to authenticate the user against the first login package (with the highest priority). If this is unsuccessful an attempt is made to log in the user via the login package with the second highest priority, etc., until the login attempt is either successful or, after the login package with the lowest priority, the attempt has to be aborted. The first login package with which the user can be successfully authenticated "wins" (see Chapter 2.3 page 18)

Further information on the modules, their dependencies and on detailed configuration is given in the following chapters.



1.3.4 Login module

The login module is a mandatory module. This means each login package must contain a login module:

The login module determines the user information required to authenticate the user, usually their login name and password. However, it can also involve a cookie (e.g. for authentication through FirstSpirit SSO) or a ticket (e.g. for authentication through Kerberos, NTLM or the SAP portal). Based on this data the authentication is then carried out by the authentication module (see Chapter 1.3.5 page 12).

For configuring the login module see Chapter 3.1 page 26.

1.3.5 Authentication module

Depending on the login module used, the authentication module is either a mandatory module or it can be used optionally. Through the authentication module, on the basis of the login information determined (from the login module), the user can be authenticated against an external system, a FirstSpirit database or against the FirstSpirit server (see Chapter 1.3.4.2 page 12). The authentication entity depends on the authentication module chosen (e.g. "LDAP" – the user is authenticated against the LDAP server).

However, it is possible that the login information determined has already been authenticated elsewhere so that subsequent authentication is no longer necessary.

For details of configuration of the authentication module see Chapter 3.2 page 34.

1.3.6 Group module

The group module can be optionally used to read out group information for the logged in user. Group information is only available for a user if this information exists in the external system (or via FirstSpirit) and the user has already been successfully authenticated.

Use of optional group modules is only necessary if a differentiation is to be made between individual groups, for example if display of certain areas of a page is reserved for specific groups. In this case the group membership of the logged in user can be used to control the display.

For details of configuration of the group module see Chapter 3.3 page 42.



1.3.7 Attribute module

The attribute module can be optionally used to read out further attributes for the logged in user. This can involve, for example, their phone number, eMail address, room number, etc. These attributes are only available for a user if this information exists in the external system (or via FirstSpirit) and the user has already been successfully authenticated.

For details of configuration of the attribute module see Chapter 3.4 page 51.



2 Installation and Configuration

FirstSpirit DynamicPersonalization is primarily configured through the FirstSpirit ServerManager.

2.1 Installing the FirstSpirit DynamicPersonalization module

The FirstSpirit DynamicPersonalization module must first be installed via FirstSpirit ServerManager. To this end, the "Modules" menu entry is selected in the Server properties area. Click the "Install" button to open a file selection dialog. The fsm file (fs-perso.fsm) to be installed can be selected here. The successfully installed file is then displayed in the "Server Properties" dialog:

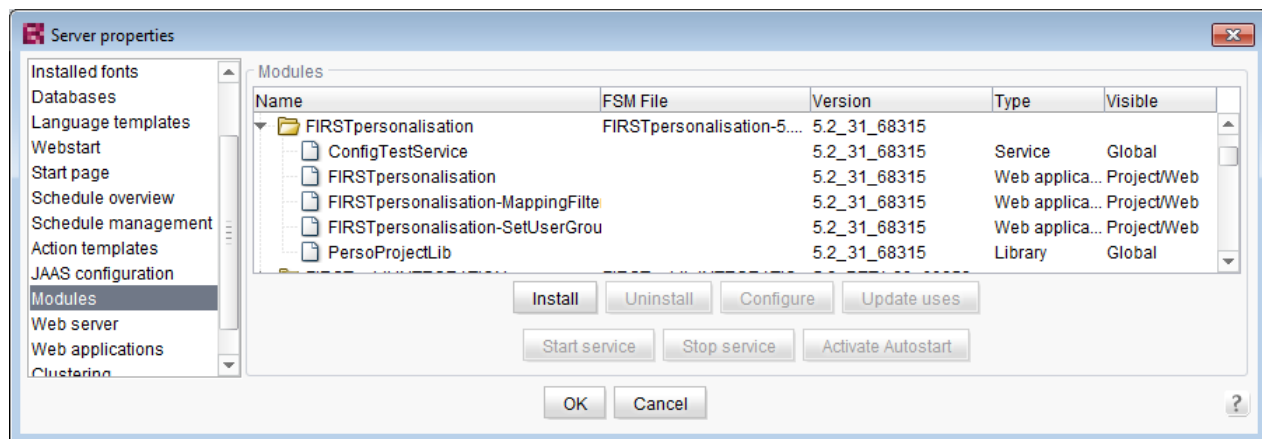


Figure 2-1: Installing the module on the FirstSpirit server

The global library "PersoProjectLib" (see Chapter 5 page 59) is part of the FirstSpirit DynamicPersonalization module and contains the classes which, following installation on the FirstSpirit server, are available within SiteArchitect and ContentCreator, in scripts and other modules.

In addition, the module contains a global service "ConfigTestService" for testing the configuration (for an example see Chapter 3.2.2 page 37) and two web applications which can be installed in the respective project.

The web application "FIRSTpersonalisation" provides JSP tags (see Chapter 4 page 56) and servlets (see Chapter 5 page 59) which can be used and opened within the project.



The web application "FIRSTpersonalisation SetUserGroups" can be used to create a test environment for the editor. The web component enables definition of user groups for the logged in user. For example, if an "Editor" is authenticated through "FirstSpirit SSO" against the FirstSpirit server, the web component provides the means for displaying the current page for a user in the "Customers" group. The security mechanisms of the preview servlet take effect within the preview environment. This means a user's permissions can be restricted through the web application but cannot be extended (this does not apply to the staging or live environment). The user of the "Editors" group can therefore, for example, view the "Customers" areas of the website, but not the current business reports of the "Managers" group for which, as an "Editor" they do not have any rights or permissions.



The "FIRSTpersonalisation SetUserGroups" web application merely provides the features to enable the definition of user groups. This must be realised in the project (evoke the servlet) by the template developer on a customer-specific basis for the respective project.

The **FIRSTpersonalisation-MappingFilter** web application can be used to protect project content from unauthorized access. If the component is added within the project, the RequestFilter in the project can be used to protect the configured URLs. The URLs defined here can only be opened if the user has special permissions (see Chapter 2.3.1 page 20).

Both components are "visible" for the "Project/Web" areas. It is therefore a "local web" component. After installation, this can be added to the different web areas (Previews, Staging, Live, ContentCreator) within the required projects (see Chapter 2.2 page 16).

A detailed description of installing a module is given in the "FirstSpirit Documentation for Administrators".



2.2 Installing the web application in the project

The web application must now be installed in the required project. Open the "Web Components" menu entry within the project properties. The web components for a project can be activated in this area.

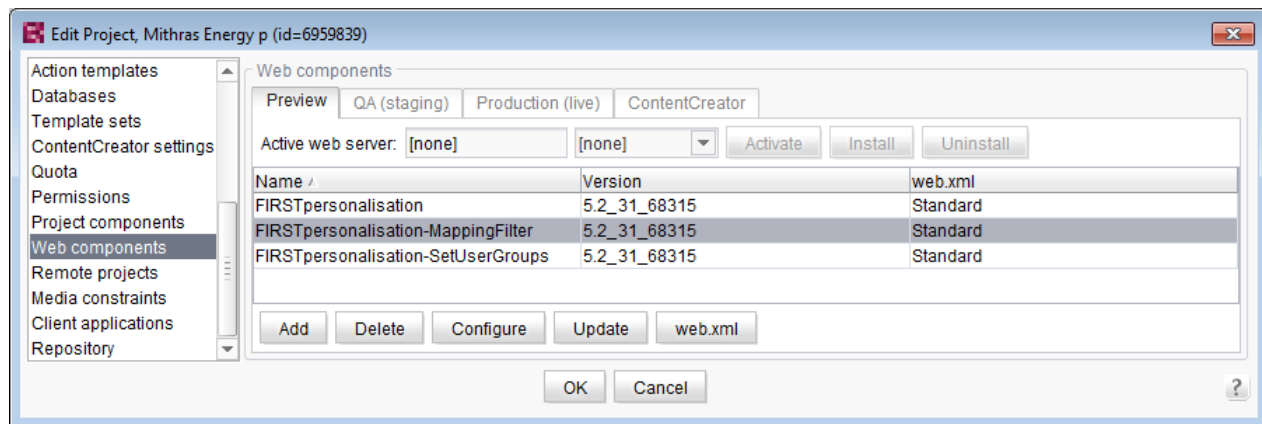


Figure 2-2: Installing the web application within the web areas

Different web areas exist. The web components for each area can be individually activated and configured via the respective tab:

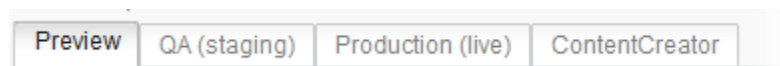


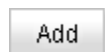
Figure 2-3: Web areas within a project

Preview: Project content location for which a preview has been requested.

QA (staging): Location for the generated project content

Production (live): Location for the published project content

ContentCreator: Location for the ContentCreator project content



Click the button to open the "Add" dialog. The list displayed contains all web components installed on the server (see Chapter 2.1 page 14).

Three web applications can be added for the FirstSpirit DynamicPersonalization module:

- **FIRSTpersonalisation**: The default web component of the FirstSpirit DynamicPersonalization module. This provides JSP tags (see Chapter 4, page 56) and servlets (see chapter 5, page 59) that can be used and called within the project.
- **FIRSTpersonalisation-MappingFilter**: In addition to the FIRSTpersonalisation function, a RequestFilter can be used to allow or deny individual groups access to particular URLs



(see Chapter 2.3.1, page 20).

- FIRSTpersonalisation-SetUserGroups: In addition to the FIRSTpersonalisation function, the SetUserGroups function can be used to set up a test environment with different user groups for an editor who is logged in (see Chapter 2.1, page 14).

After adding it to a web area it is possible to configure the components, either with a GUI generated by the component or a generic GUI (see Chapter 2.3 page 18). After configuring the components must be activated. A component within a project can be activated or deactivated for specific areas only

For further information on this dialog see "FirstSpirit Documentation for Administrators".



2.3 Configuring the web application

Configure

Click the button (cf. Figure 2-2) to configure the component within the web area. New login packages can be created in configuration dialog and existing ones can be edited:

Name	Priority
SSO_FIRSTPersonalisation_Mithras Energy p	0
NTLM_FIRSTPersonalisation_Mithras Energ...	1

Figure 2-4: Configuring the Integration web application

☒ Manual configuration

Configure

If the "Manual configuration" checkbox is selected, the "web.xml" file of the web application can be opened by clicking the button (see Chapter 2.5 page 23).





New login packages can be defined by clicking the button (see Chapter 2.4 page 23).

Name: Unique name of the login package. This name is issued by the administrator on adding a login package.

Priority: The priority (natural number ≥ 0) determines the order in which the login packages are worked through: When a user tries to log in the system first tries to log them in through the login package with the highest priority (priority "0"). If this attempt fails an attempt is made to log in the user through the login package with the second highest priority, etc., until the login attempt is either successful or, after the login package with the lowest priority, the attempt has to be aborted.

Example: Several login packages are given which are used in the order of their priority to authenticate the user. For example, the FirstSpirit DynamicPersonalization module could be configured so that a user is authenticated with first/highest priority through the "ORMapper". If this authentication attempt fails the login package with the next highest priority is used - this could be, e.g. authentication against LDAP. If this also fails the next login package is used depending on the priority (e.g. "Property File"). As long as the user cannot be successfully authenticated the login package with the next highest priority is always used until finally, after the login package with the lowest priority, the login attempt is aborted. In this case the user cannot be logged into the system.

If a login package has performed successful authentication, for example the second login package in the above example which authenticates against LDAP, the login packages with lower priority are ignored and the user is logged in.

Apart from defining the login packages, this dialog can be used to make other global settings:

Use dummy user: If this checkbox is selected (activated) the complete personalization is switched off as each user is automatically logged in as a pseudo user. This setting may be needed for testing, to view pages which may only be viewed by logged in users, but in which there is not yet any link to user checking. As a default the checkbox is not selected.

Activate group 'Everyone': If this checkbox is selected, each logged in user is automatically assigned to a specific group. This ensures that all users belong to a common group. The required group is defined using the "Group 'Everyone'" parameter. As a default the checkbox is not selected.

Group 'Everyone': The "Group 'Everyone'" input field can be used to specify the group to be used for the "Everyone" group. Any valid group can be given as the value. As a default the value for the "Everyone" group is: "*". If the "Group 'Everyone'" checkbox is selected each logged in



user is automatically a member of the "Everyone" group.

LOG4J default configuration file: This parameter gives the absolute path to the configuration file for logging and is needed in case there is no Log4j preconfigured by the application server.

SSO Cookie Domain: Domain for which the cookie is valid. The value entered here is only relevant if the "Create Cookie" checkbox was selected for the authentication module used. This setting can also be configured in the login module if an authentication module is not necessary (e.g. Kerberos or NTLM login).

SSO Cookie Time-to-Live: Time-to-live (TTL) of the cookies or time at which it is automatically deleted. The value entered here is only relevant if the "Create Cookie" checkbox was selected for the authentication module used.

SSO cookie name: Any name for the cookie. The value entered here is only relevant if the "Create Cookie" checkbox was selected for the authentication module used.

SSO Cookie Secure: Checking this box activates a security setting for the cookie. This then means that the cookie can only be sent to the server over HTTPS. The value entered here is only relevant if the "Set cookie" box has been checked for the authentication module that is being used. By default, the box is not checked.

SSO Cookie HttpOnly: Checking this box prevents access to the cookie via JavaScript. This setting can potentially provide protection against XSS provided that the browser used supports the HttpOnly attribute. The value entered here is only relevant if the "Set cookie" box has been checked for the authentication module that is being used. By default, the box is not checked.

2.3.1 Configuring the FIRSTpersonalisation-MappingFilter

The "FIRSTpersonalisation-MappingFilter" contains a RequestFilter. This filter is assigned to the /* URL mapping and filters all incoming URL requests.

Depending on the module configuration (or its "FIRSTpersonalisation" default web component), the filter first attempts to force a login. As soon as the login is successful and information about the logged in user is available, the filter configured here can be evaluated:

Before configuration, the local project application "FIRSTpersonalisation-MappingFilter" needs to be installed in the project in the same way a standard application would be installed (see Chapter 2.2, page 16).



Configuring the web component:

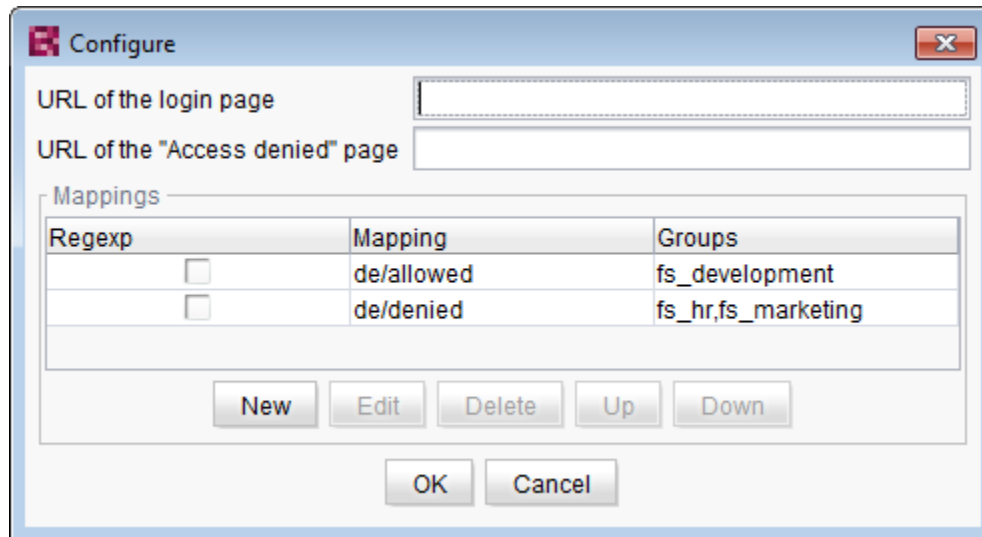


Figure 2-5: Configuring the MappingFilter (project properties)

URL of the login page: Configurable URL for the login page. The URL configured here is displayed when the conventional login process using the "FIRSTpersonalisation" default project application (for example via NTLM, Kerberos, etc.) fails.

URL of the "Access denied" page: Configurable URL for the Access Denied page. The URL configured here is displayed when the login process via the "FIRSTpersonalisation" default project application was successful, but the authenticated user does not have the access or viewing permissions to the requested content/pages.

It is possible to configure both URLs. If no URL is configured, a 403 Permission Denied status code is sent in the case of an error (failed login, insufficient permissions).

Mappings: The actual mappings for the URL RequestFilter can be defined within the table:

New: Clicking on this button configures a new mapping (see Figure 2-6).

Edit: Clicking on this button opens an existing mapping for editing (see Figure 2-6).

Delete: Clicking on this button removes an existing mapping from the table.

Up/Down: The mappings in the table are evaluated in order (from top to bottom). The first applicable filter highest up in the table is applied. The order of the mappings can be changed using the Up/Down button.



Defining an mapping:

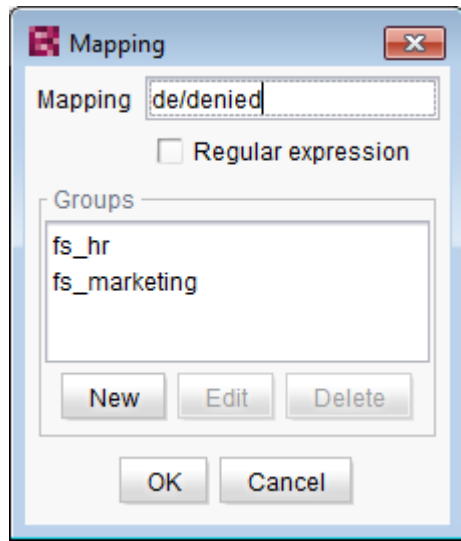


Figure 2-6: Defining the mapping for the RequestFilter

Mapping: The value entered here defines the filter. It checks whether the requested URL starts with the string configured here.


Regexp (regular expression): If the checkbox below this is selected, the value entered in the mapping (see above) is not evaluated as a string, but as a regular expression. Regular expressions can be used to define more complex URL filters.

Groups: The bottom section of the dialog is where the groups that apply to this URL area can be defined. The URL area mapping for one (or more) groups is used to allow or deny individual groups access to certain URLs.



If the list of groups remains empty, then there are no restrictions, and the user can access the URL area once he or she has been successfully authenticated.

2.4 Configuring a login package

 A new login package is configured by clicking the button (cf. Figure 2-3). Any type of module may only exist once within a login package. Which module is used is configured using the following dialog:

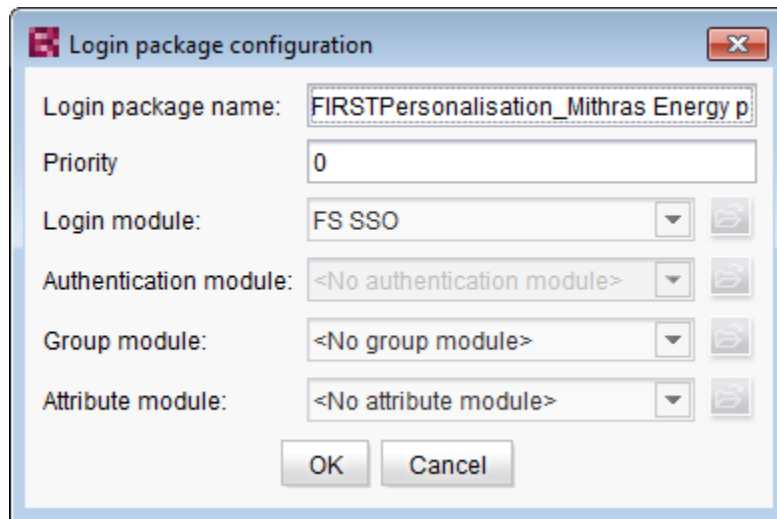


Figure 2-7: Login package configuration

The individuals are then configured by way of their parameters. Which parameters are required or are possible depends on the respective module (see Chapter 3 page 26):

Login package name and priority	(see Chapter 2.3 page 18).
Login module configuration:	(see Chapter 3.1 page 26).
Authentication module configuration	(see Chapter 3.2 page 34).
Group module configuration	(see Chapter 3.3 page 42).
Attribute module configuration	(see Chapter 3.4 page 51).

2.5 Further configuration options (web.xml)

Each FirstSpirit web application provides a configurations file "web.xml". This file does not have to be manually edited for a conventional configuration as changes can be made directly through the FirstSpirit GUI.

However, the web.xml can also be edited directly for special configuration options, for example setting up filters. (For details on editing the "web.xml" see "FirstSpirit Documentation for Administrators"). However, only the settings extending beyond configuring of the modules in



ServerManager should be made here.

2.5.1 Filter: NtlmPreAuthenticationFilter

The optional "NtlmPreAuthenticationFilter" filter is required only in connection with a configured NTLM authentication. This filter must be switched before all servlets opened by a POST Request. This filter is not required if FirstSpirit DynamicPersonalization is to run without NTLM authentication or if no HTML form data are to sent via http POST.

```
<filter>
  <filter-name>NtlmPreAuthenticationFilter</filter-name>
  <filter-class>
de.espirit.firstspirit.opt.personalisation.filters.NtlmPreAuthenticationFilter
  </filter-class>

  <init-param>
    <param-name>domainController</param-name>
    <param-value>myDomainController</param-value>
  </init-param>
</filter>

<filter-mapping>
  <filter-name>NtlmPreAuthenticationFilter</filter-name>
  <servlet-name>myServlet</servlet-name>
</filter-mapping>
```

A valid domain controller must be given for the "<param-value>" tag.

The filter mapping must be entered for each existing servlet opened by a POST Request.

2.6 Further libraries

Further libraries are required within the web application for the ExternalSAP and ExternalSAP73 group modules as well as for the ExternalSAP and ExternalSAP73 attribute modules. These must be stored manually in the directory WEB-INF/lib.

This concerns the following libraries:

- axis-1.4.jar
- axis-jaxrpc-1.4.jar
- commons-discovery-0.4.jar
- wsdl4j-1.6.2.jar

If the SAP log-on ticket is also to be read out, the following additional libraries are also required:



- iaik_jce.jar
- com.sap.security.core.jar
- com.sap.security.api.jar



3 Modules

3.1 Login modules

The login modules determine the login data (login name and password) of a user (see Chapter 1.3.4 page 12). The authentication then takes place through the selected authentication module. However, it is possible that the login data determined has already been authenticated elsewhere so that subsequent authentication is no longer necessary. This applies to all login modules except the "Request Parameter Login" module.

The following login modules are available:

- Kerberos Login
Login via Kerberos/SPNEGO (Integrated Windows Authentication, IWA)
(cf. Chapter 3.1.1 page 27).
- NTLM Login
Login via NTLM
(cf. Chapter 3.1.2 page 29).
- Anonymous NTLM Login
Login with a fixed NTLM user (anonymous login)
(cf. Chapter 3.1.3 page 31).
- Request Parameter Login
Login with the help of request parameters
(cf. Chapter 3.1.4 page 32).
- Request Header Login
Login with the help of the request header
(cf. Chapter 3.1.5 page 32).
- Portlet Login
Login via an IBM WebSphere portal server
(cf. Chapter 3.1.6 page 33).
- SAP Portlet Login
Login via an SAP portal server
(cf. Chapter 3.1.7 page 33).
- FS SSO
Login via FirstSpirit Single Sign On (cf. Chapter 3.1.8 page 33)



3.1.1 "Kerberos Login" login module

This module serves to login automatically without entering a password via Kerberos 5 and SPNEGO. Kerberos is supported on server-side by all current Unix systems and by Microsoft Active Directory since Windows 2003 / XP, here known as "Integrated Windows Authentication" (IWA), as substitution for NTLM. On client-side, it is supported by all desktop operating systems and by most of the web browsers: Mozilla Firefox, Microsoft Internet Explorer, Safari, KDE, Gnome and others.

The specification for Kerberos advises to use logins via Kerberos only via safe data channels, i.e. for example *https*, because of the theoretical risk of replay attacks in a short space of time after the login. But an encrypted login ticket cannot be read out or forged even when using unencrypted *http*.

Parameters:

useFullPrincipal: Defines whether the full Kerberos login name including @ characters and Kerberos realm (value "true") or without @ and Kerberos realm (value "false") is passed as user name e.g. to the group module. "false" is sufficient for systems whose user accounts are all entered in a Kerberos realm (corresponds to an Active Directory Domain under Windows). If logins take place from several Kerberos realms or Active Directory Domains, "true" must be given, as in most cases the pure user name is not unique over several domains.

Default value: "false"

userAgents: Here it is possible to enter a search pattern for the browser identification to activate Kerberos login for selected web browsers only, as Kerberos uses an HTTP header which does not fully conform to the standard ("WWW-Authenticate: Negotiate"); several older web browsers interpret this as an error. To use Kerberos for all web browsers, enter ". *".

Default value:

".* (Firefox|Iceweasel|Konqueror|MSIE|Opera|Safari|Shiretoko) . *"

sendAccepted: This parameter can be used to influence the HTTP status code in the case of a handshake. If the handshake is successful, OK (200) is sent. ACCEPTED (202) was sent prior to FirstSpirit Version 5.2R2. Depending on the infrastructure in use, this status (202) may not be accepted as "valid", something which may cause indexing problems, for example (the SharePoint crawler in Windows is a known



problem). This default behavior has therefore been changed as of 5.2R2. It is possible to restore the old behavior by setting the parameter `sendAccepted` to the value `true` (default value: false). It is only possible to set this parameter using a corresponding entry in the `jaas.conf` file:

```
sendAccepted="true"
```

3.1.1.1 Configuration of the Kerberos Login module

The following configuration is required in addition to the settings of the ServerManager configuration when using the Kerberos Login module:

A unique Kerberos Service Principal Name (SPN) must be assigned to each host name of a web server, which results as follows:

```
HTTP/hostname.domain.tld@REALM
```

"HTTP" is also to be used as identifier for https!

"hostname.domain.tld" is the complete DNS host name including the domain, as it is entered into the web browser for using the web server on client side.

"REALM" is the so called Kerberos realm, which is mostly conform to the domain, but written always completely in capital letters.

One or more keys must be created for one SPN by different crypto methods which are combined into a Kerberos-Keytab file. This Kerberos-Keytab file must be passed to the application server as Java property when starting. If Tomcat is used as application server, add the following parameter in `CATALINA_OPTS`, in the case of other application servers Java properties can be entered maybe directly:

```
-Djava.security.auth.login.config=/path/to/jaas.conf
```

The file `jaas.conf` must contain the following lines:

```
com.sun.security.jgss.accept {  
  com.sun.security.auth.module.Krb5LoginModule required  
    principal="HTTP/www.mydomain.net@MYDOMAIN.NET"  
    keyTab="/path/to/krb5-www_mydomain_net-HTTP.keytab"  
    useKeyTab="true"  
    storeKey="true"  
    isInitiator="false"  
    doNotPrompt="true"  
    debug="false";  
}
```



```
};
```

For "principal" enter the previously mentioned SPN, which contains the host name of the web server, as it is used by the browser. During the setup phase the parameter "debug" should be set to "true" to be able to read Kerberos error messages in the stdout/stderr logging of the application server (in the case of Tomcat in catalina.out). In addition, a file /etc/krb5.conf or c:\windows\krb5.ini on the application server is required depending on the DNS configuration of the local network. For further parameters of the file jaas.conf and for the creation of the Kerberos Keytab file which is indicated for the parameter "keyTab" and the content of the krb5.conf file see *FirstSpirit Documentation for Administrators*, Chapter "Kerberos ticket (integrated Windows login)".

3.1.2 "NTLM Login" login module

Automatically authenticates a user without entering a password through NTLM. This module does not require an authentication module.

Only NTLM v1 is supported, but not NTLM v2, which is used as standard since Windows 2003. In fact, Windows 2003 can be switched to NTLM v1, but this is not advisable because of security reasons. In this case, Kerberos should be used (Chapter 3.1.1 page 27), which is also labelled as "Integrated Windows Authentication" (IWA) by Microsoft.

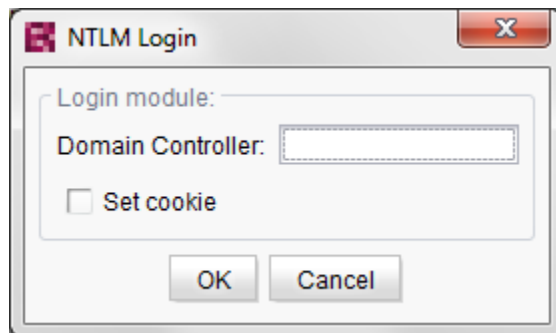


Figure 3-1: "NTLM Login" login module

Domain controller: Name of the domain controller to be used. This parameter is used to give the Windows domains approved for the login. Domain servers can be optionally given. Several domains can be entered, which are all checked one after the other for the login. The separator used is ; .

Set cookie: If this checkbox is selected a cookie is created for authentication through SSO (for concept see Chapter 1.3.1 page 7). The cookie (domain, name, Time-to-live) is configured using the web application (see Chapter 2.3 page 18).



For details of NTLM authentication, see also Chapter 2.5.1 page 24
"NtlmPreAuthenticationFilter".



3.1.3 "Anonymous NTLM Login" login module

Authenticates a user through NTLM by way of the configured user name, password and domain. This module does not require an authentication module.

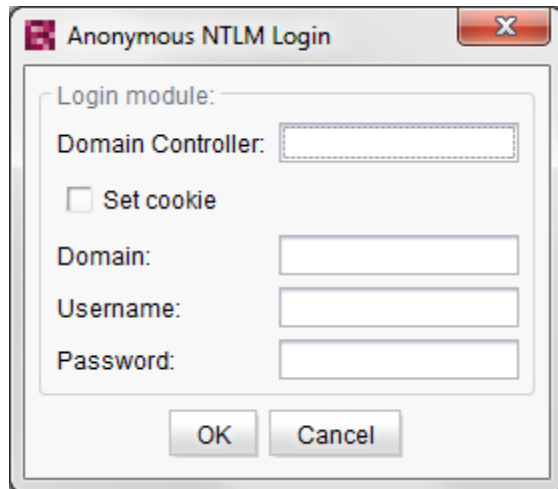


Figure 3-2: "Anonymous NTLM Login" login module

Domain controller: Name of the domain controller to be used (see Chapter 3.1.2 page 29).

Set cookie: If this checkbox is selected a cookie is created for authentication through SSO (for concept see Chapter 1.3.1 page 7). The cookie (domain, name, time-to-live) is configured using the web application (see Chapter 2.3 page 18).

Domain: Name of the domain to be used.

User name: Name of the user to be logged in or authenticated.

Password: Password with which the user to be logged in or authenticated.

For details of NTLM authentication, see also Chapter 2.5.1 page 24
"NtlmPreAuthenticationFilter".



3.1.4 "Request Parameter Login" login module

The "Request Parameter Login" login module reads out the user name and password from the HTTP Request and can be used, for example, in combination with a login servlet. The "login" and "password" parameters are expected in the request.



This module requires an authentication module which checks the login data contained in the request. For information on authentication modules see Chapter 3.2 from page 34.

Parameters: This module does not require any configuration parameters.

3.1.5 "Request Header Login" login module

The "Request Header Login" login module reads the user name from the request header. This module does not require an authentication module as only user names of already authenticated users are expected in the header.

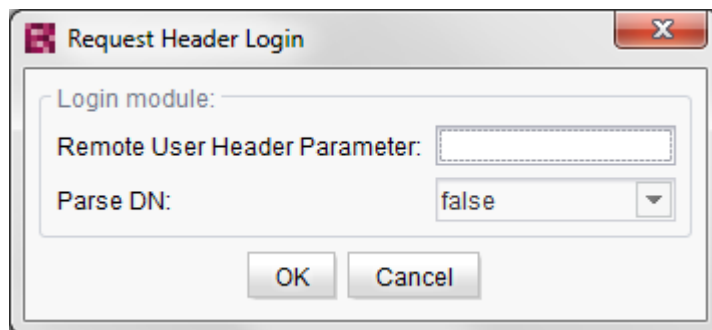


Figure 3-3: "Request Header Login" login module

Remote user header parameter: Name of the request header which contains the user name. The parameter is optional; if it is not defined the user name is read out of the request's remoteUser attribute.

Parse DN: States whether the whole header value is to be interpreted as the user name (value: true) or only the section between the first "=" and the next comma is to be used (value: false). This is required in cases in which there is a complete DN in the header which contains the user name as the first entry.



Example of a header value with complete DN:

```
cn=brown,ou=reporting,o=mycompany,c=US
```

3.1.6 "Portlet Login" login module

This login module reads out the user name from a linked IBM WebSphere portal server. This module does not require an authentication module as the user has already authenticated themselves through the portal.

Parameters: This module does not require any configuration parameters.

3.1.7 "SAP Portlet Login" login module

This login module reads out the user name from a linked SAP portal server. This module does not require an authentication module as the user has already authenticated themselves through the portal.

Parameters: This module does not require any configuration parameters.

3.1.8 "FS SSO" login module

FirstSpirit provides the option of configuring a Single Sign On (SSO) for the start page, SiteArchitect, ContentCreator, ServerMonitoring and ServerManager. A user must then only log in once. After logging in the login information is used by the other applications, provided they are configured (for concept details see Chapter 1.3.1 page 7). With the "FS SSO" login module this login information can also be used in the FirstSpirit DynamicPersonalization module, where the user can also be logged in. This module does not require an authentication module as the user has already authenticated themselves through the FirstSpirit server.

Parameters: This module does not require any configuration parameters.



3.2 Authentication modules

If login data has been determined by a login module but has not yet been authenticated, an authentication module is required which performs this task.

The following authentication modules are available:

- JDBC
Authentication against user account information that is stored in a database and can be accessed via JDBC
(cf. Chapter 3.2.1 page 35)
- LDAP
Authentication against an LDAP server
(cf. Chapter 3.2.2 page 37).
- ORMapper
Authentication against a data source from FirstSpirit Content Store
(cf. Chapter 3.2.3 page 39).
- Property file
Authentication against a property file
(cf. Chapter 3.2.4 page 40).
- Permission service
Authentication against the FirstSpirit system service "Permission Service"
(cf. Chapter 3.2.5 page 41).



3.2.1 "JDBC" authentication module

The "JDBC" authentication module is used to authenticate user account information that is stored in a database and can be accessed via JDBC.

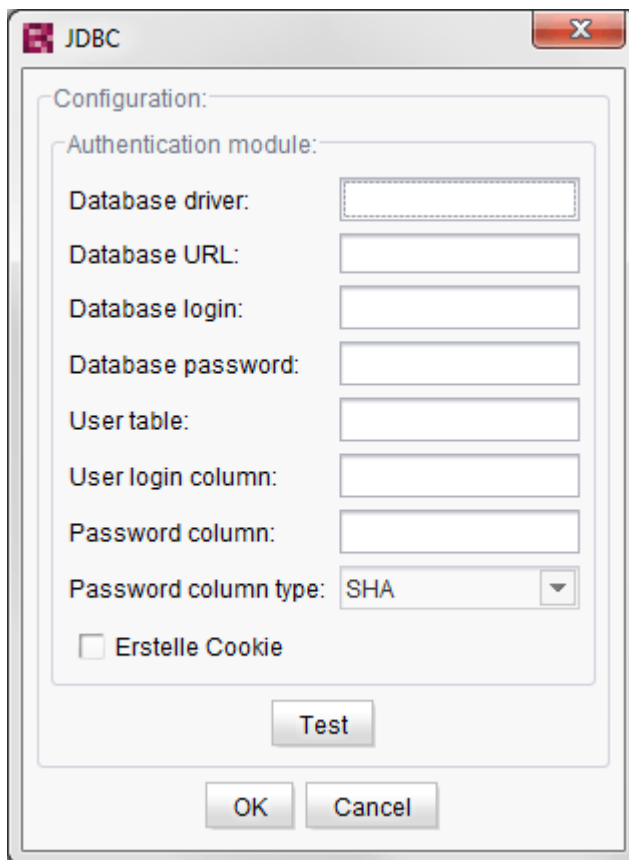


Figure 3-1: "JDBC configuration" authentication module

Database driver: Provides information on the complete class name of the JDBC database driver used; for example: `com.mysql.jdbc.Driver`.

Database URL: Provides information on the JDBC URL for a database server and a database existing there; for instance: `jdbc:mysql://myServer:3306/mydb`

Database login: Valid login name of a database user. This account is used by FirstSpirit Server to establish a connection to the database at runtime.

Database password: Valid password for the login name under "Database login".



User table: Information on a table containing the user account information for authentication.

User login column: Information on a table range (column) for the user login information.

Password column: Information on a table range (column) for the user password information. A specific encoding for the content of this table range is applied that depends on the "Password column type" setting.

Password column type: The "Password column" authentication data may be encoded based on certain algorithms (SHA, MD5-Tomcat, MD5-Base64) or may be contained in clear text. These algorithms need to be taken into account when comparing the authentication information.

Create cookie: If this checkbox is selected, a cookie is created for this authentication via SSO (see Chapter 1.3.1 page 7 for the concept). The configuration of cookies (domain, name, expiration) is handled using the web application (see Chapter 2.3 page 18).

A rectangular button with a light gray background and a thin border, containing the word "Test" in a dark gray sans-serif font.

The configuration of the authentication module can be tested by clicking on this button. The "ConfigTestService" global service must be started to do this (see Chapter 2.1, page 14). Starting and stopping services is done either via FirstSpirit ServerManager (Server properties / Modules) or via FirstSpirit ServerMonitoring (FirstSpirit / Control / Services).



3.2.2 "LDAP" authentication module

The "LDAP" authentication module is used for authentication against an LDAP server.

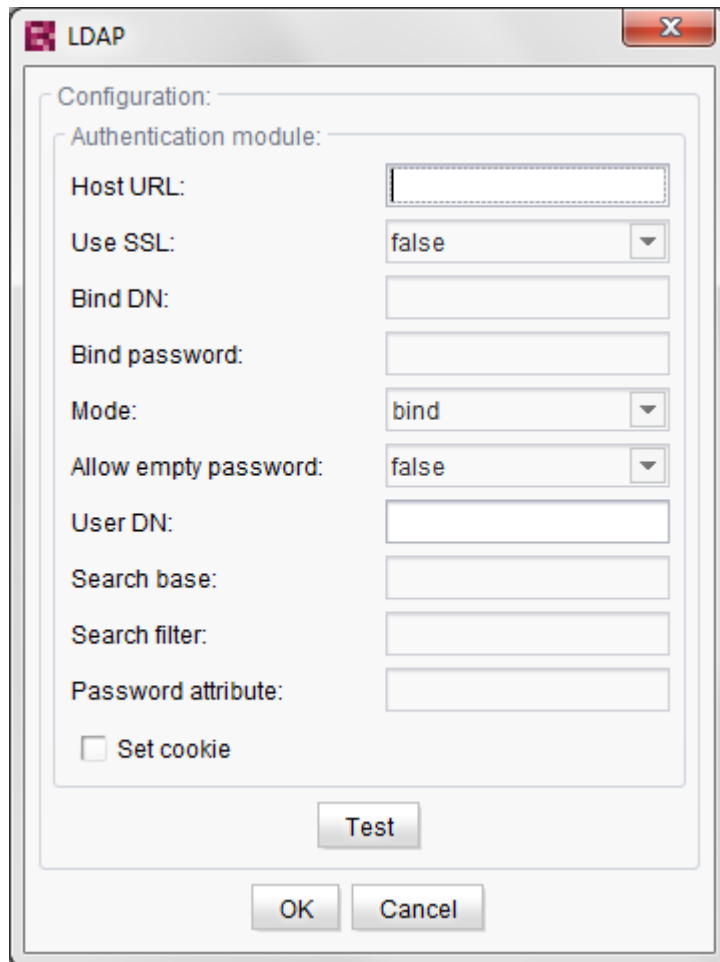
The image shows a "LDAP" configuration dialog box. It has a title bar with a red "X" button. The dialog is divided into two main sections: "Configuration:" and "Authentication module:". Under "Authentication module:", there are several fields and dropdown menus: "Host URL:" (text input), "Use SSL:" (dropdown menu with "false" selected), "Bind DN:" (text input), "Bind password:" (text input), "Mode:" (dropdown menu with "bind" selected), "Allow empty password:" (dropdown menu with "false" selected), "User DN:" (text input), "Search base:" (text input), "Search filter:" (text input), and "Password attribute:" (text input). At the bottom of the "Authentication module:" section is a checkbox labeled "Set cookie". Below this section is a "Test" button. At the very bottom of the dialog are "OK" and "Cancel" buttons.

Figure 3-4: "LDAP Configuration" authentication module

Host URL: A comma-separated list of LDAP servers

Use SSL: states whether SSL is to be used (value: true) or not. (Value: false).

Bind DN: DN of a user with which an initial context of the LDAP server can be obtained. The parameter is only required in conjunction with "search_bind" and "search_compare" (cf. "Mode").

Bind password: Password for the DN given under "Bind DN". The parameter is only required in conjunction with "search_bind" and "search_compare" (cf. "Mode").



Mode: Gives the type of authentication. Possible values: `bind`, `search_bind` and `search_compare`:

Bind: Authentication (so-called "Bind") is performed against the LDAP server using the login name and password extended to form a complete DN. The name and password are sent to the LDAP server. To this end the "Distinguished Name" (DN), i.e. the unique key for identification of the user within the LDAP server, must be known. If the DN exists the transferred password is checked with the help of the "Bind" operation.

Search & Bind: If a user's "Distinguished Name" (DN) is not known it can be looked for within a sub-tree of the LDAP server. This requires the definition of a search filter and a start node (see description of "Search Path Base" and "Search Filter"). If a start node is found a "Bind" (with regard to the LDAP server) is executed. (see LDAP Bind).

Search & Compare: Analogous to "Search & Bind", in this mode the complete DN of the user is determined by way of a search in the LDAP tree. However, after a suitable node has been found, a "Bind" operation is not performed. Instead the password entered is compared with any (previously defined) LDAP attribute (see description of "Password Attribute").

Allow empty password: States whether empty passwords may be used or not.

User DN: A "#" separated list of DNs, each of which must contain the character string `$USER_LOGIN$`. The character string is then automatically replaced by the respective user name. The parameter is not optional if the `mode` parameter was assigned the value "bind".

Example:

```
cn=$USER_LOGIN$,cn=Recipients,ou=E-SPIRIT,o=e-Spirit
```

or

```
cn=$USER_LOGIN$,cn=Recipients,ou=E-SPIRIT,o=e-Spirit#cn=$USER_LOGIN$,cn=others,ou=E-SPIRIT,o=e-Spirit
```

Search path base: Gives the DN of the start node from which the LDAP tree is to be searched. The parameter is only required in conjunction with "`search_bind`" and "`search_compare`".

Example:

```
cn=Recipients,ou=E-SPIRIT,o=e-Spirit
```

Search filter: A filter to be used to search in the LDAP tree. The respective user name can also be added within the filter by the character string `$USER_LOGIN$`. The parameter is only required in conjunction with "`search_bind`" and "`search_compare`".



Example:

```
(sn=$USER_LOGIN$)
```

Password attribute: An attribute whose value is compared with the entered password. The parameter is only required in conjunction with "search_compare".

Set cookie: If this checkbox is selected a cookie is created for authentication through SSO (for concept see Chapter 1.3.1 page 7). The cookie (domain, name, time-to-live) is configured using the web application (see Chapter 2.3 page 18).

Test

The configuration of the authentication module can be tested by clicking the button. To this end the global service "ConfigTestService" must be started (see Chapter 2.1 page 14). Services are started and stopped either through FirstSpirit ServerManager (Server Properties / Modules) or through FirstSpirit ServerMonitoring (FirstSpirit / Control / Services).

3.2.3 "ORMapper" authentication module

The "ORMapper" authentication module can be used to authenticate against a table from a FirstSpirit database.

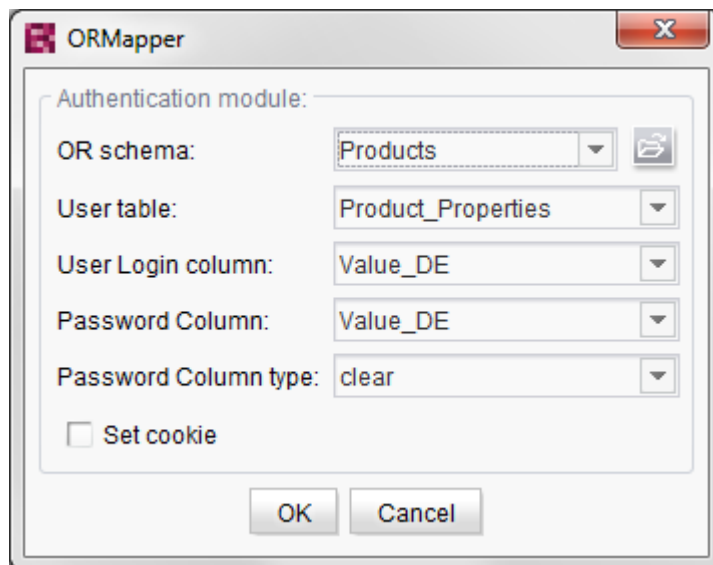


Figure 3-5: "ORMapper" authentication module

OR schema: Name of the configuration schema to be used.

User table: Name of the table with the user information.



User login column: Name of the table column which contains the user name.

Password column: Name of the table column which contains the user password.

Password column type: Type of password encryption.

Possible values: `clear`, `SHA`, `MD5-Tomcat` and `MD5-Base64`

Create cookie: If this checkbox is selected a cookie is created for authentication through SSO (for concept see Chapter 1.3.1 page 7). The cookie (domain, name, time-to-live) is configured using the web application (see Chapter 2.3 page 18).

For further information on configuration of the "OR-Schema" parameter, please refer to the "FirstSpirit DynamicDatabaseAccess" module documentation.

Note: FirstSpirit Server already includes a simple relational database system (Apache Derby), which is available immediately after the server is installed. However, this database is not suited for production mode and therefore should only be used for testing. If the internal Derby database is to be combined with the DynamicPersonalization module for test scenarios, the configuration must be modified so that it can also be called from the web application. An example of how to configure the Derby database to use external processes is available in the Documentation for Administrators (Chapter 4.9.7 "Examples for linking different database systems").

3.2.4 "PropertyFile" authentication module

This module is used to authenticate against a property file.

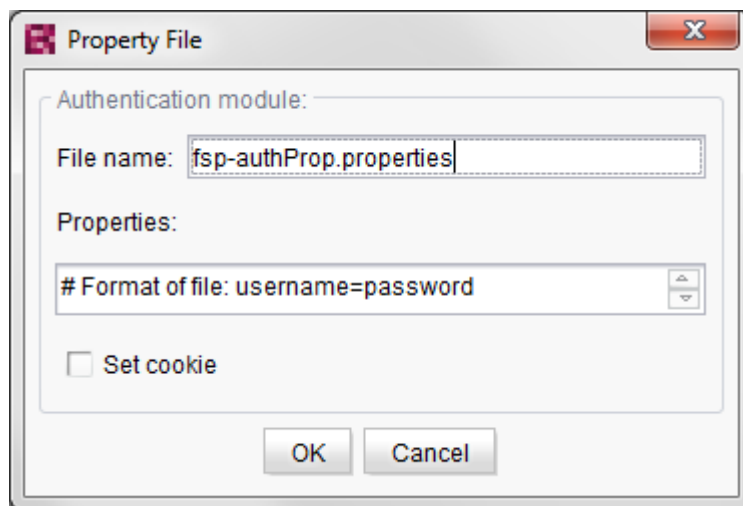


Figure 3-6: "Property File" authentication module



File Name: Name under which the file is to be saved.

Properties: User name/password pairs. Each line corresponds to a user. The user name is separated from the password by "=".

Set cookie: If this checkbox is selected a cookie is created for authentication through SSO (for concept see Chapter 1.3.1 page 7). The cookie (domain, name, time-to-live) is configured using the web application (see Chapter 2.3 page 18).

3.2.5 "Permission Service" authentication module

The "Permission Service" authentication module can be used to authenticate against the system service "Permission Service" (for details of concept see Chapter 1.3.2 page 8). The authentication is based on the content of the selected User XML file ("NAME.users" information in the service configuration file).

Further information is provided in the FirstSpirit "Documentation for Administrators".

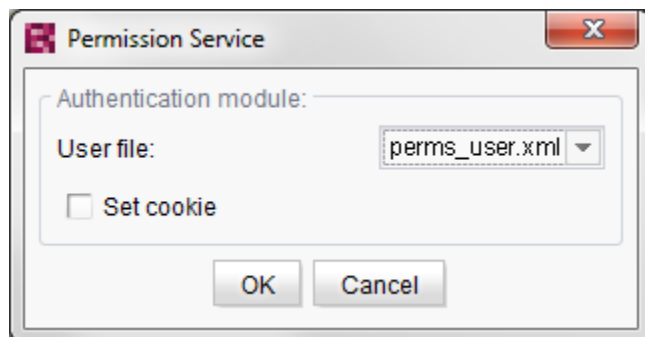


Figure 3-7: "Permission Service" authentication module

User file: File name of the selected user file.

Set cookie: If this checkbox is selected a cookie is created for authentication through SSO (for concept see Chapter 1.3.1 page 7). The cookie (domain, name, time-to-live) is configured using the web application (see Chapter 2.3 page 18).

3.3 Group modules

The group modules are used to read out group information for the logged in user. Use of the optional group modules is only necessary if a differentiation is to be made between groups. This is the case if display of certain areas of a page is reserved for certain groups. In this case the group membership of the logged in user can be used to control the display.

The following group modules are available:

- JDBC
Obtain group information from a database which can be accessed via JDBC
(cf. Chapter 3.3.1 page 43).
- LDAP group
Obtain group information from an LDAP server (group attributes are assigned to the user)
(cf. Chapter 3.3.2 page 44).
- LDAP group iterate
Obtain group information from an LDAP server (group attributes are assigned to the group object)
(cf. Chapter 3.3.3 page 45).
- ORMapper
Obtain group information from a data source from the FirstSpirit Content Store
(cf. Chapter 3.3.4 page 47).
- Portlet
Obtain group information from an IBM WebSphere portal server
(cf. Chapter 3.3.5 page 48).
- SAP portlet
Obtain group information from an SAP portal server
(cf. Chapter 3.3.6 page 48).
- FS SSO groups
Obtain information through FirstSpirit single sign-on
(cf. Chapter 3.3.7 page 48).
- Permission service
Obtain group information through the FirstSpirit system service "Permission Service"
(cf. Chapter 3.3.8 page 48).



- ExternalSAP
Obtain group information through a linked SAP portal server using its web service (cf. Chapter 3.3.93.3.10 page 49).
- ExternalSAP73
Obtain group information through a linked SAP portal server (> version 7.3) using its web service (cf. Chapter 3.3.10 page 49).

3.3.1 “JDBC” group module

The "JDBC" group module is used to include user group information from a database table and can be accessed via JDBC.

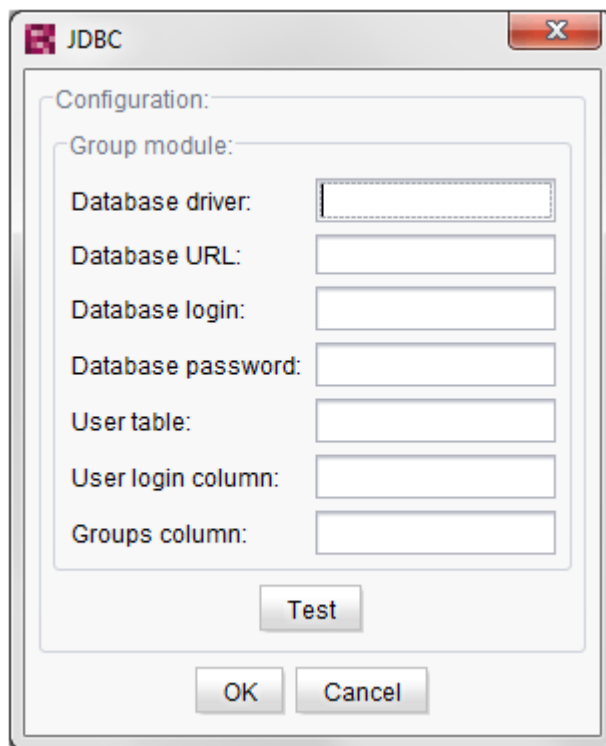


Figure 3-2: JDBC group module

Database driver: Provides information on the complete class name of the JDBC database driver used; for example: `com.mysql.jdbc.Driver`.

Database URL: Provides information on the JDBC URL for a database server and a database existing there; for instance: `jdbc:mysql://myServer:3306/mydb`.

Database login: Valid login name of a database user. This account is used by FirstSpirit Server



to establish a connection to the database at runtime.

Database password: Valid password for the login name under "Database login".

User table: Information on a table that contains the user account data.

User login column: Information on a table range (column) for the user login information.

Groups column: Information about a table area (column) for the user group information. The group column must be in the same table as the user data. Group names must be entered here with a comma separating them (no spaces).

Test

The configuration of the module can be tested by clicking on this button. The "ConfigTestService" global service must be started to do this (see Chapter 2.1 page 14). Services are started and stopped either through FirstSpirit ServerManager (Server Properties / Modules) or through FirstSpirit ServerMonitoring (FirstSpirit / Control / Services).

3.3.2 "LDAP Group" group module

The "LDAP Group" group module is used to obtain the user's group information through an LDAP server. The information in the LDAP should be structured so that a list of group attributes is available for each user.

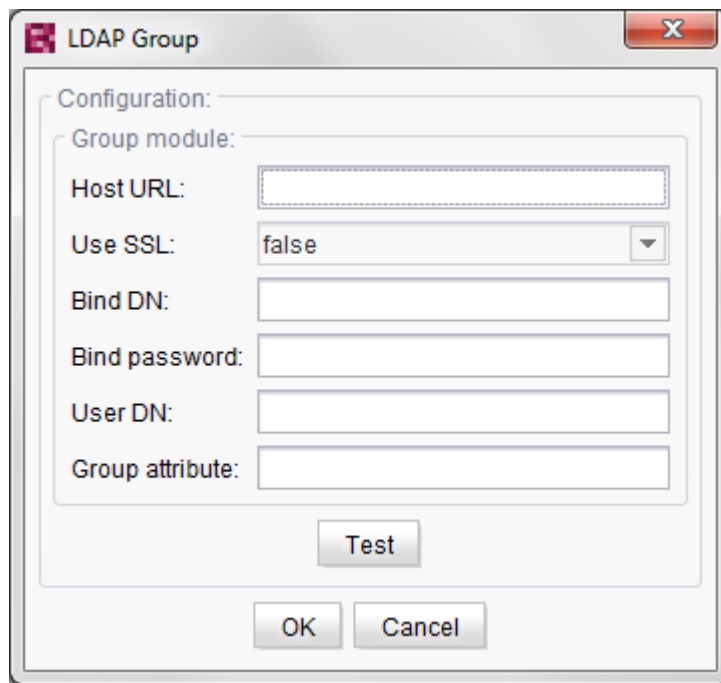


Figure 3-8: "LDAP Group" group module



Host URL: A comma-separated list of LDAP servers.

Use SSL: states whether SSL is to be used (value: true) or not. (Value: false).

Bind DN: DN of a user with which an initial context of the LDAP server can be obtained.

Bind password: Password for the DN given under "Bind DN".

User DN: A "#" separated list of DN's, each of which must contain the character string \$USER_LOGIN\$. The character string is then automatically replaced by the respective user name.

Example:

```
cn=$USER_LOGIN$,cn=Recipients,ou=E-SPIRIT,o=e-Spirit
```

Group attribute: The groups to which the user belongs are included in this attribute. Group evaluation is not case sensitive.

Test

The configuration of the module can be tested by clicking the button. To this end the global service "ConfigTestService" must be started (see Chapter 2.1 page 14). Services are started and stopped either through FirstSpirit ServerManager (Server Properties / Modules) or through FirstSpirit ServerMonitoring (FirstSpirit / Control / Services).

3.3.3 "LDAP Group Iterate" group module

The "LDAP Group Iterate" group module is used to obtain the user's group information through an LDAP server. The group module is used if LDAP has been configured as follows:

the user entries contain *no information* about the groups to which they belong
only the group objects themselves contain a list of the corresponding users

In this case it is necessary to iterate across all groups and extract the groups which contain the current user as a member.



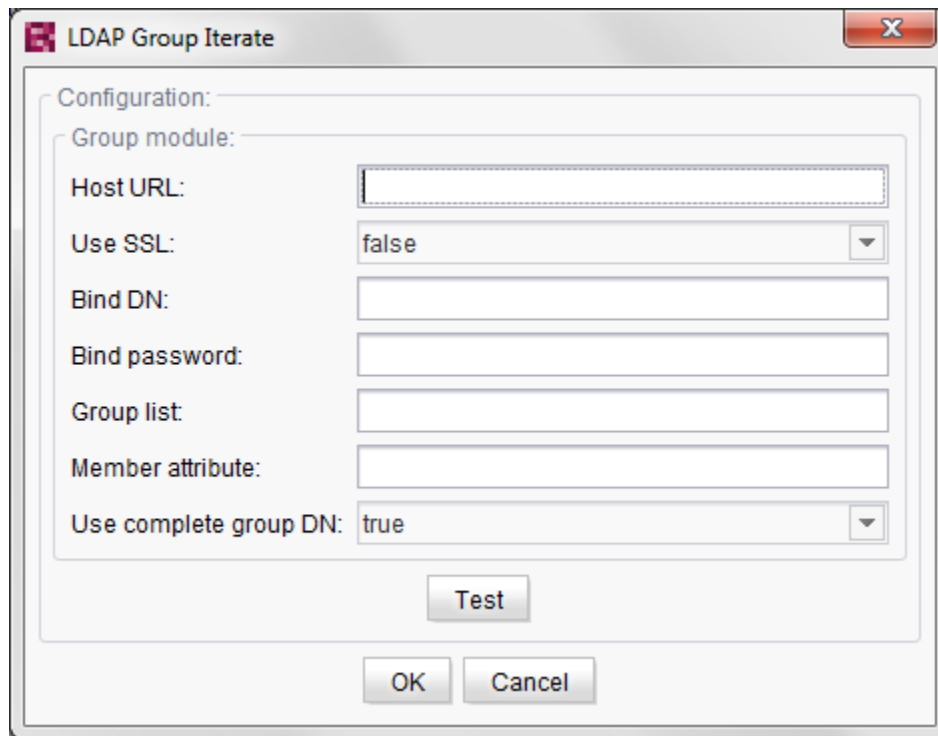


Figure 3-9: "LDAP Group Iterate" group module

Host URL: A comma-separated list of LDAP servers.

Use SSL: states whether SSL is to be used (value: true) or not. (Value: false).

Bind DN: DN of a user with which an initial context of the LDAP server can be obtained.

Bind password: Password for the DN given under "Bind DN".

Group list: A "#" separated list of DN's which represent the groups in which the users could be listed as members.

Member attribute: Attribute containing the users who are members of the group.

Use complete group DN: States whether the complete DN of the group is to be used as the group name (value: true) or not (value: false). If this is not the case the last value of the DN only is used as the group name.

Test

The configuration of the module can be tested by clicking the button. To this end the global service "ConfigTestService" must be started (see Chapter 2.1 page 14). Services are started and stopped either through FirstSpirit ServerManager (Server Properties / Modules) or through FirstSpirit ServerMonitoring (FirstSpirit / Control / Services).



3.3.4 "ORMapper" group module

The "ORMapper" group module is used to read out a user's groups from a table from the FirstSpirit database.

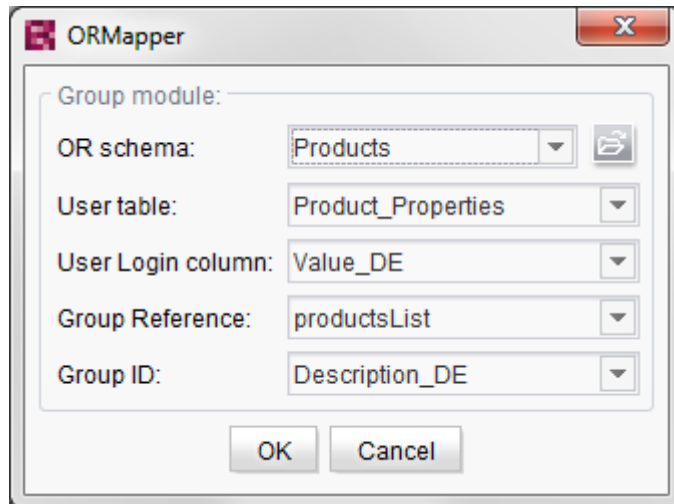


Figure 3-10: "ORMapper" group module

OR schema: Name of the configuration schema to be used.

User table: Name of the table with the user information.

User login column: Name of the table column which contains the user name.

Group reference: The reference between the user table and the group table.

Group ID: Table column of the group table which contains the group name



For further information on configuration of the "OR-Schema" parameter, please refer to the "FirstSpirit DynamicDatabaseAccess" module documentation.

3.3.5 "Portlet" group module

This group module reads out the group information from a linked IBM WebSphere portal server.

Parameters: This group module does not require any configuration parameters.

3.3.6 "SAP Portlet" group module

This group module reads out the group information from a linked SAP portal server. To this end the "SAP Business Package for FirstSpirit™" module must be installed.

Parameters: This group module does not require any configuration parameters.

3.3.7 "FS SSO Groups" group module

The "FS SSO Groups" group module can be used to read out the group information of the user logged in (to the FirstSpirit server).

For details of the configuration of FirstSpirit Single Sign-On see FirstSpirit "Documentation for Administrators".

Parameters: This group module does not require any configuration parameters.

3.3.8 "Permission Service" group module

The "Permission Service" group module can be used to obtain the group information for a user through the system service "Permission Service" (for details of concept see Chapter 1.3.2 page 8). The group information of a user is based on the content of the selected User XML file ("NAME.users" information in the service configuration file).

Further information is provided in the FirstSpirit "Documentation for Administrators".



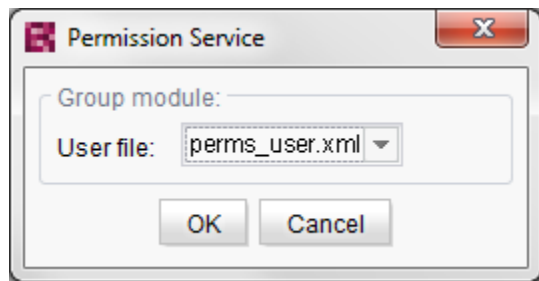


Figure 3-11: "Permission Service" group module

User file: File name of the selected user file.

3.3.9 "ExternalSAP" group module

This group module reads out the group information from a linked SAP portal server using the server's web service. It can only be configured manually.

url: Server URL for the web service (mandatory field)

authUser: User name for web service authentication (mandatory field)

authPass: Password for web service authentication (mandatory field)

fallback_groups: Fallback groups to be set for a user where no specific group has been set (mandatory field if "use_fallback_group_if_anonymous_user=true" or "read_groups=false")

use_fallback_group_if_anonymous_user: Whether fallback groups should be set for anonymous users (value: true/false)

read_groups: Whether group information should be read out (value: true/false).

3.3.10 "ExternalSAP73" group module

This group module reads out the group information from a linked SAP portal server (> version 7.3) using the server's web service. It can only be configured manually.

url: Server URL for the web service (mandatory field)

authUser: User name for web service authentication (mandatory field)

authPass: Password for web service authentication (mandatory field)



fallback_groups: Fallback groups to be set for a user where no specific group has been set (mandatory field if "use_fallback_group_if_anonymous_user=true" or "read_groups=false")

use_fallback_group_if_anonymous_user: Whether fallback groups should be set for anonymous users (value: true/false)

read_groups: Whether group information should be read out (value: true/false).



3.4 Attribute modules

The attribute modules are used to read out attributes for the logged in user. Specification of an attribute module is optional and is only necessary if further attributes are to be read out for the user.

The following attribute modules are available:

- LDAP
Obtain attributes from an LDAP server (cf. Chapter 3.4.1 page 52).
- ORMapper
Obtain attributes from a data source from the FirstSpirit Content Store (cf. Chapter 3.4.2 page 54).
- ExternalSAP
Obtain attributes for a user from a linked SAP portal server using the server's web service (cf. Chapter 3.4.3 page 54).
- ExternalSAP73
Obtain attributes for a user from a linked SAP portal server (> version 7.3) using the server's web service (cf. Chapter 3.4.4 page 55).



3.4.1 "LDAP" attribute module

The "LDAP" attribute module is used to obtain a user's attributes from an LDAP server.

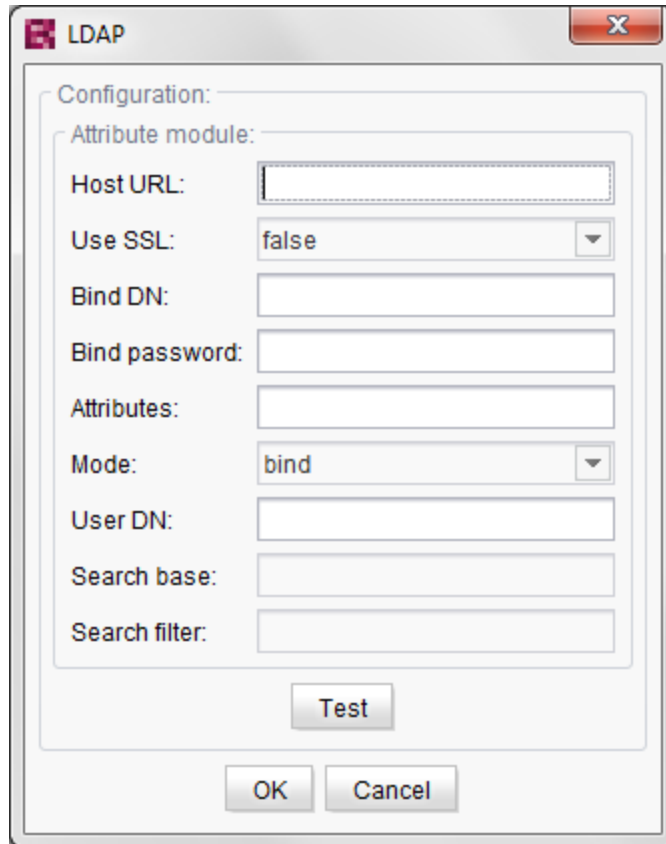
The image shows a dialog box titled "LDAP" with a standard Windows-style title bar (minimize, maximize, close buttons). The dialog contains a "Configuration:" section with a "Attribute module:" label. Below this, there are several input fields and dropdown menus: "Host URL:" (a text box), "Use SSL:" (a dropdown menu currently showing "false"), "Bind DN:" (a text box), "Bind password:" (a text box), "Attributes:" (a text box), "Mode:" (a dropdown menu currently showing "bind"), "User DN:" (a text box), "Search base:" (a text box), and "Search filter:" (a text box). At the bottom of the configuration section is a "Test" button. Below the configuration section are "OK" and "Cancel" buttons.

Figure 3-12: "LDAP" attribute module

Host URL: A comma-separated list of LDAP servers

Use SSL: states whether SSL is to be used (value: true) or not. (Value: false).

Bind DN: DN of a user with which an initial context of the LDAP server can be obtained. The parameter is only required in conjunction with "search_bind" and "search_compare" (cf. "Mode").

Bind password: Password for the DN given under "Bind DN". The parameter is only required in conjunction with "search_bind" and "search_compare" (cf. "Mode").

Attributes: Comma-separated list of attributes to be read out.

Mode: Specifies the method to be used to get to user nodes within the LDAP tree. Possible values: `bind` and `search_compare`:



Bind: Authentication (so-called "Bind") is performed against the LDAP server using the login name and password extended to form a complete DN. The name and password are sent to the LDAP server. To this end the "Distinguished Name" (DN), i.e. the unique key for identification of the user within the LDAP server, must be known. If the DN exists the transferred password is checked with the help of the "Bind" operation.

Search & Compare: Analogous to "Search & Bind", in this mode the complete DN of the user is determined by way of a search in the LDAP tree. However, after a suitable node has been found, a "Bind" operation is not performed.

User DN: A "#" separated list of DNs, each of which must contain the character string \$USER_LOGIN\$. The character string is then automatically replaced by the respective user name. The parameter is not optional if the mode parameter was assigned the value "bind".

Example:

```
cn=$USER_LOGIN$,cn=Recipients,ou=E-SPIRIT,o=e-Spirit
```

or

```
cn=$USER_LOGIN$,cn=Recipients,ou=E-SPIRIT,o=e-Spirit#cn=$USER_LOGIN$,cn=others,ou=E-SPIRIT,o=e-Spirit
```

Search path base: Gives the DN of the start node from which the LDAP tree is to be searched. The parameter is only required in conjunction with "search_compare".

Example:

```
cn=Recipients,ou=E-SPIRIT,o=e-Spirit
```

Search filter: A filter to be used to search in the LDAP tree. The respective user name can also be added within the filter by the character string \$USER_LOGIN\$. The parameter is only required in conjunction with "search_compare".

Example:

```
(sn=$USER_LOGIN$)
```

Test

The configuration of the authentication module can be tested by clicking the button. To this end the global service "ConfigTestService" must be started (see Chapter 2.1 page 14). Services are started and stopped either through FirstSpirit ServerManager (Server Properties / Modules) or through FirstSpirit ServerMonitoring (FirstSpirit / Control / Services).



3.4.2 "ORMapper" attribute module

The "ORMapper" attribute module is used to read out a user's attributes from a table from the FirstSpirit database.

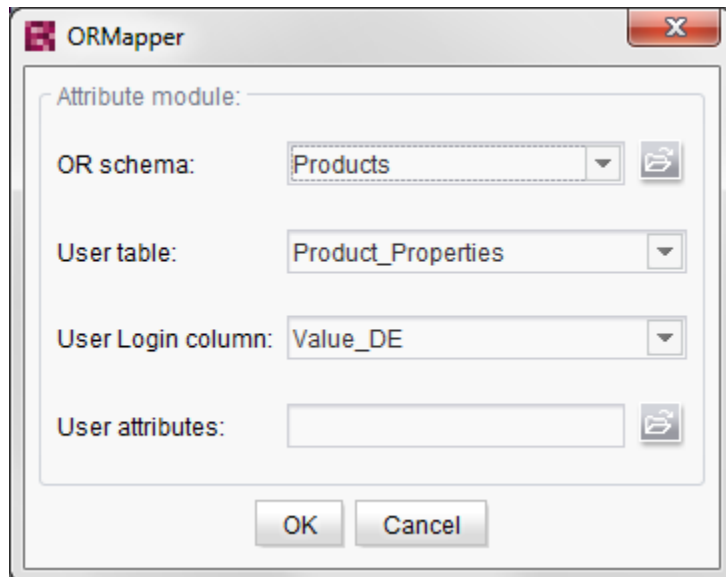


Figure 3-13: "ORMapper" attribute module

OR schema: Name of the configuration schema to be used.

User table: Name of the table with the user information.

User login column: Name of the table column which contains the user name.

User attributes: List of attributes to be read out.

For further information on configuration of the "OR-Schema" parameter, please refer to the "FirstSpirit DynamicDatabaseAccess" module documentation.

3.4.3 "ExternalSAP" attribute module

This attribute module reads out the attributes for a user from a linked SAP portal server using the server's web service. It can only be configured manually.

url: Server URL for the web service (mandatory field)

authUser: User name for web service authentication (mandatory field)



authPass: Password for web service authentication (mandatory field)

fallback_locale: Fallback language to be set for a user in cases where no language has been set (e.g. "en". Mandatory field if "use_fallback_locale_if_anonymous_user=true" or "read_attributes=false")

use_fallback_locale_if_anonymous_user: Whether the fallback language should be used for anonymous users (value: true/false)

read_attributes: Whether attribute information should be read out (value: true/false)

3.4.4 "ExternalSAP73" attribute module

This attribute module reads out the attributes for a user from a linked SAP portal server (> version 7.3) using the server's web service. It can only be configured manually.

url: Server URL for the web service (mandatory field)

authUser: User name for web service authentication (mandatory field)

authPass: Password for web service authentication (mandatory field)

fallback_locale: Fallback language to be set for a user in cases where no language has been set (e.g. "en". Mandatory field if "use_fallback_locale_if_anonymous_user=true" or "read_attributes=false")

use_fallback_locale_if_anonymous_user: Whether the fallback language should be used for anonymous users (value: true/false)

read_attributes: Whether attribute information should be read out (value: true/false)

standardAttributes: These attributes are freely configurable and are queried directly (e.g. LOCALE, DEPARTMENT).

namespaceAttributes: These attributes are freely configurable and are queried directly. In contrast to the standardAttributes, they are prefixed by the namespace (e.g. com.sap).



4 Servlets

The FirstSpirit DynamicPersonalization module provides two servlets.

With the first servlet it is possible to use request parameters to log into the module. It is useful to use the servlet if the "Request Parameter Login" login module is used (see Chapter 3.1.4 page 32).

The second servlet enables the currently logged in user to log out.

"Login Servlet": Log in user with request parameters

(cf. Chapter 4.1 page 56)

"Logout Servlet": Log out user

(cf. Chapter 4.2 page 58)



In order to prevent Open-Redirect attacks, external redirects are prohibited in projects which use the FirstSpirit DynamicPersonalization module. I.e. only relative URLs are possible, e.g.

start.jsp

../area/index.html

../area/search.jsp)

Exception: Redirects to the same host or a host in the same domain are allowed. A forwarding (redirect) page must be created for redirecting to an external URL.

4.1 "Login Servlet"

The "Login Servlet" enables a user to be logged into the module. The user name and password are usually entered in an HTML form. The request parameters transferred by the form are used by the servlet to log in the user.

The login servlet is evoked with:

```
<%= request.getContextPath() %>/do.login
```



Request parameter:

Parameter	Expected value
login	User name
password	User password
loginPackage	Name of the login package to be used for the login
login_ok_url	URL (if login is successful) <u>Note:</u> In order to prevent Open-Redirect attacks, external redirects are prohibited in projects which use the FirstSpirit DynamicPersonalization module. I.e. only relative URLs are possible, e.g. start.jsp ../area/index.html ../area/search.jsp) Exception: Redirects to the same host or a host in the same domain are allowed. A forwarding (redirect) page must be created for redirecting to an external URL.
wrong_login_url	URL (if login is wrong) External redirects are prohibited (see note above).

Simple example:

```
<form
  method="POST"
  action="<%= request.getContextPath() %>/do.login">

  User name:
  <input
    type="text"
    name="login"
    value="" />

  Password:
  <input
    type="password"
    name="password"
    value="" />

  <input
    type="hidden"
    name="login_ok_url"
    value="$CMS_REF(ptLoginOk, abs:1)$" />
  <input
    type="hidden"
    name="wrong_login_url"
    value="$CMS_REF(ptLoginFailed, abs:1)$" />
```



```
<input
  type="submit"
  value="Login" />

</form>
```

4.2 "Logout Servlet"

The "Logout Servlet" is used to log out the currently logged in user. The only parameter required is "redirect_url", which defines which URL is to be displayed after the logout.

The login servlet is evoked with:

```
"<%= request.getContextPath() %>/do.logout".
```

Simple example:

```
<form
  method="POST"
  action="<%= request.getContextPath() %>/do.logout">

  <input
    type="hidden"
    name="redirect_url"
    value="/logout.jsp" />

  <input
    type="submit"
    value="Logout" />

</form>
```



In order to prevent Open-Redirect attacks, external redirects are prohibited in projects which use the FirstSpirit DynamicPersonalization module.

I.e. only relative URLs are possible, e.g.

```
start.jsp
```

```
../area/index.html
```

```
../area/search.jsp)
```

Exception: Redirects to the same host or a host in the same domain are allowed. A forwarding (redirect) page must be created for redirecting to an external URL.



5 Tags

The FirstSpirit DynamicPersonalization tags can be used to display content user-specifically. The tag library includes tags for automatic login, hiding protected content and display of user information.

A page usually always contains the `<fsp:loginRequired>` tag or the `<fsp:authorize>` tag. The `<fsp:loginRequired>` tag re-routes to a login page if the current user is not yet logged in. There the user could log in using a form and would then be returned to the previously selected page. The `<fsp:authorize>` tag on the other hand automatically authenticates in the background (subject to corresponding selection of the FirstSpirit DynamicPersonalization modules) and then displays the content of the page.

The FirstSpirit DynamicPersonalization Taglib provides a large number of tags:

`<fsp:loginRequired>`

Force login

(see Chapter 5.2 page 60).

`<fsp:authorize>`

Automatic authentication

(see Chapter 5.3 page 61).

`<fsp:userInfo>`

Output the user name

(see Chapter 5.4 page 62).

`<fsp:userGroups>`

Output user groups

(see Chapter 5.5 page 63).

`<fsp:userAttributes>`

Output user attributes

(see Chapter 5.6 page 63).

`<fsp:isAuthorized>`

Display content if the user is authorised

(see Chapter 5.7 page 65).

`<fsp:isNotAuthorized>`

Display content if the user is not authorised

(see Chapter 5.8 page 68).



<fsp:logout>

Log out user

(see Chapter 5.9 page 68).

5.1 Prefix

In order to be able to use FirstSpirit DynamicPersonalization the relevant tag library must be given in the JSP pages. This document uses the prefix "fsp" for FirstSpirit DynamicPersonalization tags.

Example for integration in JSP pages:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="FIRSTpersonalisation" prefix="fsp" %>
```

Here it must be noted that the URI for FirstSpirit is always "FIRSTpersonalisation".



If the "fsp" prefix is changed to another value this prefix must be used for the individual tags, i.e. "<myPrefix:ref>" instead of "<fsp:ref>".

5.2 <fsp:loginRequired>

With the <fsp:loginRequired> tag the page content are only displayed if a logged in user has been determined. If the user is not logged in they are forwarded to the address given in the "loginUrl" attribute to log in there. For this forwarding to function properly it is important that the <fsp:loginRequired> tag is located at the top of the page (in front of any type of HTML or comments).



The <fsp:loginRequired> tag should always be used after the <fsp:authorize> tag. This is especially important if login modules which use the single sign-on are used (e.g. "FS SSO").



Attributes:

Attribute	Meaning	Mandatory parameters
loginUrl	<p>URL of the login page to which the user is to be forwarded if they are not yet logged in.</p> <p>After successful login via the login servlet the user is returned to the original page.</p>	Yes

Example:

```
<fsp:loginRequired loginUrl="../../login.jsp" />
```

5.3 <fsp:authorize>

The <fsp:authorize> tag carries out automatic user authentication. The methods used to do this are defined by the configured login package. The login package configuration is described in Chapter 2.4 (starting on page 23) and Chapter 3 (starting on page 26).

If automatic authentication is not successful (HTTP 401 response), the user can be forwarded to a defined address by the "loginUrl" attribute to authenticate manually at that location.

Attributes:

Attribute	Meaning	Mandatory parameters
force	<p>If the "false" value is used for the "force" attribute, authentication is not carried out if a user is already logged in. If the value is "true", authentication is always carried out again, even if a user is already logged in.</p> <p>If the attribute is not specified, the evaluation automatically uses the default value "false".</p>	No



loginUrl	<p>URL of login page to which the user is forwarded if automatic authentication fails. In this case, in addition to the HTTP 401 response ("Unauthorized"), an HTML meta redirect to the "loginUrl" defined here is returned to give the user a manual login option.</p> <p>After logging in successfully using the login servlet, the user is forwarded to the original page.</p>	Yes
----------	--	-----

Example:

```
<fsp:authorize>
  force="false"
  loginUrl="../../login.jsp" />
</fsp:authorize>
```

5.4 <fsp:userInfo>

The <fsp:userInfo> tag returns the user name of the currently logged in user in the "login" variable. If no user is logged in the variable contains an empty string. The tag has no parameters whatsoever.

Variables:

Variable	Meaning	Return data type
login	Returns the user name of the currently logged in user or an empty string.	java.lang.String

Example:

```
<fsp:userInfo>
  User name: <%= login %>
</fsp:userInfo>
```



5.5 <fsp:userGroups>

The <fsp:userGroups> tag returns the group names of the groups to which the currently logged in user belongs in the "groupname" variable. The content of the tag is opened as often as there are groups for the logged in user. The "groupname" variable can output precisely one group name with each run.

Variables:

Variable	Meaning	Return data type
groupname	Returns one group name per run for all the groups of the currently logged in user.	java.lang.String

Example:

```
<fsp:userGroups>
  Group name: <%= groupname %>
</fsp:userGroups>
```

5.6 <fsp:userAttributes>

The <fsp:userAttributes> tag returns the name and value of the user attributes in the "attributename" and "attributevalue" variables. If the "attributes" tag attribute is not given, all the user's attributes are run through. If a comma-separated list of attribute names is given with the "attributes" tag attribute , the tag runs through the given attributes only.

Attributes:

Attribute	Meaning	Mandatory parameters
attributes	Comma-separated list of attributes to be run through by the tag. If the "attributes" attribute is not given, all the user's attributes are run through by the tag.	No



Variables:

Variable	Meaning	Return data type
attributename	Name of the attribute	java.lang.String
attributevalue	Value of the attribute	java.lang.String

Example:

```
<fsp:userAttributes attributes="mail,phone">  
  Attribute name: <%= attributename %>  
  Attribute value: <%= attributevalue %>  
</fsp:userAttributes>
```



5.7 <fsp:isAuthorized>

The <fsp:isAuthorized> tag only displays the included content if the user fulfils the criteria specified by the attributes.

Attributes:

Attribute	Meaning	Mandatory parameters
userRange	<p>Gives the quantity (set) of users to which the tag should relate.</p> <p>Possible values are:</p> <p><code>all</code> The tag allows all users. The data of the "users" and "groups" attributes is ignored.</p> <p><code>loggedIn</code> The tag only allows logged in users and to this end evaluates the "users" and "groups" attributes.</p> <p><code>notLoggedIn</code> The tag allows users who are not logged in only. The data of the "users" and "groups" attributes is ignored.</p> <p>Default value: <code>loggedIn</code></p>	No



Attribute	Meaning	Mandatory parameters
users	Contains a comma-separated list of the allowed users.	No
	<u>Examples:</u>	
	<code>users="Benutzer1,Benutzer2"</code> User1 and User2 are allowed.	
	<code>users=""</code> Logged in users are not allowed.	
	<code>users="*"</code> All (logged in) users are allowed.	
	<code>users="Benutzer1,*"</code> All (logged in) users are allowed; specification of User1 is superfluous.	
	Default value: *	
groups	Contains a comma-separated list of the approved user groups.	No
	<u>Examples:</u>	
	<code>groups="Gruppel,Gruppe2"</code> Users in the groupsGroup1 and Group2 are allowed.	
	<code>groups=""</code> Logged in users are not allowed as groups are not allowed (this also applies to users who are not assigned to any groups).	
	<code>groups="*"</code> All (logged in) users who are assigned to at least one group are allowed.	
	<code>groups="Gruppel,*"</code> All (logged in) users who are assigned to at least one group are allowed; specification of Group1 is superfluous.	
	Default: *	



Attribute	Meaning	Mandatory parameters
exclude	<p>Excludes users or groups if the value is set to "true". The attribute is evaluated in conjunction with <code>userRange="all"</code> or <code>userRange="loggedIn"</code> only.</p> <p><u>Examples:</u></p> <div> <code>userRange="all"</code> <code>exclude="true"</code> This configuration is used to exclude all users, i.e. the tag content are not shown to any users. </div> <div> <code>userRange="loggedIn"</code> <code>exclude="true"</code> <code>users="Benutzer1"</code> Only <code>User1</code> is excluded. All other (logged in) users can see the tag content. </div> <div> <code>userRange="loggedIn"</code> <code>exclude="true"</code> <code>users="*"</code> This configuration is used to exclude all logged in users, i.e. neither logged in nor not logged in users can see the content of the tag. For the case <code>userRange="loggedIn"</code> the tag continues to refer to the set of logged in users, even if <code>exclude="true"</code> is set! This means: </div> <div> <code>user="*"</code> <code>exclude="true"</code> </div> <p>means→ exclude all logged in users but do NOT→ allow all not logged in users</p> <p>The function of the tag can be interpreted as follows: "Only use logged in users as the basic set of any allowed users. From this, exclude all users (<code>user="*" exclude="true"</code>). (i.e. do not allow any users.)"</p> <p>Default: false</p>	No



Example:

```
<fsp:isAuthorized
  userRange="loggedIn"
  users="admin"
  groups="projektleiter"
  exclude="false">

  Protected content

</fsp:isAuthorized>
```

5.8 <fsp:isNotAuthorized>

The <fsp:isNotAuthorized> tag is the exact opposite or negation of the <fsp:isAuthorized> tag (see Chapter 5.7 page 65). The <fsp:isNotAuthorized> tag has the same attributes as the <fsp:isAuthorized> tag. The meaning of all attributes is identical. The difference between the two tags is that the <fsp:isNotAuthorized> tag displays its content exactly when the <fsp:isAuthorized> tag conceals its content (i.e. if the user does not satisfy the given criteria). If, on the other hand, <fsp:isAuthorized> displays its content then <fsp:isNotAuthorized> conceals its content. Summarising it can be stated that the content of both tags should never be displayed simultaneously; instead, either the content of <fsp:isAuthorized> is displayed or the content of <fsp:isNotAuthorized>.

Example:

```
<fsp:isNotAuthorized
  userRange="loggedIn"
  users="admin"
  groups="projektleiter"
  exclude="false">

  Unprotected content

</fsp:isNotAuthorized>
```

5.9 <fsp:logout>

The <fsp:logout> tag logs out the currently logged in user.

Example:

```
<fsp:logout />
```



6 Functional Enhancement

This chapter deals with possibilities for specifically adapting the functionality of FirstSpirit DynamicPersonalization to the requirements of a project.

6.1 Session variables

The following session variables are available after successful login:

Variable	Meaning	Data type
FIRSTpersonalisation.user	The current user (see Chapter 6.2 "Interface")	de.espirit.firstspirit.opt.personalisation.User
FIRSTpersonalisation.usergroups	Groups of the current user as a list of strings	java.util.List<java.lang.String>

6.2 Interface "User"

The `de.espirit.firstspirit.opt.personalisation.User` interface makes the following methods available:

`getLogin():`

Return the login name of the user
(cf. Chapter 6.2.1 page 70).

`getGroups():`

Return the groups of the user
(cf. Chapter 6.2.2 page 71).

`addGroup(String):`

Add a group to the user
(cf. Chapter 6.2.3 page 71).

`setGroups(List<String>):`

Set the user's groups
(cf. Chapter 6.2.4 page 71).



isInGroup(String):

Check whether the user is a member of the transferred group

(cf. Chapter 6.2.5 page 72).

isInGroups(String):

Check whether the user is a member of at least one of the transferred groups

(cf. Chapter 6.2.6 page 72).

getAttributes():

Return all the user's attributes

(cf. Chapter 6.2.7 page 72).

getAttribute(String):

Return a specific attribute value of the user

(cf. Chapter 6.2.8 page 73).

addAttributes(Map<String, String>):

Add several attributes to the user

(cf. Chapter 6.2.9 page 73).

setAttribute(String, String):

Add an attribute to the user

(cf. Chapter 6.2.10 page 73).

clearAttributes():

Reset the user's attributes

(cf. Chapter 6.2.11 page 74).

6.2.1 "getLogin()" method

The "getLogin()" method returns the user's login name as a string.

Method signature:

```
public java.lang.String getLogin();
```

Method signature (abbreviated form):

```
getLogin():String
```



6.2.2 "getGroups()" method

The "getGroups()" method returns the groups to which the user belongs as a list of strings.

Method signature:

```
public java.util.List<java.lang.String> getGroups();
```

Method signature (abbreviated form):

```
getGroups():List<String>
```

6.2.3 "addGroup(String)" method

The "addGroup(String)" method can be used to add a group to the user. The group information for a user is determined using the group module. If the user is to have additional other group assignments (e.g. from an external database), these groups can be added using the "addGroup(String)" method. The name of the group to be added is used as the transfer parameter.

Method signature:

```
public void addGroup(java.lang.String groupName);
```

Method signature (abbreviated form):

```
addGroup(String):void
```

6.2.4 "setGroups(List<String>)" method

The "setGroups(List<String>)" method can be used to set a user's groups. Any group assignments are not adopted, i.e. the groups are "reset".

Method signature:

```
public void setGroups(java.util.List<java.lang.String> groups)
```

Method signature (abbreviated form):

```
setGroups(List<String>):void
```



6.2.5 "isInGroup(String)" method

The "isInGroup(String)" method can be used to check whether a user belongs to a certain group or not. The method returns a Boolean value. "true" is returned if the user belongs to the transferred group and "false" if they do not belong to the group.

Method signature:

```
public boolean isInGroup(java.lang.String groupName)
```

Method signature (abbreviated form):

```
isInGroup(String):boolean
```

6.2.6 "isInGroups(String)" method

The "isInGroups(String)" method can be used to check whether a user belongs to one of the given groups or not. A comma-separated list of group names must be transferred to the method. The method returns a Boolean value. "true" is returned if the user belongs to at least one of the transferred groups and "false" if they do not belong to any of the groups.

Method signature:

```
public boolean isInGroups(java.lang.String groupNames)
```

Method signature (abbreviated form):

```
isInGroups(String):boolean
```

6.2.7 "getAttributes()" method

The "getAttributes()" method returns all the attributes of a user as a map. Each attribute is a key value pair of the map and consists of the attribute name ("key") and the attribute value ("value").

Method signature:

```
public java.util.Map<java.lang.String, java.lang.String> getAttributes()
```

Method signature (abbreviated form):

```
getAttributes():Map<String, String>
```



6.2.8 "getAttribute(String)" method

The "getAttribute(String)" method is used to return the value of a specific attribute of the user.

Method signature:

```
public java.lang.String getAttribute(final java.lang.String name)
```

Method signature (abbreviated form):

```
getAttribute(String):String
```

6.2.9 "addAttributes(Map<String, String>)" method

The "addAttributes(Map<String, String>)" method can be used to add several attributes to the user. The quantity (set) of attributes to be added is the quantity (set) of key value pairs of the transferred map.

Method signature:

```
public void addAttributes(java.util.Map<java.lang.String, java.lang.String>  
attributes)
```

Method signature (abbreviated form):

```
addAttributes(Map<String, String>):void
```

6.2.10 "setAttribute(String, String)" method

The "setAttribute(String, String)" method can be used to add precisely one attribute to the user. The first transfer parameter is the attribute name ("key") and the second is the attribute value ("value").

Method signature:

```
public void setAttribute(java.lang.String name, java.lang.String value)
```

Method signature (abbreviated form):

```
setAttribute(String, String):void
```



6.2.11 "clearAttributes()" method

The "clearAttributes()" method is used to reset the user's attributes. After it is evoked the attributes set is "empty".

Method signature:

```
public void clearAttributes()
```

Method signature (abbreviated form):

```
clearAttributes():void
```



7 Configuring with JOSSO (Java Open Single Sign-On) ²

To perform the following steps JOSSO must already be installed on the web server.

7.1 Configuring personalization

The "Request Header Login" login module of the personalization which reads out the user name from the request header is required for the configuration (see Chapter 3.1.5 page 32).

This module does not require an authentication module as only user names of already authenticated users are expected in the header.

Any group and attribute modules can then be selected.

7.2 Configuring the FirstSpirit project

7.2.1 Via structure variables or project settings

The following variables must be defined in the project for configuration with JOSSO. The variables can either be defined within the project's Site Store or through the project settings. The given variable names are a recommendation.

Variable name	Value
ps_josso_loginUrl	http://MYSERVER/josso/signon/usernamePasswordLogin.do (MYSERVER is to be replaced with the real server URL)
ps_josso_logoutUrl	http://MYSERVER/josso/signon/logout.do (MYSERVER is to be replaced with the real server URL)
ps_josso_backTo	Page (within the project) which is to be opened after successful login/logout.

² <http://www.josso.org/>



ps_josso_onError	Page (within the project) which is to be opened after an error.
------------------	---

7.2.2 Configuring the personalized pages (Page Store)

Each personalized JSP content page must contain the following code at the start of the page.

```
<%@ taglib uri="FIRSTpersonalisation" prefix="fsp" %>
<fsp:authorize/>
```

7.2.2.1 Configuration example (login form)

```
<form
  name="login"
  method="post"
  action="$CMS_VALUE(ps_josso_loginUrl) $">
  <input
    type="hidden"
    name="josso_cmd"
    value="login" />
  <input
    type="hidden"
    name="josso_back_to"
    value="$CMS_VALUE(ps_josso_backTo) $" />
  <input
    type="hidden"
    name="josso_on_error"
    value="$CMS_VALUE(ps_josso_onError) $" />
  <input
    type="text"
    name="josso_username"
    value="" />
  <input
    type="password"
    name="josso_password"
    value="" />
  <input
    type="submit"
    value="Login" />
</form>
```



7.2.2.2 Configuration example (logout form)

```
<form
  name="jossologout"
  method="post"
  action="$CMS_VALUE(ps_josso_logoutUrl) $">
  <input
    type="hidden"
    name="josso_back_to"
    value="$CMS_VALUE(ps_josso_backTo) $" />
  <input
    type="submit"
    value="Logout" />
</form>
```



8 Legal notices

The module "FirstSpirit™ DynamicPersonalization" is a product of the e-Spirit AG, Dortmund, Germany.

When using this module only the licence agreed between the e-Spirit AG and the user is valid.

You can find information about third-party software which is potentially used for the module but not produced by the e-Spirit AG, their own licences and - as the case may be - information about updates on the start page of each FirstSpirit server in the area "Legal notices".

