ImCMS Documentation

Release 1.0.0

imCodePartnerAB

Jun 16, 2023

Contents

CHAPTER 1

Overview of ImCMS

In this article:

1.1 Introduction

New CMS future is available now with new ImCMS 6. It provide new great feature and make users life easier.

1.2 What’s new in ImCMS 6

ImCMS 6 provide new powerful on place editors system that make easier edit well known images, text-fields, menus, loops in the several click without reloading page.

Now in ImCMS 6 available new searching system that called Solr, it provide fast and easy searching among all documents in the system.

New ImCMS provide easy access and management of existing documents with new modern Admin Panel.

Looks at our documentation to find more powerful features that provided in ImCMS 6.

CHAPTER 2

Setup

2.1 System Requirements

2.1.1 Browsers

ImCMS should work in next browsers:

- Firefox (Latest)
- Chrome (Latest)
- Safari (Latest)
- IE10+

2.1.2 Local Development and Hosting

- Java 8 or higher
- Tomcat 8 or higher
- MySQL 5.7 or higher

2.2 Before You Start

It is required to have MySQL database server installed for ImCMS.

2.2.1 Setting up database

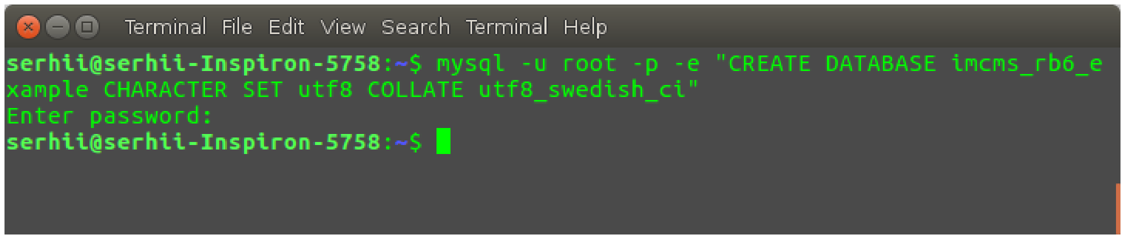
ImCMS works fine with **MySQL 5.7+** but may work on higher versions too. You need to create new database on your database server.

Example:

To create a MySQL database, a command like the following might be appropriate:

```
mysql -u <user> -p -e "CREATE DATABASE <db-name> CHARACTER SET utf8 COLLATE utf8_
↪swedish_ci"
```

Here is an example of such command execution:



```
serhii@serhii-Inspiron-5758:~$ mysql -u root -p -e "CREATE DATABASE imcms_rb6_e
xample CHARACTER SET utf8 COLLATE utf8_swedish_ci"
Enter password:
serhii@serhii-Inspiron-5758:~$
```

After you write the password for mysql user new database is created and is ready to use. Don't forget your DB name, it will be needed soon for next ImCMS configuration.

2.3 Start With ImCMS as a User

It is quite simple to to start working with ImCMS. But since you've prepared database, there are few more things that are required.

In this article:

- *Installing Java*
- *Installing Tomcat*
- *ImCMS Configuration*
- *Running ImCMS*

2.3.1 Installing Java

To use ImCMS you have to have java8 installed on your machine. You can use either OracleJDK or OpenJDK.

2.3.2 Installing Tomcat

Tomcat is a servlet container for running web applications such as ImCMS. First of all, download latest Tomcat 8 from official site and then simply extract an archive into any folder. We will call the result tomcat directory with it's full path **tomcat_home**.

2.3.3 ImCMS Configuration

Next step - download latest ImCMS. Then put downloaded file here: `tomcat_home/webapps/`. You can rename this file, so result will look like this: `tomcat_home/webapps/imcms.war`.

Before making any other changes, it is required to configure ImCMS to use your database. Open downloaded file by any archive manager, and then open this file: `/WEB-INF/conf/server.properties`. Find property `JdbcUrl` and change it by setting up correct database name like this: `JdbcUrl = jdbc:mysql://localhost:3306/<db-name>?characterEncoding=utf8` Then edit database login section:

```
# Database login
User = #your database user name#
Password = #your database user password#
```

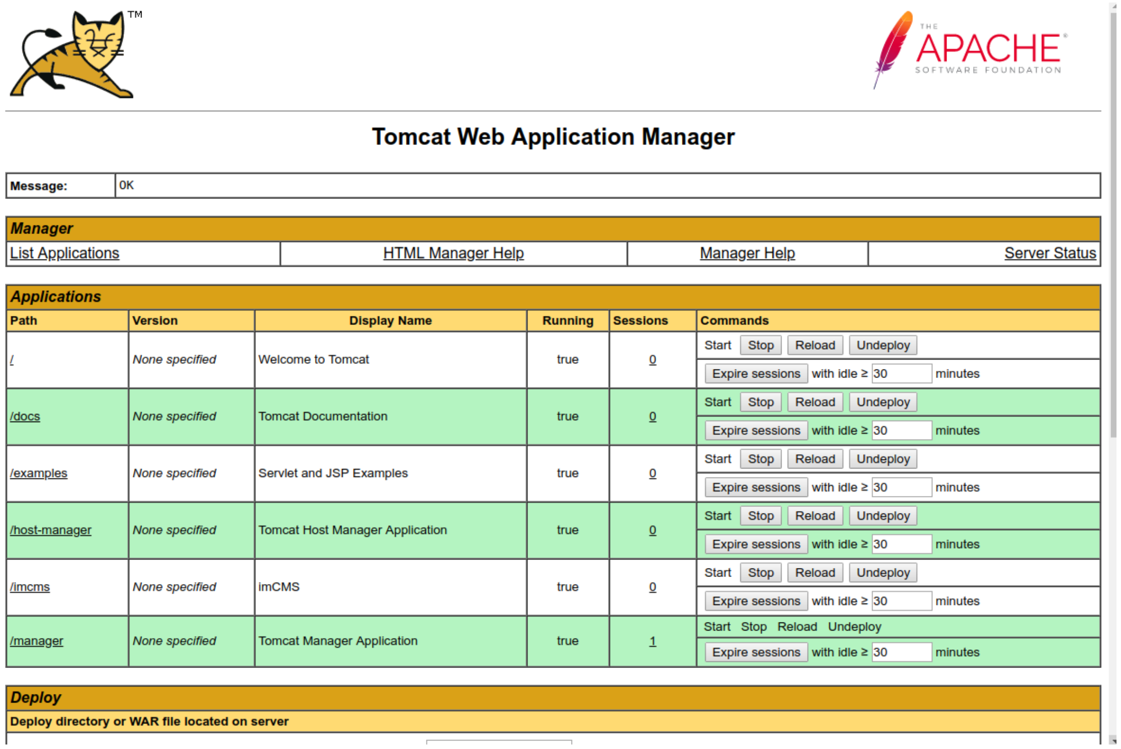
Save the file and basic ImCMS configuration finished.

2.3.4 Running ImCMS

After basic configuration done, you can run ImCMS on Tomcat. To do so, go to this path: `tomcat_home/bin` and run startup script in terminal:

```
sh startup.sh
```

In the end of result message you should receive this: **Tomcat started.** Tomcat started on standard path, check in your browser: http://localhost:8080/manager/ You should see this:



Tomcat Web Application Manager

Message:	OK

Manager

List Applications	HTML Manager Help	Manager Help	Server Status

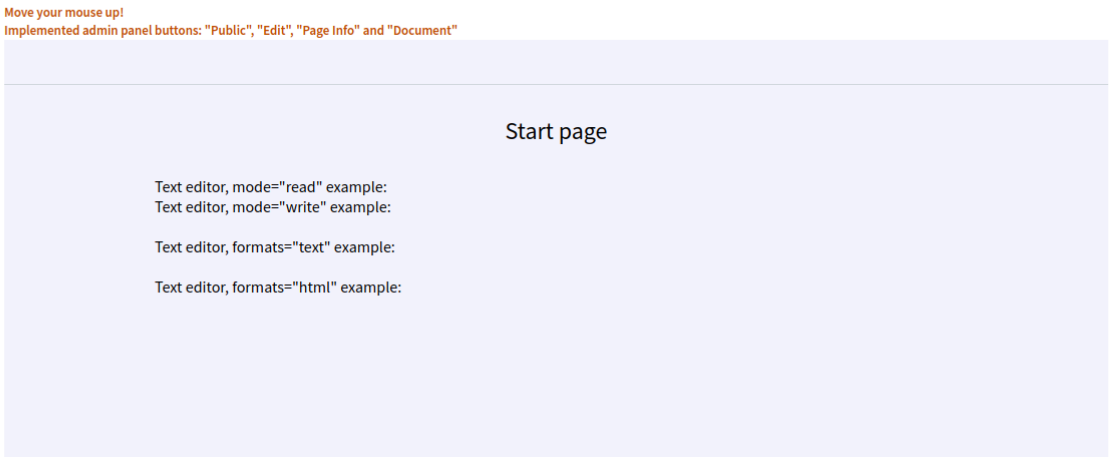
Applications

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/imcms	None specified	imCMS	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy

Deploy directory or WAR file located on server

In *Applications* section find path **/imcms** (or how you renamed downloaded file) with state *Running* - true. Click on path and you should see this:


Move your mouse up!
Implemented admin panel buttons: "Public", "Edit", "Page Info" and "Document"

Start page

Text editor, mode="read" example:
Text editor, mode="write" example:

Text editor, formats="text" example:

Text editor, formats="html" example:

If not, check each step twice to be sure that you've done all things right.

2.4 Start With ImCMS as a Developer

In this article:

- *Required things*
- *Download sources*
- *Building application*

2.4.1 Required things

There are some additional requirements if you want to develop ImCMS:

- git
- Maven 3+
- Your favorite IDE, text editor or even only terminal

2.4.2 Download sources

At first you need to clone ImCMS from github by this link: https://github.com/imCodePartnerAB/imcms.git

2.4.3 Building application

Maven is used as a build tool so run next command from project root after sources downloaded:

```
mvn clean
```

Success execution means that all was done right.

Then package it:

```
mvn clean package
```

Your first package should fail with this message:

Warning: *build.properties* file was just created, fill in required properties and run execution again

Open this file (it is located in project root directory) `build.properties`. Besides a lot of defaults, find this:

```
db-host = localhost
db-name = imcms
db-user = root
db-pass =
```

And write down your database host, name, user and password.

It's not the end! There are a lot of tests that should be executed on maven's package phase. Test DB is needed for this purposes. Go to `src/test/resources/test.server.properties` and write down correct values for next properties:

```
JdbcDriver = com.mysql.jdbc.Driver
JdbcUrl = jdbc:mysql://localhost:3306/imcms_test?characterEncoding=utf8&useSSL=false
User = root
Password = root
```

Pay attention to `JdbcUrl` property - there is a DB name after `localhost:3306/`, by default it is `imcms_test`, so you can create DB with such name (just like in *Before You Start* section) or with another name - then simply put this name into `test.server.properties` file instead of default DB name.

Since you've done, execute maven clean package again:

```
mvn clean package
```

All tests should be executed successfully, application built and ready to use. Deploy it to Tomcat and run.

CHAPTER 3

Tutorial

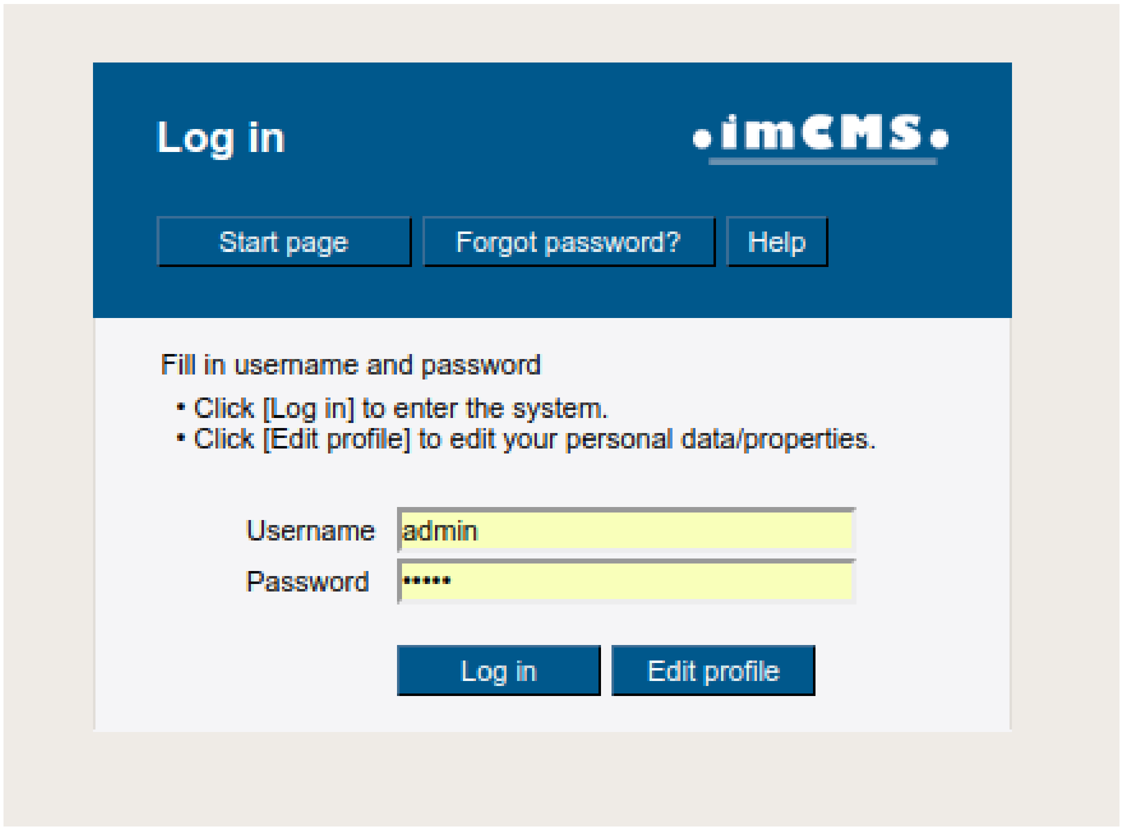
3.1 Logging in

When ImCMS is running, it's URL consist of next parts:

- Protocol - `http` or `https`
- Host - `localhost` if running locally, or any another if it is dev/prod
- Port - usually `:8080` or another *(optional)*
- Context path - could be anything after previous parts *(optional)*

The simplest variation is `https://imcode.com`, and for local is `http://localhost:8080/imcms`

There is a URL reserved for logging in - `/login`. Simply add it after ImCMS URL, so you can get this (using previous examples): `https://imcode.com/login`. You should see next page (this is an old design):



Fill in username and password

- Click [Log in] to enter the system.
- Click [Edit profile] to edit your personal data/properties.

Username	admin
Password	•••••

By default login and password is the same - `admin`.

There is possibility to set next url after login:

`next_meta` - id of desired document `next_url` - URL that user will be redirected to after successful logging in. Can be taken from current page URL before redirect, so user will be returned back.

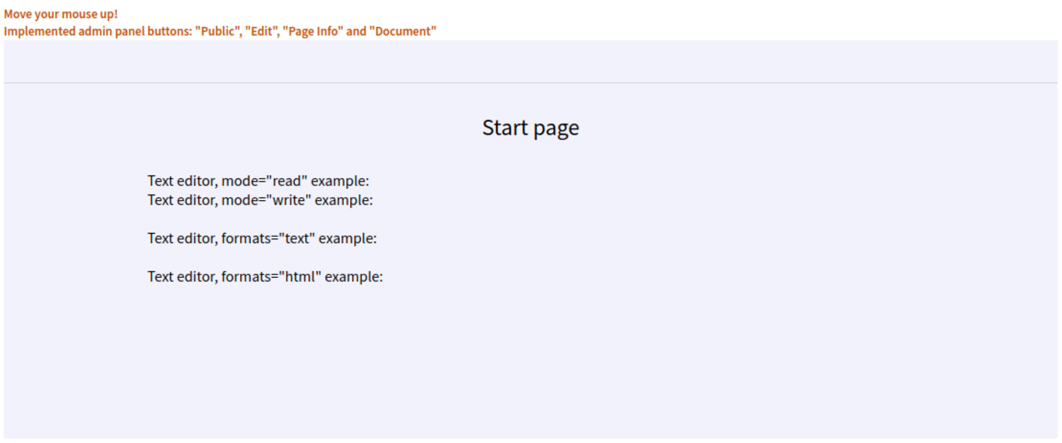
Examples:

http://imcms.dev.imcode.com/login?next_meta=1003

http://imcms.dev.imcode.com/login?next_url=http://imcms.dev.imcode.com/test-page-one-headline

3.2 First Web Site

1. Follow the *setup guide* to run ImCMS.
2. Now, if everything is ok you will get next page:



Move your mouse up!
Implemented admin panel buttons: "Public", "Edit", "Page Info" and "Document"

Start page

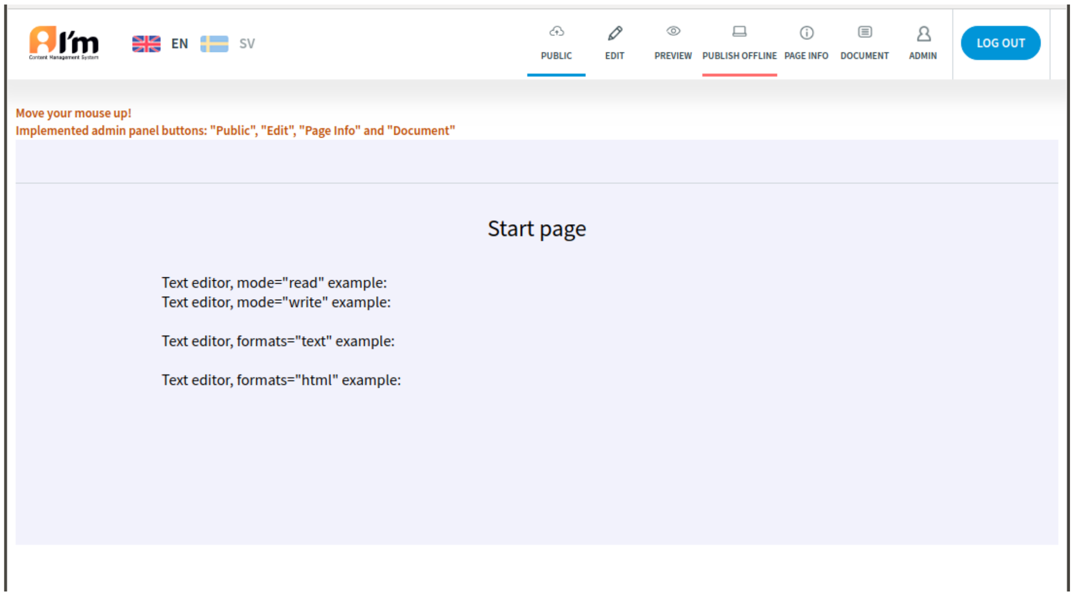
Text editor, mode="read" example:
Text editor, mode="write" example:

Text editor, formats="text" example:

Text editor, formats="html" example:

it means that ImCMS system has started and now working perfectly.

3. Lets login as admin. Check *Logging in* section.
4. If everything is okay you will see same page, but with **Admin Panel** (see *Admin Panel* section).



Move your mouse up!
Implemented admin panel buttons: "Public", "Edit", "Page Info" and "Document"

Start page

Text editor, mode="read" example:
Text editor, mode="write" example:

Text editor, formats="text" example:

Text editor, formats="html" example:

5. Lets create new document. Follow *Document Management* guide for that.
6. Great - now you know how to create new page, but you can ask the question: *How should I fill content on my new page?* - This is the time to use **Edit Mode**. It gives an ability to manage all editable content you can find in *Content Management* section. The simplest one is text editing, we will try to write our first *Hello World*.

   - Enable *Edit Mode* ✎ EDIT on the admin panel
   - Find pencil icon ✎ on the page and click it
   - Type "Hello World"

   - On Text Editor panel find button with icon and click it
   - Text has been saved - reload page to be sure

7. Check *Role Management* section to know how to manage access to your document.

CHAPTER 4

Design

Design in ImCMS consists of 2 large concepts: Templates and Tags. Templates are used for the HTML layout of your pages, whereas tags are reusable dynamic components used for embedding navigation, forms, lists, and so on in your templates.

4.1 Templates

Describes how to create/modify templates in ImCMS

In this article:

- *Create new template*
- *Modify template*
- *Summary*

4.1.1 Create new template

1. Create new file with `jsp` extension in `WEB-INF/templates` directory.
2. Add following code to created file

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>Template</title>
    <meta charset="utf-8"/>
</head>
<body>
    Hello World with New Template!
</body>
</html>
```

3. Run application using **Tomcat** (make a package if you are using maven, of package only this new file if you are using your IDE).
4. Login into ImCMS as Admin. Check *Logging in* section.
5. Open **Page info** on **Admin Panel** (see *Admin Panel guide*), choose **Appearance** tab and select your file in **Template** list.
6. Save changes by clicking **"OK"** button

Modify template

Modify your templates in `/WEB-INF/templates` directory until you're done. Keep your templates as simple as possible. Try to use one template per document, move common used parts into separated JSP files with their including. Logic can be moved to functions in TLDs. Use JSTL tags over scriptlets.

4.1.2 Summary

Now you know how to create, modify and use templates.

4.2 Tag Engine

4.2.1 Text Tag

In this article:

- *Introduction*
- *Use in template*

Introduction

Each web-page contains piece of text. It can be description of image of information about page. Usually it can be altered, edited and even deleted. Since text can be stored in database ImCMS provide easy access to it via `text` tag.

Use in template

For configure `text` tag in template just look at the code below.

```
<imcms:text no="1" pre="<div>" post="</div>"/>
```

Available list of tag attributes:

Attribute	Type	Description
no	Integer	Identifier for current text
document	Integer	Identify the linked document (default - current document)
version	Integer	Identify version of text
placeholder	String	The text that was showed if native content are empty
pre	String	Text or html tag that would be added before text tag
post	String	Text or html tag that would be added after text tag
mode	String	Possible values: `read` - means that text won't be editable `write` - editable text, just as without `mode` attribute
formats	String	If set, format switch won't be able. Possible values: `text` - formatting panel will have only simple text editor options, content won't be represented as HTML `html` - formatting panel will have HTML editor options, content will be represented as HTML `cleanhtml` - formatting panel will have HTML editor options, content will be represented as HTML and all tags will be checked according to tags whitelist - system property `text.editor.html.tags.whitelist`, where tags are separated by semicolon. Not allowed tags will be removed from content.
label	String	Text label that is connected to current text tag
showlabel	String	Set `true` if you want to see text label near text tag content in admin edit mode
showEditToSuperAdmin	String	Set `true` if you want to that be text field will be show only for super admin

Example:

```
<%@taglib prefix="imcms" uri="imcms" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>Template</title>
    <meta charset="utf-8"/>
</head>
<body>
    <imcms:text no="1" document="1001" pre="<div>" post="</div>" placeholder="<i>this␣
↪text is empty</i>"
                label="Test text tag" showlabel="true" formats="html",␣
↪showEditToSuperAdmin="true"/>
```

(continues on next page)

(continued from previous page)

```
</body>
</html>
```

4.2.2 Menu Tag

In this article:

Introduction

Well-known, an integral part of the page is menu. It navigate users overall web-site, and describe what kind of content was presented on. Since ImCMS based on document-presented pages each menu item can be present as specific document link

Use in template

There are several important part configure menu fully:

- HTML structure such as `ul-li` list or `div-div` list
- Information about menu-nesting

Default 2-level structured menu shows below:

```
<imcms:menu no='1' docId="1001">
    <ul>
        <imcms:menuloop>
            <imcms:menuitem>
                <li>
                    <imcms:menuitemlink>
                        ${menuitem.document.headline}
                    </imcms:menuitemlink>
                    <!-- sub menu definition -->
                    <imcms:menuloop>
                        <imcms:menuitem>
                            <div>
                                <imcms:menuitemlink>
                                    ${menuitem.document.headline}
                                </imcms:menuitemlink>
                            </div>
                        </imcms:menuitem>
                    </imcms:menuloop>
                </li>
            </imcms:menuitem>
        </imcms:menuloop>
    </ul>
</imcms:menu>
```

Available list of tag attributes:

Example:

```
<%@taglib prefix="imcms" uri="imcms" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>Template</title>
    <meta charset="utf-8"/>
</head>
<body>
    <imcms:menu no='1' docId="1001">
        <ul>
            <imcms:menuloop>
                <imcms:menuitem>
                    <li>
                        <imcms:menuitemlink>
                            ${menuitem.title}
                        </imcms:menuitemlink>
                        <!-- sub menu definition -->
                        <imcms:menuloop>
                            <imcms:menuitem>
                                <div>
                                    <imcms:menuitemlink>
                                        ${menuitem.title}
                                    </imcms:menuitemlink>
                                </div>
                            </imcms:menuitem>
                        </imcms:menuloop>
                    </li>
                </imcms:menuitem>
            </imcms:menuloop>
        </ul>
    </imcms:menu>
</body>
</html>
```

Second example: with use nested, disable nested in the menu and in each menuItem

```
<imcms:menu no='1' docId="1001" pre="<div><ul>" post="</ul></div>" nested="true"␣
↪label="something" showlabel="true">
               <li><imcms:menuitemlink>${menuitem.title}</imcms:menuitemlink></li>
</imcms:menu>
```

Third example: generate automatic html menu items

```
<imcms:menu index='1' nested="true" wrap="span, b, i" attributes="wcag, data, class",␣
↪treeKey="20"/>
```

4.2.3 Loop Tag

In this article:

Introduction

Probably everyone faced with a situation where same content should displayed several times. Of course this problem can be solve with well-known standard JSP tag `forEach`, but it is not enough if count cycle of content changed very often. That is why ImCMS provide own cycle tag that called `loop`. `loop` tag works like the `forEach` tag, but the main feature of it is visual editor that provide agile configuration of content's count, etc.

Use in template

For configure `loop` tag in template just look at the code below.

```
<imcms:loop no="1" pre="<div>" post="</div>" label="Loop tag example">
    <imcms:loopentry>
        <imcms:loopitem>
            ...some content that will be repeated
        </imcms:loopitem>
    </imcms:loopentry>
</imcms:loop>
```

Available list of tag attributes:

Attribute	Type	Description
no	Integer	Identifier for current loop
pre	String	Text or html tag that would be added before loop tag
post	String	Text or html tag that would be added after loop tag
label	String	Label that is used to describe loop tag

Example:

```
<%@taglib prefix="imcms" uri="imcms" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>Template</title>
    <meta charset="utf-8"/>
</head>
<body>
    <imcms:loop no="1" pre="<div>" post="</div>" label="Loop tag example">
        <ul><imcms:loopentry>
            <li><imcms:loopitem>
                <c:set var="loopEntryRef" value="${loopitem.loopEntryRef}"/>
```

(continues on next page)

(continued from previous page)

```
                <c:out value="Entry no: ${loopEntryRef.entryNo}"/>
                <c:out value="Loop no: ${loopEntryRef.loopNo}"/>
                <div class="figure">
                    <imcms:image no="3" document="${document.id}" style="max-
↪width:100px;"/>
                    <div class="description">
                        <imcms:text no="3" document="${document.id}"/>
                    </div>
                </div>
            </imcms:loopitem></li>
        </imcms:loopentry></ul>
    </imcms:loop>
</body>
</html>
```

4.2.4 Image Tag

In this article:

- *Introduction*
- *Use in template*

Introduction

Today, maybe, there are no pages that contains only text content. It can contains topic, or news and of course it contain logo image, or preview image. That is why it is very important to flexibly configure images. ImCMS provide own `image` tag that give easy access to image, stored in the system.

Use in template

For using image in template all that needed are insert image-tag in the desired place.

Available list of tag attributes:

Attribute	Type	Description
no	Integer	Identifier for current image
document	Integer	Identify the linked document (default - current document)
version	Integer	Identify version of image
id	String	Html attribute `id`
pre	String	Text or html tag that would be added before image tag
post	String	Text or html tag that would be added after image tag
styleClass	String	Add html attribute `class` to image
style	String	Add html attribute `style` to image

Example:

```
<%@taglib prefix="imcms" uri="imcms" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>Template</title>
    <meta charset="utf-8"/>
</head>
<body>
    <imcms:image no="1" document="1001" pre="<div>" post="</div>"/>
</body>
</html>
```

4.2.5 Login Tag

In this article:

- *Introduction*
- *Use in template*

Introduction

`Login` tag it is a simple on page logging, that help sign in user without redirect on base sing in page.

Use in template

For configure `login` tag in template just consider the code below.

```
<imcms:login>
    <imcms:loginname attributes="placeholder='Enter your login'"/>

    <imcms:loginpassword/>
</imcms:login>
```

Example:

```
<%@taglib prefix="imcms" uri="imcms" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>Template</title>
    <meta charset="utf-8"/>
</head>
<body>
    <imcms:login>
        <div class="field">
            <label>Login</label>
            <imcms:loginname attributes="placeholder='Enter your login'"/>
        </div>
        <div class="field">
```

(continues on next page)

(continued from previous page)

```
                <label>Password</label>
                <imcms:loginpassword/>
            </div>
            <input type="hidden" name="login" value="login"/>

            <div class="field">
                <button class="positive" type="submit">Login</button>
            </div>
        </imcms:login>
</body>
</html>
```

4.2.6 Register Tag

In this article:

- *Introduction*
- *Use in template*

Introduction

`Register` tag it is a simple on page sign up provider, that help register user in the ImCMS system without redirect on base sing up page.

Use in template

For configure `register` tag in template just consider the code below.

```
<imcms:registration>
    <imcms:registrationlogin/>
    <imcms:registrationemail/>
    <imcms:registrationname/>
    <imcms:registrationsurname/>
    <imcms:registrationpassword1/>
    <imcms:registrationpassword2/>
    <button class="positive" type="submit">Register</button>
</imcms:registration>
```

Example:

```
<%@taglib prefix="imcms" uri="imcms" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>Template</title>
    <meta charset="utf-8"/>
</head>
<body>
<imcms:registration>
```

(continues on next page)

(continued from previous page)

```
    <div class="field">
        <label>Login</label>
        <imcms:registrationlogin/>
    </div>
    <div class="field">
        <label>Email</label>
        <imcms:registrationemail/>
    </div>
    <div class="field">
        <label>Name</label>
        <imcms:registrationname/>
    </div>
    <div class="field">
        <label>Second Name</label>
        <imcms:registrationsurname/>
    </div>
    <div class="field">
        <label>Password</label>
        <imcms:registrationpassword1/>
    </div>
    <div class="field">
        <label>Repeat password</label>
        <imcms:registrationpassword2/>
    </div>
    <div class="field">
        <button class="positive" type="submit">Register</button>
    </div>
</imcms:registration>
</body>
</html>
```

4.2.7 Pager Tag

In this article:

- *Introduction*
- *Use in template*

Introduction

For today in the WEB world nobody can‘t image complex web-page with repeated material without simple pager. ImCMS make developers life easier. ImCMS `pager` tag that split repeated content on separate pages.

Use in template

For configure `pager` tag in template. Consider code below.

```
<imcms:pager visibleItemCount="6">
    ${firstPagerItem} <!--use firstPagerItem here to hold link on first page-->
    <imcms:pageritem>
        ${pagerItem} <!--use pagerItem here-->
    </imcms:pageritem>
```

(continues on next page)

(continued from previous page)

```
    ${lastPagerItem} <!--use firstPagerItem here to hold link on last page-->
</imcms:pager>
```

Note: `pager` tag must be inside tag, that implements `IPageableTag` interface.

Available list of tag attributes:

Attribute	Type	Description
visibleItemCount	Integer	Optional attribute. Describe how many page links will be shown at once

Example:

Consider example below. Since `search` tag implements `IPageableTag` - `pager` tag can be inserted into it.

```
<%@taglib prefix="imcms" uri="imcms" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>Template</title>
    <meta charset="utf-8"/>
</head>
<body>
<imcms:search searchRequest="" skip="0" take="20">
    <ul class="simple-post-list">
        <imcms:searchitem>
            <li>
                <div class="post-info">
                    <a href="${pageContext.request.contextPath}/${searchItem.
↪foundDocument.alias}">${searchItem.foundDocument.headline}</a>

                    <div class="post-meta">
                            ${searchItem.foundDocument.modifiedDatetime}
                    </div>
                </div>
            </li>
        </imcms:searchitem>
    </ul>
    <imcms:pager visibleItemCount="6">
        <ul class="pagination pull-right">
            <li><a href="${firstPagerItem.link}">a</a></li>
            <imcms:pageritem>
                <c:choose>
                    <c:when test="${pagerItem.showed}">
                        <li class="active"><a href="${pagerItem.link}">${pagerItem.
↪pageNumber}</a>
                        </li>
                    </c:when>
                    <c:otherwise>
                        <li><a href="${pagerItem.link}">${pagerItem.pageNumber}</a></
↪li>
```

(continues on next page)

(continued from previous page)

```
                    </c:otherwise>
                </c:choose>
            </imcms:pageritem>
            <li><a href="${lastPagerItem.link}">1</a>
        </ul>
    </imcms:pager>
</imcms:search>
</body>
</html>
```

4.2.8 Search Tag

In this article:

- *Introduction*
- *Use in template*
- *Paging integration*

Introduction

ImCMS support powerful and flex search engine that integrate with well-known Apache Solr. Each document in the system index in solr. Each content item on page indexing too: ImCMS provide searching by text content, document title, document alias, document menu description. Of course searching depends on current language, that user has already been selected.

Use in template

For using search in template all that needed are insert search-tag in the desired place.

```
<imcms:search searchRequest="" skip="0" take="20">
    <imcms:searchitem>
        Some action with ${searchItem} here
    </imcms:searchitem>
</imcms:search>
```

Available list of tag attributes:

Attribute	Type	Description
searchRequest	String	Optional property. Set request is searching in
skip	Integer	Optional property. It is describing how many items in result should be skipped
take	Integer	Optional property. It is describing how many items in result should be taken

Example:

```
<%@taglib prefix="imcms" uri="imcms" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>Template</title>
    <meta charset="utf-8"/>
</head>
<body>
<imcms:search searchRequest="" skip="0" take="20">
    <ul class="simple-post-list">
        <imcms:searchitem>
            <li>
                <div class="post-info">
                    <a href="${pageContext.request.contextPath}/${searchItem.
↪foundDocument.alias}">${searchItem.foundDocument.headline}</a>

                    <div class="post-meta">
                            ${searchItem.foundDocument.modifiedDatetime}
                    </div>
                </div>
            </li>
        </imcms:searchitem>
    </ul>
</imcms:search>
</body>
</html>
```

Paging integration

By default `search` tag provide paging. It is mean that `pager` tag can be inserted in to `search` tag body. (see also *Pager Tag* section).

For configure `search` tag to using paging look at example above.

Example:

```
<imcms:search searchRequest="" skip="0" take="20">
    <ul class="simple-post-list">
        <imcms:searchitem>
            <li>
                <div class="post-info">
                    <a href="${pageContext.request.contextPath}/${searchItem.
↪foundDocument.alias}">${searchItem.foundDocument.headline}</a>

                    <div class="post-meta">
                            ${searchItem.foundDocument.modifiedDatetime}
                    </div>
                </div>
            </li>
        </imcms:searchitem>
    </ul>
    <imcms:pager visibleItemCount="6">
        <ul class="pagination pull-right">
```

(continues on next page)

(continued from previous page)

```
            <li><a href="${firstPagerItem.link}">1</a></li>
            <imcms:pageritem>
                <c:choose>
                    <c:when test="${pagerItem.showed}">
                        <li class="active"><a href="${pagerItem.link}">${pagerItem.
↪pageNumber}</a>
                        </li>
                    </c:when>
                    <c:otherwise>
                        <li><a href="${pagerItem.link}">${pagerItem.pageNumber}</a></
↪li>
                    </c:otherwise>
                </c:choose>
            </imcms:pageritem>
            <li><a href="${lastPagerItem.link}">2</a>
        </ul>
    </imcms:pager>
</imcms:search>
```

CHAPTER 5

API-Documentation

Imcms has the main interface for call API is - ImcmsService; Init ImcmsService - Imcms.getServices();

titlesonly service-beans/index core/index

CHAPTER 6

Content Management

6.1 Document Management

6.1.1 Base Management

In this article:

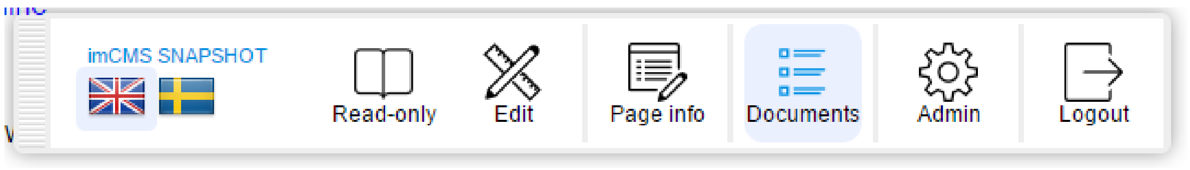
Introduction

This article describes basic document creating and configuring.

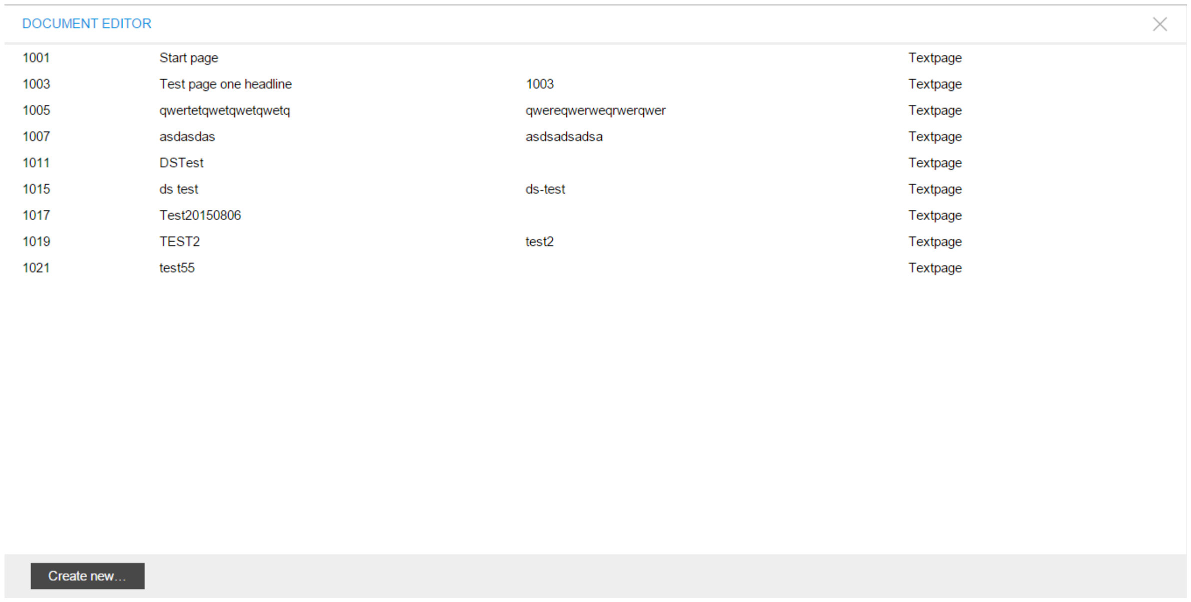
Create Document

Today there are two way of document creation:

1. Create document in document manager: - Click on *Documents* on *Admin Panel*



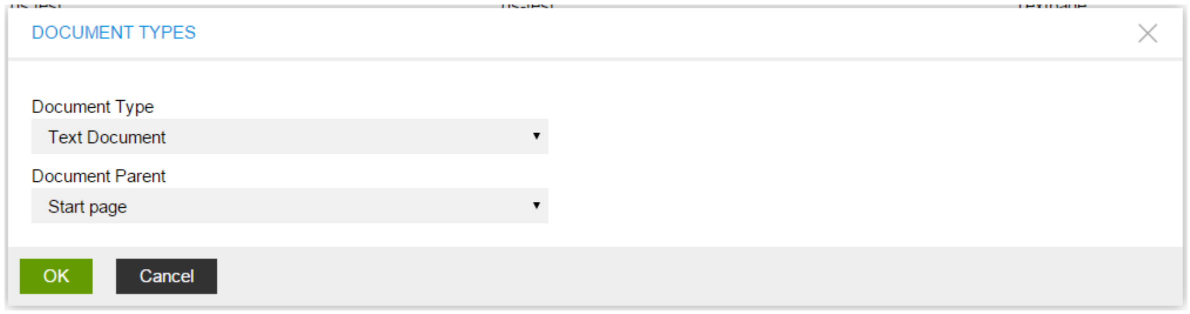
- In opened window click on *Create new…* button to open *new document dialog*


DOCUMENT EDITOR

1001	Start page		Textpage
1003	Test page one headline	1003	Textpage
1005	qwertetqwetqwetqwetq	qwereqwerweqrwerqwer	Textpage
1007	asdasdas	asdsadsadsa	Textpage
1011	DSTest		Textpage
1015	ds test	ds-test	Textpage
1017	Test20150806		Textpage
1019	TEST2	test2	Textpage
1021	test55		Textpage

Create new…

- Opened window is *document prototyping* window: **Document Type** - document type such as Text Document, or File Document, etc; **Document Parent** - document's parent document, that will be inherited for current document.


DOCUMENT TYPES

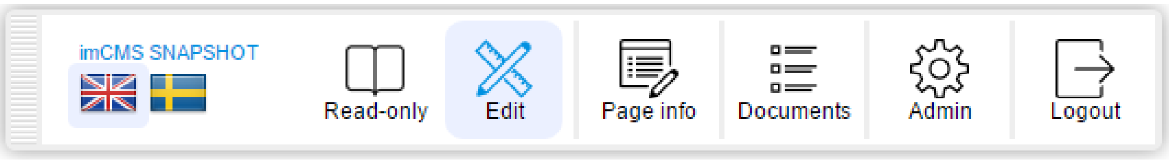
Document Type: Text Document

Document Parent: Start page

OK Cancel

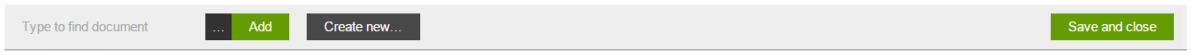
- Click *OK* button, and configure document in opened window.

2. Create document in menu manager * Enable **Edit Mode** as shown below



- Find **Menu Editor** label on the left side of page an click on it for open **Menu Editor**

- Click on **Create new…** button

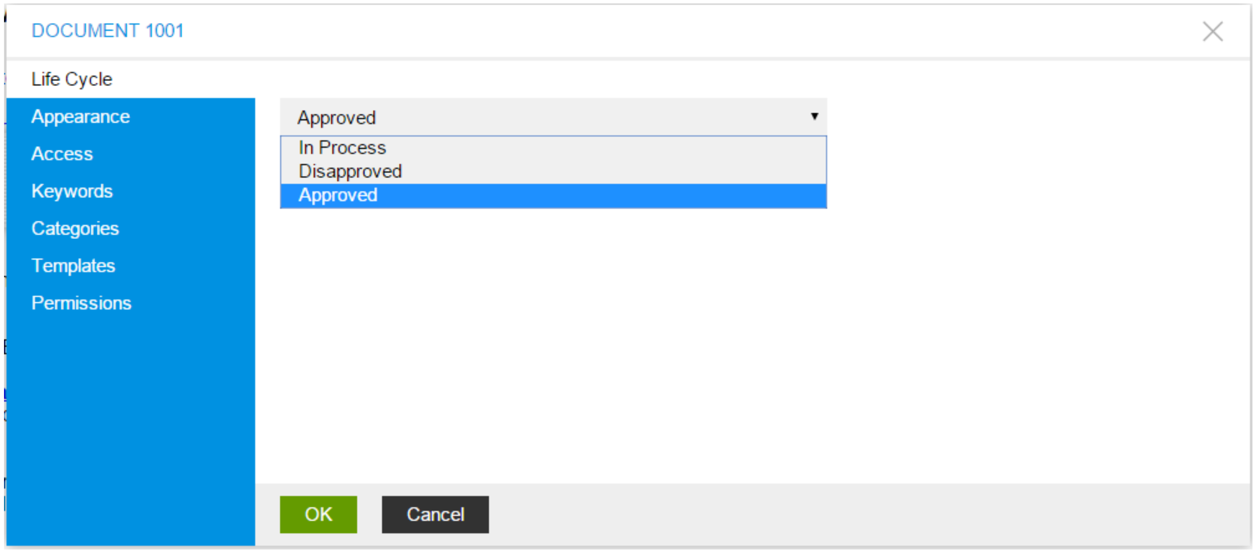


- Follow steps from Create Document in Document Manager section.

Edit Life Cycle

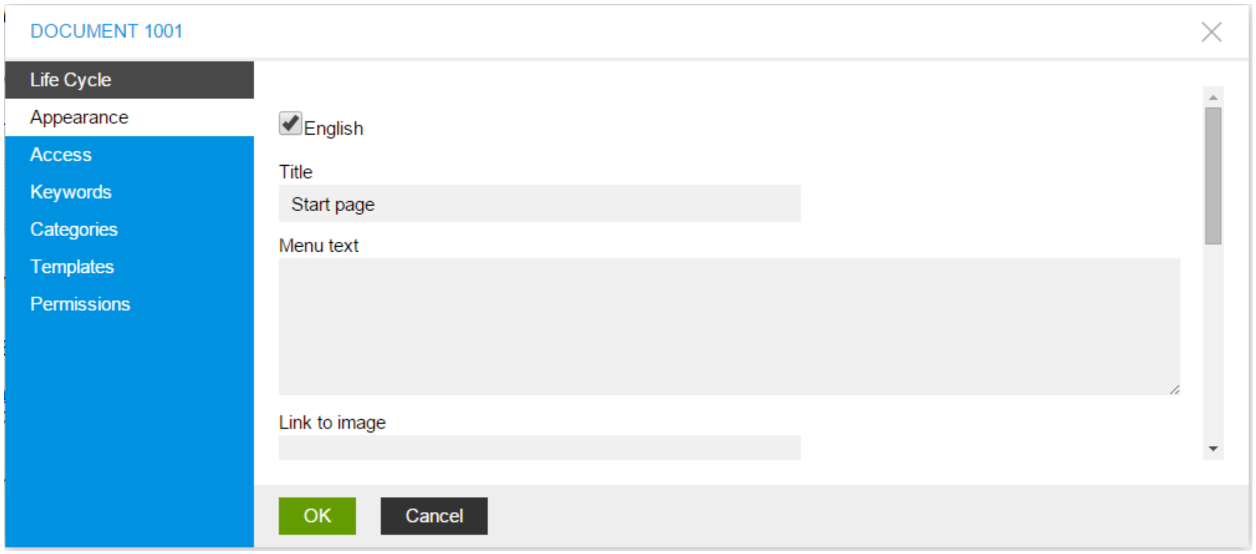
ImCMS provide base document managing, that help change document status. There are 3 types of life cycle exists:

- In Process - it is mean that document has just been created and it is preparing now.
- Approved - this status says that document is ready to use.
- Disapproved - document is disabled, and cannot be accessed.



Edit Appearance

This editor section provides access to manage documents alias, how document will be opened from menu (in new window, in same frame, etc) document name, description, link image for all available in ImCMS system languages.



[✓ English]

Title: Start page

Menu text:

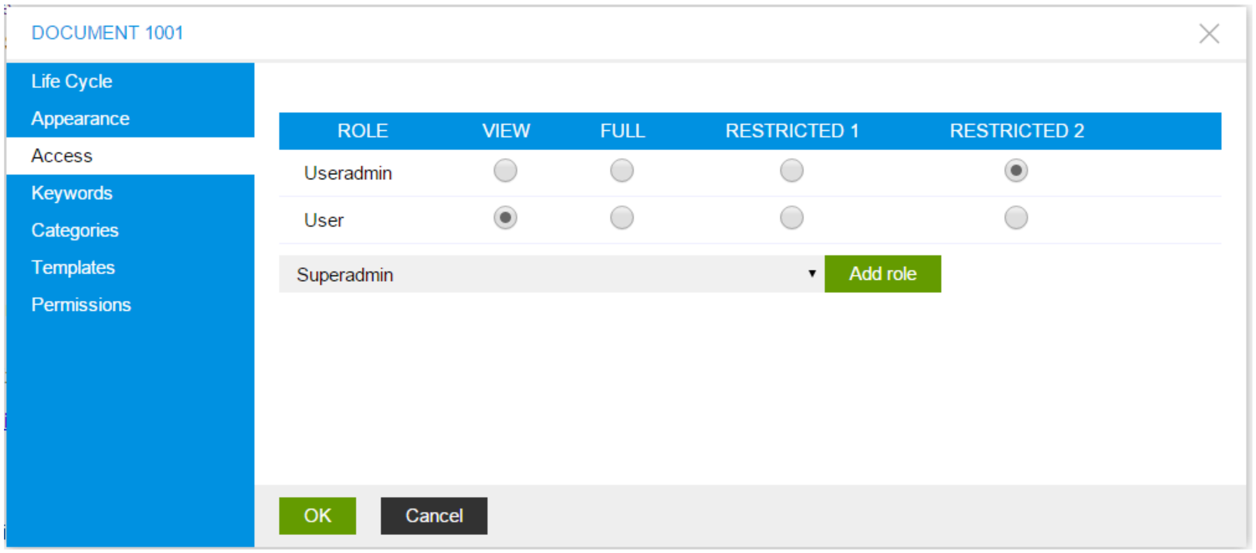
Link to image:

Edit Access

Since ImCMS is care about document securing it provide editor section for configure user-role linking with permission set for current document. There are five types of roles:

- NONE - no access to current page
- VIEW - role with this permission can only view content on this page
- RESTRICTED 1 - first custom permission set
- RESTRICTED 2 - second custom permission set
- FULL - role with this permission have access to any configuration of current document (usual it is **admin**)

More information about custom permission sets configuration is in *Text Document Management* section

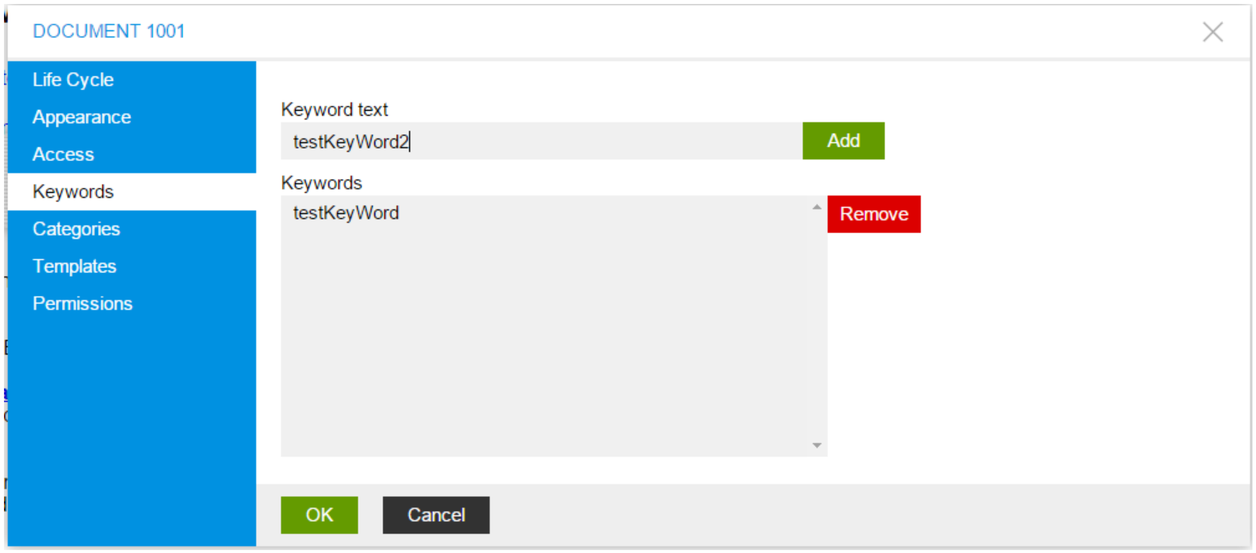


ROLE	VIEW	FULL	RESTRICTED 1	RESTRICTED 2
Useradmin	○	○	○	●
User	●	○	○	○

Superadmin

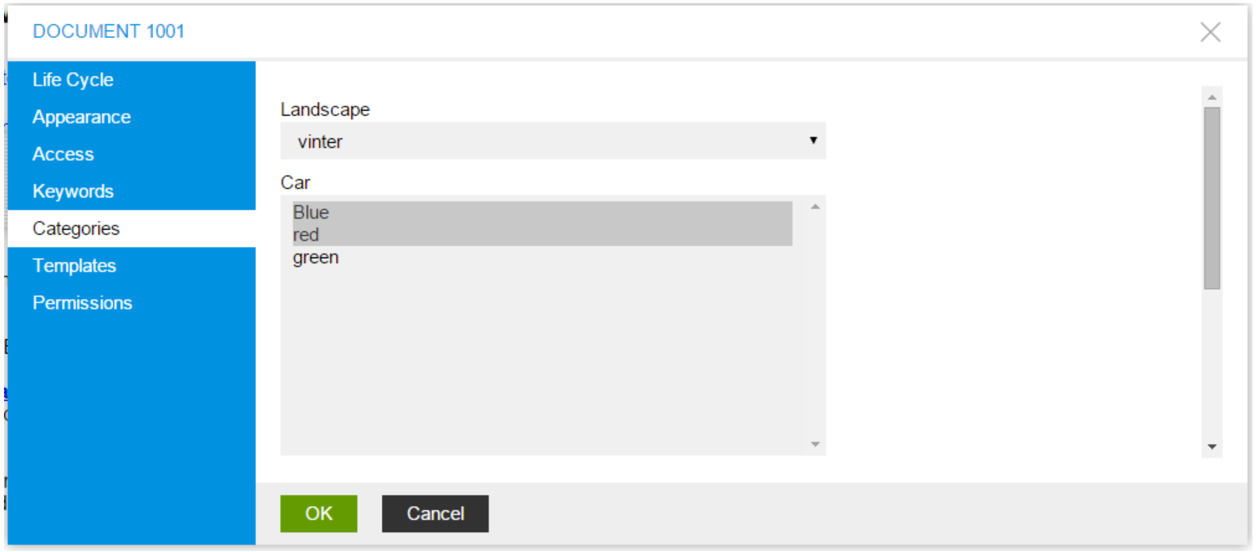
Edit Keywords

Each document in ImCMS system is indexed by the search system, that called *Solr*. That's why it is very important to mark document with special keywords that make searching easily.



Edit Categories

Since ImCMS provide categories, each document can categorized. It should be noted, that one document can be assigned to several categories at the same time if category type support multiply selecting.



Document Versioning

Imcms has versioning feature. Which can be disabled or enabled in properties.

If versioning is enabled: To apply changes press publish button on admin panel. Until that plain users. Will not seee any changes. Also you can preview new content before making publish by appropriate button on admin panel.

If versioning is disabled: No additional actions required.

Warning: If previously versioning was disabled by properties, all existed documents created during that period would be immediately published after change even if versioning will be enabled again. In order to make local changes to go to public version and press `Publish` after that versioning will start work on that document.

6.1.2 Text Document Management

In this article:

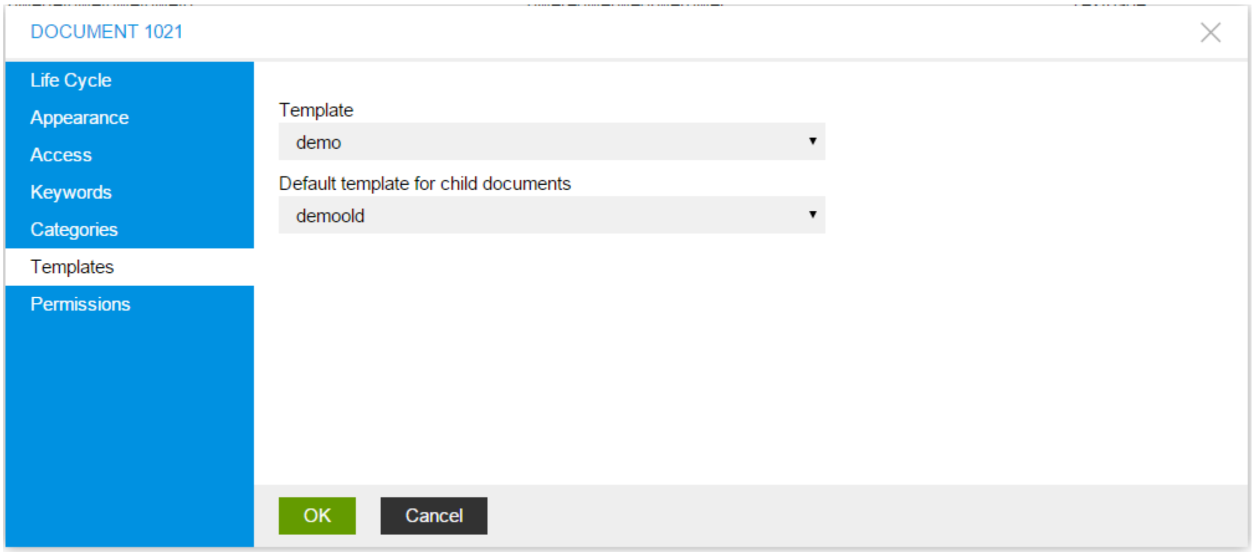
Introduction

This article describes basic text document configuration.

Edit Templates

ImCMS provide base document managing, that help change document status. There are 3 types of life cycle exists:

- In Process - it is mean that document has just been crated and it is preparing now.
- Approved - this status says that document is ready to use.
- Disapproved - document is disabled, and cannot be accessed.

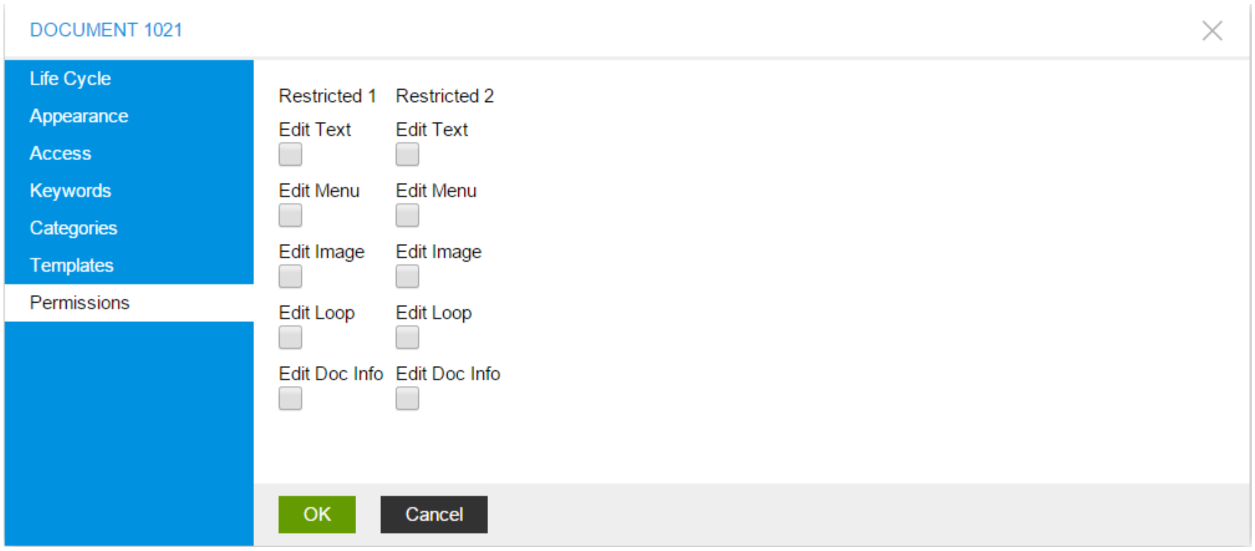


Template
demo

Default template for child documents
demoold

Edit Permissions

This section provide access to configuring custom permission sets those mentioned in *Base* section.



6.1.3 File Document Management

In this article:

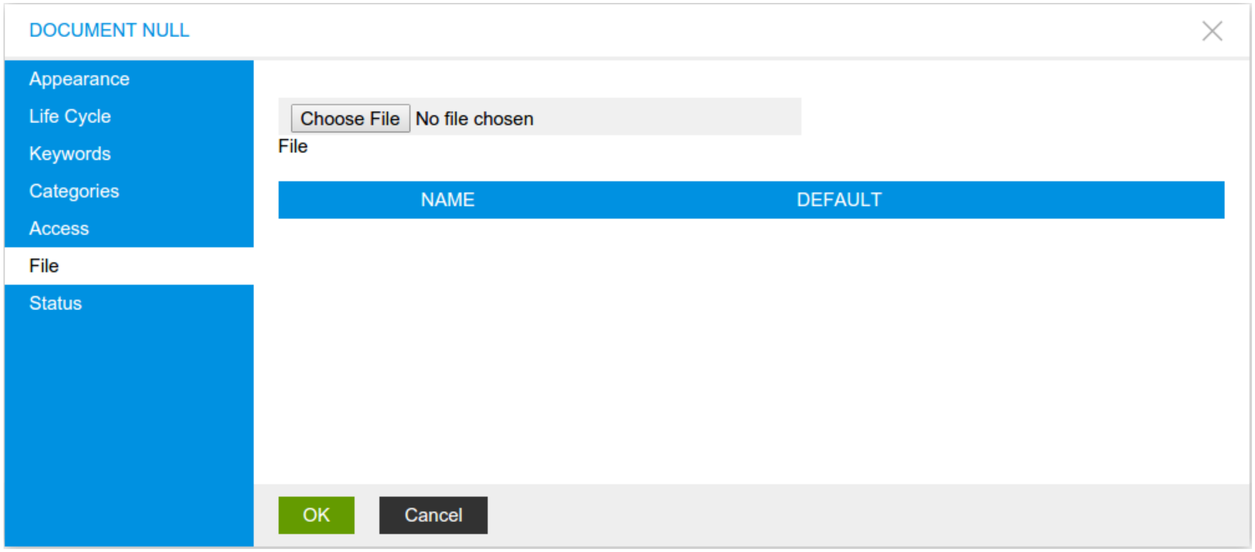
- *Introduction*
- *Edit File*

Introduction

This article describes basic file document configuration.

Edit File

ImCMS provide own section in base editor for file document. It provides access to loading file.



6.1.4 Url Document Management

In this article:

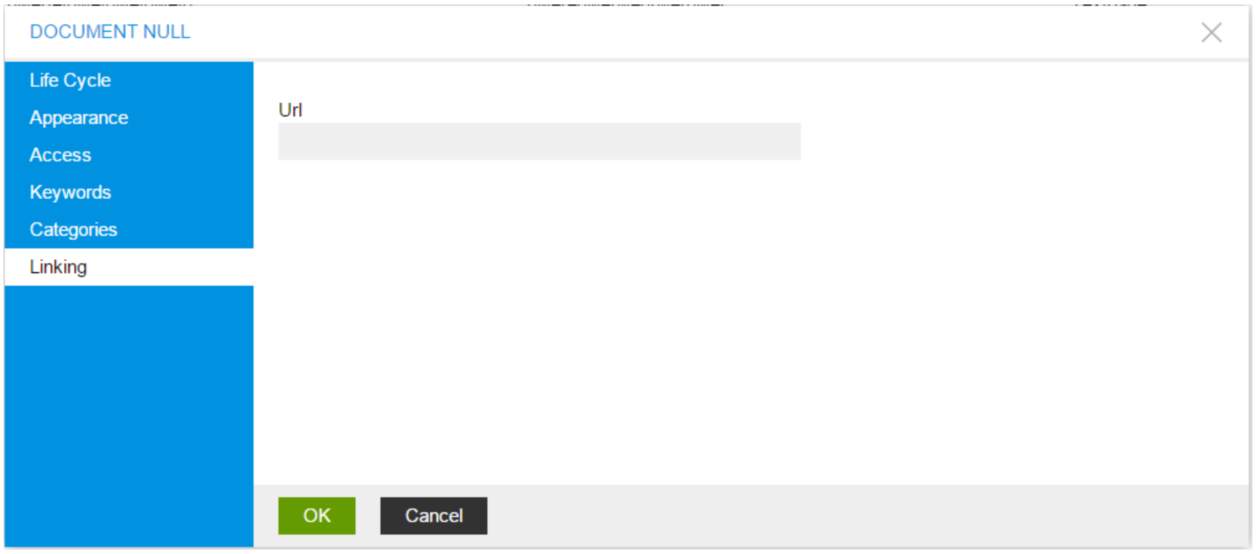
- *Introduction*
- *Edit Linking*

Introduction

This article describes basic url document configuration.

Edit Linking

ImCMS provide own section in base editor for file document. It provides access to edit document's link.



6.2 Admin Panel

In this article:

- *Introduction*
- *Panel Features*

6.2.1 Introduction

ImCMS 6 provide new modern Admin Panel with several features.

6.2.2 Panel Features

- 📖 ReadOnly Mode - Enable read only mode and hide any on-place editors.

- Edit Mode - enable edit mode - all available in template editor will show on page.
- PageInfo - show current document information and give ability for edit it.
- Document Editor - list all presented documents in the system.
- Admin Manager - open backend admin manager.
- Logout - logout current user from ImCMS system.

6.3 Text Management

In this article:

- *Introduction*
- *Open Editor*
- *Editor Features*

6.3.1 Introduction

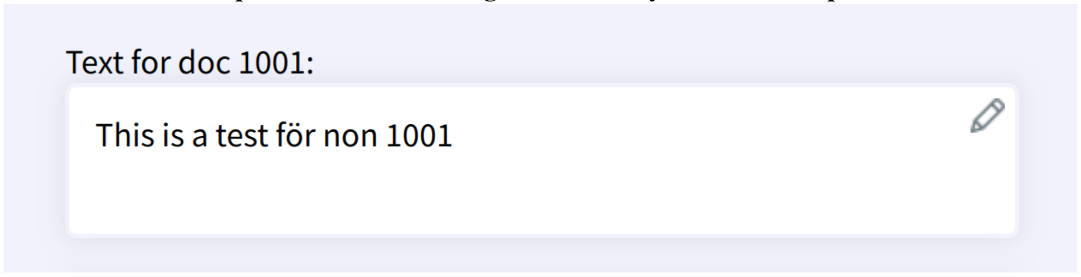
ImCMS 6 provides text and html editing feature - on-place text-editor, that can help easily change text on page.

6.3.2 Open Editor

In ImCMS, there are three modes of text editor: plain text, html and what-you-see-is-what-you-get editor.

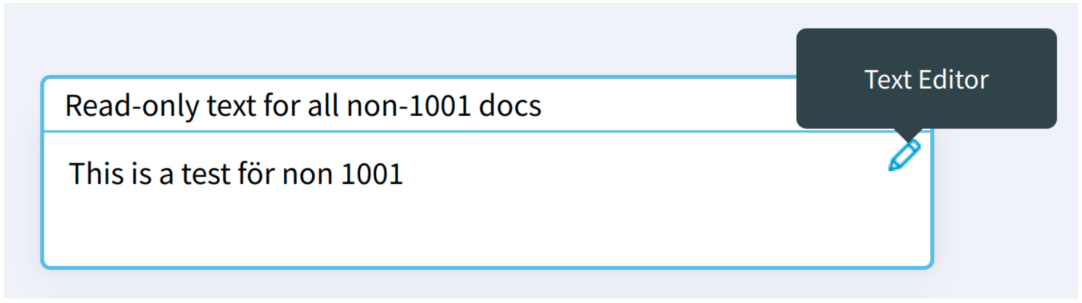
There are several ways to open **Text Editor** :

1. Over text editor's *label*:
   - **Go Edit Mode on admin panel. On each text tag in this mode you will see the pencil icon:**



Text for doc 1001:

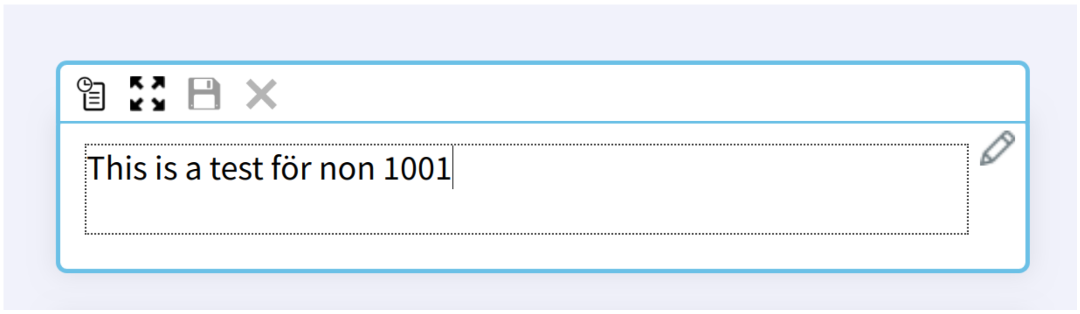
This is a test för non 1001



Read-only text for all non-1001 docs

This is a test för non 1001

   - **Click on it:**


This is a test för non 1001

2. Over *direct editing* feature - go to /api/admin/text?meta-id={id-of-doc-here}&index={index-number-of-text}, example: http://imcms.dev.imcode.com/api/admin/text?meta-id=1001&index=1
3. Currently, instead *no*, we have to use *index*

6.3.3 Editor Features

ImCMS text editor is TinyMCE v4. All information about this text editor presented on official site.

But ImCMS customize a bit this great editor and provides own features:

- `Image Browser` - this feature gives access to default ImCMS Image Editor. You can add and edit images in text
- `Text History` - you can review the history of all changes for current text
- W3 `W3C Validation` - validates current text/html
- T `Switch to Plain Text Editor Mode` - you can switch to this mode for plain text edition, all TinyMCE features will be disabled
- `</>` `Switch to HTML Edit Mode` - you can switch to this mode for HTML code edition, all TinyMCE features will be disabled
- `Switch to TinyMCE Editor Mode` - you can switch from text/html mode back to TinyMCE editor
- `HTML Content Filtering Policy` - approach to filter HTML content. It has three options:
  - Restricted:
    - `head`, `script`, `embed`, `style` : tags + content between tags are removed
    - `html`, `body`, `doctype` : only tags are removed
    - `class`, `style` and unknown attributes are removed. Known attributes are `src` `href` `rel` `alt` `align` `width` `height` `border` `cellspacing` `cellpadding` `target` `title` etc. are kept
  - Relaxed:
    - `head`, `script`, `embed`, `style` : tags + content between tags are removed
    - `html`, `body`, `doctype` : only tags are removed
    - `class`, `style` and unknown attributes are kept
  - Everything Is Allowed

6.4 Image Management

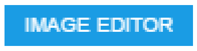
In this article:

6.4.1 Introduction

ImCMS 6 provide feature - on-place image-editor, that can help easily change image on page.

6.4.2 Open Editor

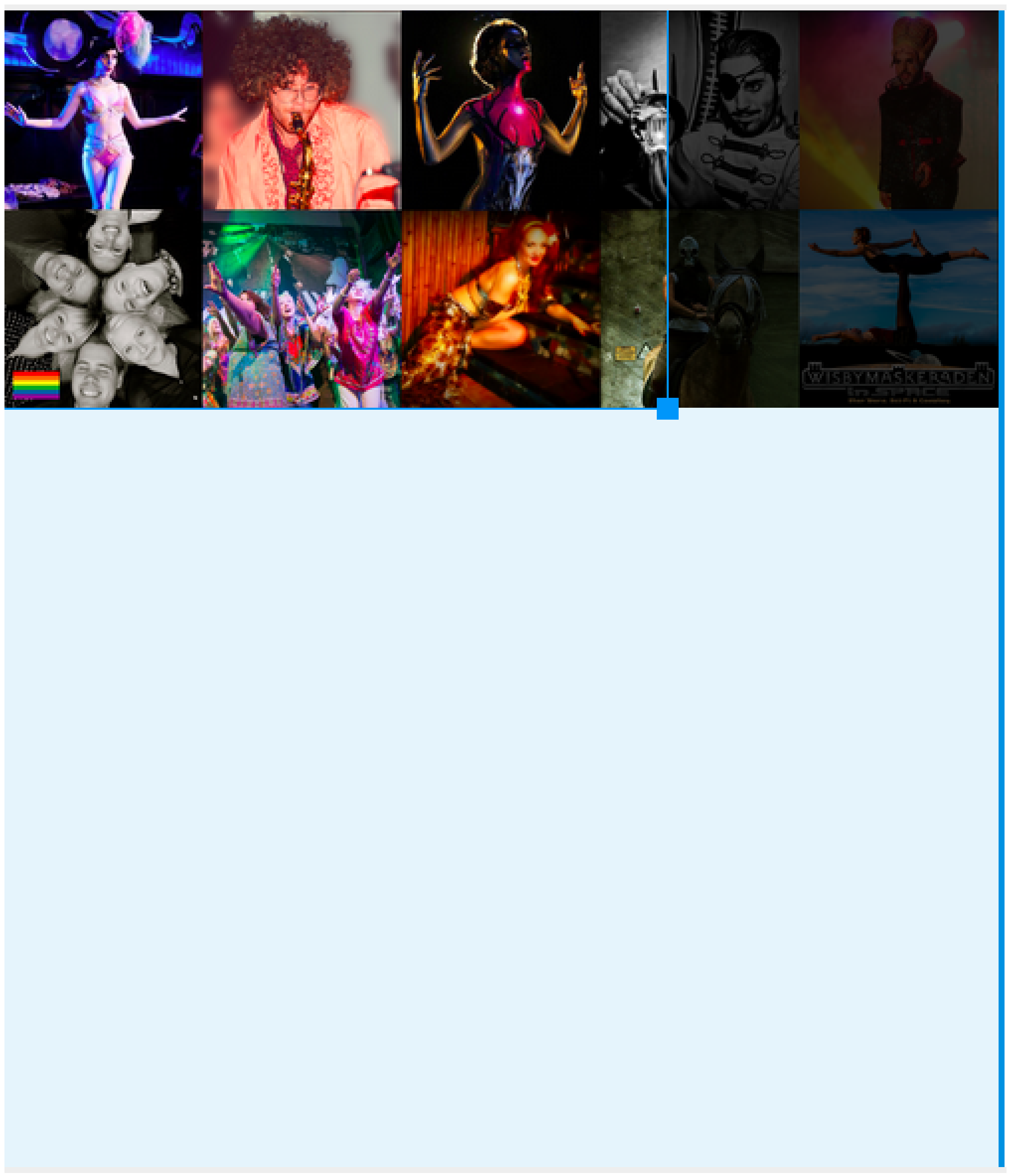
- First enable **Edit Mode**.
- To open image editor Find on the left side of the page blue label with text **Image Editor** (as shown below) and click on it to open the editor. When label hovered - the linked image is highlighted.



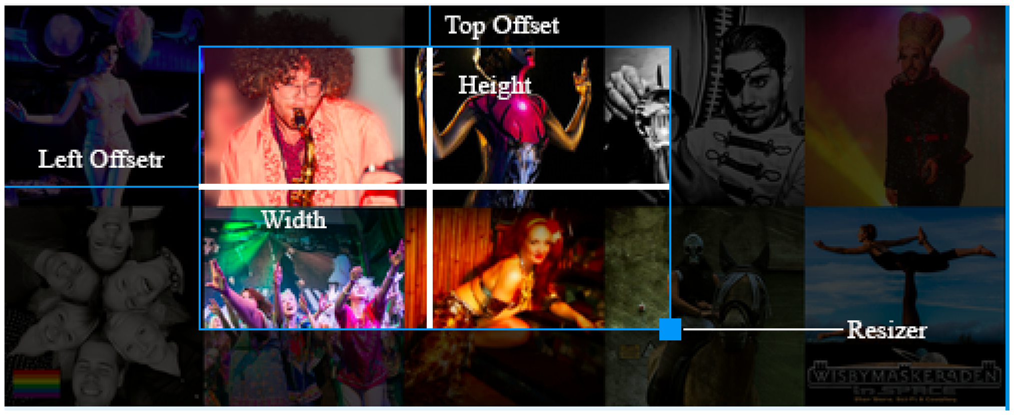
6.4.3 Using Editor

There are several base part are in the image editor:

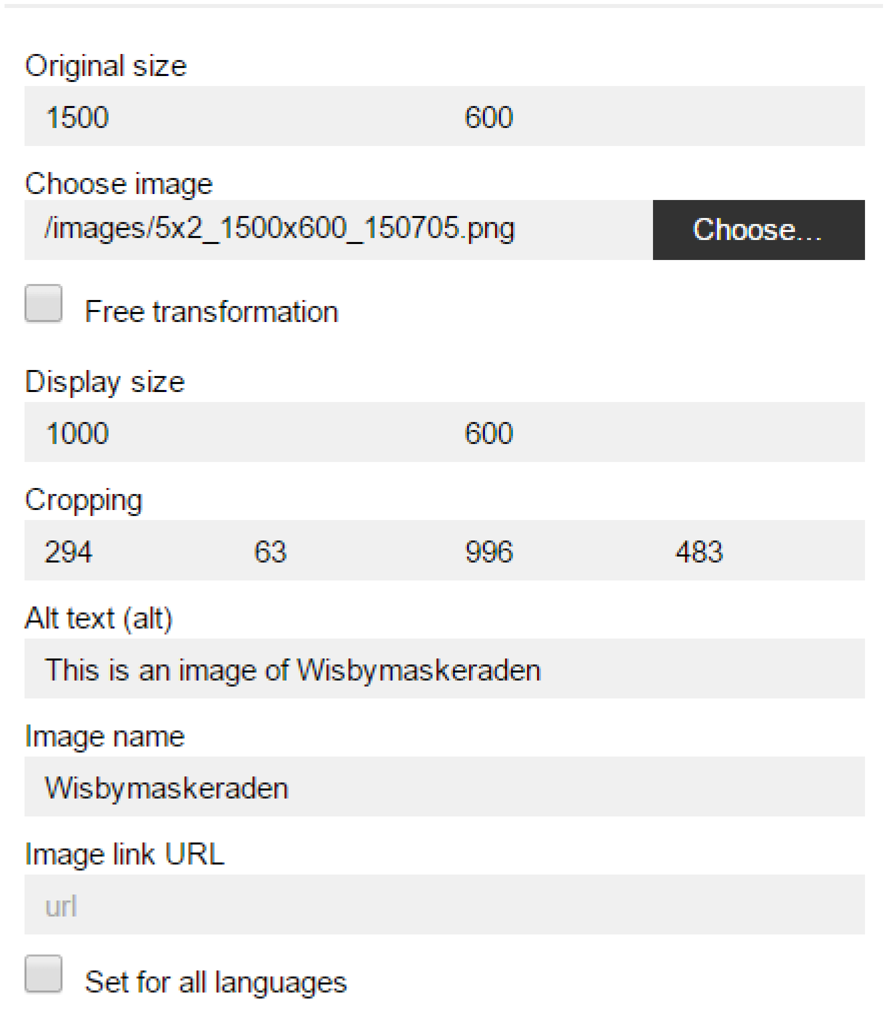
- `Image Viewer` - it include scaled or full size image with image crapper over it.



- `Image Cropper` - part of Image Viewer, the main function of it it is crop image. All features of it shown on image bellow.



- `Image Info` - numeric configuration part of editor, it provide information about image size, size of cropped area, etc.



Original size
1500 600

Choose image
/images/5x2_1500x600_150705.png

☐ Free transformation

Display size
1000 600

Cropping
294 63 996 483

Alt text (alt)
This is an image of Wisbymaskeraden

Image name
Wisbymaskeraden

Image link URL
url

☐ Set for all languages

- `Image Chooser` - it is a part of the Image Info and presented as button that open File-Content Manager.

6.5 Loop Management

In this article:

6.5.1 Introduction

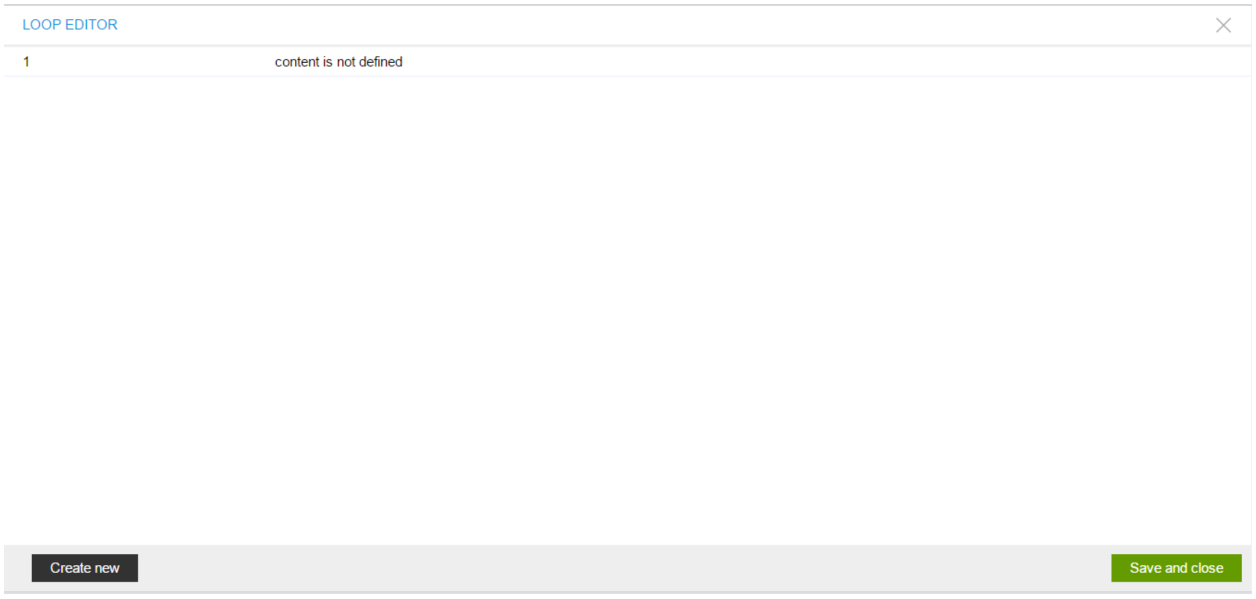
ImCMS 6 provide feature - on-place loop-editor, that can help easily mange loop content on page.

6.5.2 Open Editor

- First enable **Edit Mode**.
- To open image editor Find on the left side of the page blue label with text **Loop Editor** (as shown below) and click on it to open the editor. When label hovered - the linked loop area is highlighted.

6.5.3 Using Editor

Since *Loop* it is just a cycle's realization all managing reduces to adding or removing content iteration.4



1 content is not defined

6.6 Menu Management

In this article:

- *Introduction*
- *Add Menu Item*
- *Order Menu Items*

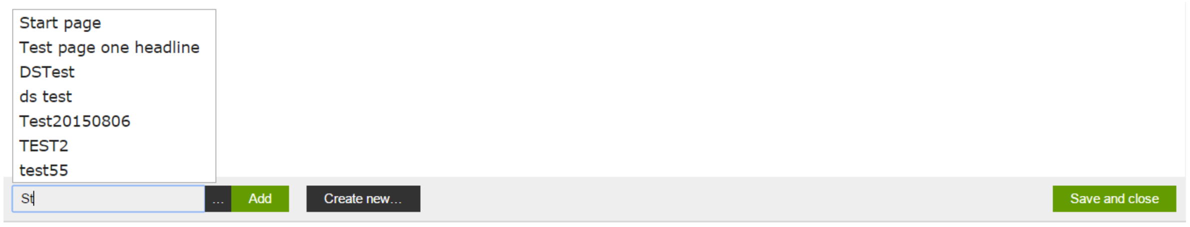
6.6.1 Introduction

This section describes how manage menu - add menu item, sorting and ordering existing menu items, etc.

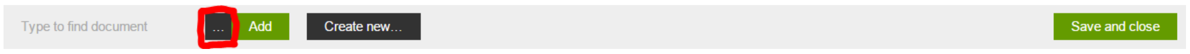
6.6.2 Add Menu Item

ImCMS Menu Editor provide several way for adding document to menu:

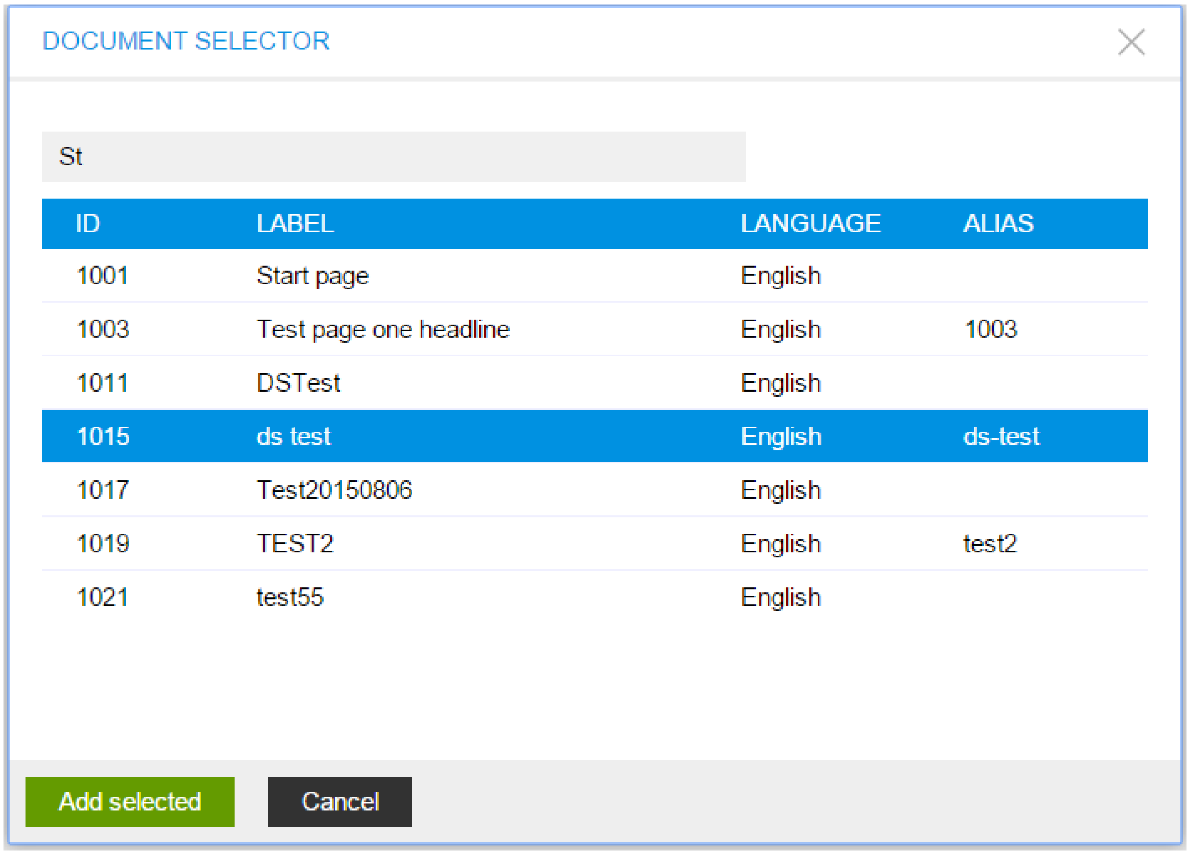
1. From search box that located in the left-bottom corner of menu-editor. When required document had been found click on the green button with text *Add*.



2. From search dialog. To open search dialog click on marked button (figure below).



In opened window type document name, and from presented result select required document and then click *Add selected* button.

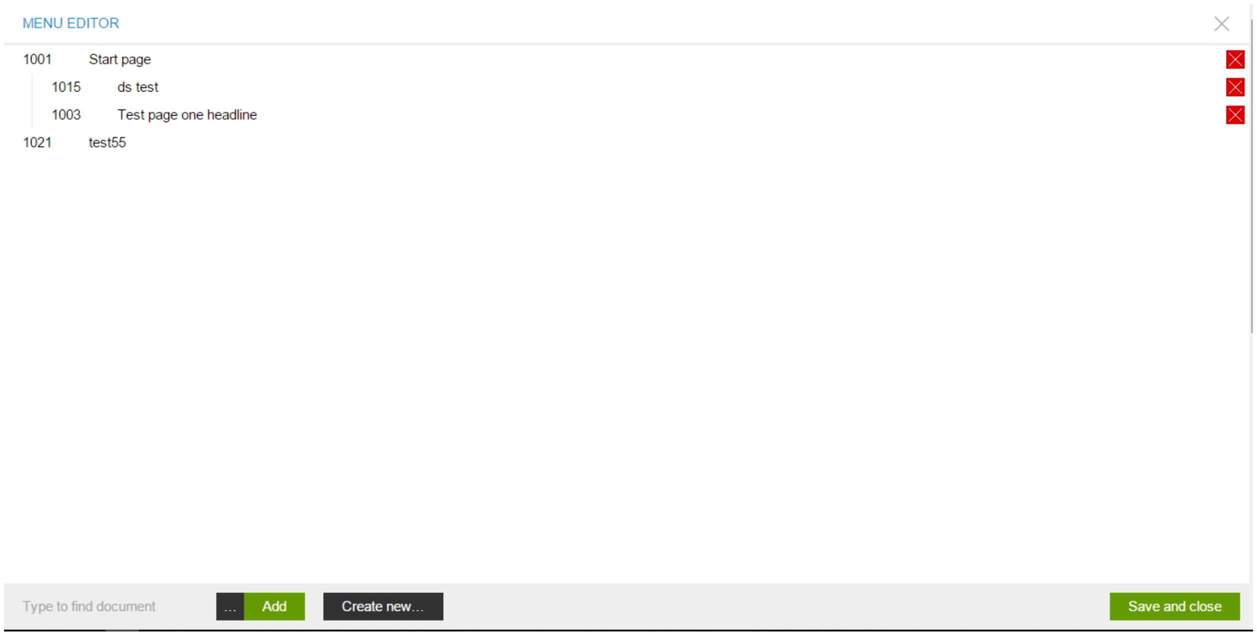


ID	LABEL	LANGUAGE	ALIAS
1001	Start page	English	
1003	Test page one headline	English	1003
1011	DSTest	English	
1015	ds test	English	ds-test
1017	Test20150806	English	
1019	TEST2	English	test2
1021	test55	English	

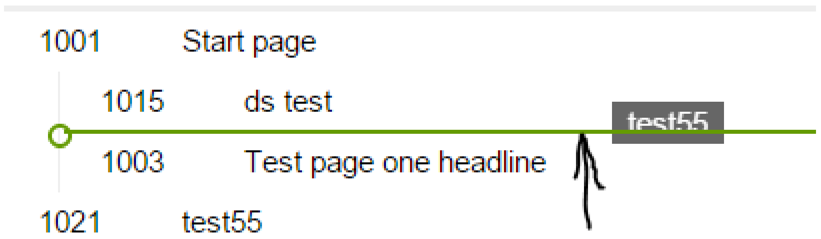
3. From new document. To create new document from menu editor consider *Base Document Management* section.

6.6.3 Order Menu Items

ImCMS menu has tree-structure. Each item can include several sub-item and can be included to another parent item.



ImCMS provide easy way to order menu-item by drag-n-dropping. To order items click on required one and drug and drop it into/above/below another one.



6.7 Category Management

In this article:

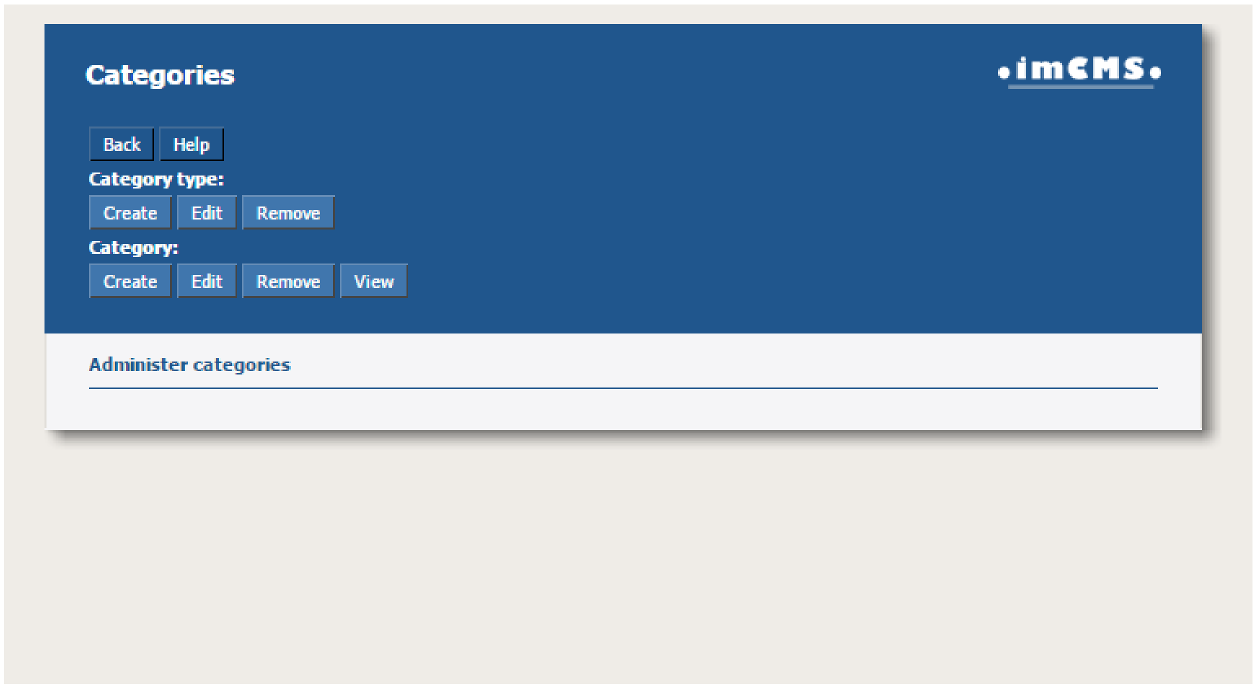
- *Introduction*
- *Create Category Type and Category*

6.7.1 Introduction

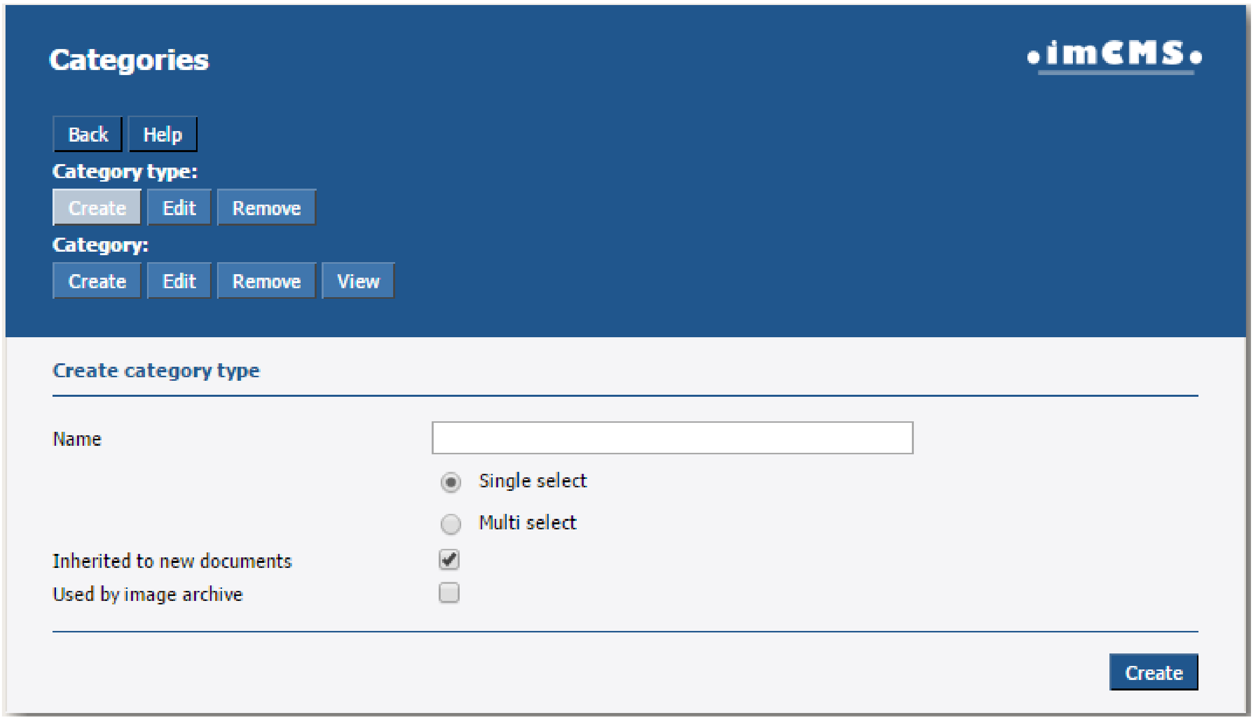
To classify elements in the system ImCMS provide well known things that called categories. The main concept is that the base type of categories is **Category Type**. It can include several categories and applied for all elements in the system that support classifying (such as document, images, etc). Each category type can be simple or complex. Simple category type provide only one category selecting, complex provide multiply category selecting.

6.7.2 Create Category Type and Category

Category editor located on `/servlet/AdminCategories`



To create new category first create new category type.



Name	
	● Single select
	○ Multi select
Inherited to new documents	✓
Used by image archive	☐

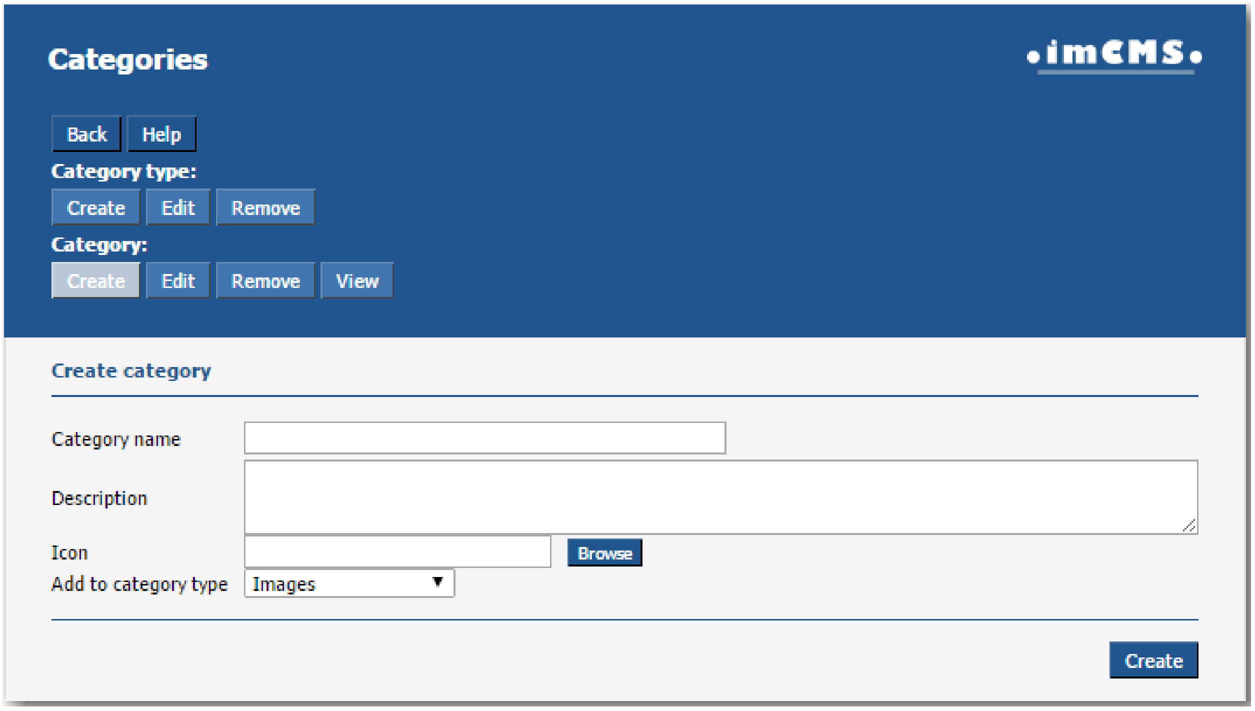
`Name` - future category type name;

`Single select` - only one category can be selected from this category type;

`Multi select` - provide multiple category selecting;

`Inherited to new documents` - enable inheritance for new documents, that is mean that all new documents, create from prototype document, inherited already chosen category for this category type;

Now, create category as show on image below



Category name	
Description	
Icon	
Add to category type	Images

`Category name` - name of created category;

`Description` - short description of created category;

`Icon` - brows image for category if needed;

`Add to category type` - parent category type;

6.8 Role Management

In this article:

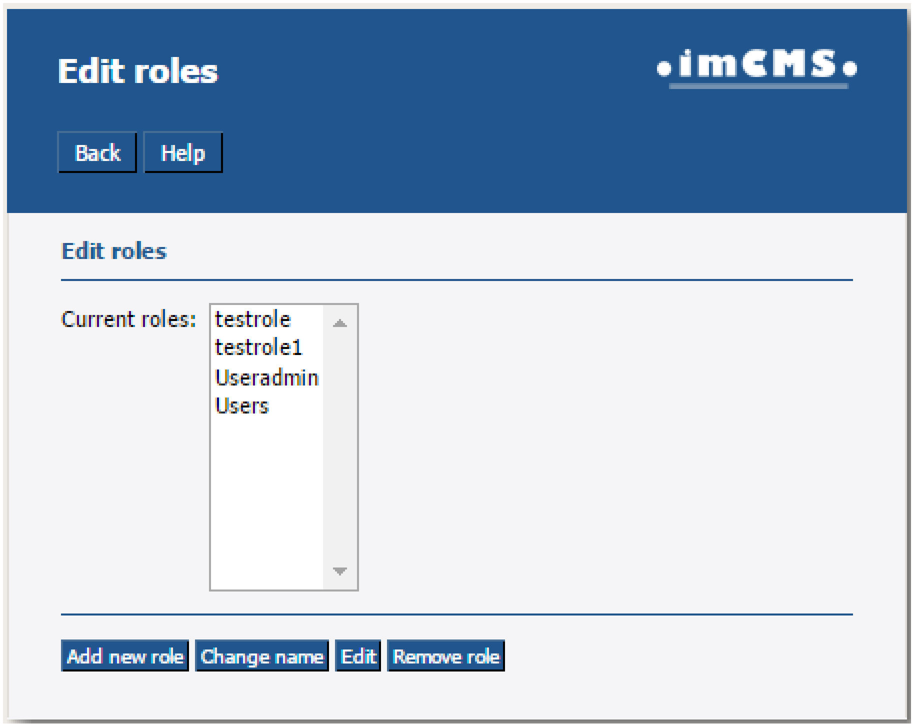
- *Introduction*
- *Create Role*

6.8.1 Introduction

To classify users in the system ImCMS provide well known things that called **Role**. Also roles coupled with permissions and give each group of users that have special role some permission set.

6.8.2 Create Role

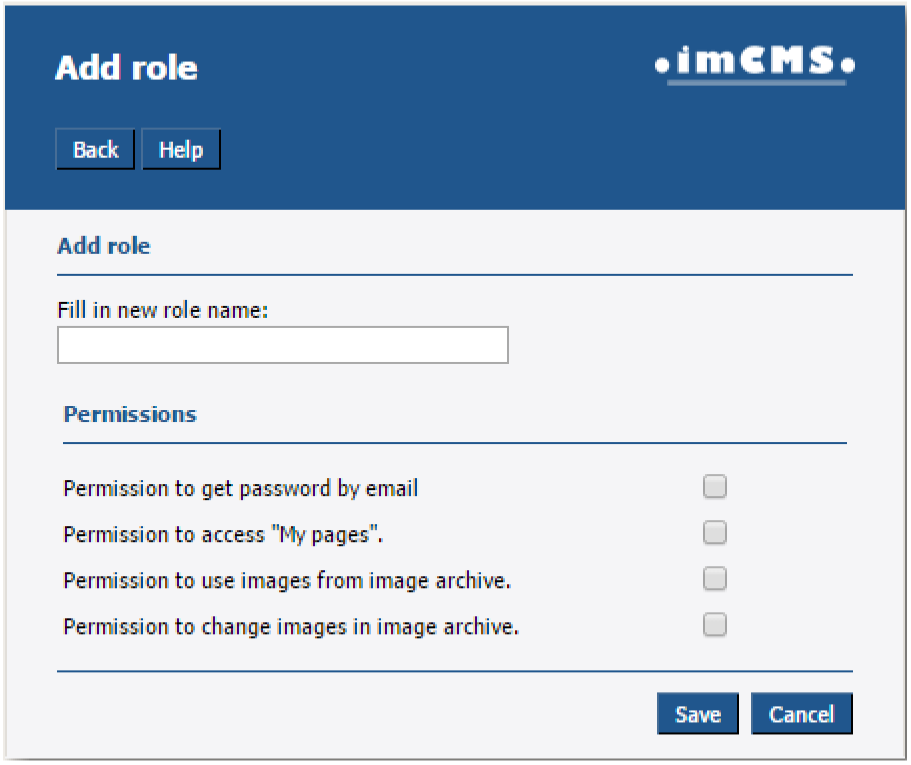
Role editor located on address `<domain-name>/servlet/AdminRoles` and looks like on figure below.



Edit roles

Current roles: testrole, testrole1, Useradmin, Users

To create new role click on button `Add new role`, then, in opened window fill name of new role and choose basic permissions.



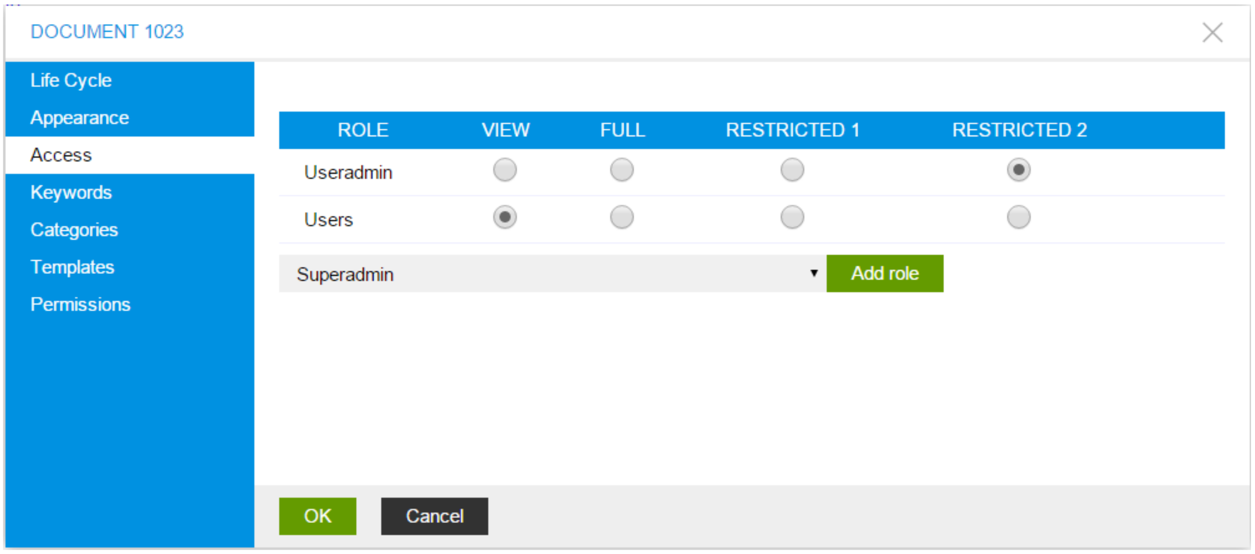
Add role

Fill in new role name:

Permissions

Permission to get password by email	☐
Permission to access "My pages".	☐
Permission to use images from image archive.	☐
Permission to change images in image archive.	☐

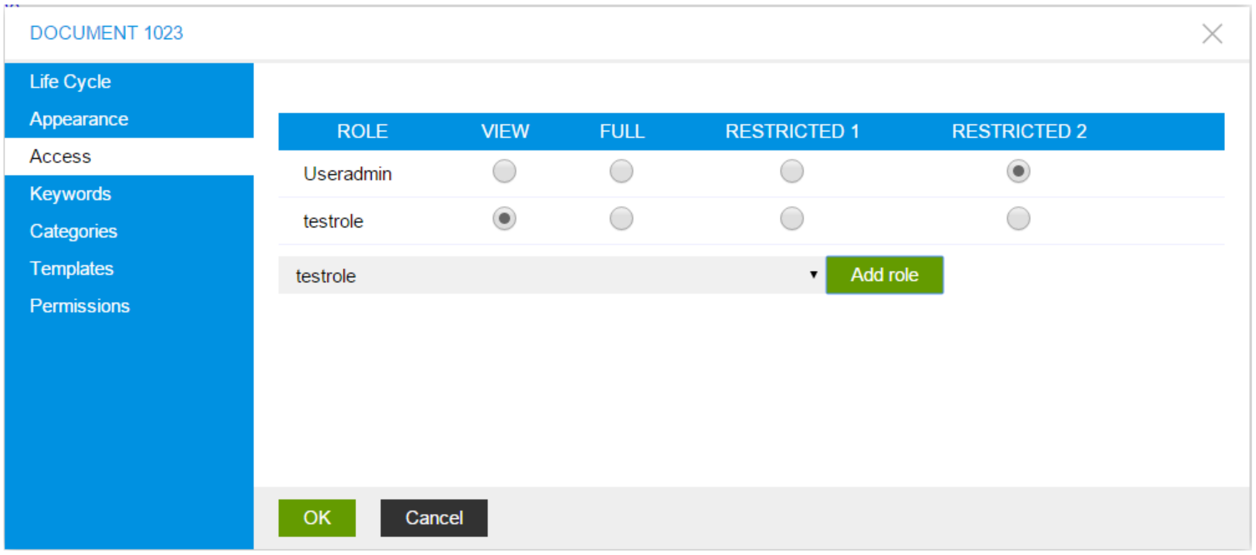
After the role has been created open PageInfo dialog and open *Access* section as has shown below.



ROLE	VIEW	FULL	RESTRICTED 1	RESTRICTED 2
Useradmin	○	○	○	●
Users	●	○	○	○

Superadmin

Remove role **Users** if it has been presented and add created role from the list.



ROLE	VIEW	FULL	RESTRICTED 1	RESTRICTED 2
Useradmin	○	○	○	●
testrole	●	○	○	○

testrole

CHAPTER 7

Advanced Configuration

7.1 JDBC

`JdbcDriver` - this option describes the java class that provide connection to a database.

`JdbcUrl` - in other words - connection string. It specified by the database type.

`User` - login user name to the database.

`Password` - login password to the database.

`MaxConnectionCount` - maximum connection count to the database.

7.2 Image Processing

In this article:

- *Introduction*
- *Setup*

7.2.1 Introduction

All image processing in ImCMS powered by well known tools called ImageMagick

7.2.2 Setup

1. Download ImageMagic distribution and install it if it has not been installed yet.
2. For Windows users the path to Image Magick should be declared in base ImCMS configuration that located in the directory **"tomcat/webapps/imcms/WEB-INF/conf"**.

7.3 Apache Solr

In this article:

- *Introduction*
- *Configuration*

7.3.1 Introduction

ImCMS provides great feature that calls Solr - it is powerful searching system. Solr presents as stand alone server that can be run with or separate to main system. Solr is highly reliable, scalable and fault tolerant, providing distributed indexing, replication and load-balanced querying, automated failover and recovery, centralized configuration and more. Solr powers the search and navigation features of many of the world's largest internet sites.

7.3.2 Configuration

By default way solr has integrated with ImCMS system yet and works perfectly without any additional configuration. But ImCMS support remote Solr server and all that is needed is put url address to Solr in ImCMS configuration.

Example:

```
# Remote SOLr server URL
# Type: Http(s) URL; optional
# Unless specified imCMS uses embedded SOLr server.
SolrUrl = http://urltosolrserver.com/
```

7.4 Language

7.4.1 General actions with language

- get default language: .. code-block:: java

  DocumentLanguage defaultLang = Imcms.getServices().getDocumentLanguages().getDefault();

- get document's enabled languages: .. code-block:: java

  Imcms.getServices().getDocumentMapper().getDocument("current document's id").getMeta().getEnabledLanguages();

  This code will return the `Set<DocumentLanguage>`, which can be used for choosing the right language and setting it.

- set document's language: .. code-block:: java

  Imcms.getServices().getDocumentMapper().getDocument("current document's id").setLanguage(DocumentLanguage language);

- set user's language (not recommended): .. code-block:: java

  Imcms.getUser().getDocGetterCallback().setLanguage(DocumentLanguage language, boolean isDefaultLang);

- set only one(or more, doesn't matter) language on admin panel in `server.properties` file:

  You have to set in value `AvailableLanguages`, language which you would want to see in admin panel. Currently, necessary use 2 letters language codes (en;sv) with ';' delimiter

7.5 Properties

In this article:

- *Introduction*
- *Server properties*
- *Properties in other files*

7.5.1 Introduction

In ImCMS we have convenient service for working with `*.properties` files - `imcode.util.PropertyManager`. This class has own cache of properties files for quick access.

Server properties

While starting application, ImCMS automatically read its root path and goes to `WEB-INF/conf/server.properties` to read server properties. Only after this moment we can read server properties from cache. If you get `FileNotFoundException` or `NullPointerException` from PropertyManager, there are two explanations:

1. You point wrong path to properties file.
2. You want to read some properties from any file (and server too) before ImCMS read its root path.

So you have to check arguments (first of all). If not helps, there are two solutions:

- Wait when ImCMS reads it's root path automatically (recommended).
- Set root path manually if ImCMS do not do that yet.

ImCMS sets path from system root to `target` folder. Try to find root path from something like:

```
ServletContext servletContext = filterConfig.getServletContext();
String rootPath = servletContext.getRealPath("/");
```

And then set root by next method:

```
PropertyManager.setRoot(String rootPath);
```

or if you find path by another way in `File` type:

```
PropertyManager.setRoot(File rootPath);
```

! Be sure that root path is set by you or ImCMS before read any properties !

If root path is set, properties from `server.properties` can be accessed by next methods:

- read the value of `property` from server properties: .. code-block:: java

  PropertyManager.getServerProperty(String property);

- returns server properties in `Properties` type for next use: .. code-block:: java

PropertyManager.getServerProperties();

Properties in other files

! Be sure that root path is set by you or ImCMS before read any properties !

If you need to read some properties from another files, you may use next methods:

- read the value of `property` from properties file by specified path: .. code-block:: java

  PropertyManager.getPropertyFrom(String path, String property);

- returns properties which lies by specified path in `Properties` type for next use: .. code-block:: java

  PropertyManager.getPropertiesFrom(String path);

7.6 SCSS

If you want ImCMS to compile your *.scss files, do the following:

- Create new file named *client.config.js* with following content in this directory: *src/main/webapp*

```
/**
 * Client configuration used by SCSS compiler. You have to declare relative
 * compiled files destination path. And also you have to declare input entries,
 * where key is output file name and value is relative path to the source file.
 *
 * @example
 *  module.exports = {
 *      entry: {
 *          main: './css/test.scss'
 *      },
 *      destination: 'dist-css'
 *  };
 *
 *  // Result files structure would be: './dist-css/main.css'
 */
module.exports = {
    entry: {
        main: './css/test.scss'
    },
    destination: 'dist-css'
};
```

- Create some .scss file for test purposes, e.g. test.scss under css directory
- Configuration have two parts (notice the comment inside *client.config.js* file):
  - `destination` is the relative path to where all compiled css should be
  - `entry` is key-value pairs, where key is the name of future file, and the value is the path to file that had to be compiled. All it's dependencies (@import's) will be bundled together
- Configure it with paths and files you want to test
- Add this code to maven:

```
<plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>exec-maven-plugin</artifactId>
    <version>1.6.0</version>
    <executions>
        <execution>
            <id>install webpack</id>
            <phase>package</phase>
            <goals>
                <goal>exec</goal>
            </goals>
            <configuration>
                <executable>npm</executable>
                <workingDirectory>${project.build.directory}/${project.build.
↪finalName}</workingDirectory>
                <arguments>
                    <argument>install</argument>
                </arguments>
            </configuration>
        </execution>
        <execution>
            <id>build scss</id>
            <phase>package</phase>
            <goals>
                <goal>exec</goal>
            </goals>
            <configuration>
                <executable>npm</executable>
                <workingDirectory>${project.build.directory}/${project.build.
↪finalName}</workingDirectory>
                <arguments>
                    <argument>run</argument>
                    <argument>build:scss</argument>
                </arguments>
            </configuration>
        </execution>
    </executions>
</plugin>
```

That's all you need, on next `package` phase in maven scss will be compiled into css, check in maven destination directory. If something vent wrong, check here, I've managed it to work.

Also latest nodeJS+NPM is required to be installed on a machine where you want to use it!

CHAPTER 8

Image Archive

8.1 Overview

In this article:

8.1.1 Introduction

For advanced working with images in imCMS, the imCode represents addon Image Archive (IA). IA was created to make work convenient with next things:

- Images are stored in one general web-service
- Users can view existing images in this web-service, change image's info, delete or upload new
- User has access to his own images
- Use images from IA in imCMS (of course)

8.1.2 IA Configuration Steps

There are two steps to get started use Image Archive:

- *Configure IA server*
- *Configure IA client*

8.2 Configure IA server

In this article:

8.2.1 Introduction

Image Archive is divided into two parts: server and client. Both of them needs to be configured. Lets see how to configure the server side.

To set up Image Archive server we have application container:

https://svn.imcode.com/imcode/customers/imagearchive/trunk

Set it up with next properties:

8.2.2 Database Configuration

- For SQL Server: .. code-block:: properties

  jdbc-driver = net.sourceforge.jtds.jdbc.Driver jdbc-url = jdbc:jtds:sqlserver://localhost:1433/;AppName=imCMS;DatabaseName=imcms hibernate-dialect = com.imcode.imcms.addon.imagearchive.util.SQLServerDialect

- For MySQL: .. code-block:: properties

  jdbc-driver = com.mysql.jdbc.Driver jdbc-url = jdbc:mysql://localhost:3306/iarch_new?characterEncoding=utf8 hibernate-dialect = org.hibernate.dialect.MySQL5InnoDBDialect

- For both: .. code-block:: properties

  jdbc-username = jdbc-password =

8.2.3 Hibernate Configuration

Automatically validates or exports schema DDL to the database when the `SessionFactory` is created.

```
hibernate-hbm2ddl-auto =
```

Possible values:

- **validate**: validate that the schema matches, make no changes to the schema of the database, you probably want this for production
- **update**: update the schema to reflect the entities being persisted
- **create**: creates the schema necessary for your entities, destroying any previous data
- **create-drop**: create the schema as in create above, but also drop the schema at the end of the session. This is great in early development or for testing.

8.2.4 Image Archive Own Configuration

URL to imCMS application that makes use of this Image Archive, as seen by the clients browser, in form:

<host> [":" <port>] "/" <context-path>

For example: `test.com/imcms` or `http://localhost:8080/skurup`

```
imcms-root-url =
```

Path where all the images that are uploaded to Image Archive will be stored, can be relative or absolute.

For example: `/var/image_archive`

Warning: Be sure that user have rights to change folder content.

```
storage-path =
```

Path where temporary images that are being processed are stored.

For example: `/tmp` or `C:/tmp`

Warning: Be sure that user have rights to change folder content.

```
temp-path =
```

ImageMagick is a software suite for creating, editing and composing images. It can be downloaded from http://www.imagemagick.org. This path should lead to where ImageMagick is installed, and is required only on windows. For linux leave it empty.

For example: `C:/program files/imagemagick-6.4.9-q16`

```
image-magick-path =
```

Maximum size of an uploaded image in bytes. By default 250 MB.

```
max-image-upload-size = 262144000
```

Maximum size of an uploaded ZIP archive in bytes. By default 250 MB.

```
max-zip-upload-size = 262144000
```

URL path to login, in imCMS, relative to context path.

```
imcms-login-url-path = login
```

Name for a directory within libraries folder, that will contain each users personal library. This directory will be automatically created.

```
imcms-users-library-folder = users
```

Images from Image Archive that are being used by imCMS will be stored here, can be relative or absolute.

```
imcms-images-path =
```

Next two properties may be empty:

Path to libraries, can be relative or absolute. Each folder in this directory will become a library in Image Archive - these folders can be created using imCMS file manager. Each library can contain one or more raw images which can be activated in Image Archive.

```
imcms-libraries-path =
```

Absolute or relative paths separated by “;”. Each path will become a library in Image archive, can be used for gaining access to old Image Archive.

```
imcms-old-library-paths =
```

8.3 Configure IA client

In this article:

- *Introduction*
- *Database and Hibernate Configuration*
- *Image Archive Own Configuration*

8.3.1 Introduction

Image Archive is divided into two parts: server and client. Both of them needs to be configured.

Client side includes into existing project by maven dependency:

```
<dependency>
    <groupId>com.imcode.imcms.addon.imagearchive</groupId>
    <artifactId>client</artifactId>
    <version>1.0-SNAPSHOT</version>
    <type>war</type>
</dependency>
<dependency>
    <groupId>com.imcode.imcms.addon.imagearchive</groupId>
    <artifactId>client</artifactId>
    <version>1.0-SNAPSHOT</version>
    <type>jar</type>
    <classifier>classes</classifier>
</dependency>
```

With this dependencies client’s .war and required .jar are ready to use after some configuration. Most of next properties already available in project, but for full info lets see all required properties for client.

8.3.2 Database and Hibernate Configuration

- For SQL Server: .. code-block:: properties

  jdbc-driver = net.sourceforge.jtds.jdbc.Driver jdbc-url = jdbc:jtds:sqlserver://localhost:1433/;AppName=imCMS;DatabaseName=imcms hibernate-dialect = com.imcode.imcms.addon.imagearchive.util.SQLServerDialect

- For MySQL: .. code-block:: properties

jdbc-driver = com.mysql.jdbc.Driver jdbc-url = jdbc:mysql://localhost:3306/iarch_new?characterEncoding=utf8 hibernate-dialect = org.hibernate.dialect.MySQL5InnoDBDialect

- For both: .. code-block:: properties

  jdbc-username = jdbc-password =

- Hibernate Configuration

  Automatically validates or exports schema DDL to the database when the `SessionFactory` is created.

  ```
  hibernate-hbm2ddl-auto =
  ```

  Possible values:

  – **validate**: validate that the schema matches, make no changes to the schema of the database, you probably want this for production

  – **update**: update the schema to reflect the entities being persisted

  – **create**: creates the schema necessary for your entities, destroying any previous data

  – **create-drop**: create the schema as in create above, but also drop the schema at the end of the session. This is great in early development or for testing.

8.3.3 Image Archive Own Configuration

URL to the separate image archive application, as seen by the clients browser, in form:

<host> [":" <port>] "/" <context-path> "/archive"

For example: `localhost:8080/client/archive` or `http://www.skurup.se/archive`

```
ImageArchiveUrl =
```

URL to Image Archive server.

For example: `http://skurup-imagearchive.dev.imcode.com` or `http://localhost:8081`

```
ia-server-url =
```

IDs of the roles that are allowed to see the "Choose from image archive" button in image edit page, delimited by ",". If not specified, everyone is allowed.

```
ImageArchiveAllowedRoleIds = 2
```

Path where all the images that are uploaded to Image Archive will be stored, can be relative or absolute.

For example: `/var/image_archive`

Warning: Be sure that user have rights to change folder content.

```
storage-path =
```

Path where temporary images that are being processed are stored.

For example: `/tmp` or `C:/tmp`

Warning: Be sure that user have rights to change folder content.

```
temp-path =
```

Path to images, in file system and URL.

```
ImageArchiveImagePath = archivedimages/
ImageArchiveImageUrl = /archivedimages/
```

ImageMagick is a software suite for creating, editing and composing images. It can be downloaded from http://www.imagemagick.org. This path should lead to where ImageMagick is installed, and is required only on windows. For linux leave it empty.

For example: `C:/program files/imagemagick-6.4.9-q16`

```
image-magick-path =
```

Maximum size of an uploaded image in bytes. By default 250 MB.

```
max-image-upload-size = 262144000
```

Maximum size of an uploaded ZIP archive in bytes. By default 250 MB.

```
max-zip-upload-size = 262144000
```

URL path to login, in imCMS, relative to context path.

```
imcms-login-url-path = login
```

Name for a directory within libraries folder, that will contain each users personal library. This directory will be automatically created.

```
imcms-users-library-folder = users
```

Next two properties may be empty:

Path to libraries, can be relative or absolute. Each folder in this directory will become a library in Image Archive - these folders can be created using imCMS file manager. Each library can contain one or more raw images which can be activated in Image Archive.

```
imcms-libraries-path =
```

Absolute or relative paths separated by “;”. Each path will become a library in Image archive, can be used for gaining access to old Image Archive.

```
imcms-old-library-paths =
```

CHAPTER 9

BankId

9.1 BankId Application Settings

To use BankId logging you need to add next authentication properties to your application properties:

```
authentication-configuration = cgi,loginPassword
```

Next property sets the authentication method URL:

```
cgi-metadata-url = *url*
```

where **url** is different:

- for testing: https://m00-mg-local.testidp.funktionstjanster.se/samlv2/idp/metadata/0/0
- for production: https://m00-mg-local.idp.funktionstjanster.se/samlv2/idp/metadata/0/0

Than need define role name for set of created configurations:

```
cgi-user-role-name = *roleName*
```

You can define and use **roleName** whatever you want, like BankID, CGIuser etc.

And the last one - server name, like **http://localhost:8080** or **https://www.somesite.se** :

```
server-name = *actual server name*
```

9.2 Generate Test BankId

In this guide tells about test bankId for testing environment

Attention: You can generate test bankId only with exist BankId account.

1. Install BankId Application. Instruction for different devices you can find here.
2. Login with your real/legal bankId on https://demo.bankid.com .
3. Go into "Issue BankID for Test".
4. Generate person: https://fejk.se/ .
5. Fill out form "Issue Mobile BankID" and submit.
6. Send activation code.

Note: With the one BankId you can login and generate more, so you should have a test bankId on your computer to be able to generate new bankId's.

See also:

If you have some problems with generation test bankId, or other questions, please visit this page.

9.3 Login

To login with test bankId, follow next steps:

1. Start BankID Security App and select New BankID You can also select New BankID under Settings in the security app.
2. Enter personal ID number and Activation Code Enter your ID number and the activation code below, in BankID Security App.

Attention: Your Activation Code is valid for 10 minutes. If you leave this page the activation is aborted.

3. Choose a Security Code Choose a personal Security Code with at least 6 digits in BankID Security App. You will use this code whenever you use Mobile BankID.