[EN] 05. Changing the Configuration of the Reader Application

Changing the Configuration of the Reader Application

Configuring the Reader Application

In this chapter, there are instructions on the configuration of some mechanisms related to the reader application.

Configuring the CAS Service

The dLibra system makes it possible to use a central login system created with the use of the CAS server and the LDAP service. CAS is available as another identity provider in file **dlibra-webapp/WEB-INF/conf/user-providers.xml**. The following fragment should be uncommented:

```
<pl.psnc.dlibra.web.comp.user.CasUserInformationProvider>
   <loginPage>http://dlibra.psnc.pl/cas/login?service=${prevpage}</loginPage>
   <logoutPage>${homepage}main?action=LogoutAction</logoutPage>
   <loginPagePosition>2</loginPagePosition>
   <methodNameResourceKey>uip.cas.name</methodNameResourceKey>
   <ldapConfiguration>WEB-INF/conf/ldap.properties</ldapConfiguration>
   <userAttributesDefaults>
     <property name="givenName" value="dLibra"/>
     <property name="cn" value=""/>
     <property name="mail" value=""/>
   </userAttributesDefaults>
   <emailAttributeName>mail</emailAttributeName>
 </pl.psnc.dlibra.web.comp.user.CasUserInformationProvider>
```

Next, the user should:

- change the value of the loginPage tag in such a way that it points to the login page of the used CAS server, and
- in the userAttributesDefaults tag, enter the names of the attributes which are to be used for creating dynamic property groups in the Administrator Application of the dLibra system; the value of the value attribute of every property tag determines the default value for the attribute;
- if every user has an email address specified as the value of an attribute in the used LDAP register, the name of that attribute can be entered in the **emailAttributeName** tag; then, the dLibra system will be able to use the information about the users’ email addresses;
- the name of an identity provider can be changed, by overwriting, in WEBAPP_xx.xml, the value of the text label located in methodNameResourceKey.

Next, the content of file **dlibra-webapp/WEB-INF/conf/ldap.properties** should be completed. The file contains parameters which make it possible to use the LDAP service.

Configuring News Import

The reader application makes it possible to publish various types of messages on the home page. Those news can be added with the use of the administration panel of the reader application or by means of the mechanism of RSS news import. A sample import process is discussed below.

In order to import news from an RSS available online, the user should uncomment the “periodic-task” tag containing the configuration of the NewsImportTask in the periodic.xml file

```
<periodic-task logicClass="pl.psnc.dlibra.web.comp.periodic.NewsImportTask"
         executeOnStart="yes">
 <description>Checks feed url given in configuration, and imports news from that feed.</description>
 ....
</periodic-task>
```

The configuration of that task consists of the following parameters:

- feed.url – the address of the RSS/Atom channel containing the imported news;
- add.post.link.to.blog – if the value of that parameter is “true”, then for every news, a link to the page on which it was originally published will be added;
- full – which element of an entry in the RSS channel is to be used as the full text of the news; in the case of RSS channels, it will usually be the “description” element, but Atom channels also have two tags with the content of an entry: “content” and “summary”; the admissible values are feed_description and feed_content;
- short – which element of the channel is to be imported as the summary of a news item; the admissible values are feed_description and feed_content;
- publish.in.all.languages – if the value of that parameter is “true”, the news will be imported in all the interface languages installed in the reader application;
- blog.language – the language in which the imported news is written;
- start.date – only those entries will be imported which have been published after the date (dd.MM.YYYY) entered here;

by default, that task is carried out once a day; whenever the reader application detects that a new news item has appeared in the RSS channel, it will import it within the framework of the nearest performance of that task.

Adding Bookplates

In the reader application of the dLibra system, bookplates are represented as images with a link to the selected page (library, institution, etc.) which appear on the page with the description of a publication (*publication, docmetadata*). Bookplates are added on the basis of the values of particular attributes in metadata. For example, a bookplate may be displayed depending on the value of the “Permissions” attribute – for the “Library Institution X” value, an image which identifies that institution will be displayed, together with a link to the home page of that institution, “*www.instytucjabibliotecznax.com*”.

A bookplate is added in the following manner:

1. In the **webapp/exlibris** directory, the graphics file with the bookplate of the library should be entered.
2. File **webapp/WEB-INF/conf/exNbris.xml** contains the following entries:

```
<exlibrises>
 <attribute rdfname="Rights">
  <attribute-value value="Biblioteka Uniwersytecka w Poznaniu">
   <exlibris>
    <img>bu-poznan.png</img>
    <link>http://lib.amu.edu.pl/</link>
   </exlibris>
  </attribute-value>
 </attribute>
</exlibrises>
```

- The XML “*attribute*” element and value “*rdfname*” (the RDF name) indicate the publication metadata attribute the value of which will represent the bookplate; there can be many “attribute” elements.
- Next, within the framework of the “*attribute*” element, the “*attribute-value*” elements should be defined; they represent the values of the publication metadata attribute which identify bookplates. Those values are entered in the XML “value” field of the “*attribute-value*” element. In the example of the default configuration given above, the attribute value is “Biblioteka Uniwersytecka w Poznaniu” Recognizing that value in the publication metadata will cause an appropriate bookplate to be displayed on the publication metadata description page in the reader application (the “*docmetadata*” or “*publication*” page). Every “*attribute-value*” element is responsible for one attribute value, so if we want the bookplate to be recognized by means of various name variants, a separate “*attribute-value*” element should be created for every name.
- In the end, the “*exlibris*” elements are determined in the “*attribute-value*” element. An “*exlibris*” element has to subordinate elements, “img” and “link”; the first one is the name of a graphics file (added in the first step to the /exlibris folder), and the second one indicates the link to the page to which the user will be redirected when the bookplate image has been clicked.

The “*attribute*, “*attribute-value*”, and “*exlibris*” elements can be defined many times, so many bookplates can be added for every determined value of a metadata attribute,

For example:

```
<exlibrises>
 <attribute rdfname="Rights">
  <attribute-value value="Biblioteka Uniwersytecka w Poznaniu">
   <exlibris>
    <img>bu-poznan.png</img>
    <link>http://lib.amu.edu.pl/</link>
   </exlibris>
  </attribute-value>
  <attribute-value value="BUP">
   <exlibris>
    <img>bu-poznan.png</img>
    <link>http://lib.amu.edu.pl/</link>
   </exlibris>
  </attribute-value>
  <attribute-value value="Uniwersytet im. A.Mickiewicza i Politechnika Poznaska">
   <exlibris>
    <img>uam.png</img>
    <link>http://uam.edu.pl/</link>
   </exlibris>
   <exlibris>
    <img>put.png</img>
    <link>http://www.put.poznan.pl/</link>
   </exlibris>
  </attribute-value>
 </attribute>
</exlibrises>
```

In that configuration, the “Biblioteka Uniwersytecka w Poznaniu” bookplate will appear for the “**Permissions**” attribute both when it has the “Biblioteka Uniwersytecka w Poznaniu” value and when it has the “BUP” value. Moreover, if the same “**Permissions**” attribute has the “Uniwersytet A. Mickiewicza i Politechnika Poznanska” value, two bookplates (images with links) will be displayed, for the Adam Mickiewicz University and for the Poznan University of Technology.

The exlibris.xml configuration will only begin to apply after the servlet environment of the reader application (Apache Tomcat) has been restarted. Incorrect settings may cause exceptions and system log errors to appear.

Configuring the JCR Backup System

In order to change the default JCR backup creation settings in the system, the user should edit the *periodic.xml* file in the /WEB-INF directory of the reader application. The entry concerning the appropriate periodic task is looks as follows:

```
<periodic-task logicClass="pl.psnc.dlibra.web.comp.periodic.JCRBackupTask" executeOnStart="yes">
                <expression>0 0 23 * * ?</expression>
                <properties>
                        <property>
                                <name>jcr.backup.dir</name>
                                <value>[backup directory path]</value>
                                <description>Path to JCR backup destination directory. Path should end with '/'.
</description>
                        </property>
                        <property>
                                <name>copies.amount</name>
                                <value>[amount of copies]</value>
                                <description>Specifies how many different copies of backup can be stored at
once</description>
                        </property>
                </properties>
                <description>Task is responsible for backup of JCR stored data</description>
</periodic-task>
```

Two parameters can be changed in the task: jcr.backup.dir and copies.amount. By means of the first parameter, the administrator indicates the directory in which backups are to be placed. Every single backup copy is an XML file named jcr-backup_[the date of the creation of the copy].xml, for example, *jcr-backup_2010.12.25.08.00.00.xml*.

The second parameter determines the maximum number of backup copies. The periodic task is carried out periodically – by default, every 24 hours. Thus, when the maximum number of copy files has been created, each subsequent file overwrites the oldest file from the *jcr.backup.dir* directory.

The copies.amount can have the following values:

- for parameter > 0, the task creates a specific number of backup copy files;
- for parameter = 0, the task does not create any backup copy files; and
- for parameter < 0, the task creates an unlimited number of backup copy files.

A change of parameters in periodic tasks requires a restart of the container (for example, Apache Tomcat), of the reader application.

> **Uwaga**
>
> When the parameters mentioned above are filled in incorrectly, the task may function incorrectly, and exceptions can be reported in the logs.

Restoring Data from JCR Backup Copies

In order to recover data from a backup copy XML file, it should be placed in an appropriate directory. That directory is configured in the */WEB-INF/conf/jcr.properties* file of the reader application, at parameter *jcr.restore.dir*.

A backup copy is automatically restored at a restart of the container (for example, Apache Tomcat) of the reader application.

Caution! When the data from a backup copy file have been recovered, the file is automatically removed.

Configuring Thumbnails

The “context-param” parameter from the /WEB-INF/web.xml file of the reader application is responsible for configuring the thumbnails displayed in the reader application. The file looks as follows:

```
<context-param>
                <description>
                        ...
                </description>
                <param-name>thumbnails.settings</param-name>
                <param-value>{
                        "recommended": {206;232;1.0;false;image/png},
                        "coll_home_recommended": {206;232;1.0;false;image/png},
                        "collection_description": {390;400;1.0;false;image/png},
                        "edition": {242;132;1.0;false;image/png},
                        "docmetadata": {225;335;1.0;false;image/png},
                        "result_item": {206;232;1.0;false;image/png},
                        "example": {200;200;1.0;false;image/png}
                }</param-value>
</context-param>
```

The parameter values are entered between the ***<param-value></param-value>*** tags. The parameter consists of **id_miniatury:zestaw_parametrów_miniatury** pairs The parameter set consists of four parts which are separated with the semicolon (;) character, so various thumbnail profiles can be defined, for example: “recomended”, “col_home_recommended”, or “collection_description”, and we can choose a suitable size of the image. The URL below illustrates the schema of the access to a thumbnail:

```
https://{domena}/image/{typ_obiektu}/thumbnail:{id_miniatury}/{id}
```

where:

- **{domena}** is the Internet domain of our library, for example, demo.dl.psnc.pl;
- **{typ_obiektu}** is the object type for which the thumbnail will be generated (for example, edition, publication, or collection);
- **{id_miniatury}** is the ID of the thumbnail defined in thumbnails.settings; and
- **{id}** is the ID of the object (edition, publication, or collection).

In the **example** presented above, the parameter set for the example thumbnail is: 200, 200,1.0, false, and image/png. The subsequent parts mean:

- the width of the output graphics file (in the example, it is 200 pixels);
- the height of the output graphics file (in the example, it is 200 pixels);
- the degree of the compression of the output graphics file (in the example, it is 1.0);
- whether the input graphics file is to be cropped in proportion to its dimensions (in the example, the value is “false”); and
- the format of the output graphics file (in the example, the value is “image/png”).

The first two values determine the dimensions of the output graphics file of the thumbnail for the reader application regardless of the dimensions of the input graphics file entered in the Editor Application. Thus, if the graphics file in the Editor Application has been entered in dimensions: width = 500 pixels, height = 200 pixels, then the image will be appropriately scaled in the reader application, to the dimensions configured in the "*thumbnails.settings*" parameter.

That can be useful when the thumbnail files entered by editors are very detailed, and that degree of detail is not necessary when they are displayed in the reader application; moreover,when the dimensions are decreased, the memory volume needed for sending them is lowered, which may accelerate the speed of the data transfer.

The next parameter is related to the last one, and it is only taken into account when the thumbnail has the "image/jpeg" format of the output graphics file. It makes it possible to further decrease the sent file, thanks to the lossy compression of the JPEG format. The optimal compression is set with the use of a floating-point number, in the (0,1) range, for example, 0.1, 0.25, or 0.3.

The next parameter determines the way in which the input image is to be adjusted to the dimensions of the thumbnail entered in the "*thumbnails.settings*" parameter. When the flag is set to "true", the original graphics file will be decreased proportionally and cropped to the area defined by the first two parameters. When the flag is set to "false", the original graphics file will be adjusted to fit in the area defined by the "**thumbnails.settings**" thumbnail parameters.

The last parameter determines the type of the output graphics file format of the thumbnail. The two most popular formats, JPEG and PNG, are supported. When the JPEG format is chosen, the parameter of the degree of compression (described above) is also significant. The formats are identified by MIME types. For PNG, it is "imagne/png" and for JPEG – "image/jpeg".

A change of thumbnail display parameters requires a restart of the container (for example, Apache Tomcat), of the reader application.

> **Uwaga!**
>
> When incorrect parameters are entered, thumbnails may not be displayed, and errors may be reported in the logs.

Configuring the "Recommended" Component

The basic configuration of the "Recommended" component is in the */WEB-INF/components.xml* file of the reader application.

Here is a sample configuration entry of the component:

```
<component name="pl.psnc.dlibra.web.comp.pages.components.RecommendedComponent">
        <properties>
                <property>
                        <name>RecommendedId</name>
                        <value>&recommendedId;</value>
                </property>
                <property>
                        <name>RecommendedCount</name>
                        <value>20</value>
                </property>
                <property>
                        <name>reverseOrder</name>
                        <value>true</value>
                </property>
        </properties>
</component>
```

The configuration has two basic parameters:

- *RecommendedId* – indicates the ID of the collection which contains recommended publications;
- *RecommendedCount* – indicates how many publications are to be randomly selected for display in the component; and
- *reverseOrder* – reverses the default order of display

When it has its default values, the first parameter puts in the value of the recommendedId entity which is declared at the beginning of the document, in line:

```
<!ENTITY recommendedId "3">
```

The default value of the entity is "3", and it points to the ID of the collection which is automatically created during installation.

From that collection, the publications which are to be displayed in the component are randomly selected, and only those of the randomly selected publications are displayed to which a thumbnail is assigned. When the collection does not contain any publications which fulfill the abovementioned conditions, the component will not be displayed.

> The value of the recommendedId entity is also used in the configuration of the *CollectionsStructureComponent* component in the HiddenCollectionsIds parameter in which the collections which are **invisible** in the collection tree are defined.

Both the value assigned to the entity and the parameter values can be changed so as to indicate collections chosen by the administrator.

The next parameter, RecommendedCount, is responsible for the number of publications to be selected randomly and presented in the component. If the component does not find an appropriate number of publications with thumbnails in the collection, it will display those it has been able to find.

Adding a Component to a Page

The next step of the configuration of the component is adding it to a selected page. Pages are defined in the /WEB-INF/pages.xml configuration file of the reader application. In order to add the component to the home page, the user should find the declaration of the page (the <page> tag) named “main” in the pages.xml document.

The following entry should be made in that declaration, between the <components></components> tags (the entry should be already in the file by default):

```
<component name="pl.psnc.dlibra.web.comp.pages.components.RecommendedComponent">
        <place>main</place>
        <position>4</position>
</component>
```

The configuration and changes in the components.xml and pages.xml files do not require a restart of the container of the reader application.

Configuring the Self-Archiving Functionality (Only for Dedicated Implementations with the Self-Archiving Function Switched On)

This section is about the configuration of the self-archiving mechanism, **pubcreator.properties**, in the reader application.

The first configuration file is */WEB-INF/conf/pubcreator.properties*; it contains the following parameters:

- class.name
- max.file.size
- upload.temp.dir
- main.files.suggesters.use
- main.files.suggesters
- default.main.files.lookup.filter.regexp(x)

The class.name and main.files.suggesters parameters are not subject to changes.

The *max.file.size* parameter allows administrators to introduce limits of the size of the content files of created publications. The file size is given in bytes. In this case, value “0” means that the size is not limited.

One important parameter is *upload.temp.dir*. In that parameter, the administrator determines the path to the temporary file directory. Publication content files are also stored in that directory until they are sent to the server. When the process of creating a publication has finished, the files are automatically removed from the directory. If a user does not complete the process of creating a publication in his or her session, the temporary files sent by that user to the reader application will be removed by the appropriate periodic task. **The setting of that parameter is important for the correct functioning of the self-archiving mechanism.**

The remaining parameters, that is, *main.files.suggesters.use*, *main.files.suggesters*, and *default.main.files.lookup.filter.regexp(x)* are for configuring the suggestion options of the main content file. In the current version of the dLibra system, there is a suggestion mechanism based on file names. *default.main.files.lookup.filter.regexp(x)*, parameters, where x are the subsequent numbers from set N={0,1,...n}, determine the regular expressions which recognize the potential main content files of a publication.

Configuring the Attribute Form – pubc-metadata.xml

File */WEB-INF/conf/pubc-metadata.xml* is for configuring the attribute form which appears in the second step of the publication creator. That file determines, among other things, the number, type, and order of metadata attributes.

The basic form of the file looks as follows:

```
<pubc-metadata>
        <attribute name="title" required="true" pname="true">
                <position>1</position>
                <rdf-name>Title</rdf-name>
        </attribute>
        <attribute name="author" required="true">
                <position>2</position>
                <rdf-name>Creator</rdf-name>
        </attribute>
        <attribute name="abstract">
                <position>3</position>
                <rdf-name>Abstract</rdf-name>
        </attribute>
        <attribute name="subject">
                <position>4</position>
                <rdf-name>Subject</rdf-name>
        </attribute>
        <attribute name="date" required="true">
                <position>5</position>
                <rdf-name>Date</rdf-name>
                <input-type name="DATE">
                </input-type>
        </attribute>
        <attribute name="licence" alias="true">
                <position>6</position>
                <rdf-name>Rights</rdf-name>
        </attribute>
        <attribute name="comments" alias="true">
                <position>7</position>
                <rdf-name>Description</rdf-name>
                <multiple>false</multiple>
                <input-type name="TEXTAREA"/>
        </attribute>
</pubc-metadata>
```

The main element of the XML structure is *pubc-metadata* which contains attribute elements which represent fields in the form. Every attribute element has its own set of elements and attributes.

The *attribute* elements are:

- name – the name of a field in the form; it must be unique;
- required – whether the field is required;pname – whether the field has the function of the name of a publication; and
- alias – whether the field is an attribute alias.

The subordinate elements of the *attribute* element are:

- rdf-name – the RDF name of the attribute onto the values of which the field values are mapped;
- position – the position of the field on the displayed list;
- multiple – whether many values can be assigned to the field (the default value of that element is “true”); and
- input-type – the type of the field in the form.

The “name” attribute of the “attribute” element must be unique for every defined field. The administrator can use that attribute in correlation with the “alias” attribute set to the “true” value because by default the name of a field is taken directly from the name of the dLibra metadata attribute indicated with the “rdf-name” element (see above); in this case, however, the user can set his or her own name for the filed in the label files of the “Pub licationUploadComponent” file. Such a label begins with the “PublicationUploadComponent” prefix and ends with the name determined in the attribute, for example, “PublicationUploadComponent.title”.

For more information about labels, see here.

The “*required*” attribute determines whether the field must be filled in in order for the publication to be accepted. When the user does not enter any value in that field, an appropriate message will be displayed in the creator. The “*pname*” attribute determines whether the data taken from the field will play the role of the name of the publication; a field marked in this way with the attribute should be a required field. If there is no field with the “pname” attribute set to the “true” value, the name of the created publication will be taken from the first required field.

> **Uwaga**
>
> If no field in the form is defined as the publication name field and no field in the form is marked as required, every attempt at creating the publication will end with an error in the reader application.

The first important subordinate element of the “attribute” element is “rdf-name”. It combines the indicated field of the form with an attribute in the metadata schema of the digital library. All values entered in the field are associated with a specific metadata attribute through the RDF name of the attribute. Sample RDF names are: “Title”, “Creator”, “Abstract”.

> **Uwaga**
>
> In order for the form to function correctly, a specific metadata schema attribute must be assigned to every defined field of the form.

The “position” subordinate element determines the position of the field on the list of form fields. The two additional subordinate elements, “multiple” and “input-type”, are not required. The first one determines if many values can be added to the field (the default value of that element is “true”), and the second one determines the type of the field. There are three types to choose from: TEXT (standard text field), TEXTAREA (big text field), and DATE (date checkbox); they are entered in the “name” attribute of the element. By default, the TEXT type is selected for every field.

For example:

```
<input-type name="DATE">
</input-type>
```

If a DATE-type field is defined, the following subordinate elements can be placed between the <input-type ...></input-type> tags:

- “*startYear*” – the earliest year for selection (by default, 1945);
- “*endYear*” – the latest year for selection (by default, the current year); and
- “*separator*” – the separator which appears between the parts (day, month, year) of the date in the metadata attribute value of the created publication.

Here is a sample configuration of the date field:

```
<input-type name="DATE">
  <startYear>1990</startYear>
  <endYear>2010</endYear>
  <separator>:</separator>
</input-type>
```

The last issue worth noting is the management of controlled metadata attributes. The fields which refer to controlled attributes in the RDF name are presented as expandable lists from which the user can choose one of the proposed items. Attributes are set as controlled in the Administrator Application. For more information about that subject, see here

> Entering any changes in the *pubcreator.properties* or *pubc-metadata.xml* file requires a restart of the container of the reader application.

Configuring the Rules

Before adding a publication, every user must accept the Rules, the content of which can be adjusted to the needs of the digital library. The content of the “Rules” is located on the “regulations” help page and can be changed in the administration panel of the reader application.

The Rules can also be completely disabled in the publication creator, by adding an entry concerning the “*PublicationUploadComponent*” in the “*components.xml*” file.

```
<component name="pl.psnc.dlibra.web.comp.pages.components.PublicationUploadComponent">
                <properties>
                        <property>
                                <name>regulationsAcceptRequired</name>
                                <value>false</value>
                        </property>
                </properties>
</component>
```

A change of the “components.xml” does not require a restart of the container of the reader application.

Other Configuration Stages

In order for the self-archiving functionality to operate correctly, there must be appropriate settings in the dLibra server. The configuration of the self-archiving functionality can be checked here.

Configuring the Mechanism of Adding Public Tags (Only for Dedicated Implementations with the Public Tag Function Switched On)

The basic parameters of the action of adding public tags are configurable. To configure them, the user should open the action configuration file, /WEB-INF /actions.xml. It contains the following entry:

```
<action name="pl.psnc.dlibra.web.comp.pages.actions.AddBookmarkTagAction">
        <properties>
                <property>
                        <name>accept.all.tags</name>
                        <value>false</value>
                </property>
                <property>
                        <name>max.tag.length</name>
                        <value>100</value>
                </property>
                <property>
                        <name>min.tag.length</name>
                        <value>3</value>
                </property>
        </properties>
 </action>
```

In the public tag moderation process, they are kept for a period of time in which the moderator can set their status to the “accepted” or “unaccepted” value. Only accepted public tags are added to the “Tag” attribute of the selected publication. The “accept.all.tags” parameter determines whether the sent proposal of a public tag is to be automatically set as “accepted”, which status can be changed by the moderator. By default, the “accept.all.tags” parameter has the “false” value, which means that the added public tags are “unaccepted” by default.

The two following parameters determine the minimum (“min.tag.length”) and maximum (“max.tag.length”) number of characters for a public tag. If a tag is not within the limit, it is not added to the proposal of key words. Setting the value of any of those parameters to “0” will cause the parameter to not be taken into account when checking the number of tag characters.

Configuration of the Display of Protected PDF Files

The described configuration allows the user to determine the protection in the PDF document displayed by the reader application with greater precision. In order to begin modifying the parameters for protecting PDF files, the user should display the “settings.xml” file from the */WEB-INF/formats/pdf* for edition.

The file should contain the following entries:

```
<properties>
<!--
 In case of any problems with securing PDF files you may try to use alternative format writer based on iText.
-->
<!--
 <entry key="secured.format.writer.class">pl.psnc.dlibra.web.comp.formats.writers.ItextSecuredPdfFormatWriter<
/entry>
-->
<entry key="secured.format.writer.class">pl.psnc.dlibra.web.comp.formats.writers.SecuredPdfFormatWriter</entry>
<entry key="handler.id">pdf:secured</entry>
<entry key="handled.mime.types">application/pdf</entry>
<entry key="handles.secured">true</entry>
<entry key="handles.normal">false</entry>
<entry key="owner.pass"></entry>
<entry key="user.pass"></entry>
<entry key="print.degraded">true</entry>
<entry key="print">false</entry>
<entry key="modify.annotations">false</entry>
<entry key="fill.in.form">false</entry>
<entry key="extract.for.accessibility">false</entry>
<entry key="assemble.document">false</entry>
<entry key="extract.content">false</entry>
</properties>
```

The basic parameters are "**owner.pass**" and "**user.pass**". They determine, respectively: the password of the owner of the PDF document and the password of the user of the PDF document. The user password allows document browsing, and the owner's password is needed for introducing changes in those documents which allow that.

As regards the remaining parameters, there are (with the omission of the general parameters of the configuration of the display mechanism):

- *assemble.document* – determines whether the user can modify – that is, turn, remove, and add – pages in a protected file;
- *print.degraded* – determines whether the user can print the document in a degraded quality;
- *print* – determines whether the user can print the document;
- *modify.annotations* – determines whether the user can modify notes in a text or enter data in interactive forms;
- *fill.in.form* – determines whether the user can enter data in interactive forms;
- *extract.for.accessibility* – determines whether the content of a document can be downloaded to be made available for people with visual impairment;
- *extract.content* - określa czy użytkownik może zaznaczać i kopiować treść dokumentu
- *modify* – determines whether the user can modify a downloaded document.

All the parameters from the list have boolean-type values ("true" or "false")

If the PDF protection does not produce the expected results, an alternative mechanism of generating protected PDFs can be used, by entering value "*pl.psnc.dlibra.web.comp.formats.writers.ItextSecuredPdfFormatWriter*" (in the XML comment) in the "**secured.format.writer.class**" parameter, instead of the default "*pl.psnc.dlibra.web.comp.formats.writers.SecuredPdfFormatWriter*" value.

A change of those parameters requires a restart of the container (Apache Tomcat) of the reader application.

Configuring the Main Collection for the Reader Application

Since version 5.0 of the dLibra system, administrators can indicating any collection from the collections in the digital library to be the main collection, and they can connect many reader applications to one dLibra server. In order to configure the main collection for a reader application, the /WEB-iNF/web.xmi file must be opened for edition. It contains the following entry (by default, in the XML comment):

```
<context-param>
        <description>
                Configurable main dLibra collection id
        </description>
        <param-name>main.collection.id</param-name>
        <param-value>[id of collection]</param-value>
</context-param>
```

The entry should be "uncommented", and the "[id of collection]" should be replaced with the numerical ID of the collection (the unique ID set in the collection database). In the end, the servlet environment (for example, Tomcat) should be restarted.

Changes in the “Contact” Form

In order to change the default contact values in the form, the user should open the “WEBAPP_en.xml” (for English) and “WEBAPP_ pl.xml” (for Polish) from the “/WEB-INF/components/resources” directory for edition. The files may contain the following entries:

```
<entry key="ContactComponent.Name">Instytucja partnerska BC</entry>
<entry key="ContactComponent.Address">ul. Nieznana 16, 61-895 Pozna, POLSKA</entry>
<entry key="ContactComponent.URL">http://institution.com/</entry>
<entry key="ContactComponent.Mail">mail@institution.com</entry>
<entry key="ContactComponent.Phone">telefon: (+48) 61-333-33-32, faks: (+48) 61-333-33-33</entry>
```

Particular values for the XML elements correspond to the values from the form. The changes in those files do not require a restart of the servlet container.

As regards the contact form, one should also remember about the email address to which the messages entered by the user are sent. That address can be configured in the “/WEB-INF/actions.xml” file, by changing the value for the declared “SendMailActiorr” action.

```
<action name="pl.psnc.dlibra.web.comp.pages.actions.SendMailAction">
        <properties>
                <property>
                        <name>to.web.mail</name>
                           <value>[adresy mail, na ktory maja byc wysylane zgloszenia]</value>
                </property>
                    ...
        </properties>
</action>
```

Any number of semicolon- or comma-separated email addresses can be placed between the “value” tags.

The Settings of the Management of Watermarks (Only for Dedicated Implementations with the Public Tag Function Switched On)

Watermarks are displayed for **protected JPG presentations** (a special publication tag in the dLibra system, which is created by indicating the directory with JPG files as the main file). It is done with the use of a special Java applet.

A watermark is an image which is put over the content of a presentation, with the effect of opacity. We recommend the use of a graphics file in the PNG format which makes it possible to save an image with a transparent background. Such an image is also enlarged to take as much space as possible. However, it is centered and not spread to the proportions of the main image. One can also obtain the effect of a watermark smaller than the main image, by adding a transparent border of selected width in the watermark graphic.

Apart from an image, the name of the logged in user can also be automatically put in a watermark. Watermark configuration is located in the “**WEB-INF /web.xml file**”, in the section about the “**watermarkServlet**” servlet. The following parameters can be defined there:

- “`image.path`” – the path to the image which is to be displayed as the watermark;
- “`opacity`” –the strength of the opacity effect; it must be a number from the (0,1) range; for value 0, the image is invisible, completely transparent, and for value 1, it completely covers the content;
- “`text.userName.position`” – the position indicates the upper left-hand corner of the rectangle in which the name of the user will be placed; the position is entered as coordinates in the watermark image, where the 0,0 point is the upper left-hand corner of the image;
- “`text.userName.size`” – the size of the rectangle in which the name of the user will be entered, also in relation to the size of the watermark; if the size of the font is not defined, then the size which will make it possible to fit in the name in the rectangle will be automatically set;
- “`text.userName.align`” – the position of the name of the user in the defined rectangle; possible values: left, center, and right;
- `"text.userName.wordwrap"` – makes text wrapping possibel when the username is long;
- “`text.userName.font`” – the font of the username; the font availability depends on what fonts are installed on the user’s computer; usually, however, Arial, Courier New, Monospaced, Verdana, and Tahoma fonts are available;
- “`text.userName.font.style`” – makes it possible to enable bold and italic type in the username;
- “`text.userName.font.color`” - – the color of the username, in the hexadecimal format; and
- “`text.userName.font.size`” – the font size; if it is not defined, the maximum size which allows the whole name to fit in will be chosen.

Making the Site Map Available to Indexing Bots

The site map is a protocol which makes it easier for indexing bots to collect information about the most important pages in the reader application. Within the framework of the “**SitemapGeneratingTask**” periodic task, the reader application generates absolute URLs to those pages and saves them in accordance with the following assumptions:

- **/sitemap/** – the directory in which all files containing the addresses of the most important pages of the reader application are saved; the files are divided so as to not exceed 50,000 addresses;

- **sitemapindex.xml** – the file in which the paths to all files in the **/sitemap/** directory are aggregated; and
- **robots.txt** – the file which contains the necessary information for the indexing bots; the "**SitemapGeneratingTask**" periodic task adds a new line to it with the absolute URL to the "**sitemapindex.xml**" file, for example, Sitemap: {domena}/sitemapindex.xml.

In order to change the default page map generation settings in the system, the user should edit the "periodic.xml" file in the "/WEB-INF" directory of the reader application. The entry concerning the appropriate periodic task is looks as follows:

```
        <periodic-task logicClass="pl.psnc.dlibra.web.comp.periodic.SitemapGeneratingTask"
                 executeOnStart="yes" >
                <description>Generates sitemap.</description>
                <expression>0 0 0 1/10 * ?</expression>
                <properties>
                        <property>
                                <name>urlParams</name>
                                <value>Title;Creator</value>
                        </property>
                        <property>
                                <name>overwriteRobotsFile</name>
                                <value>true</value>
                        </property>
                </properties>
        </periodic-task>
```

In the task, you can set the values of the following parameters:

- "urlParams" – the RDF names of the metadata attributes the values of which, in the case of publications and editions, will be added to the URL address; for the sample value "Title;Creator" of the "urlParams" parameter, we can expect the following result for "Pan Tadeusz" item: {domena} /publication/100/pan-tadeusz-adam-mickiewicz; and
- "overwriteRobotsFile" – the information about whether the "**SitemapGeneratingTask**" periodic task is to overwrite the "**Sitemap**" line in the "robots. xml" file; the possible values are "true" or "false"; if the set value is "false", the user should remember to manually add the "**Sitemap**" value in the "robots.txt" file.

Managing Linking Publications

Linking publications make it possible to enter, in a digital library, the information about an object which is made available by another website. In the default configuration, when the user wants to see the content of such a publication, the user will first see a notice the information about the external content and the target link which has to be called. If we want the user to be redirected to the target address without having the notice displayed, we should add the configuration presented below to the "`WEB-INF/components.xml`":

```
    <component name="pl.psnc.dlibra.web.comp.pages.components.ContentBrowserComponent">
        <properties>
            <property>
                <name>linkedPublication</name>
                <value>redirect</value>
            </property>
        </properties>
    </component>
```