05. Zmiany w konfiguracji Aplikacji Czytelnika

Konfiguracja mechanizmu dodawania tagów publicznych Zmiany w konfiguracji Aplikacji Czytelnika

- Konfiguracja aplikacji czytelnika
  - Konfiguracja usługi CAS
  - Konfiguracja importu wiadomości
  - Dodawanie exlibrisów
  - Konfiguracja systemu kopii zapasowych JCR
  - Konfiguracja miniatur
  - Konfiguracja komponentu "Polecane"
  - Konfiguracja funkcjonalności Self-Archiving (tylko dla dedykowanych wdrożeń z włączoną funkcją Self-Archiving)
  - Konfiguracja wyświetlania zabezpieczonych plików PDF
  - Konfiguracja kolekcji głównej dla Aplikacji Czytelnika
  - Zmiany w formularzu "Kontakt"
  - Ustawienia obsługi znaków wodnych (tylko dla dedykowanych wdrożeń z włączoną funkcją tagów publicznych)
  - Udostępnianie Mapy Strony robotom indeksującym
  - Obsługa publikacji linkujących
  - Aktywacja reCAPTCHA

Konfiguracja aplikacji czytelnika

W niniejszym rozdziale znajdują się instrukcje związane z konfiguracją niektórych mechanizmów związanych z działaniem aplikacji czytelnika.

Konfiguracja usługi CAS

dLibra umożliwia wykorzystanie centralnego systemu logowania stworzonego za pomocą serwera CAS i usługi katalogowej LDAP. CAS jest dostępny jako kolejny dostawca tożsamości w pliku **dlibra-webapp/WEB-INF/conf/user-providers.xml**. Należy odkomentować fragment poniższy fragment:

```
<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>
```

Następnie należy:

- zmienić zawartość znacznika loginPage tak aby wskazywał na stronę logowania wykorzystywanego serwera CAS.
- W znaczniku userAttributesDefaults wpisać nazwy atrybutów, które mają być wykorzystywane przy tworzeniu dynamicznych grup właściwości w aplikacji Administratora systemu dLibra. Wartość atrybutu value każdego znacznika property określa wartość domyślną dla danego atrybutu.
- Jeżeli w wykorzystywanym rejestrze LDAP każdy użytkownik posiada adres email wyspecyfikowany jako wartość konkretnego atrybutu, nazwę tego atrybutu można wpisać w znacznik **emailAttributeName**. Wówczas dLibra będzie w stanie wykorzystać informacje o adresie email użytkowników.
- Jeżeli chcą Państwo zmienić sposób w jaki dany dostawca tożsamości jest nazywany należy w nadpisać (w WEBAPP_xx.xml) wartość etykiety tekstowej, której nazwa znajduje się w methodNameResourceKey.

Następnie należy uzupełnić zawartość plik **dlibra-webapp/WEB-INF/conf/ldap.properties** - zawiera on parametry umożliwiające wykorzystanie usługi katalogowej LDAP.

Konfiguracja importu wiadomości

Aplikacja czytelnika pozwala na publikowanie na stronie głównej różnego rodzaju wiadomości. Wiadomości te można dodawać za pośrednictwem panelu administracyjnego aplikacji czytelnika bądź wykorzystując mechanizm importu wiadomości z kanału RSS. Poniżej omówiona został sposób w jaki import taki można przeprowadzić.

Aby zaimportować wiadomości z dostępnego w sieci kanału RSS należy w pliku periodic.xml odkomentować znacznik periodic-task zawierający konfigurację klasy
NewsImportTask.

```
<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>
```

Konfiguracja tego zadania składa się z następujących parametrów:

- feed.url - adres kanału RSS/Atom zawierającego importowane wiadomości.
- add.post.link.to.blog - jeżeli wartość tego parametru zostanie ustawiona na true do każdej wiadomości zostanie dodany odnośnik do strony na której wiadomość była pierwotnie publikowana.
- full - który element wpisu w kanale RSS ma zostać użyty jako pełny tekst wiadomości. W przypadku kanałów RSS przeważnie będzie to element description, jednak dla kanałów w formacie Atom oprócz istnieją dwa znaczniki zawierające treść wpisu: content i summary. Dopuszczalne wartości to : feed_description, feed_content.
- short - który element kanału ma zostać zaimportowany jako skrót wiadomości. Dopuszczalne wartości to : feed_description, feed_content.
- publish.in.all.languages - jeżeli parametr ten będzie miał wartość true wiadomość zostanie zaimportowana we wszystkich zainstalowanych w aplikacji czytelnika językach interfejsu.
- blog.language - język w jakim napisane są importowane wiadomości.
- start.date - zostaną zaimportowane tylko wpisy opublikowane po wpisanej tu dacie (data w formacie dd.MM.YYYY)

Zadanie to jest domyślnie wykonywane raz na dobe. Gdy tylko aplikacja czytelnika wykryje, że w kanale RSS pojawiła się nowa wiadomość zaimportuje przy najbliższym wykonaniu tego zadania.

Dodawanie exlibrisów

Exlibrisy w aplikacji czytelnika systemu dLibra reprezentowane są jako obrazki z odnośnikiem do wybranej strony (biblioteki, instytucji itp.), które pojawiają się na stronie opisu publikacji (*publication, docmetadata*). Exlibrisy są dodawane wg wartości konkretnych atrybutów umieszczonych w metadanych. Np. exlibris może zostać wyświetlony w zależności od wartości atrybutu Prawa - dla wartości "Instytucja Biblioteczna X" będzie wyświetlony obrazek identyfikujący daną instytucję oraz odnośnik do strony głównej tej instytucji "*www.instytucjabibliotecznax.com*".

Procedura dołączania exlibrisu wygląda następująco:

1. W katalogu **webapp/exlibris** należy umieścić plik graficzny z ekslibrisem biblioteki
2. Następnie w pliku **webapp/WEB-INF/conf/exlibris.xml** znajdują się następujące wpisy:

```
<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>
```

- Poprzez element XML *attribute* i wartość *rdfname* (nazwa RDF) wskazywany jest atrybut metadanych publikacji, którego wartość będzie reprezentowała exlibris; elementów "attribute" może być wiele.
- Następnie w ramach elementu *attribute* definiuje się elementy *attribute-value*, które reprezentują wartości atrybutu metadanych publikacji identyfikujące exlibrisy. Te konkretne wartości wprowadza się w polu XML *value* elementu *attribute-value*. W podanym wyżej przykładzie domyślnej konfiguracji wartością atrybutu jest "Biblioteka Uniwersytecka w Poznaniu". Rozpoznanie tej wartości w metadanych publikacji spowoduje wyświetlenie odpowiedniego exlibrisa na stronie opisu metadanych publikacji w aplikacji czytelnika (strona *docmetadata lub publication*) . Każdy element *attribute-value* odpowiada za jedną wartość atrybutu, stąd jeśli chcemy aby exlibris był rozpoznawany za pomocą różnych wariantów nazw, to należy stworzyć osobny element *attribute-value* dla każdej nazwy.
- Ostatecznie wewnątrz elementu *attribute-value* określa się elementy *exlibris*. Element *exlibris* posiada dwa podrzędne elementy *img* oraz *link*, z których pierwszy jest nazwą pliku graficznego
(dodanym w kroku pierwszym do folderu */exlibris*), drugi zaś wskazuje na odnośnik do strony, na która zostanie przekierowany użytkownik po kliknięciu na obrazek exlibrisu.

Każdy z elementów: *attribute*, *attribute-value* i *exlibris* może być definiowany wielokrotnie, co pozwala na dodanie wielu exlibrisów w ramach każdej określonej wartości atrybutu metadanych.

Przykładowo:

```
<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>
```

W przypadku takiej konfiguracji, dla atrybutu **Praw**, exlibris *Biblioteki Uniwersyteckiej w Poznaniu* pojawi się zarówno, gdy w atrybucie podana będzie wartość "Biblioteka Uniwersytecka w Poznaniu" jak i "BUP". Dodatkowo jeśli w tym samym atrybucie **Praw** pojawi się wartość "Uniwersytet im. A. Mickiewicza i Politechnika Poznańska", wówczas wyświetlą się dwa exlibrisy (dwa obrazki z odnośnikami) odpowiednio dla UAM i PP.

Konfiguracja exlibris.xml zacznie działać dopiero po restarcie środowiska serwletów aplikacji czytelnika (Apache Tomcat). Nieprawidłowe ustawienia mogą w efekcie prowadzić do wyświetlania wyjątków i błędów w logach systemu.

Konfiguracja systemu kopii zapasowych JCR

Aby zmienić domyślne ustawienia systemu tworzenia kopii zapasowych JCR należy wyedytować plik *periodic.xml* znajdujący się w katalogu /WEB-INF Aplikacji Czytelnika. Wpis dotyczący odpowiedniego zadania okresowego przedstawia się następująco:

```
<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>
```

W zadaniu można zmienić dwa parametry: *jcr.backup.dir* oraz *copies.amount*.
Pierwszym parametrem administrator wskazuje katalog, do którego mają być umieszczane pliki kopii zapasowych. Każda pojedyncza kopia zapasowa to plik w formacie .xml, o nazwie jcr-backup_[data utworzenia kopii].xml, np. *jcr-backup_2010.12.25.08.00.00.xml*.

Drugi parametr określa maksymalną liczbę kopii zapasowych. Zadanie okresowe wykonuje się cyklicznie, domyślnie co 24 godz., i stąd po utworzeniu przez nie maksymalnej ilości plików kopii, każdy kolejno utworzony nadpisuje najstarszy plik kopii z katalogu *jcr.backup.dir*.

Możliwe wartości parametru *copies.amount*:

- dla parametru > 0, zadanie tworzy określoną liczbę plików kopii zapasowych;
- dla parametru = 0, zadanie nie tworzy żadnych plików kopii zapasowych;
- dla parametru < 0, zadanie tworzy nielimitowaną liczbę plików kopii zapasowych.

Zmiana parametrów w zadaniach okresowych wymaga restartu kontenera (np. Apache Tomcat) Aplikacji Czytelnika.

> **Uwaga**
>
> Błędne wypełnienie wspomnianych parametrów może spowodować nieprawidłowe działanie zadania i wypisywanie wyjątków w logach.

Odzyskiwanie danych z kopii zapasowych JCR

Aby odzyskać dane z utworzonego pliku .xml kopii zapasowej należy umieścić go w odpowiednim katalogu. Katalog ten jest konfigurowany w pliku */WEB-INF/conf/jcr.properties* Aplikacji Czytelnika pod parametrem *jcr.restore.dir*.
Kopia zapasowa jest automatycznie odzyskiwana podczas ponownego uruchomienia kontenera (np. Apache Tomcat) Aplikacji Czytelnika.
Uwaga! Plik kopii zapasowej, po odzyskaniu z niego danych, jest automatycznie usuwany.

Konfiguracja miniatur

Za konfigurację miniatur wyświetlanych w Aplikacji Czytelnika odpowiedzialny jest parametr (context-param) z pliku */WEB-INF/web.xml* Aplikacji Czytelnika, który przedstawia się następująco:

```
<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>
```

Wartości parametru wstawia się między znacznikami ***<param-value></param-value>***. Parametr składa sie z par **id_miniatury:zestaw_parametrów_miniatury**. Zestaw parametrów jest czteroczęściowy i poszczególne jego części oddzielone są od siebie znakiem ';'. Istnieje zatem możliwość zdefiniowania różnych profili miniatur np.: recomended, col_home_recommended, collection_description itd. Dzięki temu możemy sobie wybrać jakie wymiary ma mieć nasz obrazek w zależności od potrzeb. Schemat dostępu do miniatury przedstawia poniższy url:

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

gdzie:

- **{domena}** - domena internetowa naszej biblioteki np. demo.dl.psnc.pl,
- **{typ_obiektu}** - typ obiektu, dla którego będziemy generować miniaturkę(np.: edition, publication, collection)
- **{id_miniatury}** - id miniatury zdefiniowanej w thumbnails.settings
- **{id}** - id obiektu(wydania, publikacji lub kolekcji)

W przedstawionym wyżej przykładzie zestaw parametrów dla miniatury **example** wynosi: 200, 200, 1.0, false oraz image/png. Każda część oznacza kolejno:

- szerokość wynikowego pliku graficznego (w przykładzie równa 200 pikseli);
- wysokość wynikowego pliku graficznego (w przykładzie równa 200 piskeli);
- stopień kompresji wynikowe pliku graficznego (w przykładzie równa 1.0);
- czy wejściowy plik graficzny ma być obcięty proporcjonalnie do wymiarów (w przykładzie równa 'false');
- format wynikowego pliku graficznego (w przykładzie jest to 'image/png');

Pierwsze dwie wartości określają jakie wymiary będzie miał wyjściowy plik graficzny miniatury dla Aplikacji Czytelnika, bez względu na wymiary pierwotnego pliku graficznego - wprowadzonego w Aplikacji Redaktora. Jeśli więc przykładowo plik graficzny w Aplikacji Redaktora został wprowadzony w wymiarach szerokość = 500 pikseli i wysokość = 200 pikseli, to w Aplikacji Czytelnika będzie on odpowiednio przeskalowany do wymiarów skonfigurowanych w parametrze *thumbnails.settings*. Może to być przydatne w przypadku, gdy wprowadzane przez redaktorów pliki miniatur są dużej szczegółowości, niepotrzebnej przy wyświetlaniu w Aplikacji Czytelnika; zmniejszenie ich wymiarów zmniejsza zarazem ilość pamięci potrzebnej do ich przesyłu, co może dodatnio wpłynąć na szybkość transferu danych.

Kolejny parametr jest powiązany z ostatnim i jest brany pod uwagę jedynie w przypadku, gdy miniatura ma ustawiony format wyjściowego pliku graficznego na 'image/jpeg'. Dzięki temu dodatkowo można zmniejszyć przesyłany plik stosując zalety kompresji stratnej formatu JPEG. Sensowną kompresje ustala się za pomocą liczby zmiennopozycyjnej w przedziale (0,1),
np. 0.1, 0.25, 0.3 itd.

Następny parametr określa sposób w jaki pierwotny obraz graficzny ma być dostosowany do wymiarów miniatury wprowadzonych w parametrze *thumbnails .settings*. Ustawienie flagi na 'true' spowoduje, że pierwotny plik graficzny zostanie proporcjonalnie zmniejszony i następnie obcięty do obszaru zdefiniowanego przez pierwsze dwa parametry. Ustawienie flagi na 'false' (domyślne) spowoduje, że pierwotny plik graficzny zostanie wpisany w obszar zdefiniowany przez parametry miniatury *thumbnails.settings*.

Ostatni z parametrów określa typ wynikowego formatu graficznego miniatury. Wspierane są dwa najpopularniejsze formaty: JPEG oraz PNG. W przypadku wybrania formatu JPEG znaczenie ma również parametr stopnia kompresji (opisany wyżej). Poszczególne formaty identyfikowane są przez typy MIME, tj. dla PNG - 'image/png' oraz dla JPEG - 'image/jpeg'.

Zmiana parametrów wyświetlanej miniatury wymaga restartu kontenera (np. Apache Tomcat) Aplikacji Czytelnika.

> **Uwaga!**
>
> Wprowadzenie nieprawidłowych parametrów może zaowocować niewyświetlającymi się miniaturami oraz błędami w logach.

Konfiguracja komponentu "Polecane"

Podstawowa konfiguracja komponentu "Polecane" znajduje się w pliku */WEB-INF/components.xml* Aplikacji Czytelnika.

Przykładowy wpis konfiguracyjny komponentu przedstawia się następująco:

```
<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>
```

Konfiguracja posiada dwa podstawowe parametry:

- *RecommendedId* - parametr wskazujący ID kolekcji zawierającej publikacje polecane;
- *RecommendedCount* - parametr wskazujący ile publikacji ma być wylosowanych do wyświetlenia w komponencie;
- *reverseOrder - parametr odwracający domyślną kolejność wyświetlania.*

Pierwszy parametr w swoich domyślnych wartościach wstrzykuje wartość encji *recommendedId*, która jest deklarowana
na początku dokumentu w linii:

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

Domyślnie wartością encji jest '3', która wskazuje na ID automatycznie utworzonej, przy instalacji, kolekcji.
Ze wskazanej tak kolekcji losowane są publikacje, które mają być wyświetlone w komponencie, przy czym wyświetlane są wyłącznie te, które posiadają przypisaną miniaturę. W przypadku braku w kolekcji jakichkolwiek publikacji spełniających to założenie, komponent się nie wyświetli.

Wartość encji *recommendedId* wykorzystywana jest również w konfiguracji komponentu *CollectionsStructureComponent* w parametrze *HiddenCollectionsIds*, w którym określane są kolekcje **niewidoczne** w komponencie drzewa kolekcji.

Zarówno wartość przypisana encji jak i wartości parametrów mogą zostać zmienione, tak aby wskazywać na kolekcje wybrane przez administratora.

Kolejny parametr *RecommendedCount* odpowiada za liczbę publikacji, które mają być wylosowane i prezentowane w komponencie. Jeśli komponent nie odnajdzie we wskazanej kolekcji odpowiedniej liczby publikacji z miniaturami, wyświetli te, które udało mu się odnaleźć.

Dodanie komponentu na stronę

Kolejnym krokiem konfiguracji komponentu jest dodanie go na wybraną stronę. Strony definiuje się w pliku konfiguracyjnym */WEB-INF/pages.xml* Aplikacji Czytelnika. Aby dodać komponent na stronę główną, należy w dokumencie *pages.xml* znaleźć deklarację strony (znacznik <page>) o nazwie *main*.
W deklaracji tej, między znacznikami <components></components> należy umieścić wpis (domyślnie powinien on już być obecny w komentarzu w pliku):

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

Wprowadzanie zmian i konfiguracja zarówno w obszarze pliku *components.xml* jak i *pages.xml* nie wymaga zrestartowania kontenera Aplikacji Czytelnika.

Konfiguracja funkcjonalności Self-Archiving (tylko dla dedykowanych wdrożeń z włączoną funkcją Self-Archiving)

Niniejsza sekcja przedstawia konfigurację mechanizmu Self-Archiving w Aplikacji Czytelnika.

pubcreator.properties

Pierwszym plikiem konfiguracyjnym jest */WEB-INF/conf/pubcreator.properties*, w którym zawarte są następujące parametry:

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

Parametry *class.name* oraz *main.files.suggesters* nie podlegają zmianom.

Parametr *max.file.size* umożliwia administratorom wprowadzenie limitów na wielkość pliku treści tworzonych publikacji. Wielkość pliku podaje się w bajtach. Wartość 0 w tym wypadku oznacza brak limitu na wielkość.

Istotnym parametrem jest *upload.temp.dir*, w którym administrator ustala ścieżkę do katalogu plików tymczasowych. W katalogu tym przechowywane są pliki z treścią publikacji do momentu przesłania ich do serwera. Po zakończeniu procesu tworzenia publikacji, pliki te są automatycznie usuwane z katalogu. Jeśli użytkownik w ramach swojej sesji nie dokończy procesu tworzenia publikacji, wówczas pliki tymczasowe, przesłane przez niego do Aplikacji Czytelnika, usunie zadanie okresowe. **Ustawienie tego parametru jest istotne do prawidłowego działania mechanizmu Self-Archiving**.

Reszta parametrów, tj. *main.files.suggesters.use*, *main.files.suggesters* oraz *default.main.files.lookup.filter.regexp*(x) służy do konfiguracji opcji podpowiadania pliku głównego treści. W obecnej wersji dLibry opracowany został mechanizm podpowiadania działający w oparciu o nazwę plików. Parametry *default.main.files.lookup.filter.regexp*(x), gdzie x stanowią kolejne liczby ze zbioru N={0,1,...n}, określają wyrażenia regularne rozpoznające potencjalne pliki główne treści publikacji.

Konfiguracja formularza atrybutów - pubc-metadata.xml

Plik */WEB-INF/conf/pubc-metadata.xml* służy do konfiguracji formularza atrybutów, który pojawia się w 2-gim kroku kreatora publikacji. W pliku tym określa się m.in. liczbę, rodzaj oraz kolejność wyświetlania atrybutów metadanych.

W podstawowej formie plik przedstawia się następująco:

```
<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>
```

Głównym elementem struktury XML jest *pubc-metadata*, który zawiera elementy *attribute* reprezentujące pola w formularzu.

Każdy element *attribute* posiada własny zestaw elementów oraz atrybutów.

Atrybuty elementu *attribute* :

- name - nazwa pola w formularzu, musi być unikalna;
- required - czy pole jest wymagane;
- pname - czy pole spełnia funkcję nazwy publikacji;
- alias - czy pole jest aliasem dla atrybutu.

Elementy podrzędne elementu *attribute*:

- rdf-name - nazwa RDF atrybutu, na którego wartości mapowane są wartości pola;
- position - pozycja pola na wyświetlanej liście;
- multiple - czy pole może mieć przypisanych wiele wartości (domyślnie 'true');
- input-type - typ pola w formularzu.

Atrybut nazwy *name* elementu *attribute* musi być unikalny dla każdego zdefiniowanego pola. Administrator może go wykorzystać w korelacji z atrybutem *alias* ustawionym na 'true', domyślnie bowiem nazwa dla pola jest pobierana bezpośrednio od nazwy atrybutu metadanych dLibra, wskazanego elementem *rdf-name* (patrz wyżej), w tym wypadku można jednakże ustawić własną nazwę dla pola w plikach etykiet komponentu *PublicationUploadComponent*. Etykieta taka rozpoczyna się od prefiksu "PublicationUploadComponent" a kończy nazwą określoną w atrybucie *name*, np. "PublicationUploadComponent.title". Więcej na temat etykiet można przeczytać **tutaj**.

Atrybut *required* określa czy wypełnienie pola jest wymagane do zaakceptowania tworzonej publikacji.
W sytuacji, gdy użytkownik nie wprowadzi w to pole żadnej wartości wyświetlony zostanie w kreatorze odpowiedni komunikat.
Atrybut *pname* określa czy dane brane z pola będą spełniały rolę nazwy dla publikacji; pole tak oznaczone tym atrybutem powinno być jednocześnie polem wymaganym. Jeśli nie istnieje żadne pole z atrybutem *pname* ustawionym na 'true', wówczas w trakcie tworzenia publikacji jego nazwa zostanie wzięta z pierwszego w kolejności pola wymaganego.

> **Uwaga**
>
> Jeśli żadne pole w formularzu nie zostało zdefiniowane jako pole nazwy publikacji oraz nie istnieje przy tym żadne pole oznaczone jako wymagane, wówczas każda próba utworzenia publikacji zakończy się błędem w Aplikacji Czytelnika.

Pierwszym istotnym elementem podrzędnym elementu *attribute* jest *rdf-name*, który łączy wskazane pole formularza z atrybutem w schemacie metadanych biblioteki cyfrowej. Wszystkie wpisywane w pole wartości są kojarzone z konkretnym atrybutem metadanych poprzez nazwę RDF atrybutu. Przykładowymi nazwami RDF są: Title, Creator, Abstract itp.

> **Uwaga**
>
> Do prawidłowego funkcjonowania formularza potrzebne jest przypisanie każdemu zdefiniowanemu polu formularza konkretnego atrybutu schematu metadanych.

Element podrzędny *position* określa pozycję pola na liście pól formularza. Dwa dodatkowe podrzędne elementy, tj. *multiple* oraz *input-type* nie są wymagane. Pierwszy określa czy do danego pola można dodać wiele wartości (domyślnie 'true'), drugi określa jakiego typu jest pole formularza. Można wybrać spośród trzech typów: TEXT (standardowe pole tekstowe), TEXTAREA (duże pole tekstowe) oraz DATE (pole wyboru daty), które wpisuje się w atrybucie *name* elementu. Domyślnie dla każdego pola wybierany jest element typu TEXT.

Przykładowo:

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

Jeśli zdefiniowane pole jest typu DATE wówczas można między znacznikami <input-type ...></input-type> umieścić podrzędne elementy:

- *startYear* - najwcześniejszy możliwy rok do wyboru (domyślnie 1945);
- *endYear* - najpóźniejszy możliwy rok do wyboru (domyślnie rok aktualny);
- *separator* - separator, który pojawia się po między poszczególnymi częściami daty (dniem, miesiącem, rokiem) w wartości atrybutu metadanych stworzonej publikacji.

Przykład konfiguracji pola daty:

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

Ostatnią kwestią wartą zaznaczenia jest obsługa kontrolowanych atrybutów metadanych. Pola, które w nazwie RDF odwołują się do atrybutów kontrolowanych prezentują się jako listy rozwijalne, w których użytkownik może wybrać jedną z proponowanych pozycji. Atrybuty ustawia się jako kontrolowane w Aplikacji Administratora. Więcej na ten temat można znaleźć **tutaj**.

> Wprowadzenie zmian tak w pliku *pubcreator.properties* jak i *pubc-metadata.xml* wymaga restartu kontenera Aplikacji Czytelnika.

Konfiguracja regulaminu

Przed przystąpieniem do dodawania publikacji każdy użytkownik jest zobowiązany do zaakceptowania regulaminu, którego treść można zmienić na potrzeby własnej biblioteki cyfrowej. Treść regulaminu znajduje się na stronie pomocy *regulations* i może ona zostać zmieniona w panelu administracyjnym Aplikacji Czytelnika.
Istnieje również możliwość całkowitego wyłączenia regulaminu z kreatora publikacji, wystarczy w tym celu do pliku *components.xml* dodać wpis dotyczący komponentu *PublicationUploadComponent*:

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

Wprowadzenie zmiany w pliku *components.xml* nie wymaga wymagać restartu kontenera Aplikacji Czytelnika.

Inne etapy konfiguracji

Do prawidłowego działania funkcjonalności Self-Archiving potrzebne są jeszcze ustawienia w serwerze dLibra.
Konfigurację Self-Archiving w serwerze dLibra można sprawdzić tutaj.

Konfiguracja wyświetlania zabezpieczonych plików PDF

Omawiana konfiguracja umożliwia precyzyjniejsze określenie zabezpieczenia w wyświetlanym przez Aplikację Czytelnika dokumencie PDF.
Aby rozpocząć modyfikację parametrów zabezpieczania plików PDF należy wyświetlić do edycji plik *settings.xml* z katalogu */WEB-INF/formats/pdf*.
Plik ten powinien zawierać wpisy:

```
<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>
```

Podstawowymi parametrami są **owner.pass** oraz **user.pass**, które określają kolejno: hasło właściciela dokumentu PDF oraz hasło użytkownika dokumentu PDF (hasło użytkownika pozwala przeglądać dokument, hasło właściciela jest potrzebne do wprowadzania zmian w dokumentach, które na to pozwalają).
Z pozostałych parametrów można wymienić (pomijając ogólne parametry konfiguracji mechanizmu wyświetlania):

- *assemble.document* - parametr określa czy użytkownik może w zabezpieczonym pliku modyfikować strony, tj. obracać, usuwać i dodawać nowe
- *print.degraded* - określa czy użytkownik może drukować dokument w zubożonej jakości
- *print* - określa czy użytkownik może drukować dokument
- *modify.annotations* - określa czy użytkownik może modyfikować adnotacje w tekście lub wypełniać danymi interaktywne formularze
- *fill.in.form* - określa czy użytkownik może wypełniać danymi interaktywne formularze.
- *extract.for.accessibility* - określa czy można pobierać treść w dokumencie na potrzeby dostępności dla osób niewidomych
- *extract.content* - określa czy użytkownik może zaznaczać i kopiować treść dokumentu
- *modify* - określa czy użytkownik może modyfikować pobrany dokument

Wszystkie wymienione w liście parametry przyjmują wartości typu *boolean* (true|false).

> ⚠ W przypadku, gdyby zabezpieczenie PDFów nie dawało oczekiwanych rezultatów wówczas można użyć alternatywnego mechanizmu generowania zabezpieczonych PDFów, poprzez użycie w parametrze **secured.format.writer.class** wartości *pl.psnc.dlibra.web.comp.formats.writers.ItextSecuredPdfFormatWriter* (umieszczona w komentarzu XML), zamiast domyślnie wprowadzonej *pl.psnc.dlibra.web.comp.formats.writers.SecuredPdfFormatWriter*

Zmiana parametrów wymaga restartu kontenera Aplikacji Czytelnika (tj. Apache Tomcat).

Konfiguracja kolekcji głównej dla Aplikacji Czytelnika

Od wersji 5.0 administratorzy mają możliwość wskazania dowolnej kolekcji, spośród istniejących w bibliotece cyfrowej, która będzie spełniała rolę kolekcji głównej. Opcja ta idzie w parze z możliwością podłączenia wielu Aplikacji Czytelnika pod pojedynczy Serwer dLibra. Aby skonfigurować dla danej Aplikacji Czytelnika jej kolekcję główną wystarczy otworzyć do edycji plik */WEB-INF/web.xml*, w którym znajduje się następujący wpis (domyślnie w komentarzu XML):

```
<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>
```

Wpis należy "odkomentować" zaś w miejsce *[id of collection]* należy wrzucić numeryczne ID kolekcji (unikalne ID ustawiane w bazie danych kolekcji). Po wszystkim należy zrestartować środowisko serwletów (np. Tomcat).

Zmiany w formularzu "Kontakt"

Aby zmienić domyślne wartości kontaktowe w formularzu należy otworzyć do edycji pliki *WEBAPP_en.xml* (dla języka angielskiego) oraz *WEBAPP_pl.xml* (dla języka polskiego), z katalogu */WEB-INF/components/resources*. W plikach tych można znaleźć wpisy w postaci:

```
<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>
```

Łatwo zauważyć, iż poszczególne wartości dla elementów XML odpowiadają wartościom z formularza. Zmiany w tych plikach nie wymagają ponownego uruchomienia kontenera serwletów.

Inną rzeczą, o której warto pamiętać w kontekście formularza kontaktu, jest adres mail, na który wysyłane są wpisane przez użytkownika wiadomości. Adres ten można skonfigurować w pliku /WEB-INF/actions.xml, zmieniając wartość dla deklarowanej akcji *SendMailAction*:

```
<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>
```

Między znacznikami *value* można umieścić dowolną liczbę adresów mail oddzielonych średnikami lub przecinkami.

Ustawienia obsługi znaków wodnych (tylko dla dedykowanych wdrożeń z włączoną funkcją tagów publicznych)

Znaki wodne wyświetlane są dla **zabezpieczonych prezentacji JPG** (specjalny typ publikacji w dLibrze, który tworzy się poprzez wskazanie jako plik główny katalogu zawierającego pliki JPG). Odbywa się to za pomocą specjalnego appletu Java.

Znak wodny to obrazek, który jest nakładany na wyświetlaną treść prezentacji z efektem przezroczystości. Zalecamy wykorzystać grafikę w formacie PNG, który pozwala zapisać obraz z przezroczystym tłem. Obrazek ten jest zawsze powiększany tak, aby zajmował maksymalną powierzchnię. Nie jest jednak rozciągany do proporcji głównego obrazu, tylko wyśrodkowany. Można jednak uzyskać efekt znaku wodnego mniejszego niż główny obraz, poprzez dodanie przezroczystego obramowania o wybranej szerokości w grafice znaku wodnego.

Oprócz obrazka, w znaku wodnym może być automatycznie wstawiona nazwa zalogowanego użytkownika. Konfiguracja znaków wodnych znajduje się w pliku **WEB-INF/web.xml**, w sekcji poświęconej servletowi **watermarkServlet**. Można tam zdefiniować następujące parametry:

- `image.path` - sadaścieżka do obrazka, który ma być wyświetlany jako znak wodny
- `opacity` - siła efektu przezroczystości, musi być liczbą z przedziału (0, 1). 0 - znak niewidoczny, zupełnie przezroczysty, 1 - znak zupełnie przesłania treść
- `text.userName.position` - pozycja wskazuje lewy górny róg prostokąta, w którym będzie wpisana nazwa użytkownika. Pozycja jest podawana jako współrzędne w ramach obrazka znaku wodnego, przy czym punkt 0,0 to lewy górny róg obrazka.
- `text.userName.size` - wielkość prostokąta, w który będzie wpisana nazwa użytkownika, również w stosunku do rozmiaru znaku wodnego. Jeśli wielkość czcionki nie jest zdefiniowana, zostanie automatycznie ustawiona maksymalna wielkość, która pozwoli zmieścić nazwę w tym prostokącie.
- `text.userName.align` - położenie nazwy użytkownika w ramach zdefiniowanego prostokąta. Możliwe wartości: `left` (na lewo), `center` (na środku), `right` (na prawo).
- `text.userName.wordwrap` - wartość `true/false`, pozwala włączyć zawijanie tekstu w przypadku długich nazw użytkownika.

- `text.userName.font` - czcionka nazwy użytkownika. Dostępne czcionki są zależne od czcionek zainstalowanych na komputerze użytkownika, ale zwykle można wykorzystać czcionki Arial, Courier New, Monospaced, Verdana, Tahoma.
- `text.userName.font.style` - pozwala włączyć pogrubienie i pochylenie w nazwie użytkownika.
- `text.userName.font.color` - kolor nazwy użytkownika, w formacie szesnastkowym.
- `text.userName.font.size` - wielość czcionki. Jeśli nie jest zdefiniowana, zostanie wybrana maksymalna wielkość, przy której zmieści się cała nazwa.

Udostępnianie Mapy Strony robotom indeksującym

Mapa Strony(Sitemap) jest protokołem umożliwiającym robotom indeksującym łatwiejsze zgromadzenie informacji o najważniejszych stronach w Aplikacji Czytelnika. Aplikacja Czytelnika w ramach zadania okresowego **SitemapGeneratingTask** generuje bezwzględne adresy url do stron i zapisuje je według poniższych założeń:

- **/sitemap/** - Katalog, w którym zapisywane są wszystkie pliki zawierające adresy do najważniejszych stron Aplikacji Czytelnika. Pliki dzielone są w taki sposób, by nie przekroczyć 50 000 adresów.
- **sitemapindex.xml** - Plik, w którym agregowane są ścieżki do wszystkich plików znajdujących się w katalogu **/sitemap/**.
- **robots.txt** - Plik, który zawiera niezbędne informacje dla robotów indeksujących. Zadanie okresowe **SitemapGeneratingTask** dodaje do niego linię z bezwzględnym adresem url do pliku **sitemapindex.xml**. Np. **Sitemap: {domena}/sitemapindex.xml**.

Aby zmienić domyślne ustawienia generowania mapy strony należy wyedytować plik periodic.xml znajdujący się w katalogu /WEB-INF Aplikacji Czytelnika. Wpis dotyczący odpowiedniego zadania okresowego przedstawia się następująco:

```
<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>
```

W zadaniu można ustawić wartości następujących parametrów:

- urlParams - Nazwy RDF atrybutów metadanych, których wartości, w przypadku publikacji i wydań, zostaną dodane do adresu url. Dla przykładowej wartości parametru urlParams: Title;Creator możemy spodziewać się następującego wyniku w przypadku pozycji "Pan Tadeusz": {domena} /publication/100/pan-tadeusz-adam-mickiewicz
- overwriteRobotsFile - Informacja o tym czy zadanie okresowe **SitemapGeneratingTask** ma nadpisywać linię "**Sitemap**" w pliku robots.xml. Możliwe wartości to true lub false. W przypadku wartości ustawionej na false, należy pamietać aby dodać w pliku robots.txt wartość "**Sitemap**" ręcznie.

Obsługa publikacji linkujących

Publikacje linkujące pozwalają na umieszczenie w bibliotece cyfrowej informacji o obiekcie, który jest udostępniany przez jakiś inny serwis internetowy. W domyślnej konfiguracji, gdy użytkownik chce zobaczyć treść takiej publikacji, najpierw zobaczy planszę z informacją o zewnętrznej treści oraz docelowym linkiem, który musi wywołać. Jeśli nie chcemy, aby ta plansza była wyświetlana, tylko żeby użytkownik od razu został przekierowany na docelowy adres, należy dodać poniższą konfigurację w `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>
```

Aktywacja reCAPTCHA

Aby aktywować usługę google reCAPTCHA należy wejść na stronę https://www.google.com/recaptcha/, kliknąć przycisk "My reCAPTCHA" i po przekierowaniu zalogować się na googlowskie konto (konto jest wymagane do aktywacji usługi). Po zalogowaniu widok będzie przedstawiał wcześniej stworzone reCaptche oraz możliwość stworzenia kolejnej. Aby stworzyć nową reCaptche należy:

1. Przejść do arkusza "Register a new site".
2. W polu tekstowym label wpisać nazwę dla tworzonej reCaptchy.
3. Wybrać opcję "reCAPTCHA **v2**" oraz "Checkbox".
4. W polu "Domains" wpisać adres biblioteki, które będzie używała reCAPTCHA, np. "moja.biblioteka.pl".
5. Zaznaczyć checkbox "Accept the reCAPTCHA Terms of Service."
6. Potwierdzić przyciskiem "Register".
7. Po przekierowaniu wartości pól "Site key" oraz "Secret key" dodać do konfiguracji w pliku "WEB-INF/web.xml" w miejscach oznaczonych jako "siteKey" oraz "secretKey".
8. Usunąć znaki komentujące, tj. "<!--" oraz "-->".
9. Wyłączyć podstawowe CAPTCHA, wartość parametru captcha.enabled ustawić na "false".

Po wprowadzeniu zmian konfiguracja powinna wyglądać w następujący sposób(w miejscach oznaczonych jako "siteKey" oraz "secretKey" wpisane klucze ze strony google):

web.xml

```
<context-param>
    <description>Whether to display CAPTCHA on forms</description>
    <param-name>captcha.enabled</param-name>
    <param-value>false</param-value>
</context-param>
<context-param>
        <description>
            CAPTCHA settings:
                length - number of characters
                width - width of the result image
                height - height of the result image
                transformations (optional) - a comma-separated list of transformations
            Possible transformation values include 'noise', 'ripple', 'shadow' and 'block'.
        </description>
        <param-name>captcha.settings</param-name>
        <param-value>6;175;50;noise,shadow</param-value>
    </context-param>
    <context-param>
        <description>
            reCAPTCHA settings:
                site verify url - recaptcha endpoint
                site key - used in the HTML code your site serves to users.
                secret key - Used for communication between dlibra site and Google. Be sure to keep it a secret.
        </description>
        <param-name>recaptcha.settings</param-name>
        <param-value>https://www.google.com/recaptcha/api/siteverify;siteKey;secretKey</param-value>
    </context-param>
```