Evidencia publikačnej činnosti

Technická dokumentácia

Tím 16 - 2009/2010

Posledná revízia: 14/04/2010

Zodpovedné osoby:

Roman Táborský, Michal Maslíš

Cieľové osoby:

Roman Táborský

Michal Maslíš

Ladislav Clementis

Miroslav Mikluš

Michal Námešný

Branislav Lukáč

Cieľ dokumentu

Cieľom tohto dokumentu je dokumentovať systém epca.form z hľadiska

- Návrhu
- Implementácie

Táto dokumentácia popisuje štruktúru systému epca.form, základné myšlienky, ktoré sú v pozadí tohto systému a a implementačné detaily, ktoré nie sú zrejmé zo zdrojových kódov systému.

Tento dokument sa nevenuje úvodnej analýze problémovej oblasti a analýze dostupných prostriedkov. Dokumentácia týchto tém je dostupná na http://labss2.fiit.stuba.sk/TeamProject/2009/team16is-si/download/Projektova_dokumentacia.pdf.

Obsah

Návrh systému

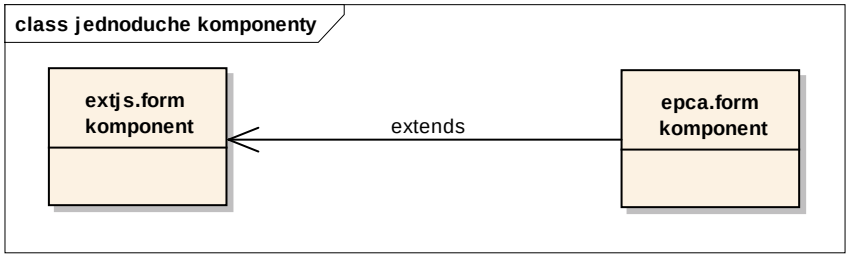
Typy komponentov

Systém využíva dva pohľady na komponenty. Jednoduché komponenty sú základné stavebné kamene formulárov a reprezentujú vizuálne prvky ako TextBox a ComboBox z knižnice ExtJS, ale sú obohatené o funkcionalitu ktorá ich umožňuje používať v spojení s formátmi MARC.

Zložené komponenty predstavujú pohľad na komponent, ktorý je poskladaný z iných komponentov. Najčastejší prípad predstavujú opakovateľné komponenty a komponenty epca.form.FieldSet ktoré zapuzdrujú zložené komponenty.

Jednoduché komponenty

Jednoduché komponenty sú realizované pomocou extjs.Extend mechanizmu, ktorý umožňuje rozšíriť ExtJS triedu o ďalšiu funkcionalitu.



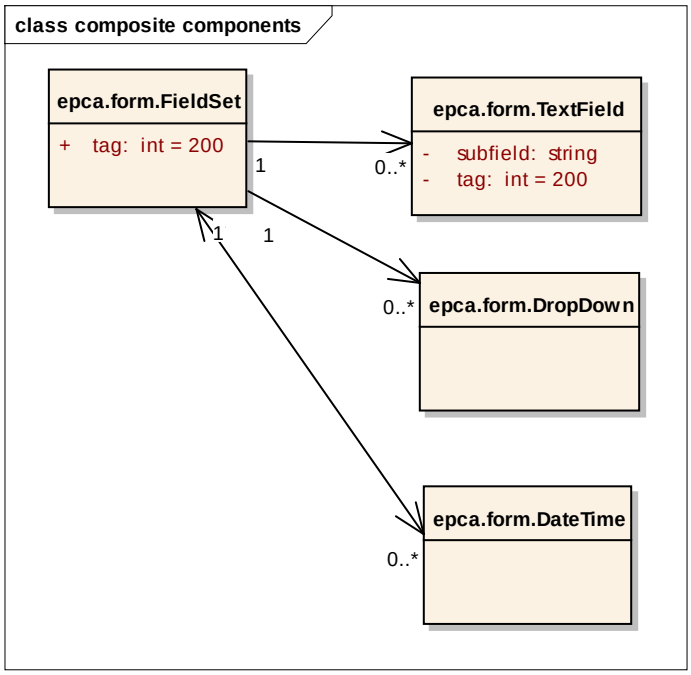
Obrázok 1 - Jednoduché komponenty

Tieto vlastnosti sú určené na naplnenie a získanie MARC hodnoty z komponentu, na podporu opakovateľnosti a na nastavenie dátových zdrojov pre čerpanie z databázy.

Pre všetky nastavenia komponentov viď. **Error! Reference source not found.**

Zložené komponenty

Pohľad na zložené komponenty je pohľad na jednotlivé tagy formátu UniMarc. Tento pohľad je zaujímavý hlavne z pohľadu dizajnéra formulárov, pretože umožňuje vkladať už predpripravené komponenty pre jednotlivé tagy a nie je nutné ich ručne tvoriť



Obrázok 2 - Zložené komponenty

Dátová štruktúra pre mapu tagov

Pre účely evidencie bibliografických záznamov a publikačnej činnosti je pri vytváraní zložitých komponentov najjednoduchšie vychádzať z reprezentácie dát v tejto oblasti, a to z formátu Unimarc/Marc21 a definovať zložené komponenty pre jeho jednotlivé tagy.

Požadovaná funkcionalita pre tento model je:

Zoznam tagov

Vstup - UNIMARC formát

Výstup - Zoznam všetkých tagov z daného záznamu, nepotrebujeme vedieť ich podpolia, dokonca ani všetky atribúty, ale musia byť: id, title, type

```
result = [ {
id : '100',
title: 'Všeobecné údaje spracovania#Všeobecná data zpracování',
type: 'Q',
repeatable : 'false' //resp. 0
},{
...
]
```

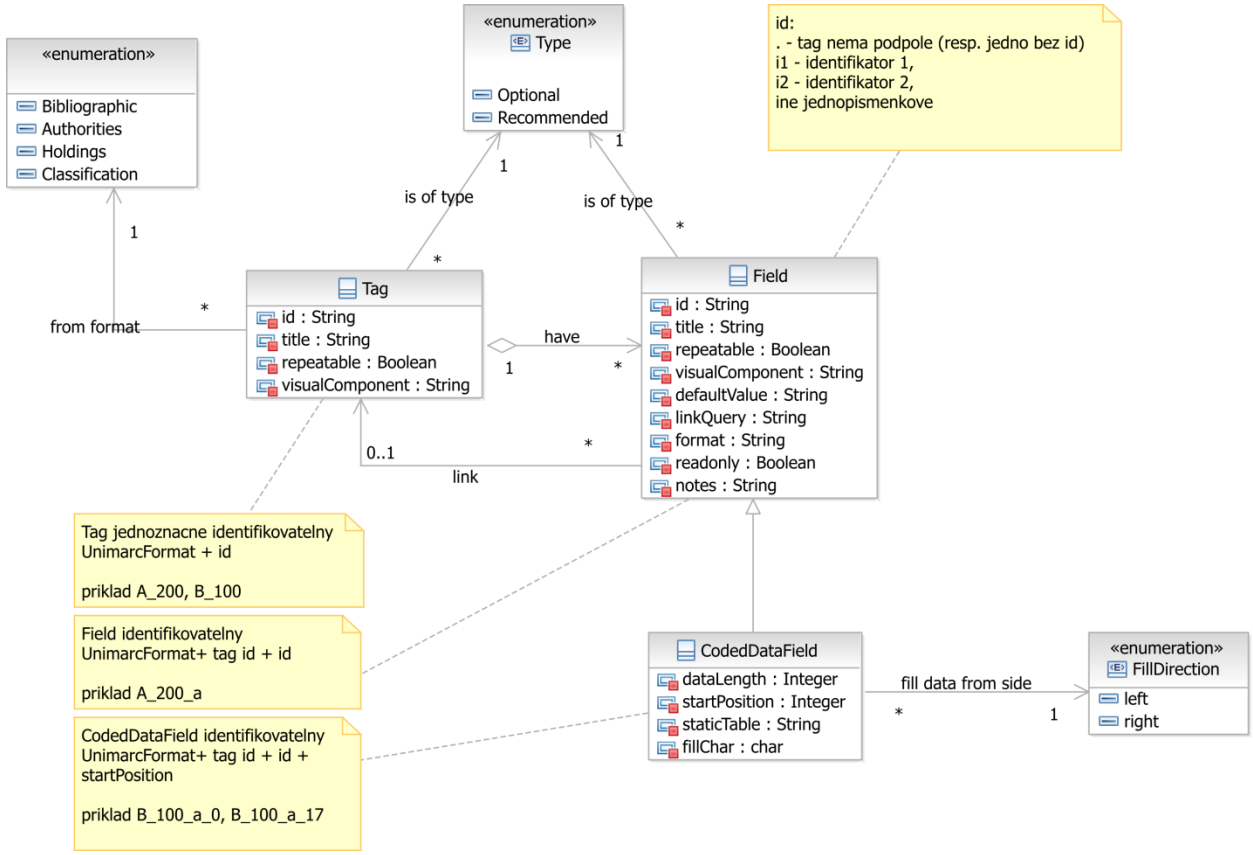
Výber tagu

Vstup - Unimarc formát, tág id (číslo tagu)

Výstup - Tág so všetkými podpoliami

```
result = {
id : '100',
title: 'Všeobecné údaje spracovania#Všeobecná data zpracování',
type: 'Q',
repeatable : 'false', //resp. 0
field: [{
id: 'a',
title: 'Dátum uloženia do súboru#Datum uložení do souboru'
startPosition: 0,
dataLength: 8,
},
{...},
...]
}
```

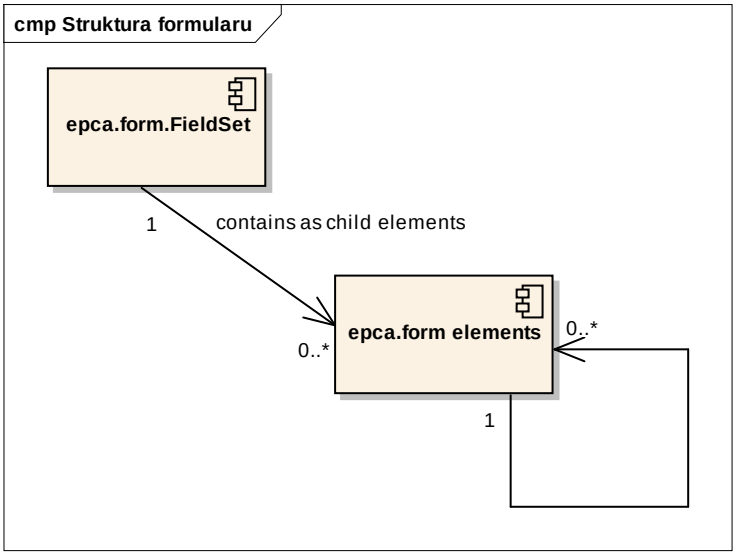
```
{
linkTag: 'B_200'
}
```



Obrázok 3- Dátový model pre mapu tagov

Štruktúra formuláru

Formulár vo výslednom produkte je realizovaný ako strom.



Obrázok 4 - Štruktúra formuláru

Na diagrame epca.form.FieldSet predstavuje hlavný kontajner pre celý formulár. Tento kontajner môže obsahovať ďalšie komponenty, či už kompozitné alebo jednoduché.

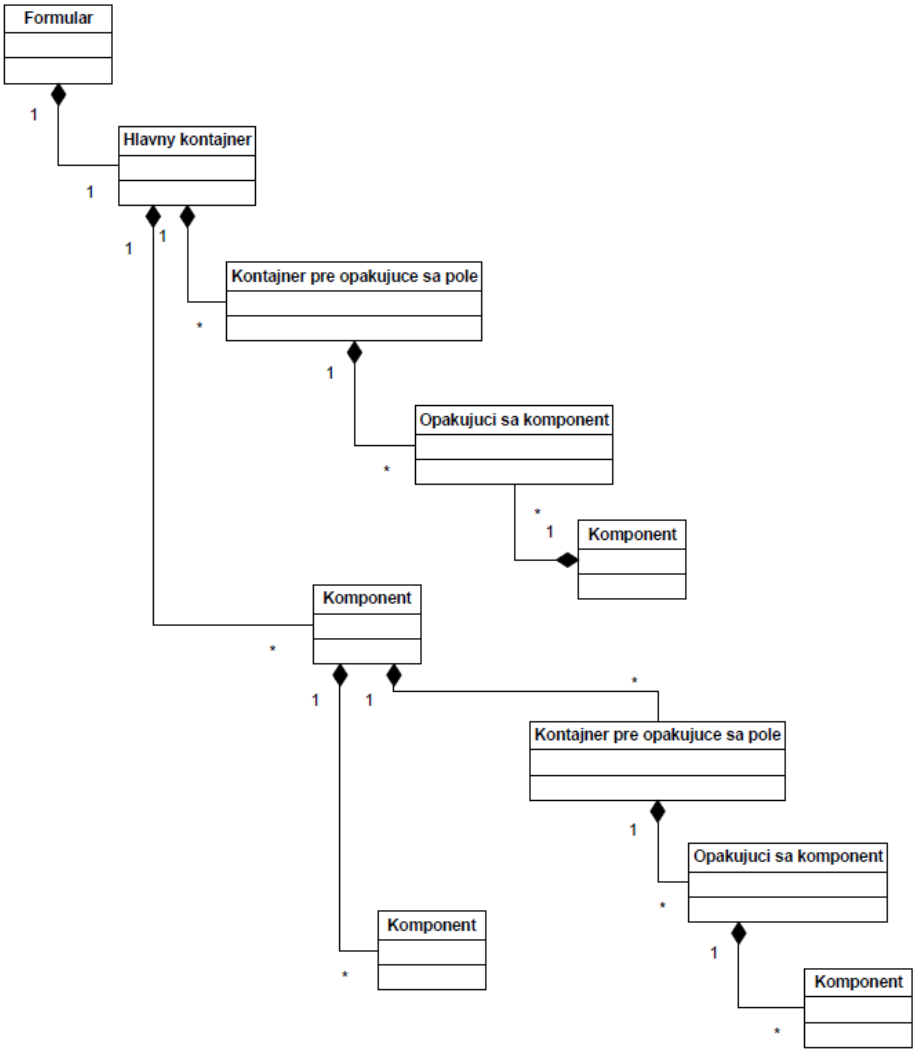
Koncepty použité pri návrhu systému

Stromová štruktúra

Pohľad na formulár ako na stromovú štruktúru umožňuje veľmi jednoduché programovanie funkcionality ako komunikácia medzi jednotlivými komponentmi a posielanie dát, keďže na väčšinu týchto procesov sa vieme pozrieť rekurzívnym spôsobom a tým umožniť prechod jednotlivých akcii nahor po komponentovom strome a zabezpečiť tak distribúciu do celého formuláru.

Táto funkcionalita je obsiahnutá v metódach getMarc a setMarc.

Tento koncept sa odvíja od pôvodného návrhu formulárovej štruktúry, ktorý sa ale mierne zmenil, vzhľadom na implementačné možnosti Javascriptu a ExtJS.



Obrázok 5 - Pôvodný model formulára

Opakovateľnosť

Vo formulároch, hlavne pri bibliografických záznamoch, sa objavujú prvky, ktoré sa v zázname objavujú viackrát. Z tohto dôvodu systém rieši opakovateľnosť na úrovni jednotlivých komponentov. Opakovať vieme jednotlivé grafické prvky na najnižšej úrovni rovnako ako aj zoskupenia prvkov.

Opakovateľné komponenty

Komponenty sme rozšírili o funkcionalitu opakovateľnosti (možnosť ich pridávať - kopírovať a odoberať). Tieto komponenty sa vyznačujú tým, že sa môžu vo formulári vyskytovať aj viac ako raz. Tieto komponenty sa na našej programovej úrovni vyznačujú tým, že názov tried má predponu „Repeatable". Sú to:

- epca.form.RepeatableField
- epca.form.RepeatableFieldSet (nevyužíva sa, namiesto neho epca.form.RepeatableEncap)
- epca.form.RepeatableEncap

Všade, kde sa využíva opakovateľnosť, sa využívajú práve tieto opakovateľné komponenty, aj keď môžu mať opakovateľnosť deaktivovanú. Opakovateľnosť týchto komponentov je po inicializácii predvolene deaktivovaná. Z tohto dôvodu majú všetky tieto komponenty dve funkcie:

- MakeRepeatable(): aktivuje opakovateľnosť
- MakeNonRepeatable(): deaktivuje opakovateľnosť

Ak chceme používať opakovateľnosť daného komponentu, musí byť:

1. Jeho opakovateľnosť zapnutá ešte pred vykreslením (vykreslia sa ikony plus, mínus)
2. Tento komponent vložený do enkapsulačného komponentu
3. Zavolaná funkcia render('main-ct') enkapsulačného komponentu (alebo nadkomponentu)

epca.form.RepeatableEncap

Ide o rozšírenie epca.form.RepeatableFieldSet, ktoré je rozšírením epca.form.FieldSet (ktoré je rozšírením Ext.Form.FieldSet). Jedná sa o enkapsulačný komponent, obsahujúci a manažujúci práve jeden opakovateľný komponent (a jeho klony - kópie).

Každý opakovateľný komponent je zaobalený v jednom enkapsulačnom komponente a každý enkapsulačný komponent obsahuje jeden opakovateľný komponent v jednej alebo viacerých inštanciách.

Enkapsulačný komponent má dve základné funkcie:

- addChild(Objekt1)
- removeChild(Objekt1)

Funkcia addChild(Objekt1) pridá do komponentu epca.form.RepeatableEncap daný „Objekt1“ ako položku FieldSet-u. Zároveň však opakovateľnému vkladanému komponentu „Objekt1“ nastaví vnútornú premennú „parent“ („rodič“) na enkapsulačný rodičovský komponent (na seba), kvôli spätnej referencii pri klonovaní a vymazávaní komponentov.

Funkcia removeChild(Objekt1) vymazáva „Objekt1“ z enkapsulačného komponentu. V prípade, že enkapsulačný komponent obsahuje poslednú položku, tento „Objekt1“, namiesto odstránenia zavolá funkciu opakovateľného komponentu „Objekt1“ Clear(), ktorá ho vynuluje. Tým je zabezpečené, že sa opakovateľný komponent nedá z nedbalosti používateľa natrvalo z enkapsulačného komponentu odstrániť, ale aspoň jeden sa v ňom vždy bude nachádzať (pri inicializácii tam samozrejme musí byť vložený, rieši sa run-time odoberanie klonov).

Funkcia Clear() opakovateľných komponentov je rekurzívna, tak že ak sa zavolá napríklad nad opakovateľným FieldSet-om, vynuluje všetky jeho komponenty. Ak je zavolaná nad Field-om, vymaže jeho obsah.

Ak chceme vytvoriť čisto enkapsulačný komponent, zavoláme:

enc = new epca.form.RepeatableFieldEncap().

epca.form.RepeatableField

Ide o rozšírenie epca.form.TextField (ktoré je rozšírením Ext.Form.TextField). Na rozdiel od epca.form.TextField je pridaná funkcionalita klonovania, aj keď pôvodná funkcionalita je zachovaná. Vizuálne sa epca.form.RepeatableField od epca.form.TextField líši tým, že má na pravej strane pridané dve ikony: plus a mínus.

Ikona plus po kliknutí vyvolá event, ktorý vezme danú inštanciu epca.form.RepeatableField (sám seba), vytvorí kópiu danej inštancie (sám seba naklonuje), vyvolá funkciu parent-a (rodiča, epca.form.RepeatableEncap) addChild() s parametrom odkazu na klon. Tento enkapsulačný komponent pomocou funkcie addChild() pridá klon medzi svoje položky.

Ikona mínus po kliknutí vyvolá event, ktorý vyvolá funkciu parent-a (rodiča, epca.form.RepeatableEncap) removeChild() s parametrom odkazu na seba. Enkapsulačný komponent ho potom z pomedzi svojich položiek odstráni. Ak je tento epca.form.RepeatableField v enkapsulačnom komponente ako posledný, enkapsulačný komponent ho iba vynuluje: zavolá jeho funkciu Clear(), ktorá mu nastaví hodnotu na prázdny reťazec.

epca.form.RepeatableFieldSet

Ak chceme použiť opakovateľný FieldSet (RepeatableFieldSet), vytvoríme: new epca.form.RepeatableFieldEncap(({title : 'Text'}), pričom „Text“ musí byť reťazec nenulovej dĺžky, a zapneme mu opakovateľnosť. Samozrejme, aj ten musí byť v enkapsulačnom komponente. Zaobalenie má potom tvar:

Enkapsulačný komponent -> Opakovateľný FieldSet -> Komponenty vo FieldSet-e

RepeatableFieldSet obsahuje iba neopakovateľné komponenty a prípadné enkapsulačné komponenty. Ak chceme v opakovateľnom FieldSete využívať napríklad RepeatableField, tento RepeatableField musí byť osobitne zaobalený tiež v neopakovateľnom enkapsulačnom komponente pod FieldSet-om. Zaobalenie má potom tvar:

Enkapsulačný komponent -> Opakovateľný FieldSet -> Komponenty vo FieldSet-e a Enkapsulačný komponent -> RepeatableField

Tento príklad je aj názorne vyobrazený v časti „Príklad použitia opakovateľnosti“.

RepeatableFieldSet sa vizuálne líši od RepeatableEncap tým, že má na hornej lište ikony plus a mínus. Tieto ikony sa správajú obdobne ako pri RepeatableField, až na to že pracujú nad celým RepeatableFieldSet-om. Ikona plus vyklonuje celý RepeatableFieldSet, zavolá funkciu addChild()

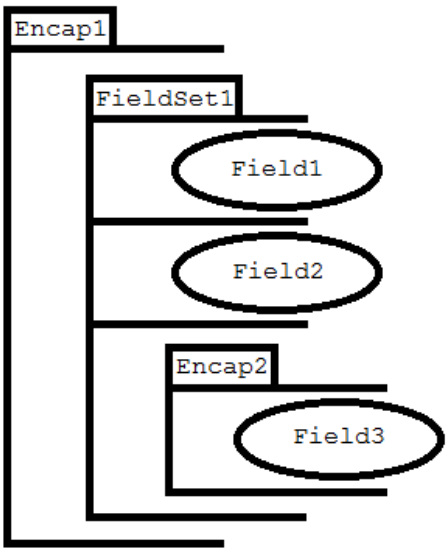
enkapsulačného komponentu s referenciou na tento klon. Ikona mínus vymaže celú inštanciu RepeatableFieldSet-u ak je ich v enkapsulačnom komponente viac ako jedna. Ak je tam posledná, volá rekurzívnu funkciu Clear().

Príklad použitia opakovateľnosti

Chceme napríklad vytvoriť opakovateľný FieldSet, obsahujúci dva neopakovateľné a jeden opakovateľný Field.

```
var Encap1 = new epca.form.RepeatableFieldEncap();              //Enkapsulacia pre opakovatelny FieldSet1
var FieldSet1 = new epca.form.RepeatableFieldEncap({title : 'Text'});    //Opakovatelny FieldSet1
var Encap2 = new epca.form.RepeatableFieldEncap();              //Enkapsulacia pre Field3

var Field1 = new epca.form.RepeatableField({fieldLabel : 'Field1'});      //neopakovatelny
var Field2 = new epca.form.RepeatableField({fieldLabel : 'Field2'});      //neopakovatelny
var Field3 = new epca.form.RepeatableField({fieldLabel : 'Field3'});      //opakovatelny

FieldSet1.MakeRepeatable();                    //nastavenie opakovatelnosti
Field3.MakeRepeatable();

Encap1.addChild(FieldSet1);
FieldSet1.addChild(Field1);
FieldSet1.addChild(Field2);
FieldSet1.addChild(Encap2);
Encap2.addChild(Field3);

Encap1.render('main-ct');
```

Náčrt tejto enkapsulácie je nasledovný:



Obrázok 6 - Náčrt opakovateľnosti

Výsledný formulár vo webovom prehliadači vyzerá nasledovne:

Obrázok 7 - Opakovateľnosť vo formulári

Prístup k databáze pomocou webových služieb

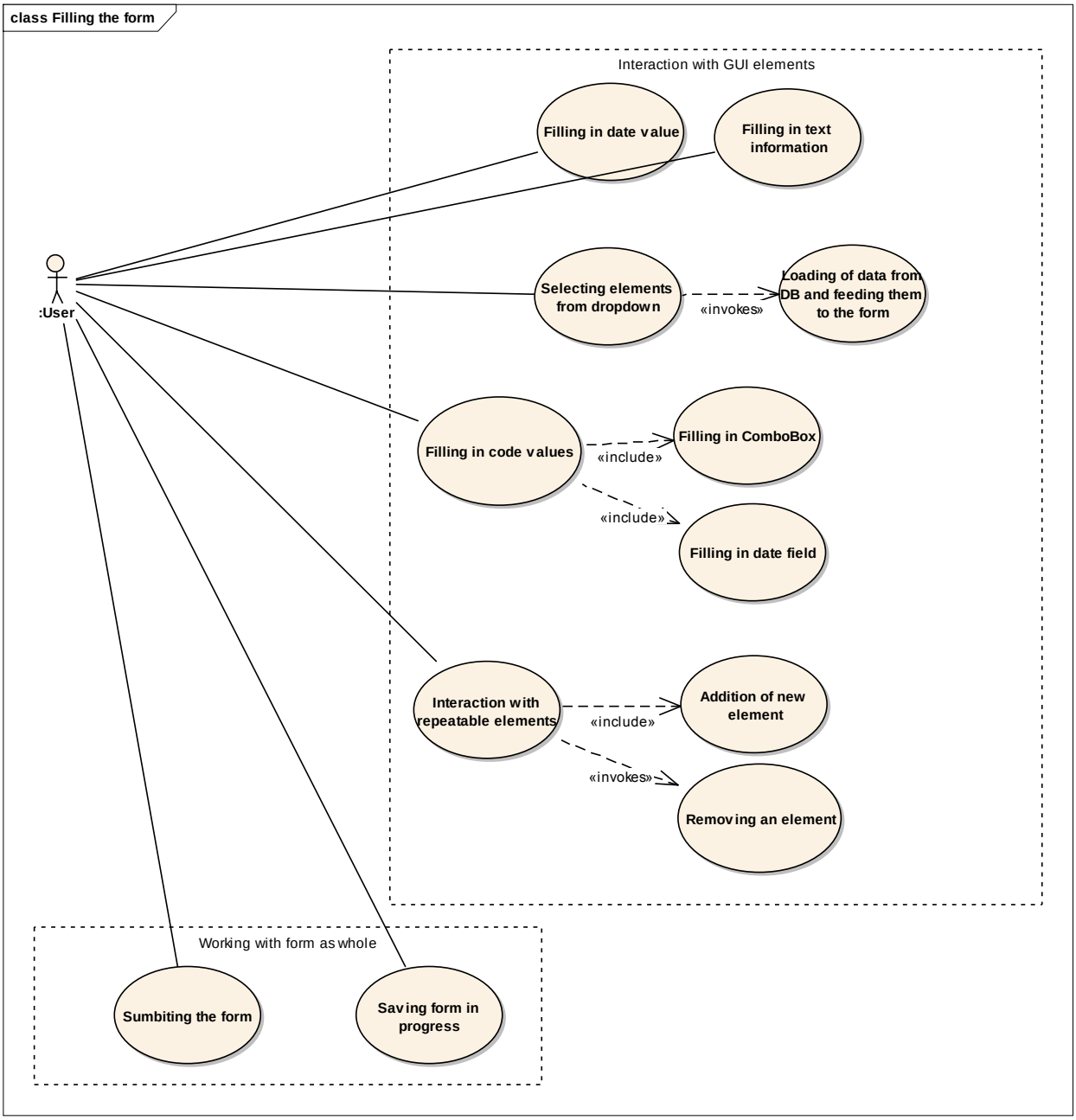
To, že sa k databáze pristupuje pomocou webových služieb, umožňuje jednoduché preprogramovanie prístupu k webovým službám, a tým umožniť ľahkú adaptáciu systému epca.form na iný účel.

Tieto webové služby slúžia na:

- Načítanie informácii z mapy tagov - epca.WsTagMap
- Načítavanie dynamických údajov vo formulároch - epca.marc_search_combobox
- Ukladanie a načítavanie dát z databázy – formuláre.

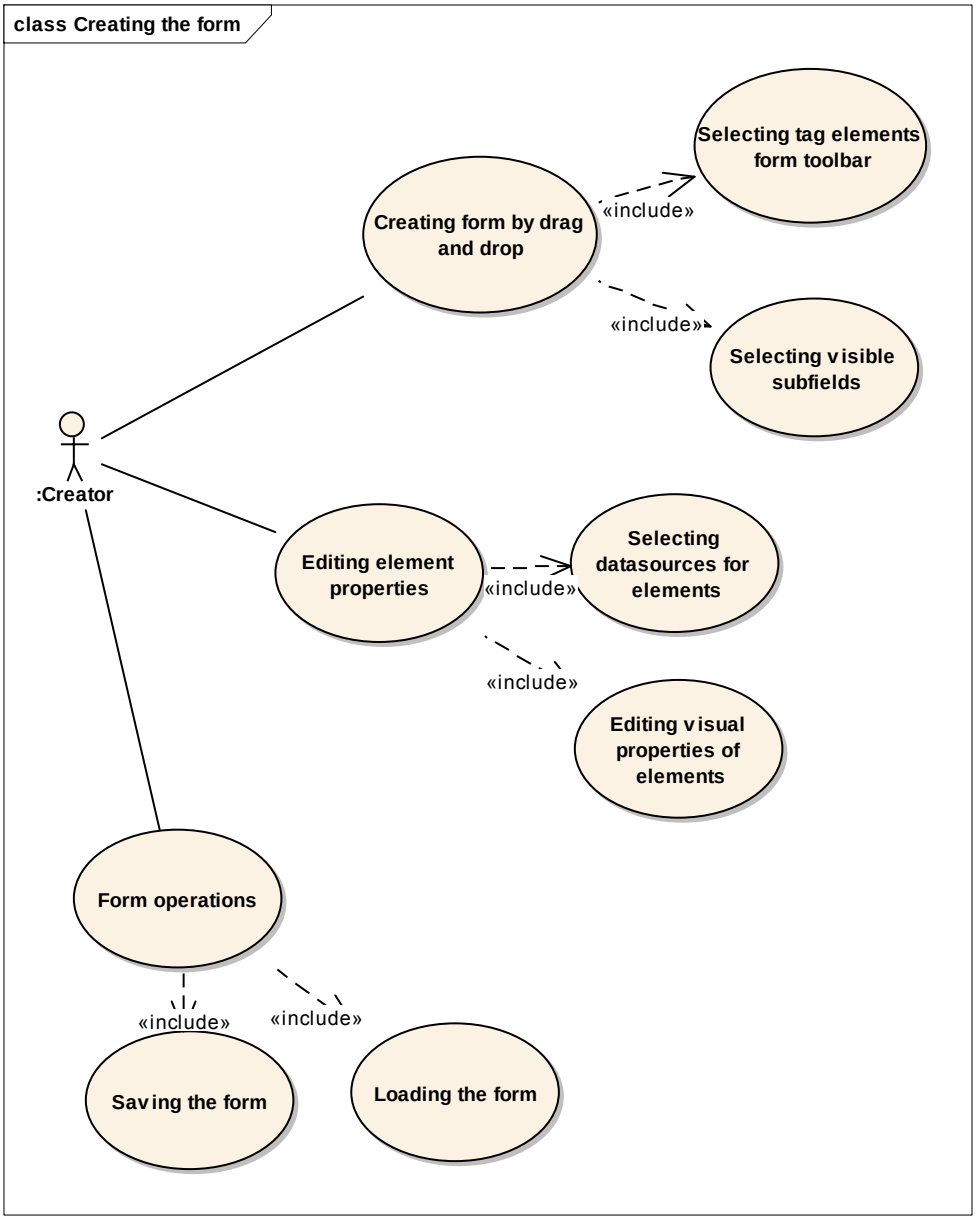
Použitie systému

Pre použitie systému sú dva základné pohľady – pohľad tvorcu formulára a pohľad človeka, čo formulár vypĺňa. Tieto dva pohľady sú dokumentované ako skupina prípadov použitia.



Obrázok 8 - Prípady použitia pre vypĺňanie formuláru

Prípady použitia pri vypĺňaní formuláru určujú aké grafické elementy sú nutné pri vykresľovaní hotového formuláru.



Obrázok 9 - Prípady použitia pri tvorbe formuláru

Prípady použitia pre tvorbu formuláru určujú aktivity, ktoré je nutné vykonávať v dizajnéri formulárov a zároveň hovoria o tom, aké vlastnosti je nutné mať možnosť nastaviť pre jednotlivé elementy vo formulári.

Vypĺňanie formuláru

Prípady použitia pri vypĺňaní formuláru sú v princípe dvoch druhov

- Interakcia s grafickými prvkami
    - Interakcia so základnými grafickými prvkami
    - Interakcia s opakovateľnými prvkami
- Interakcia s formulárom

Na základe týchto prípadov použitia sú identifikované základné grafické prvky

- TextBox
- DropDown s pripojením na databázu
- DateTime
- Panel

Funkčnosť týchto elementov je podrobne rozobraná v **Error! Reference source not found.** a **Error! Reference source not found.**.

Pre podporu práce s opakovateľnými prvkami je nutné vytvoriť grafické rozhranie, ktoré umožňuje pridávať a odoberať opakovateľné prvky.

Z toho vyplýva požiadavka na to, aby sa existujúce prvky obohatili o funkcionalitu opakovateľnosti. Vizuálne to znamená to, že dané prvky budú obohatené o tlačidlá + a -, ktoré budú realizovať funkčnosť pridávania a odoberania.

Tvorba formuláru

Prípady použitia uvedené v diagrame pre vypĺňanie formuláru budú realizované nástrojom s názvom **Error! Reference source not found.**. Tento nástroj bude realizovať funkčnosť vytvárania formulárov. Detailný popis tohto nástroja je v samostatnej kapitole **Error! Reference source not found.**.

Zoznam tried

Komponenty

epca.form.RepeatableFieldSet

Extends: epca.form.FieldSet

Táto trieda umožňuje vytvoriť fieldset, ktorý je možné opakovať. Do tohto fieldset-u je možné pridávať ďalšie komponenty. Fieldset obsahuje 2 tlačítka - plus a mínus. Po kliknutí na plus sa fieldset naklonuje, po kliknutí na mínus sa fieldset vymaže. V prípade, že je tento opakovateľný fieldset už iba jeden, po kliknutí na mínus sa nevymaže, ale iba vyprázdni.

Properties:

typ: String slúži na identifikáciu typu komponentu – Fieldset
Repeatable: boolean Ak je fieldset opakovateľný, jeho hodnota je true, ináč false
Parentencap: Component Obaľovací komponent

Methods:

Clear (): void Vyprázdni Fieldset
SetParentEncap (epca.form.RepeatableFieldEncap parentencap):void Nastavuje obaľovací komponent parentencap, aby k nemu mohol fieldset pristupovať
klonuj():void Vytvorí klon a priradí ho obaľovaciemu komponentu
afterRender (): void Vďaka tejto funkcii sa určí, čo sa zavolá po kliknutí na ikonky plus a mínus

epca.form.RepeatableField

Extends: epca.form.TextField

Táto trieda umožňuje vytvoriť textfield, ktorý je možné opakovať. Ak je textfield opakovateľný, je nutné obaliť ho komponentom RepeatableFieldEncap. RepeatableField obsahuje 2 tlačítka - plus a mínus. Po kliknutí na plus sa naklonuje, po kliknutí na mínus sa RepeatableField vymaže. V prípade, že je tento opakovateľný Repeatfield už iba jeden, po kliknutí na mínus sa nevymaže, ale iba vyprázdni.

Properties:

typ: String slúži na identifikáciu typu komponentu – Fieldset

Repeatable: boolean Ak je fieldset opakovateľný, jeho hodnota je true, ináč false
Parentencap: Component Obaľovací komponent

Methods:

MakeRepeatable(): void Funkcia nastaví RepeatableField na opakovateľný
MakeNonRepeatable(): void Funkcia nastaví RepeatableField na neopakovateľný
Clear (): void Vyprázdni Fieldset
SetParentEncap (epca.form.RepeatableFieldEncap parentencap):void Nastavuje obaľovací komponent parentencap, aby k nemu mohol fieldset pristupovať
klonuj():void Vytvorí klon a priradí ho obaľovaciemu komponentu
onIconPlus(Object e,Object icon):void Udalosť po kliknutí na ikonku plus
onIconMinus(Object e,Object icon): void Udalosť po kliknutí na ikonku mínus
getIconCt(Object el): Object vráti kontajner pre ikonky pre daný element
alignIcon(boolean isPlus): void zarovná ikonku podľa toho či je plus alebo mínus
afterRender ():void Vďaka tejto funkcii sa zobrazujú ikonky plus a mínus pri opakovateľnom fieldset-e
addIconMinus(Array iconCfg):void Pridá sa ikonka mínus. Jej vlastnosti sa nastavia podľa vstupujúcej konfigurácie
addIconPlus(Array iconCfg):void Pridá sa ikonka plus. Jej vlastnosti sa nastavia podľa vstupujúcej konfigurácie

epca.form.RepeatableFieldEncap

Extends: epca.form.RepeatableFieldSet

Táto trieda slúži na obalenie opakovateľných komponentov. Umožňuje pridávanie o odstraňovanie jednotlivých komponentov, ktoré obaľuje.

Methods:

addChild(Object child) Pridá dieťa
removeChild(Object child) odstráni dieťa

epca.marc_fieldset

Extends: Ext.form.FieldSet

Táto trieda umožňuje pracovať s MARC záznamami. Zabezpečuje napĺňanie fieldset-u na základe vstupujúceho MARC záznamu, vracia MARC záznam fieldset-u.

Properties:

tag: Number určuje príslušný MARC tag
field: String určuje príslušné pole daného tagu

Methods:

getMarc(): Object vracia Marc záznam
setMarc (Object marc) nastaví element podľa vstupujúceho Marc záznamu

epca.marc_textfield

Extends: Ext.form.TextField

Táto trieda umožňuje pracovať s MARC záznamami. Zabezpečuje napĺňanie textfield-u na základe vstupujúceho MARC záznamu, vracia MARC záznam textfield-u.

Properties:

tag: Number určuje príslušný MARC tag
field: String určuje príslušné pole daného tagu

Methods:

getMarc(): Object vracia Marc záznam
setMarc (Object marc) nastaví element podľa vstupujúceho Marc záznamu

epca.marc_search_combobox

Extends: Ext.form.ComboBox

Táto trieda umožňuje pracovať s MARC záznamami. Zabezpečuje napĺňanie combobox-u na základe vstupujúceho MARC záznamu, vracia MARC záznam combobox-u.

Properties:

tag: Number určuje príslušný MARC tag
field: String určuje príslušné pole daného tagu

Methods:

getMarc(): Object vracia Marc záznam
setMarc (Object marc) nastaví element podľa vstupujúceho Marc záznamu

epca.codeddata_datefield

Extends: Ext.form.DateField

Táto trieda umožňuje pracovať s MARC záznamami. Zabezpečuje napĺňanie datefield-u na základe vstupujúceho MARC záznamu, vracia MARC záznam datefield-u.

Properties:

tag: Number určuje príslušný MARC tag
field: String určuje príslušné pole daného tagu
position: Number pozícia kódovaného poľa
dataLength: Number dĺžka kódovaného poľa
dataFormat: String formát kódovaného poľa

Methods:

getCodedData ():String vracia Marc záznam
setCodedData (String codedData) nastaví element podľa vstupujúceho Marc záznamu

epca.codeddata_fieldset

Extends: epca.form.FieldSet

Táto trieda umožňuje pracovať s MARC záznamami kódovaných polí . Zabezpečuje napĺňanie fieldset-u na základe vstupujúceho MARC záznamu, vracia MARC záznam fieldset-u.

Properties:

tag: Number určuje príslušný MARC tag
field: String určuje príslušné pole daného tagu
dataLength: Number dĺžka kódovaného poľa

Methods:

getMarc(): Object vracia Marc záznam
setMarc (Object marc) nastaví element podľa vstupujúceho Marc záznamu

epca.codeddata_combobox

Extends: i3.ui.ComboboxST

Táto trieda umožňuje pracovať s MARC záznamami. Zabezpečuje napĺňanie combobox-u na základe vstupujúceho MARC záznamu, vracia MARC záznam combobox-u.

Properties:

tag: Number určuje príslušný MARC tag
field: String určuje príslušné pole daného tagu
position: Number pozícia kódovaného poľa
dataLength: Number dĺžka kódovaného poľa

Methods:

getCodedData ():String vracia Marc záznam
setCodedData (String codedData) nastaví element podľa vstupujúceho Marc záznamu

Evidencia

epca.evidence.EvidencePanel

Extends: Ext.Panel

Trieda na zobrazuje formulárov.

Methods:

initializeForm (): void Inicializácia

Dizajnér

epca.designer_tagtreepanel

Extends: Ext.tree.TreePanel

Methods:

processTagFields (Object tag): void Spracováva všetky polia vstupujúceho tagu. Využíva sa, po kliknutí na tag v ľavom paneli a následnom zobrazení v strednom paneli

epca.designer.DesignerPanel

Extends:Ext.Panel

Properties:

unFormat: String Unimarc formát. A: Authorities B: Bibliographic C: Classification H: Holdings
tagListTreePanel: Panel Stromový panel, ktorý obsahuje zoznam tagov
formTreePanel: Panel Stromový panel, ktorý slúži na návrh formulára

propertyGridPanel: Panel Panel vlastností. Po kliknutí na uzol stromu vo formTreePanel umožňuje nastaviť vlastnosti uzla ako napríklad názov, vizuálny komponent.
visualComponentComboBox: Ext.form.ComboBox combobox, ktorý je naplnený zoznamom dostupných našich komponentov

Methods:

initializeNewForm (String unFormat,String title): Inicializuje sa nový formulár, podľa vstupujúcich parametrov sa mu nastaví príslušný unimarc formát a titulok.
processTagList(Array tagList): void Metóda na spracovanie zoznamu tagov získaných cez webovú službu. Umiestni ich do tagListTreePanel panelu.
getTagListTreePanel ():Panel metóda vracia tagListTreePanel
getFormTreePanel():Panel metóda vracia formTreePanel
VytvorJson (Object tree_root):String Vytvorenie Json z panelu formTreePane, kde sa narhujú formuláre
getPropertyGridPanel():Panel metóda vracia propertGridPanel
updatePropertyGridPanel(Object selectedNode):void Po kliknutí na uzol stromu selectedNode vo formTreePanel aktualizuje panel propertyGridPanel, ktorý umožňuje nastaviť vlastnosti uzla ako napríklad názov, vizuálny komponent,...

Ďalšie

epca.WsTagMap

Táto trieda slúži na prácu s jednotlivými tagmi – zložené komponenty a na prácu s webovou službou, ktorá poskytuje tieto informácie.

Properties:

RepeatableValues:Array
opakovateľnosť
{
0: false,
1: true
}

TagType:Array
Typ tagu
{
R: 'mandatory',
Q: 'recommended',
O: 'optional'
}

Methods:

isRepeatable (Object value):boolean Kontrola, či value znamená opakovateľnosť value==0:return false value==1:return true
getTagList (Object unFormat,Object scope, Function callbackFunction) :void vráti tagList pre príslušný unimarc formát unFormat
processTagList (Object selectedRecord) :Object Funkcia spracuje zoznam tagov
getTagId (Object unFormat,Object tag) :String vracia ID tagu
getTag (Object unFormat, Object scope,Function callbackFunction,Object tag) : Object vracia tag
processTag:(Object selectedRecord) :Object spracuje tag pre záznam selected Record
getValue (Object recordId, Function processFunction,Object scope, Function callbackFunction) :Object Funkcia pre prácu s WS Parametre: Object recordId: id záznamu Function processFunction: funkcia, ktorá spracuje UNIMARC záznam do objektu Object scope: callbackFunction Function callbackFunction: funkcia, ktorá očakáva údaje z WS ako object

epca.MainPanel

Táto trieda definuje úvodnú obrazovku a jej jednotlivé formuláre.

Properties:

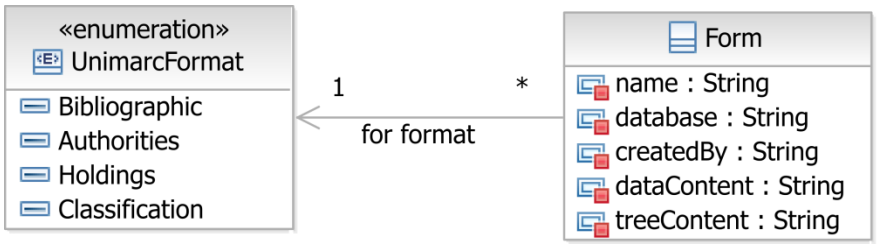
mainMenuPanel:Panel panel pre hlavné menu
buttonsPanel:Panel panel pre tlačítka
evidenceWizardPanel:Panel panel pre sprievodcu evidencie
designerWizardPanel:Panel panel pre sprievodcu dizajnéra
designerPanel:Panel panel pre dizajnér

Methods:

getMainMenuPanel ():Panel vráti mainMenuPanel
getButtonsPanel() : Panel vráti buttonsPanel
getEvidenceWizardPanel():Panel vráti evidenceWizardPanel
getDesignerWizardPanel():Panel vráti designerWizardPanel
getDesignerPanel():Panel vráti designerPanel

Epca.Form

Táto trieda definuje formulár pre EPCA.



Obrázok 10 - Dátový model pre formulár

Properties:

unFormat: Object Unimarc formát
title: Object Titulok
targetDb: Object databáza
treeContent: Object Stromový zoznam pre dizajnér, ktorý bude jednoduché načítať a upravovať v dizajnéri formulárov.
dataContent: Object Vygenerovaný formulár z treeContent, ktorý sa bude zobrazovať pre knihovníkov. Výhodou je, že nemusíme spätne parsovať tento formulár pri potrebe jeho úpravy.
createdBy:Object Referencia na objekt – rodiča

Methods:

getFormRecord(): Object vráti marc záznam formulára

Epca

Táto trieda definuje základné nastavenia, MARC formát a poskytuje prostriedky na prácu s týmito formátmi.

Properties:

UnFormat: Object Enum definuje UNIMARC formáty

Methods:

getMarcValue(Object object,Object tag, Object field) :Object vráti hodnotu poľa daného tagu pre daný objekt Object object – objekt, ktorý obsahuje tagy a ich polia Object Tag – tag, v ktorom sa nachádza pole field, ktorého hodnotu funkcia vracia Object Field – pole prislúchajúce tagu, ktoré funkcia vracia
setMarcValue(Object object, String value, Object tag, Object field) :Object nastaví marc záznam pre daný objekt Object object – objekt, ktorý obsahuje tagy a ich polia Object Tag – tag, v ktorom sa nachádza pole field, ktorého hodnotu funkcia mení Object Field – pole prislúchajúce tagu, ktorého hodnotu funkcia mení String value – nová hodnota
processMarcValue(Object object, Object property, Object newValue) :void Metóda v objekte 'object' atribútu 'property' priradí hodnotu 'newValue' Parametre:

Object object: objekt, do ktorého vloží údaj Object property: property, ktorej sa priradí newValue. Ak property neexistuje, vytvorí novú a priradí objekt. Ak existuje, vloží starú a novú hodnotu do poľa, ktorý priradí danému atribútu. Object newValue: nová hodnota
convertToObject(Object marc) :Object Skonvertuje marc záznam na objekt Parametre: Object marc: Marc záznam (pole stringov), string = tag + medzera + indikator1, indikator2 + medzera + podpolia oddelene i3.c.SF (na začiatku názov)
convertToMarc(Object object) :Object Skonvertuje objekt na marc záznam Object object: objekt, ktorý sa bude konvertovať
spracujSubtagy(Object tag) :Object spracovanie subtagov daného tagu

Epca.WsForms

Slúži na ukladanie vytvorených formulárov.

Properties:

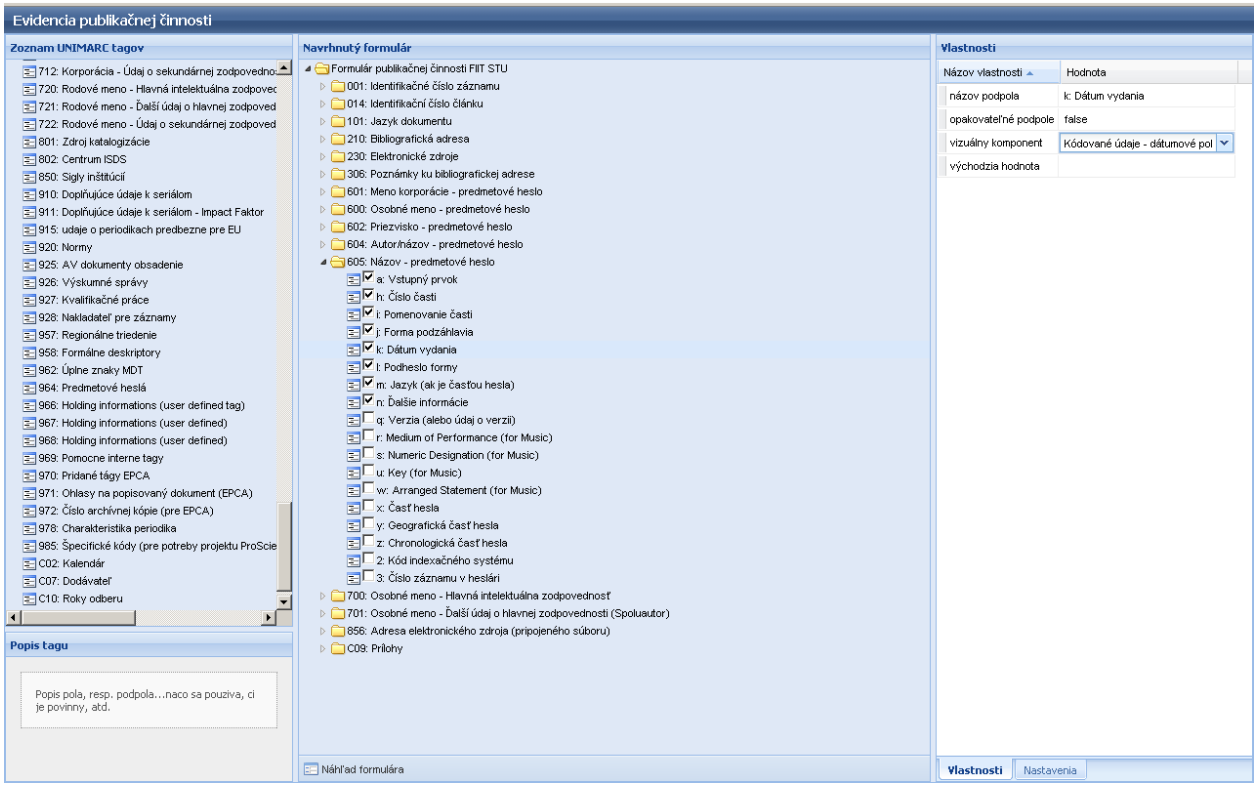
formsPrefix:String Prefix má tvar "EPCA_FORM_ "

Methods:

addForm(Object unFormat,Object formRecord) :void vkladá EPCA formulár

Dizajnér

Dizajnér je jedna z hlavných súčastí systému, keďže slúži na pohodlné vytváranie formulárov.



Obrázok 11 - Dizajnér

Dizajnér sa skladá z troch hlavných častí:

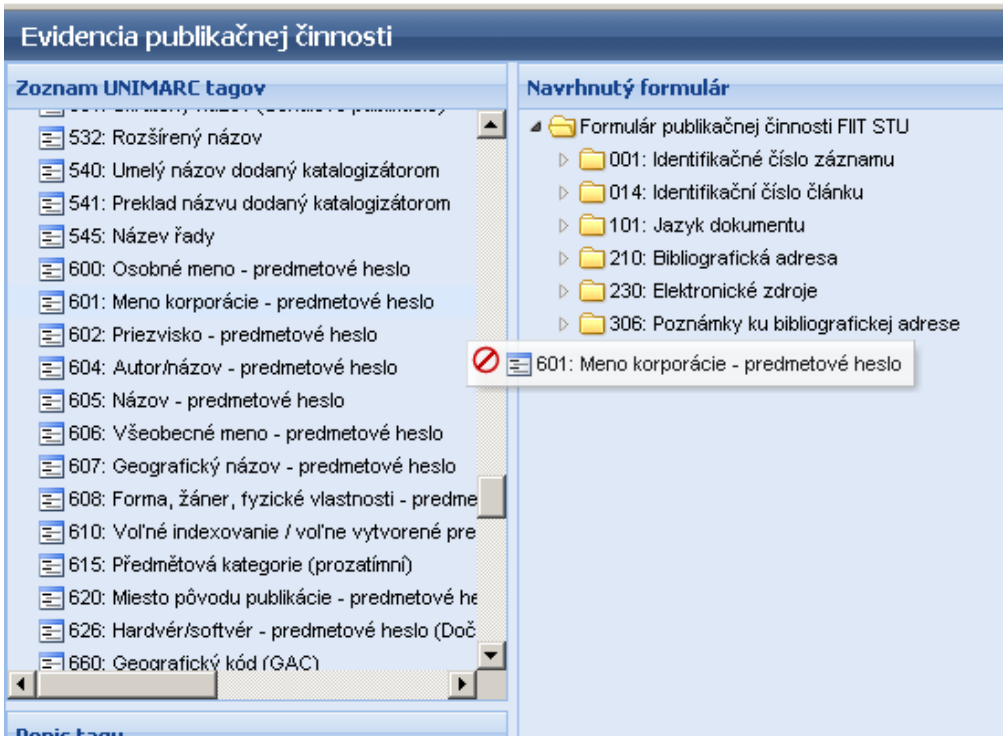
- Panel tagov
- Strom komponentov
- Panel vlastností

Tieto časti sú do výsledného panelu zložené v epca.designer.DesignerPanel.js.

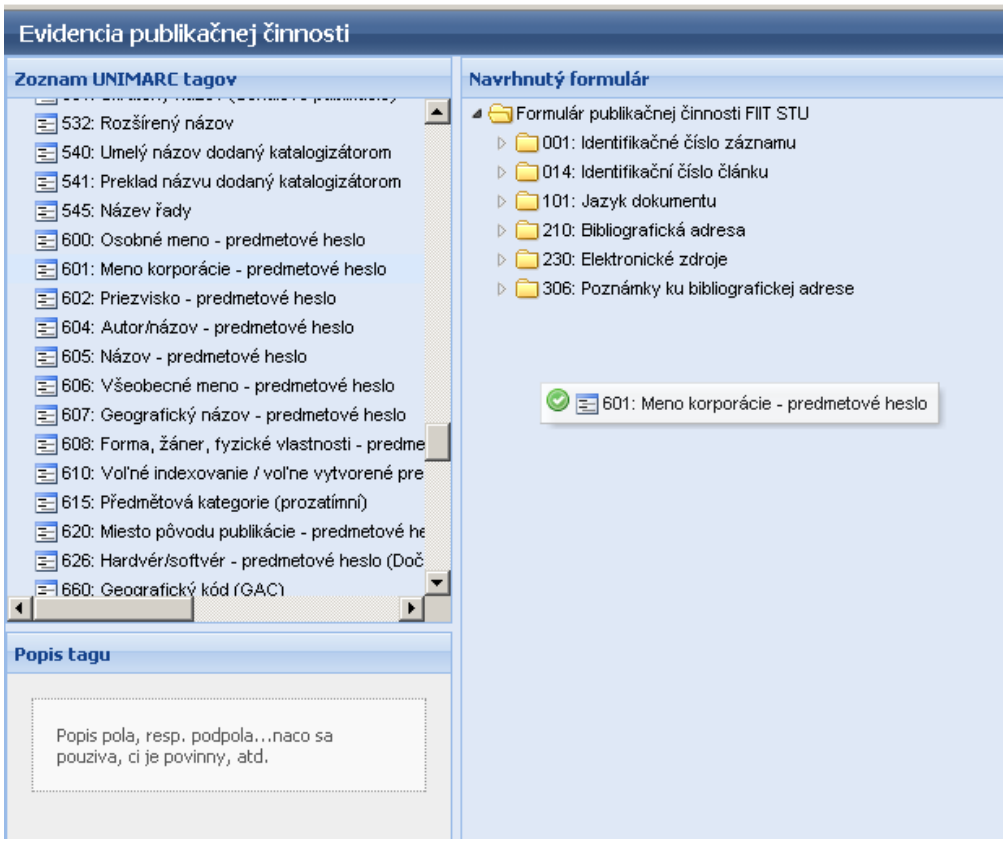
Panel tagov

V tomto paneli sa nachádzajú jednotlivé tagy, ktoré sú uložené v mape tagov. Na načítanie týchto tagov sa používa trieda epca.WsTagMap. Samotný panel je definovaný v epca.designer_tagtreepanel. Doťahovanie komponentov z webovej služby umožňuje rýchle preprogramovanie tohto panelu na iný typ formátu ako MARC respektíve iný účel ako EPCA.

Zobrazené sú jednotlivé tagy (zložené komponenty), ktorých podpolia sa zobrazia až po presunutí tagu do stredného panelu. Toto presunutie sa deje pomocou drag n drop.



Obrázok 12 - Ukážka drag n drop na nepovolenú pozíciu



Obrázok 13 - Ukážka drag n drop do stromu komponentov

Strom komponentov

Strom komponentov je časť dizajnéra, kde sa zobrazujú rozvinuté jednotlivé tagy a možnosť aktivovať jednotlivé podpolia. Po kliknutí na podpole sa v paneli vlastností objavia nastavenia pre dané konkrétne podpole.

Tento panel je definovaný v epca.designer.DesignerPanel.js ako formTreePanel v metóde getFormTreePanel.

V tomto paneli rozlišujeme tri typy uzlov:

- form
- tag
- field

Tieto uzly definujú jednotlivé zložky formulára. Form definuje formulár ako celok, tag definuje konkrétny kompozitný komponent a field definuje konkrétne podpole. Pre tieto typy uzlov sa líšia nastavenia dostupné v paneli vlastností.

Panel vlastností

V tomto paneli sa zobrazujú vlastnosti zvoleného poľa v strome komponentov. Možnosti, ktoré sú v paneli vlastností, sú definované v súbore property_config.json. Táto funkcionalita momentálne nie je naimplementovaná, jednotlivé vlastnosti pre rôzne typy uzlov sú definované v getPropertyGridPanel v súbore epca.designer.DesignerPanel.js.

Výpis modelového súboru property_config.json

```
[{
 id: 'form',
 properties: [
 {
    name: 'názov formulára',
    type: 'text',
    editable: true
 },
 {
    name: 'vytvoril',
    type: 'text',
    editable: false
 },
 {
    name: 'dátum vytvorenia',
    type: 'date',
    editable: false
 }
 ]
},
{
 id: 'tag',
 properties: [
 {
    name: 'názov tagu',
    type: 'text',
    editable: false
 },
 {
```

```
    name: 'povinný tag',
    type: 'boolean',
    editable: false
  }
  ]
},
{
  id: 'field',
  properties: [
  {
    name: 'podpole',
    type: 'text',
    editable: false
  },
  {
    name: 'názov podpola',
    type: 'text',
    editable: false
  },
  {
    name: 'opakovateľné podpole',
    type: 'boolean',
    editable: false
  },
  {
    name: 'vizuálny komponent',
    type: 'combo',
    data: [
      { id: 'epca.marc_textfield', text: 'MARC - textové pole' },
      { id: 'epca.marc_search_combobox', text: 'MARC - comboBox' },
      { id: 'epca.codeddata_combobox', text: 'Kódované údaje - comboBox' },
      { id: 'epca.codeddata_datefield', text: 'Kódované údaje - dátumové pole' }
    ],
    editable: true
  },
  {
    name: 'východzia hodnota',
    type: 'text',
    editable: true
  },
  {
    name: 'zdrojová tabuľka',
    type: 'text',
    defaultValue: 'UmbUnTablesD',
    editable: true
  },
  {
    name: 'vyhľadávací index',
    type: 'combo',
    data: [
      { id: '1', text: 'Autorské údaje', textENG: 'Personal name' },
      { id: '4', text: 'Názvové údaje', textENG: 'Title' },
      { id: '1016', text: 'Všetky polia', textENG: 'Any' },
      { id: '2426', text: 'Kód záznamu skrátený', textENG: 'Record no. short' },
      { id: '2467', text: 'Hlavný názov', textENG: 'Main title' },
      { id: '2475', text: 'Heslá', textENG: 'Subject term' }
    ],
    editable: true
  }
  ]
}
]
```

Marc Formát pre formulár

100 $aunFormat
200 $atitle

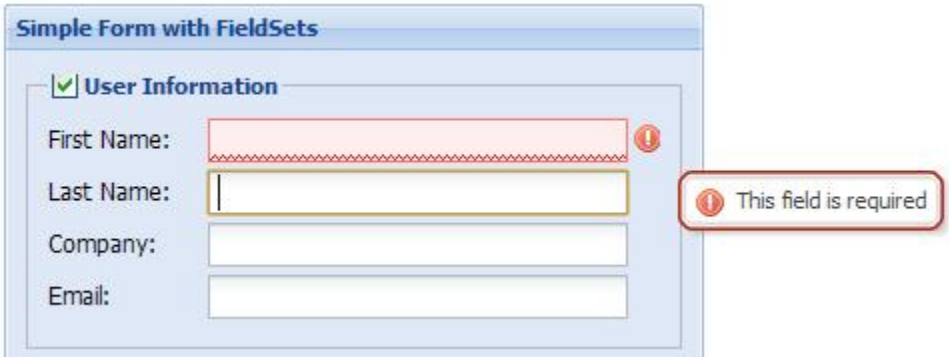
300 $atargetDb
400 $acreatedBy
500 $atreeContent
6XX $adataContent //začína 600, a postupne sa inkrementuje, aby sa dal spraviť update po jednotlivých tagoch, ak je formulár veľmi dlhý

Popis ExtJS komponentov a konceptov

V systéme sa používajú niektoré ExtJS komponenty ktoré sú využívané mimo ich štandardnej funkcionality, a preto je nutné sa im v dokumentácii venovať.

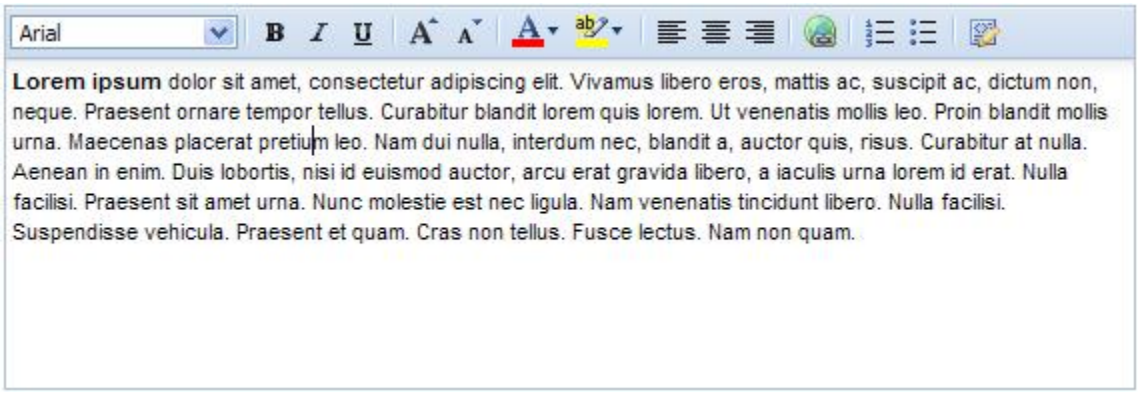
Prvky formulára

Štandardné prvky ako napríklad bežne používaný textbox má v ExtJS rozšírenú funkcionalitu, je tu pridaná možnosť validácie.



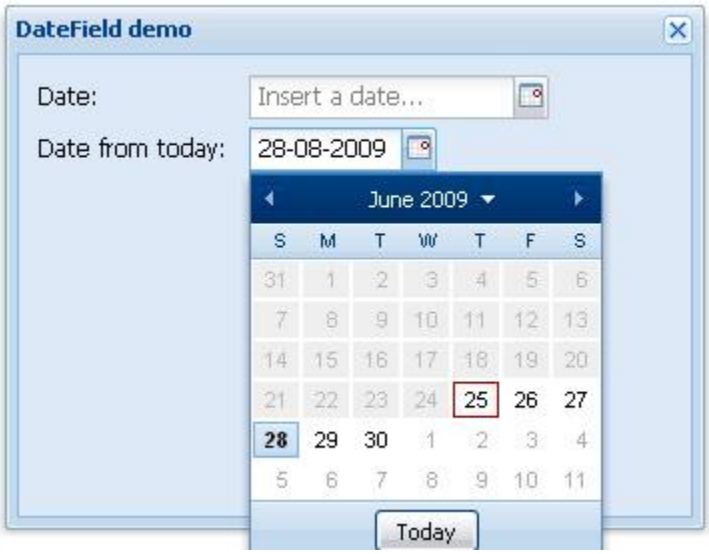
Obrázok 14 - príklad validácie

ExtJS pozná okrem textfieldu ďalšie štandardné prvky ako je radiobutton, checkbox. Pozná však aj prvok pre editáciu HTML textov, ktorý je tvorený jednoduchým DHTML editorom:



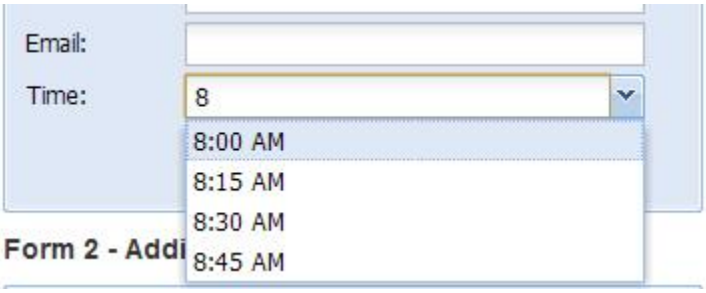
Obrázok 15 - editácia HTML textov

Ďalším zaujímavým prvkom je prvok pre editáciu dátumu a času:



Obrázok 16 – Datefield – editácia dátumu a času

ExtJS rozširuje funkcionalitu Combo boxu o možnosť automatického vyhľadávania počas písania:



Obrázok 17 – Combo box - automatické vyhľadávanie počas písania

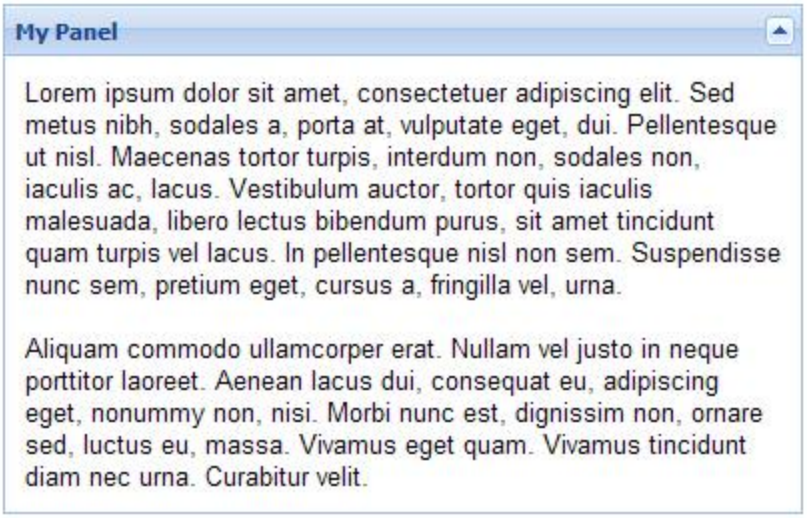
Dataview dokáže zobrazovať dáta, ktoré sú prijaté vo formáte JSON, podľa šablóny pre jednotlivé dátové položky. Datagrid umožňuje zobrazovať dáta vo vopred definovanej tabuľke. Je tu možnosť triedenia a filtrovania záznamov, zmena šírky pomocou Drag & drop.

Obrázok 18 – Datagrid – možnosť triedenia a filtrovania záznamov, zmena šírky pomocou Drag & drop

Pre zobrazenie dát stromových štruktúr sa využíva component Tree. Uzly sa dajú presúvať pomocou drag&drop, dá sa priamo editovať ich názov.

Obrázok 19 – Tree

Panel sa využíva na usporiadanie komponentov v okne aplikácie. Tento panel v sebe môže obsahovať ďalšie panely alebo komponenty, ktorých rozloženie sa dá nastaviť.



Obrázok 20 - Panel

Window je základným prvkom pri tvorbe dialógových aplikácií, má vlastnosti ako panel, no navyše obsahuje ovládacie prvky pre presúvanie, zmenu veľkosti, zatvorenie…

Obrázok 21 - Window

Použité technológie

Formát UNIMARC

UNIMARC je formát, resp. skupina formátov, slúžiaca hlavne na výmenu bibliografických údajov medzi knižnicami, ktorú má umožniť na medzinárodnej úrovni. Z toho vyplýva aj jeho predpona UNI ako univerzálny formát. Patrí do skupiny formátov typu MARC. UNIMARC definuje polia (tagy), indikátory, kódy podpolí pridelené záznamu v strojom čitateľnej štruktúre. V súčasnosti je nahradzovaný novším formátom MARC 21.

UNIMARC formáty:

- UNIMARC pre bibliografické údaje
- UNIMARC pre autority
- UNIMARC pre klasifikačné systémy
- UNIMARC pre holdingy

Základný formát záznamu

Každý záznam vo formáte UNIMARC sa skladá z jednotlivých polí s nasledovným formátom:

CXX		##		ÚDAJE

- CXX – kód poľa (tag), špecifikuje význam poľa, polia sa rozdeľujú do niekoľkých skupín podľa začiatočnej číslice.
- ## – dva indikátory, majú špecifický význam pre príslušný tag.
- údaje – údaje opisujúce dané pole, údaje sú rozdelené do podpolí s výnimkou kódovaných údajov, ktoré sú rozlíšené znakom ’$’ a jedným alfanumerickým znakom *a-z,0-9+, opäť význam jednotlivých podpolí závisí od poľa, v ktorom sa nachádzajú.

Špeciálnym prípadom polí sú kódované polia, ktoré nemajú podpolia, ale údaje predstavuje reťazec znakov definovanej dĺžky, kde každý znak na určitej pozícii má špeciálny význam.

Pre podrobnejšiu analýzu viz. http://labss2.fiit.stuba.sk/TeamProject/2009/team16is-si/download/Projektova_dokumentacia.pdf

ExtJS

Ext je JavaScript knižnica, používaná na budovanie interaktívnych webových aplikácií. Pôvodne bola táto technológia len ako rozšírenie knižnice YUI (Yahoo user interface). Grafické rozhranie knižnice obsahuje tzv. "widgets", pre tvorbu rozhrania webových aplikácií. Viac informácii o použití pri dokumentácii jednotlivých tried(Zoznam tried),informáciach o ExtJS(ExtJS) a informáciach o komponentoch (Opakovateľné komponenty, Dizajnér)

Dokumentácia ku knižnici je dostupná na http://www.extjs.com.

JSON- JavaScript Object Notation

JSON je formát slúžiaci na výmenu dát. Je založený na podmnožine programovacieho jazyka JavaScript. JSON je založený na dvoch štruktúrach:

1. množina párov názov/hodnota
2. utriedený zoznam hodnôt

Pre viac informácii viz. http://www.json.org.

Webové služby ARL

Protokol

Komunikačný protokol je založený na prenose dát protokolom HTTP/HTTPS, prostredníctvom metód GET alebo POST.

Formát požiadavky

`http://server/adresa_sluzby?method=xy&...`
namiesto method môže byť použitý alias "operation"

Parametre :

method : version | scan | search | update | command | display | hang | rights
username : login používateľa
CSPCHD : identifikácia aktuálneho sedenia (Ak nepodporuje prehliadač cookies)
OCSPCHD : identifikácia aktuálneho sedenia (Pre účely predania parametrov zo starého sedenia)
auth : autorizácia
nonce : Echo slova č.2 z predchádzajúcejpodpovede servera v danom sedení.
client_info : Nepovinný parameter informácií o klientskej aplikácií
pfmt : Formát výsledku (json,xml,text)
seqno : Poradové číslo požiadavky
ictx : Inštalačný kontext.
language : Číslo jazyka v kontexte ARL IPAC2 (1=slovenčina, 2=čeština, 3=angličtina, 4=nemčina, ..).
skin : Identifikácia skin v klientskej aplikácií
content_type : Interný parameter, ak je nastavený na 0 použije sa hlavička "Content-type: text/plain", inak predvolená "Content-type: application/x-download"
charset : Znaková sada výsledku - utf8, cp1250, latin2, default : utf8

Viac informácii v dokumentácii pre ARL.

Webové služby využívame na získavanie informácii z databázy Caché, kde sú uložené informácie o bibliografických entitách a zároveň aj informácie o štruktúre zložených komponentov (tagmap).

Perils of Code

Táto kapitola popisuje nie úplne zrejmé implementačné detaily a funkčnosť ktorá nie je štandardná – hacks and tweaks a upozorňuje na bežné chyby ktoré nastávali pri programovaní systému.

JSON a neexistujúce objekty

Pri práci so zdrojovými kódmi, čo sa týka perličiek pri kódovaní, najviac nás zaujala jedna samotná vlastnosť JavaScriptu. Ide o tú vlastnosť JavaScript Object Notation (JSON), že ak chceme používať nejaký atribút objektu, ktorý nemáme zadefinovaný, stačí ho prvý krát použiť (zapísať do neho, nie čítať), a odvtedy existuje

Teda napríklad:

```
object1 : {
    a : 'Hello ',
    c : '!'
};
object1.b = 'World';

document.write(object1.a);
document.write(object1.b);
document.write(object1.c);
```

Po spustení tohto skriptu sa na stránke vypíše "Hello World!". Táto vlastnosť, ako je vidno na príklade, nám umožňuje prácu s atribútom objektu (object1.b), ktorý nielenže nebol zadefinovaný, no dokonca ani inicializovaný. Týmto sa stáva kód flexibilnejší a voľnejší. Avšak po skúsenosti s programovaním v jazyku C# pod .NET frameworkom sme zvyknutí na to, že ak chceme používať atribút objektu, treba ho vopred vhodne zadefinovať. Vývojové prostredia podporujú hĺbkovú analýzu a kontrolu pred kompiláciou zdrojového kódu. Je to vynikajúca vec, ktorá nám pomáha vyhnúť sa mnohým nepríjemnostiam.

Ako znázorňuje príklad:

```
object1 : {
    prefixova_cast : 'Hello ',
    suffixova_cast : '!'
};
object1.stredna_cast = 'World';
document.write(object1.prefxiova_cast);
document.write(object1.stredna_cast);
document.write(object1.suffixova_cast);
```

Tento príklad nám nezobrazí nič (bez debug nástroja ani chybu). Dôvodom je preklep:

```
object1.prefIXova_cast
object1.prefXIova_cast
```

Kvôli tomu, že veľa vecí sa neošetruje vopred, že tu nie sú také pevné väzby a definície, môžu ľahko vzniknúť run-time chyby (principiálne alebo preklepové) ktorých zdroje sa ťažko vyhľadávajú.

Asynchrónnosť webových služieb

Problém ktorý vzniká pri používaní WS ARL je ten, že tieto funkcie sú asynchrónne.

```
SELECT_funkcia{
  result = poziadavka na WS.
   result =  nemáme;    // prečo? lebo asynchrónnosť
}
```

Riešením je použitie callback funkcie.

```
callback_function(result)    // sem príde result z WS, nevieme kedy
{
   tu môžeme spracovať výsledok z WS.
   Táto funkcia však nie je použiteľná pre viac typov požiadaviek
}
```

Preto je nutné volať WS takýmto spôsobom.

```
getValue: function(recordId, processFunction, scope, callbackFunction)
{
       i3.WS.getRecord({
                classn: i3.WS.getDfltUnTablesd(),
                fmt: 'LINEMARC',
                t001: recordId,
                success: function(selectedRecord, processFunction)
                {
                        //this = callbackFunction
                        this(processFunction(selectedRecord));
                },
                failure: function(errmsg, o)
                {
                        // spracovanie chyby; 'o' je objekt so ziskanym JSON
kodom
                        var errorMsg = String.format('{0}, {1}', errmsg, o);
                        Ext.Msg.alert('Error', errorMsg);
                },
                scope: callbackFunction.createDelegate(scope)
       }, processFunction.createDelegate(this));
}
```

Dynamické pridávanie atribútov

Objektu je možné dynamicky za behu pridať atribút bez toho, aby bol vopred zadefinovaný, čo sme využili napríklad pri nastavovaní vlastností pre jednotlivé uzly v strome (treeTagPanel) pomocou propertyGridPanelu:

```
selectedTreeNode.attributes["visualComponent"] =
propertyGrid.record.data.value;
```

Vytváranie vlastných komponentov

Vďaka frameworku Ext JS sme jednoducho dokázali vytvárať vlastne znovupoužiteľné komponenty nasledovným spôsobom:

```
Ext.ns('epca.form');
epca.form.SearchComboBox = Ext.extend(Ext.form.ComboBox, {
   constructor: function(config){
      ...
   },
   ...
});
// registrácia komponentu

Ext.reg('epca.marc_search_combobox', epca.form.SearchComboBox);
```

a potom môžeme už tento komponent pouziť kdekoľvek takto:

```
items:
[
   {
                        xtype: 'epca.marc_search_combobox',
                        fieldLabel: 'Priezvisko autora',
                        tag: '200',
                        field: 'b',
                        db: 'UmbUnAuth',
                        fmt: 'UNA_BASIC',
                        width: 400
   },
   ...
]
```