PROIECTARE ORIENTATĂ SPRE DOMENIU ŞI DEZVOLTARE ORIENTATĂ SPRE MASHUP BAZATE PE OPEN SOURCE JAVA METAFRAMEWORK PENTRU DEZVOLTAREA DE SOFTWARE WEB PRAGMATICĂ, FIABILĂ ŞI SIGURĂ:

TEHNOLOGII RICH CLIENT PENTRU APLICAȚII WEB

http://www.ict-romulus.eu, http://romulus.ici.ro

Rezumat: Această lucrare prezintă tehnologii Web de tip Rich Client, securitate şi fiabilitate şi prototipul tehnologiilor emergente de portal, acoperind şi subiecte de cercetare teoretice. Scopul cercetării îl reprezintă utilizarea tehnologiilor "client side" (pentru dezvoltarea de aplicații şi a tehnologiilor de scripting în combinație cu tehnologii Java.

Cuvinte Cheie: Rich Client Application, securitate, fiabilitate, scripting, Java, tehnologii emergente de portal

Abstract: This paper presents Rich Client Web technologies, security and reliability and prototype of emerging portal technologies, covering research and theoretic topics. The purpose of this research is the using of client side technologies for the development of applications and scripting technologies in combination with Java technologies.

Keywords: Rich Client Application, security, fiability, scripting, Java, emerging portal technologies

1. Introducere

Tehnologiile de scripting, tehnologii client side şi server side au fost, pe piață, pentru o lungă perioadă de timp, dar s-au bucurat de importanță şi atenție diferite. În zilele noastre, tehnologiile client side şi de scripting atrag un interes mai mare pentru că acestea permit dezvoltarea unui alt tip de aplicații foarte dificil sau chiar imposibil de a fi dezvoltat numai cu tehnologii server side.

Rich Internet Client (RIC) este o paradigmă relativ nouă în dezvoltarea de aplicații web care fac parte din aplicații client, îmbunătățind astfel dramatic posibilitățile de interacțiune a utilizatorului cu aplicația şi reducând consumul de bandă şi puterea de calcul a server-ului. De asemenea, un subiect interesant este faptul că oferă utilizatorilor posibilitatea de a crea aplicații bazate pe compunerea de aplicații simple care comunică între ele. Mecanismul de bază pentru acest lucru este portlet-ul de intercomunicare, care se obține prin mecanismele prevăzute de versiunea 2.0 a standardului de portlet-uri.

Componentele "Client side" ale unei aplicații trebuie să fie dezvoltate în alt limbaj decât un limbaj de descriere al documentului (HTML). JavaScript a fost prezent în browserele web de mai mulți ani, dar pentru o lungă perioadă de timp, utilizarea acestuia a fost scăzută şi a fost chiar considerată ca fiind o practică "rea". Odată cu apariția clienților puternici şi a dezvoltării de biblioteci (Prototype, jQuery, ExtJS, Dojo, ...), care asigură servicii de nivel mult mai ridicat decât cele prevăzute de JavaScript "simplu" şi ascunde unele dintre problemele sale de bază (în

principal datorate implementărilor diferite şi, uneori, incompatibile oferite de browsere) s-a produs o explozie de aplicaţii şi de site-uri prin intermediul acestei tehnologii RIC.

Popularitatea RIC şi JavaScript are, ca şi o consecinţă, printre altele, un interes tot mai mare în potenţialul tehnologiilor scripting pentru dezvoltarea de aplicaţii, precum şi posibilitatea de a dezvolta aplicaţii care rulează în întregime în browser-ul web al clientului.

Fiecare secţiune include trimiteri la prototipuri dezvoltate în legătură cu tehnicile şi framework-urile prezentate. Selecţia şi dezvoltarea acestor prototipuri au fost ghidate de un dublu scop:

- să demonstreze viabilitatea practică a abordărilor propuse;
- să identifice dificultăţile şi problemele potenţiale ale implementării.

Bazat pe problemele şi dificultăţile întâmpinate, va fi posibil să se propună soluţii care vor fi abordate ulterior, în scopul de a îmbunătăţi productivitatea dezvoltatorilor prin utilizarea acestor tehnologii.

Majoritatea prototipurilor au fost dezvoltate ca portlet-uri sau ca modificare a portlet-urilor actuale, utilizând capacităţile specifice ale Liferay. Există, de asemenea, prototipuri scrise ca aplicaţii JEE, folosind Roma şi suportul acordat pentru aplicaţii CRUD.

2. Analiza componentelor exclusiv client side

Cum Ajax devine, în mod rapid, cea mai bună soluţie pentru dezvoltarea unei noi generaţii de aplicaţii web extrem de interactive şi bogate în funcţionalităţi, - Rich Internet Applications (RIA) - un număr tot mai mare de dezvoltatori şi companii includ Ajax în aplicaţiile web ale acestora. Cu toate acestea, complexul cod JavaScript necesar pentru a sprijini mai multe browsere şi pentru a integra diversele componente, a condus la necesitatea unui framework adecvat de dezvoltare, care să abordeze aceste probleme. În prezent, un număr semnificativ de framework-uri JavaScript (AJAX) - ambele open source si comerciale - sunt disponibile:

- Dojo, [Echo2], int [ExtJS1], jQuery [JQu1], Midori, MochiKit, mootools [MoT1], Prototype, Pyjamas, Gooxdoo, Rialto, Rico, YUI [YUI1] - din categoria open source.
- Backbase [BaB1], Bindows, JackBe [JBe1] sau TIBCO General Interface – din categoria comercială.

Echo [Echo2] este un framework open-source pentru dezvoltarea de aplicaţii Web “bogate” (Rich client), care a fost creat de compania NextApp. Pentru informaţii suplimentare vizitaţi pagina Web Echo [Echo2].

Ext [ExtJS1] este o bibliotecă JavaScript pentru construirea de aplicaţii web interactive folosind tehnici precum AJAX, DHTML şi scripting DOM. Toate aceste caracteristici sunt explicate în detaliu pe pagina web Ext [ExtJS1].

jQuery este o bibliotecă JavaScript de dimensiuni mici (15KB), axată pe interacţiunea dintre JavaScript şi HTML. A fost lansat în ianuarie 2006, la New York BarCamp, de John Resig [JQu1], şi a crescut, în popularitate, foarte rapid (în conformitate cu Google Trends). Acesta este construit în jurul Selectorilor CSS (care oferă mecanisme puternice pentru selecţia elementelor) şi proiectat cu “concizie, comoditate, viteză şi stabilitate ca factorii de conducere”.

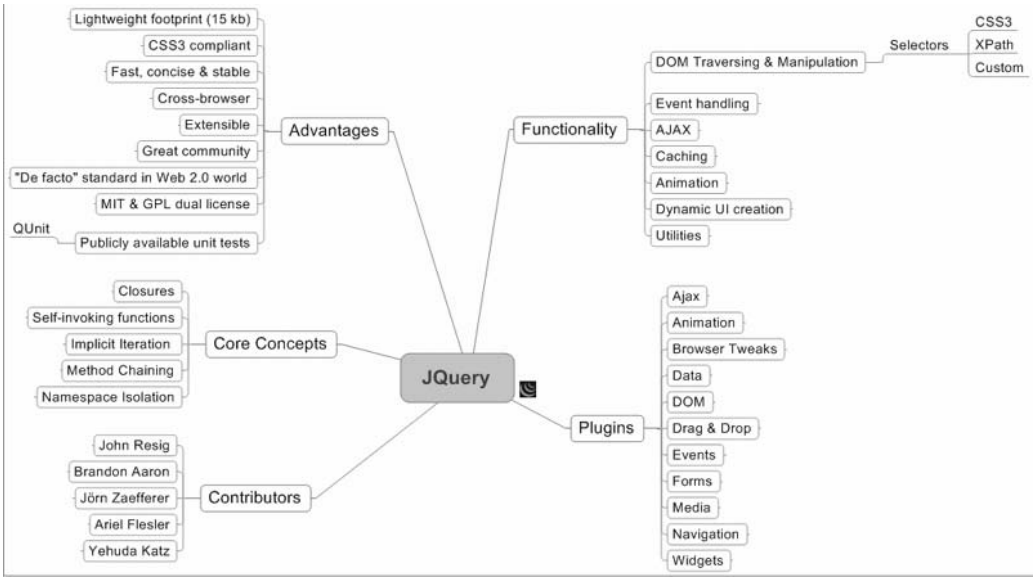

Figura 1

jQuery este unul din primele framework JavaScript "cross" browser-e: sunt acceptate IE 6.0 +, FF 2 +, Safari 2.0 +, Opera 9.0 +. De asemenea, este compatibil cu toate versiunile de CSS-uri. jQuery simplifică sarcinile legate de interfață web, cum ar fi:

- documente „orizontale" de tip HTML;
- tratarea evenimentelor;
- animații;
- adăugarea de interacțiuni AJAX la paginile web;

Operațiunea jQuery este concepută ca o operațiune în două etape, care implică:

- găsirea unui / unor element(e) HTML;
- manevrarea elementului (elor) HTML.

Mecanismul de selecție a jQuery este destul de puternic şi are asemănări cu selectoarele CSS. Selecția se poate face pe baza atributelor de identitate, clase, nume de elemente şi combinațiile de mai sus. Selecțiile pot fi reduse, ceea ce permite selectarea elementelor după poziție într-o ierarhie, într-o matrice, cu valori ale atributelor, în funcție de tipul din formular etc. Selecția este posibilă, atât pentru elementele individuale cât şi pentru seturi de elemente. Există, de asemenea, funcțiile care traversează structura arborescentă **DOM** (Document Object Model).

Pentru manipularea unor elemente există o gamă largă de posibilități disponibile, inclusiv atributul achiziție / setare (getting/setting) HTML, manevrarea codului HTML, manevrarea CSS-ului, prelucrarea de eveniment sau aplicarea de efecte, cum ar fi ascunderea, afişarea, "decolorarea" şi alte animații.

Biblioteca oferă, de asemenea, un set de funcții de utilitate comună, legate de browser-ul de informații, manevrarea obiectelor şi a vectorilor sau manipularea şirurilor sau manevrării URL-urilor.

Obiectele jQuery au "încastrate" toate funcționalitățile şi funcțiile întorc obiecte jQuery, permițând astfel un mecanism de înlănțuire puternic. "Looping" (buclarea) automat este o cerință a metodelor de jQuery, care trebuie să fie efectuate în toate elementele DOM din vector (array) (eliminând astfel necesitatea ca dezvoltatorul să rescrie codul de repetare) [JQB1]. Aceste două caracteristici fac destul de eficientă scrierea de cod JavaScript folosind jQuery sau chiar scrierea unei funcții nou construite jQuery peste acestea, este destul de eficientă.

jQuery este construit în jurul unei **arhitecturi** bottom-up, cu trei componente principale:

nucleu (core), plugin-uri şi jQueryUI.

Core JQuery (de bază), furnizează mecanismele de bază pentru a răspunde nevoilor comune în dezvoltarea web, cum ar fi [LjQ1]:

- accesarea părților unei pagini, prin intermediul unui sistem puternic de selecție (după cum s-a menționat mai sus);
- modificarea aspectului unei pagini prin intermediul unor "manipulări" CSS, oferind un mecanism uniform în browserele Web;
- alterarea conținutului unei pagini, inclusiv eliminarea şi inserarea unor porțiuni de cod HTML, schimbarea de imagini sau procesarea de liste;
- răspuns la interacțiunile utilizatorului cu o pagină, intersectându-se o mare varietate de evenimente şi asigurând mecanisme de prelucrare adecvate;
- adăugarea animației la o pagină; biblioteca jQuery facilitează acest lucru prin furnizarea unei serii de efecte, precum şi a unui set de instrumente pentru noi abilități (crafting);
- preluarea informațiilor de la server prin intermediul cererilor AJAX, eliminând complexitatea browser-elor specifice;
- simplificarea sarcinilor comune JavaScript.

Extensiile jQuery sunt create prin intermediul unor plugin-uri, care sunt construite înaintea funcționalității de bază jQuery. Lista de plugin-uri disponibile este destul de mare] [JQP1, acoperind o gamă largă de categorii, cum ar fi un eveniment, formulare, tabele, navigație, widget-uri şi altele. Plugin-urile sunt scrise atât de dezvoltatorii de bază cât şi de membrii comunității.

jQueryUI [JQU1] este bibliotecă de nivel superior, ascunzând interacțiuni jQuery de nivel inferior şi furnizând widget-uri şi componente pentru dezvoltarea de aplicații web extrem de interactive. Funcțiile din bibliotecă facilitează interacțiuni diferite realizate cu mouse-ul, cum ar fi drag & drop, selectare, sortare, redimensionare şi alte efecte de administrare (widget-uri diferite, cum ar fi armonici, cursoare sau tab-uri). Eforturile actuale sunt îndreptate către oferirea unei interfețe tematice.

Exhibit [EXH1] este un framework web scris în JavaScript, care poate fi folosit pentru a afişa date. Nu este un framework de uz general, în schimb, este axat pe afişarea datelor în mai multe formate, cum ar fi tabele interactive, hăți, grafice, etc. Este un framework pe trei nivele, şi anume este implementat în diverse JavaScript, CSS şi fişiere imagine. Ca o particularitate, este disponibil la un URL public de unde oricine poate da referințe în paginile sale HTML. Autorii Exhibit nu au nevoie să descarce niciun software, iar vizitatorii Exhibit nu au nevoie să îşi instaleze nicio extensie a browser-ului. Codul sursă Exhibit este public şi orice parte a sa poate fi suprascrisă prin scrierea de cod JavaScript suplimentar şi definiții CSS, după includerea codului Exhibit.

Pentru a utiliza Exhibit trebuie să fie furnizate două elemente: datele (de exemplu, un simplu fişier de date) şi prezentarea (un fişier HTML), care precizează modul în care datele ar trebui să fie afişate.

Datele în sine trebuie să fie în format JSON şi amplasarea acestuia trebuie să fie specificată prin intermediul unui tag (etichetă) de legătură punctând către fişierul JS care conține structura de date. Într-un caz simplu, link-ul poate indica un fişier JS static, dar poate, de asemenea, să puncteze către un URL care generează dinamic datele JSON.

Pentru a specifica modul în care datele urmează să fie prezentate, trebuie să fie adăugate unele proprietăți speciale la etichetele HTML. Arhitectura Exhibit este compusă, în principal, din stratul de date (în partea de jos) şi stratul interfață cu utilizatorul (în partea de sus).

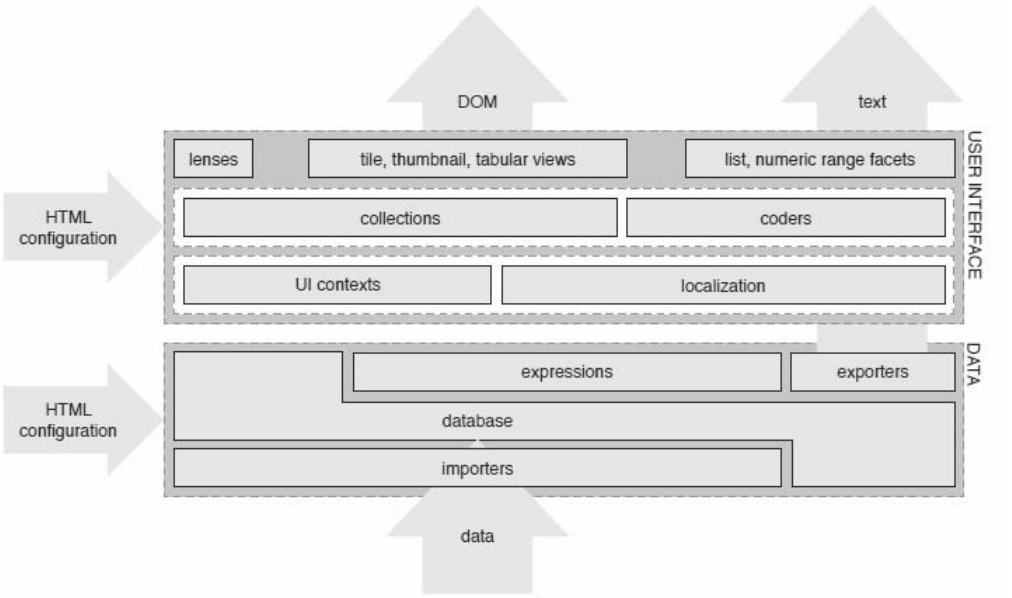

Figura 2

Stratul de date constă în patru sub-straturi: baza de date, parser-ul şi evaluatorul expresiilor de limbaj, importatorii, exportatorii.

Interfață utilizator este formată din trei componente:

- contextele UI şi resursele de localizare - care stochează setările de prezentare pentru restul stratului de interfață utilizator.
- colecții şi codări - care sunt componente ce nu se afişează pe ecran, dar determină ce date widget ar trebui să se afişeze şi cum să se afişeze.
- Widgets - care efectuează afişarea reală şi suportă interacțiunile.

Există mai multe puncte de extensibilitate: mai multe vizualizări pot fi adăugate şi mai mulți exportatori pot fi înregistrați. Panoul Browse poate fi, de asemenea, prelungit cu fațete, care nu doar listează alegerile lor, ci se specializează pentru proprietățile pe care sunt configurate, cum ar fi indicarea unui calendar pentru o proprietate de tip dată.

Componenta de localizare încapsulează resurse UI localizate, inclusiv şiruri de caractere (text), imagini, stiluri şi chiar scheme (machete).

Tehnologii Widget

Tehnologiile widget se bucură de multă atenție în ultimii ani şi au apărut o mulțime de platforme (Google, Netvibes – Rezultate obținute pe forum, Yahoo, Mac, Opera, etc.). S-a constatat că widget-urile sunt o tehnologie importantă pentru a crea sinergii (acțiune simultană pentru un scop comun) în rândul diferitelor sarcini şi pachetelor de lucrări din Romulus. În special, widget-urile au fost considerate foarte potrivite pentru dezvoltarea mashup-ului portalului, în cazul în care acestea au fost folosite ca bază pentru un tip de funcționalitate de export şi de import a portalului.

3. Dezvoltarea de aplicații personalizate, prin intermediul portlet-urilor slab cuplate

Întrucât unul dintre obiectivele lui Romulus este acela de a spori productivitatea cu tehnologii Enterprise Portal, această secțiune se concentrează pe modul în care tehnologia portlet-ului poate fi utilizată pentru dezvoltarea de aplicații personalizate.

Un portlet este o componentă web care procesează solicitările şi generează un conţinut dinamic [PoS1]. Conţinutul generat de un portlet este, de asemenea, numit fragment (de exemplu, HTML, XHTML, WML) şi poate fi cumulat cu alte fragmente pentru a forma un document complet. Un portlet poate avea aceeaşi funcţionalitate ca orice aplicaţie web. În forma sa originală, portlet-urile au fost considerate ca entităţi izolate, dar este necesar să se ia în considerare relaţiile, mai noi, dintre portlet-uri.

Conceptul de portlet-uri slab cuplate se referă la capacitatea de două portlet-uri de a schimba informaţii, dar fără a-şi pierde independenţa şi fără a depinde reciproc, asfel încât ele să nu poată lucra separat. Acest model nu are nevoie ca dezvoltatorii de portlet-uri să ştie reciproc, la momentul dezvoltării acestora, despre munca fiecăruia (reducând astfel complexitatea dezvoltării şi creşterea productivităţii). Conexiunile între portlet-uri sunt stabilite în timpul rulării. În acest fel, utilizatorii pot crea aplicaţii mai mari, compuse, chiar fără nicio altă dezvoltare. Dezvoltarea de portlet-uri slab cuplate impune disponibilitatea unui mecanism de rulare pentru portlet-ul de comunicare.

Mecanisme de comunicare inter portlet

În timp ce este posibil pentru dezvoltatori să proiecteze şi să pună în aplicare mecanisme de comunicare personalizate pentru a comunica aplicaţiilor opţiunea de "cuplare slabă", platforma JavaEE oferă metode standardizate, ca parte a specificaţiilor (caietului de sarcini) pentru portlet. Prin utilizarea unei metode standard, dezvoltatorul economiseşte timpul preţios de dezvoltare şi scade curba de învăţare pentru utilizatori.

Versiunea 1.0 [PoS1] a portlet-ului standard (JSR-168) nu a inclus suport pentru comunicare între portlet-uri. Aceasta este una dintre limitările principale ale JSR-168 şi forţează dezvoltatorii să realizeze comunicaţia prin mijloace non-standard, creând metode diferite şi adesea incompatibile pentru a comunica între portlet-uri.

Versiunea 2.0 [PoS2] a portlet-ului standard (JSR-286), lansat în iunie 2008, a fost realizată de IBM şi susţinută de către vânzătorii de principalelor portal-uri, inclusiv Liferay. Acesta rămâne compatibil şi adaugă caracteristici noi pe baza experienţei acumulate. Una dintre cele mai importante noi caracteristici este suportul pentru comunicare între portlet-uri, precizând două metode:

- parametrii prezentaţi public;
- evenimente.

Noua versiune a specificaţiilor (conform caietului de sarcini) defineşte, de asemenea, metodele standard de servire a resurselor (inclusiv a datelor binare şi solicitărilor AJAX) şi de filtrare (într-un mod similar cu servlets).

Toate aceste mecanisme de comunicare pot fi folosite pentru a dezvolta portlet-uri "slab cuplate" care pot fi combinate pentru forma aplicaţii complete (fully featured) pe partea de client. Următoarele secţiuni explică modul în care fiecare dintre aceste mecanisme lucrează şi descriu circumstanţele în care fiecare dintre ele ar trebui să fie utilizate.

Parametrii publici pentru comunicarea între portlet-uri

Parametrii publici de prezentare reprezintă cea mai simplă metodă de comunicare standard între portlet-uri şi permite utilizarea în comun a parametrilor ceruţi între diferite portlet-uri. Un dezvoltator poate declara o listă de parametri publici (ale căror nume sunt **namespaced** pentru a evita conflictele de denumire) pentru o aplicaţie portlet din portlet.xml. Portlet-urile trebuie să înregistreze (în fişierul (dosarul) descriptor **portlet.xml**) parametrii făcuţi publici pe care doresc să-i citească şi care sunt disponibili în întreg ciclul de viaţă al portlet-ului (processAction, processEvent, render şi serveResource).

Portlet-urile pot citi parametrii făcuţi publici prin utilizarea unor metode specifice, dar cum parametrii sunt amestecaţi cu parametri obişnuiţi, ei pot fi citiţi, de asemenea, prin metodele

uzuale. Parametrii făcuți publici pot fi chiar eliminați de un portlet. Figura 3 prezintă un exemplu de parametri făcuți publici lucrând pe un URL ca http://foo.com/my/path?_1_paramA=a&_2_paramB=b&_2_paramC=c&_prp_paramX=y, amestecând mai mulți parametri obişnuiți cu un parametru făcut public. Container va furniza pentru fiecare portlet parametrii adecvați.

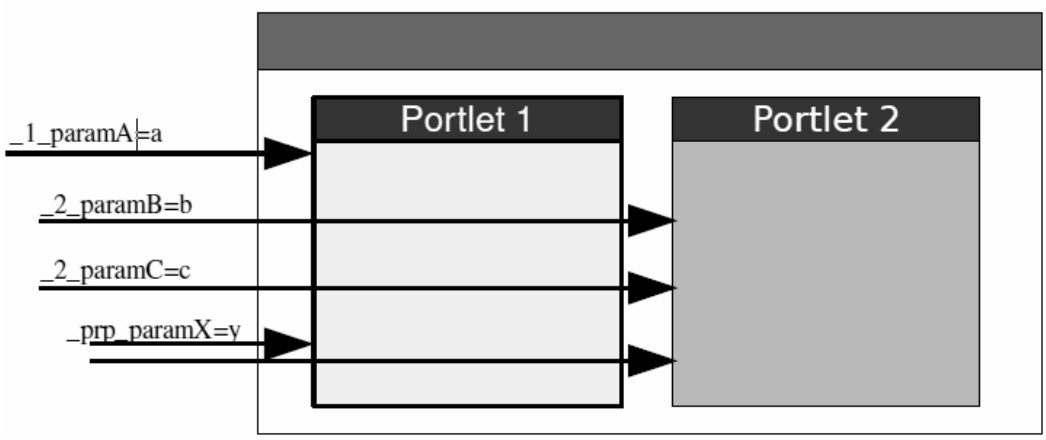

Figura 3

Evenimente pentru comunicare între portlet-uri

Mecanismul de evenimente este o metodă foarte puternică de comunicare între portlet-uri, decuplate. Ea se bazează pe un model de producător - ascultător:

- un portlet-ul generează un eveniment;
- zero sau mai multe portlet-uri pot fi ascultate şi acționează la acesta (inclusiv declanşarea de evenimente noi).

În plus, containerul (recipientul) poate genera, de asemenea, evenimente proprii, dar nu au fost standardizate încă evenimente specifice pentru container. Un portlet poate defini un eveniment în descriptorul portlet-ului lui:

Portlet-urile pot publica evenimentele din codul său **processAction.**

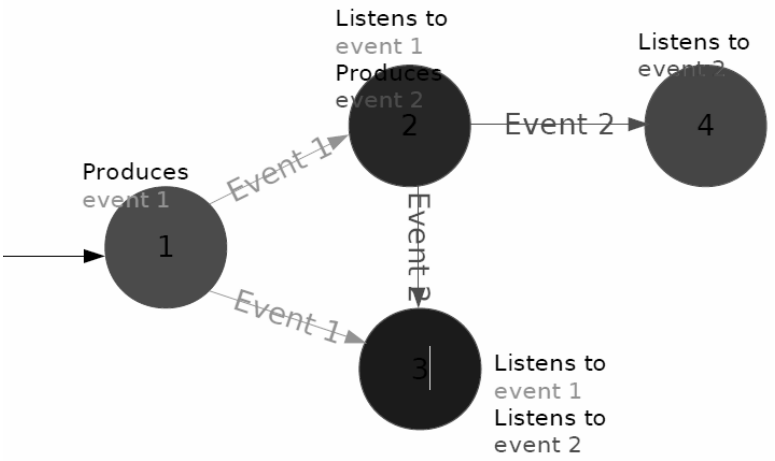

Figura 4

Publicarea unui eveniment provoacă una sau mai multe invocări de noi metode **processEvent** în acest portlet sau în altele. Din interiorul **processEvents** este permisă declanşarea de noi evenimente. Nu există nici o garanție în ordinea de livrare a evenimentelor,

aşa că dezvoltatorii nu ar trebui să facă orice presupunere despre asta. Figura 4 prezintă un exemplu de livrare eveniment. Lanţul efectiv ar putea fi 1,2,3,4, dar ar putea fi, de asemenea, 1,2,4,3 sau chiar 1,3 (eveniment 1), 2, 4, 3 (eveniment 2).

Fiecare portlet trebuie să declare în descriptorul portlet-ului ce evenimente ascultă şi ce evenimente pot fi declanşate. Odată ce un eveniment este declanşat, toate portlet-urile înregistrate pentru a asculta acest eveniment au executat metoda lor **processEvent**. Prin folosirea adnotărilor este posibil să se definească o metodă particulară care va fi executată pentru a procesa un anumit tip de eveniment. Figura 5 prezintă un exemplu de secvență (ordine) de manevrare a cererii, atunci când evenimentele sunt implicate în comunicarea portlet-urilor, şi în cazul în care un eveniment declanşat în portlet-ul B **(RESP(event(X))** face portlet-ul A să reacționeze la acest eveniment (**processEvent (X**)) şi îl schimbă în mod corespunzător.

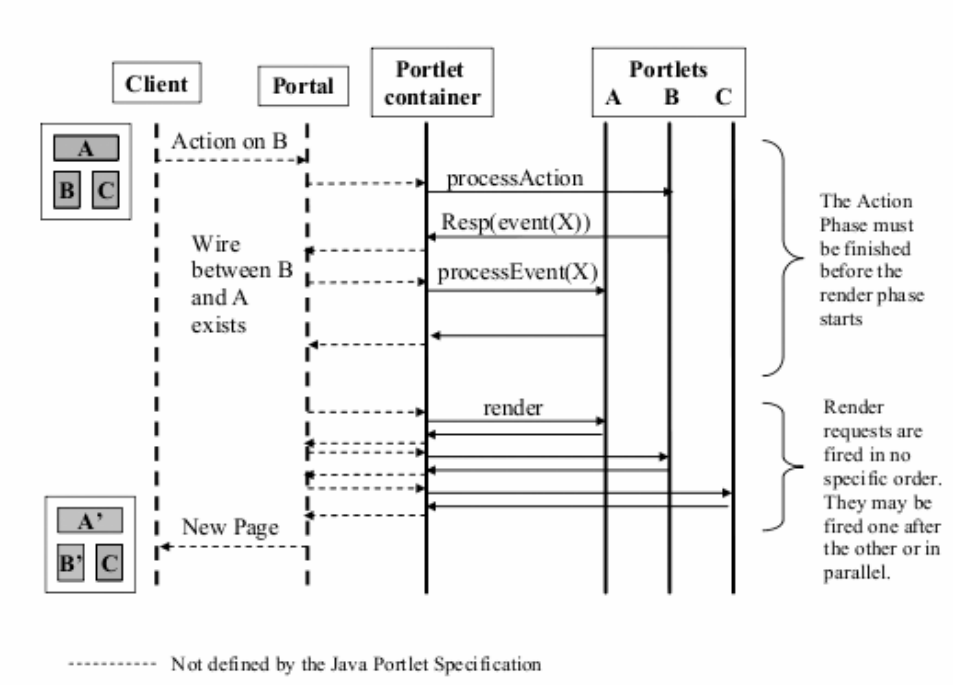

Figura 5

Comparație între parametrii de redare publică şi evenimente

Ambele mecanisme sunt o alegere potrivită pentru dezvoltarea de portlet-uri slab cuplate, şi alegerea uneia sau a celeilalte depinde, în mod considerabil, de cerințele pentru situația specială care este analizată. Recomandarea generală este alegerea parametrilor făcuți public, ca prima opțiune, datorită simplității lor.

Prototipuri de portlet-uri cuplate slab

Un singur portlet poate servi în diferite scopuri, atunci când este combinat cu portlet-uri auxiliare diferite. Luând în considerare acest lucru, doar dezvoltatorii trebuie să dezvolte blocurile "construcție" de bază şi să lase utilizatorul să le combine pentru a obține funcționalitatea dorită.

Ca un exemplu în acest sens, două portlet-uri auxiliare la portlet-ul wiki (navigare wiki şi categorii de navigare) au fost puse în aplicare şi vor fi explicate în următoarele secțiuni.

Portlet-ul categoriilor de navigare poate fi, de asemenea, folosit în conjuncție cu portlets blog. Acesta prezintă posibilitatea de a extrage funcționalitatea şi de a permite o combinație flexibilă cu alte aplicații, cu condiția de a fi de acord cu protocolul de comunicare.

Navigare wiki

Portlet-ele Wiki sunt adesea folosite ca o bază de cunoştinţe cu articole complexe. Aceste articole sunt legate împreună prin utilizarea capacităţii paginilor wiki de a avea copii. Acest lucru permite crearea unei structuri ierarhice de pagini pentru a organiza informaţiile după cum doriţi. Dar, atunci când sunt utilizate în acest fel, tehnicile elementare de navigare wiki nu sunt suficient de bune. În mod ideal, utilizatorul ar trebui să poată vedea ierarhia de pagini pentru a naviga mai uşor între ele. În timp, astfel de funcţionalităţi ar putea fi adăugate la wiki, ceea ce nu este întotdeauna necesar, deoarece aceasta depinde de modul în care portlet-ul wiki este utilizat şi autorii decid să lege paginile. O soluţie la această problemă ar fi să se creeze două variante ale aplicaţiei wiki şi să se utilizeze una sau alta, în funcţie de cazurile de utilizare specifice. Dar această metodă necesită mult mai multă muncă şi reduce astfel productivitatea globală.

Acesta este un scenariu perfect pentru aplicarea ideilor de portlet-uri "slab cuplate". În loc de a dezvolta variante ale portlet-ului wiki, el este implementat, cu funcţionalitate sa generală necesară în majoritatea situaţiilor, şi apoi este completat cu portlet-uri auxiliare suplimentare. Pentru probleme specifice descrise mai sus, soluţia este să se implementeze un portlet care este capabil să detecteze faptul că portlet-ul principal wiki este un articol care arată că face parte dintr-o ierarhie şi, dacă da, arată arborescenţa întregii ierarhii.

Figura 6 arată portlet-ul şi wiki, şi portlet-ul de navigare wiki, lucrând împreună. Făcând clic pe orice link în navigare, portlet-ul prezintă intrarea în portlet-ul principal wiki, care are încă toate opţiunile obişnuite de navigare.

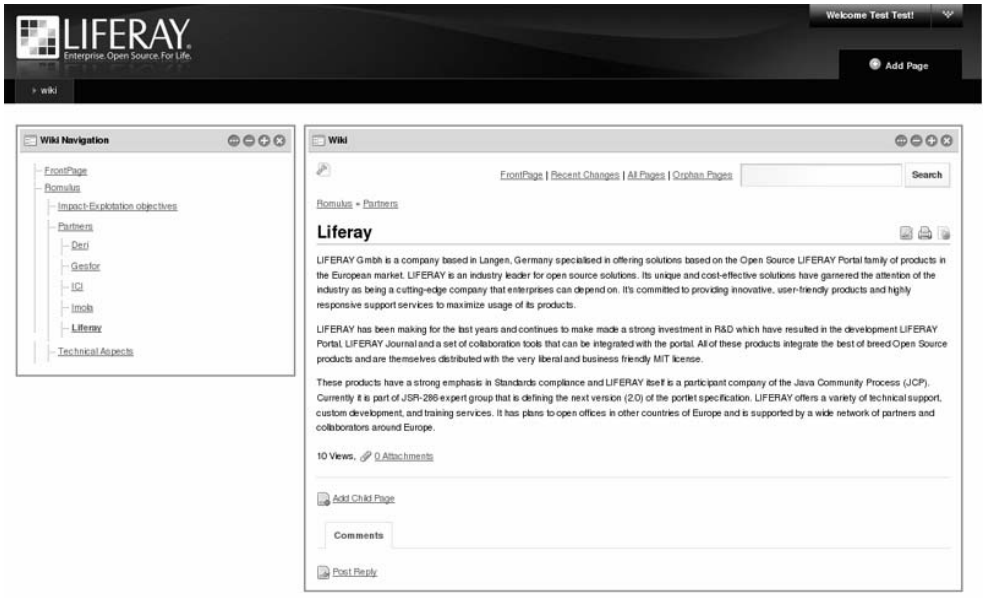

Figura 6

Funcţionalitatea de mai sus a fost implementată prin:

1. extinderea portlet-ului wiki Liferay existent;
2. dezvoltarea unui nou portlet care este în măsură să prezinte o ierarhie de pagini wiki şi oferă navigare printre ele.

Ambele portlet-uri utilizează comunicare între portlet-uri, astfel că:

- portlet-ul de navigare poate determina care pagină wiki este afişată în portlet-ul principal şi oferă un meniu de navigare;
- când un element în portlet-ul de navigare wiki este apăsat (se face clic), portlet-ul principal trebuie să determine pagina pe care a fost făcut clic şi să o prezinte.

Pentru a implementa această comunicare standard a fost folosită comunicarea între portlet-uri definită în caietul de sarcini JSR-286. Deoarece comunicarea necesară nu implică modificări ale statutului de server, cel mai potrivit mecanism pentru acest caz îl reprezintă

parametri făcuți publici, aşa cum s-a explicat în secțiunile anterioare.

În special, au fost definiți doi parametri publici de redare:

- titlul: reprezintă titlul paginii wiki pentru demonstrație;
- nodeId: portlet-ul wiki, Liferay sprijină gruparea paginilor în noduri. Deoarece nodurile diferite pot avea pagini cu acelaşi nume, este necesar să se treacă un identificator de nod, specific pentru a viza în mod unic o anumită pagină.

Acesta este singura modificare necesară la portlet-ul principal existent, a cărui complexitate nu a fost crescută. De asemenea, noul portlet este unul destul de simplu, deoarece se concentrează pe citirea titlului articolului şi a ID-urilor nodurilor precum şi pe determinarea şi evidențierea ierarhiei. Asta înseamnă că este independent de funcționalitatea obişnuită wiki, cum ar fi sintaxa wiki, wiki crearea nodului sau alte caracteristici specifice de gestionare a wiki.

Categoriile de navigare

Multe aplicații diferite, bazate pe conținut trebuie să îşi organizeze conținutul în categorii. Odată ce conținutul este organizat, acesta oferă pentru utilizator mijloace de navigare pe aceste categorii şi lista conținutului conex. Abordarea uzuală folosită în acest scenariu este aceea de a pune în aplicare funcționalitate continuă (din nou şi din nou) în fiecare portlet, care implică mult mai multă muncă şi reducerea productivității.

În plus, navigarea pe categorii este independentă de portlet şi nu există nici o modalitate de a le lega. Asta înseamnă că, dacă un end user (utilizator) trebuie să împartă aceeaşi organizație la două tipuri diferite de conținut, gestionate de portlet-uri diferite şi să navigheze atât în acelaşi timp, este necesar să pună în aplicare un portlet terț care combină funcționalitățile celor două deja existente. Acest lucru înseamnă chiar mult mai multă muncă, pentru o activitate care ar putea fi într-adevăr efectuată de către utilizator, în cazul în care, este oferită flexibilitatea necesară.

O soluție optimă pentru acest scenariu poate fi furnizată, din nou, prin utilizarea conceptului de portlet-uri slab cuplate. În primul rând, un serviciu general pentru clasificare este implementat ca un element comun care poate fi folosit de orice portlet necesitând această funcționalitate. Apoi, este pus în aplicare un portlet auxiliar de navigare pe categorii, ce arată arborele categoriilor definite şi permite utilizatorului să navigheze printre ele. Figura 7 prezintă o aplicație compusă din portlet-ul de navigare pe categorii şi portlet-ul wiki, care a fost modificat pentru a răspunde primului.

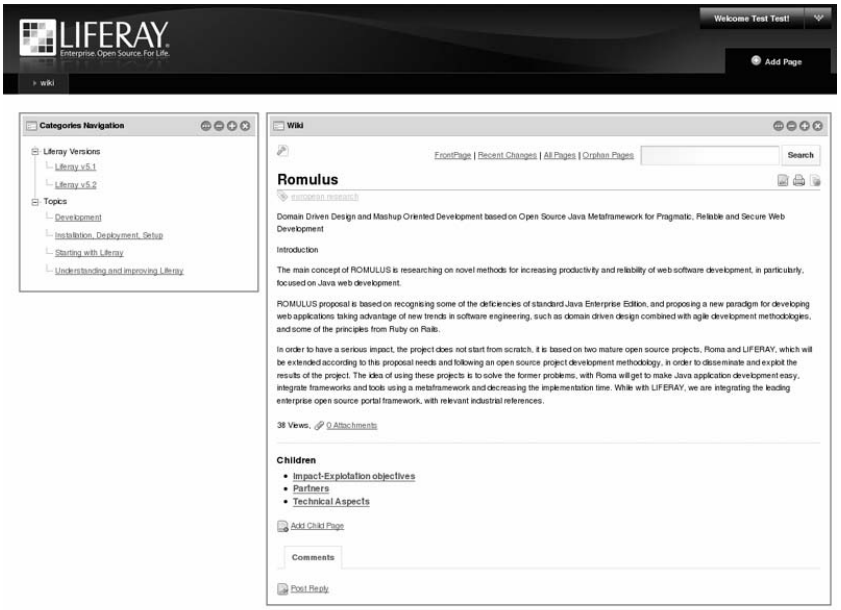

Figura 7

Această navigare a fost pusă în aplicare cu ajutorul comunicării inter portlet, aşa cum este definită de JSR-286. În special, un parametru făcut public, numit categorie, a fost definit pentru a permite portlet-ului de categorii să specifice o anumită categorie dată, astfel încât alte portlet-uri să se poată adapta punctului său de vedere de a afişa numai conținutul acestora clasificate în această categorie.

Liferay a extins două portlet-uri pentru a sprijini acest mecanism, bloguri şi wiki, astfel încât acestea prezintă doar conținut din categoria selectată (atunci când este selectată o categorie).

4. Concluzii

Ambele mecanisme de comunicare între portlet-uri, definite de standardul JSR-286, oferă o mai mare flexibilitate dezvoltatorilor pentru a crea proceduri complexe de comunicare. Dar pentru ca această comunicare să aibă loc, portlet-urile trebuie să convină între ele tipul comunicare, fie parametrii fie evenimente partajate sau să parametrizeze aceste comunicări. De exemplu, atunci când se utilizează evenimente este necesar să se definească numele evenimentului şi sarcinile (utile), în timp ce atunci când se utilizează parametrii făcuți publici este necesar să se definească numele parametrului şi înțelesul său. Prototipurile incluse în prima versiune Romulus arată cum este posibil să se definească astfel de parametrizare pentru a dezvolta portlet-urile care, atunci când sunt utilizate împreună, sunt conectate pentru a forma o aplicație cu mai multe funcționalități.

Integrarea şi dezvoltarea producției de componente gata făcute a continuat prin definirea parametrizărilor standard a acestor mecanisme de comunicare. Acest lucru permite dezvoltatorilor de portlet-uri să elaboreze portlet-uri pregătite să comunice cu orice portlet-uri terțe ce folosesc acelaşi standard. Rezultatul final s-a materializat printr-o productivitate crescută pentru dezvoltatori, datorată reutilizării crescute de portlet-uri existente şi o mai mare flexibilitate pentru utilizatorul final pentru construirea de aplicații complete prin combinarea portlet-urilor dezvoltate de către acelaşi grup de dezvoltatori sau de către diferite grupuri de dezvoltatori.

BIBLIOGRAFIE

[Echo2] Echo (framework) official web site - http://echo.nextapp.com/site/

[EXH1] Exhibit official website - http://simile.mit.edu/exhibit/

[ExtJS1] Ext (JavaScript library) official web site - http://extjs.com/products/extjs/

[JQu1] “jQuery in Action” by Bear Bibeault, Yehuda Katz

[JQB1] “Why jQuery’s Philosophy is Better”, http://blog.jquery.com/2006/08/20/why-jquerys-philosophy-is-better/

[JQP1] jQuery plugins homepage, http://plugins.jquery.com/

[JQU1] jQuery UI homepage, http://ui.jquery.com/

[JSR223] JSR223 home page - http://jcp.org/en/jsr/detail?id=223

[LjQ1] “Learning jQuery”, Jonathan Chaffer and Karl Swedberg, Ed. Packt Publishing, 2007

[PoS1] Portlet specification v 1.0 (JSR-168), http://jcp.org/en/jsr/detail?id=168

[PoS2] Portlet specification v 2.0 (JSR-286), http://jcp.org/en/jsr/detail?id=286

[SaJ2] Scripting and the Java Platform: Productivity and Performance, Roberto Chinnici, Web 2.0 Expo, April 2007, San Francisco USA

[SiJ1] "Scripting in Java" by Dejan Bosanac, 2006

[SiJ2] Scripting for the Java Platform, by John O'Conner, July 2006 - http://java.sun.com/developer/technicalArticles/J2SE/Desktop/scripting/