

Universitatea "Lucian Blaga" Sibiu
Facultatea de Științe
Specializarea: Informatică Economică

LUCRARE DE LICENȚĂ

Coordonator,
Lector univ. drd. Daniel HUNYADI
Engineering Manager Adobe, Marius Andreiana

Absolvent,
Radu Mogoș

Universitatea "Lucian Blaga" Sibiu
Facultatea de Științe
Specializarea: Informatică Economică

Extinderea frameworkului AJAX Spry pentru Dreamweaver Stiletto. Widgetul Spry Rating

Coordonator,
Lector univ. drd. Daniel HUNYADI
Engineering Manager Adobe, Marius Andreiana

Absolvent,
Radu Mogoș

Cuprins

1. INTRODUCERE.....	5
2. Crearea de extensii folosind Dreamweaver.....	6
2.1.Ce sunt extensiile și de ce se folosesc.....	6
2.2.Cum procesează Dreamweaver extensiile.....	6
2.3.Despre diferitele tipuri de extensii Dreamweaver.....	9
2.4.Interfața de programe de aplicații – lucrul cu extensii.....	11
2.5.Modelul Obiectual al Documentului (DOM).....	12
2.6.Care DOM?.....	13
3. Frameworkul Spry.....	14
3.1.Spry Effects.....	14
3.2.Spry Data.....	15
3.3.Spry Widgets.....	15
4. Widgetul Spry Rating.....	17
4.1.Despre widgetul Spry Rating	17
4.2.Anatomia widgetului Spry Rating	17
4.3.Funcționarea widgetului când suportul Javascript lipsește.....	18
4.4.Construirea unui widget Spry Rating	18
4.5.Configurarea widgetului.....	19
4.5.1. Protejarea widgetului pentru votare:.....	19
4.5.2. Setarea valorii inițiale a widgetului:.....	19
4.5.3. Trimiterea valorii votate la server:.....	19
4.5.4. Modificarea dinamică a valorii widgetului.....	20
4.5.5. Votare cu ajutorul tastelor.....	20
5. Extensia Spry Rating pentru Dreamweaver CS4.....	21
5.1.Componentele extensiei Spry Rating.....	21
5.2.Spry.DesignTime.Widget.Base.....	22
5.3.Spry.DesignTime.Widget.Manager.....	22
5.4.Spry.DesignTime.Editing.Utils.....	23
5.5.Spry.DesignTime.Widget.Rating.....	23
6. Integrarea și componentele extensiei.....	28
6.1.Inserarea și modificarea extensiei Spry Rating	28
6.2.Componentele extensiei.....	31
6.2.1. Comenzi.....	31
6.2.1.1 Modul de lucru al comenzilor.....	32
6.2.1.2 Comanda Spry Rating	34
6.2.2. Obiecte.....	35

6.2.2.1 Modul de lucru al obiectelor.....	36
6.2.2.2 Obiectul Spry Rating	36
6.2.3. Translatori.....	39
6.2.3.1 Modul de lucru al translatorilor.....	39
6.2.3.2 Translatorul de Spry Widgets	40
6.2.4. Inspectorul de proprietăți	43
6.2.4.1 Modul de funcționare al PI.....	44
6.2.4.2 PI pentru Spry Rating	45
6.2.5. Alte extensii	52
6.3.Împachetarea și distribuția extensiei Spry Rating	53
7. Concluzie.....	56

1. INTRODUCERE

Tema aleasă extinde suita de widgeturi Spry existente pentru pachetul software Dreamweaver Stiletto al firmei Adobe. Dreamweaver Stiletto va fi lansat în această toamnă împreună cu un pachet complet de widgeturi Spry, widgeturi Google gadgets și o mulțime de alte unelte pentru integrarea de soluții internet deja existente.

Frameworkul Ajax Spry¹ este o librărie Javascript ce oferă funcționalități Ajax puternice și ușor de folosit, ce permit creatorilor de pagini de internet să construiască pagini care oferă o experiență multimedia mult mai bogată utilizatorilor lor. Scopul librăriei este de a ușura integrarea de funcționalități Ajax și de a permite designerilor să creeze mult mai ușor pagini Web 2.0, în majoritatea cazurilor doar printr-un simplu click.

Librăria Spry permite încorporarea informațiilor în format XML, JSON, sau HTML în pagini ce folosesc cod HTML, CSS și o cantitate infimă de cod Javascript, fără a mai fi nevoiți să reîncarce pagina pentru a vedea modificările. Librăria Spry permite și o stilizare și integrare ușoară a widgeturilor, oferindu-le utilizatorilor elemente de conținut avansate. Construcția librăriei a fost axată încât să ofere folosirea minimală a codului HTML și Javascript pentru a implementa numeroasele widgeturi, și poate fi folosită de oricine dorește să dezvolte pagini web interactive, folosind orice unealtă pentru procesul de creație.

Motivul alegerii acestei teme îl constituie posibilitatea care mi s-a dat în a lucra la un proiect care chiar are o finalitate practică, alături de o echipă care are pregătirea necesară și resursele necesare pentru ați permite să lucrezi și să extinzi un pachet software folosit de mii sau chiar sute de mii de oameni din întreaga lume. Experiența a fost una fructuoasă, iar echipa Adobe Romania a oferit tot suportul necesar pentru finalizarea proiectului.

Ca și aplicabilitate și exemplu de integrare a unui widget Spry în ultima versiune a produsului Dreamweaver, am ales să integrez widgetul Spry Rating². Widgeturile Spry sunt componente web create folosind cod HTML, CSS și cod Javascript minim, permit customizarea lor vizuală folosind cunoștințele de HTML și CSS pe care le aveți și sunt de asemenea componente accesibile; pot fi atașate unor secvențe de taste și pot fi folosite și când suportul pentru Javascript lipsește.

Widgetul Spry Rating permite votarea, care poate fi aplicată unei pagini sau mai multor componente ale unei pagini (de exemplu: poze, muzică, votarea articolelor, votarea utilizatorilor). Widgetul poate fi configurat încât să permită o singură votare per înregistrare sau multiple votări.

1 Adobe Labs - <http://labs.adobe.com/technologies/spry/> - pentru mai multe informații și exemple complete privind librăria Spry

2 Adobe Labs - <http://labs.adobe.com/technologies/spry/samples/rating/RatingSample.html> – exemple complete de implementare a widgetului Spry Rating.

2. CREAREA DE EXTENSII FOLOSIND DREAMWEAVER

Capitolul tratează modalități de creare a extensiilor pentru pachetul software Dreamweaver(versiunea Stiletto), detaliind diferitele tipuri de extensii care pot fi create și apoi particularizând la lista de extensii folosite pentru a crea widgetul Spry Rating.

2.1. Ce sunt extensiile și de ce se folosesc

Extensiile sunt bucăți de cod, ce extind funcționalitatea pachetului Dreamweaver. În general, sunt create pentru a fi folosite în situații ce presupun repetarea multiplă a aceleiași cerințe. Situațiile generale în care sunt folosite extensiile sunt următoarele:

- automatizarea schimbărilor pe care le face utilizatorul documentului curent, cum ar fi inserarea de conținut HTML, CSS, CFML, Javascript sau orice alt cod recunoscut de Dreamweaver.
- interacționarea cu aplicația, pentru a închide/deschide automat ferestre, închide/deschide automat documente, pentru a schimba anumite taste rapizi și nu numai.
- conectarea la surse de date, ceea ce permite crearea de pagini dinamice care folosesc aceste informații din sursele respective.
- inserarea și lucrul cu blocuri de cod server în documentul curent.

2.2. Cum procesează Dreamweaver extensiile

Următoarele componente permit crearea de extensii pentru Dreamweaver:

- Un parser HTML, ce permite crearea interfeței cu utilizatorul folosită de extensie. Acesta presupune existența unui formular, câmpuri folosite de formular, posibilitatea de a adăuga imagini, elemente poziționate absolut și alte elemente HTML. Elementele HTML vor fi renderate într-o fereastră Windows/Mac. Dreamweaver are propriul său parser HTML.
- Un arbore de directoare, ce organizează și stochează fișierele care implementează elementele și extensiile sistemului.
- O serie de interfețe de programare (API) care permit accesul la funcționalități Dreamweaver folosind limbajul Javascript.
- Un interpretor Javascript ce execută codul Javascript din fișierele extensiilor.

Practic, o extensie este formată dintr-un fișier ce conține codul HTML, folosit pentru a rendera fereastra în Dreamweaver, și un fișier sau mai multe, cu codul Javascript, ce conține logica acelei extensii. Fișierele Javascript pot fi referențiate, sau codul lor poate fi introdus direct în secțiunea HEAD a fișierului HTML.

Un exemplu simplu de extensie Dreamweaver o reprezintă o comandă din meniul principal. Presupunem următoarea comandă simplă creată de noi, care include codul Javascript în același fișier HTML, fiind vorba de comenzi puține.

Apelată, comanda va afișa următoarea fereastră (fig 1.2):

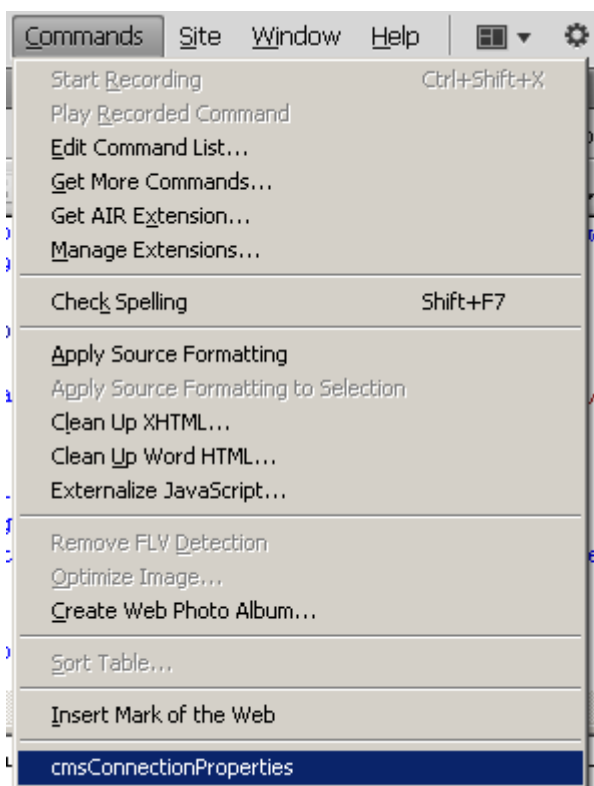


Fig 2.1 Comanda cmsConnectionProperties, cum arată apelată din meniul principal

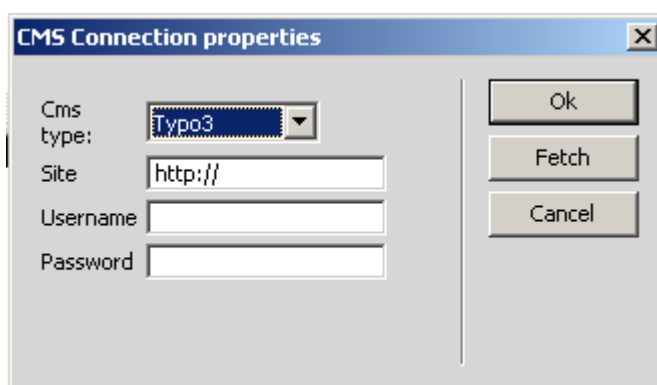


Fig 2.2 Rezultatul apelării comenzii constă în afișarea acestei ferestre.

Codul HTML necesar pentru a genera această fereastră va arăta în felul următor:

```
<form name="cmsform">
<table>
  <tr>
    <td>Cms type:</td>
    <td>
      <select name="cms">
```

```

        <option value="typo3" selected="selected">
        Typo3</option>
        <option value="wordpress">Wordpress</option>
        <option value="joomla">Joomla</option>
    </select>
    </td>
</tr>
<tr>
    <td>Site</td>
    <td><input type="text" name="site" value="" /></td>
</tr>
<tr>
    <td>Username</td>
    <td><input type="text" value="" name="username"/></td>
</tr>
<tr>
    <td>Password</td>
    <td><input type="password" name="password" value=""
/></td>
</tr>
</table>
</form>

```

Tot ceea ce Dreamweaver va întâlni în tagul form, îl va rendera în stânga ferestrei, butoanele de comanda fiind generate în secțiunea de cod Javascript, în funcția `commandButtons()`. Codul Javascript pentru generare a butoanelor și logicii formularului, este următorul:

```

<script>

function commandButtons()
{
    return new Array("Ok", "okPressed()", "Fetch", "fetch()", "Cancel",
"cancelPressed()");
}

function okPressed()
{
    // comenzi pentru butonul ok
}

function cancelPressed()
{
    // comenzi pentru butonul cancel
}

function fetch()
{
    var site = document.cmsform.site.value;

```

```
var httpReply = MMHttp.getTextCallback3("status", site, "test");  
}  
</script>
```

Interacțiunea între formular și cod se face simplu, ca în cazul oricărei alte pagini HTML ce conține cod Javascript. Putem lua valoarea elementelor după nume, folosind apeluri de genul `document.cmsform.site.value;` sau `document.forms[0].site.value` pentru elementele obișnuite (textarea, select, input), însă putem folosi și funcțiile API pentru a prelua valoarea elementelor speciale puse la dispoziție de Dreamweaver.

2.3. Despre diferitele tipuri de extensii Dreamweaver

Dreamweaver oferă o serie de extensii, care oferă diferite funcționalități, pornind de la executarea unei comenzi până la afișarea și executarea unui property inspector pe un tag sau un bloc de cod destul de complicat.

Folosite împreună, extensiile permit crearea de widgeturi sau unelte destul de compacte, care să simplifice în final enorm munca depusă de utilizatorul programului, fie el designer sau programator.

Lista de extensii oferite de Dreamweaver include:

- **Obiecte Insertbar** – aceste extensii creează modificări în bara de Inserare. Un obiect e folosit de obicei pentru a automatiza inserarea de cod într-un document. Aceste obiecte pot să conțină și un formular în care se primesc date de la utilizator, și cod Javascript care execută codul în funcție de parametrii introduși. Fișierele pentru obiecte sunt stocate în directorul Configuration\Objects⁴.

3 MMHttp.getTextCallback – este o funcție API, folosită pentru a trimite un apel asincron și pentru a returna răspunsul cererii, incluzând headerele returnate și conținutul returnat de apel, care poate fi cod HTML, XML, sau orice alt format text. Funcțiile API încep cu MMnume/MM sau dw/dreamweaver. Mai multe despre funcțiile API în capitolele următoare.

4 Fiecare tip de extensie e stocat într-un anume director preconfigurat. Pentru o bună funcționare a extensiilor, fișierele lor trebuie stocate în folderul categoriei cărora le aparțin. Stocate în altă parte, ele nu vor funcționa, deoarece Dreamweaver caută și apelul anumitor funcții specifice fiecărei extensii.

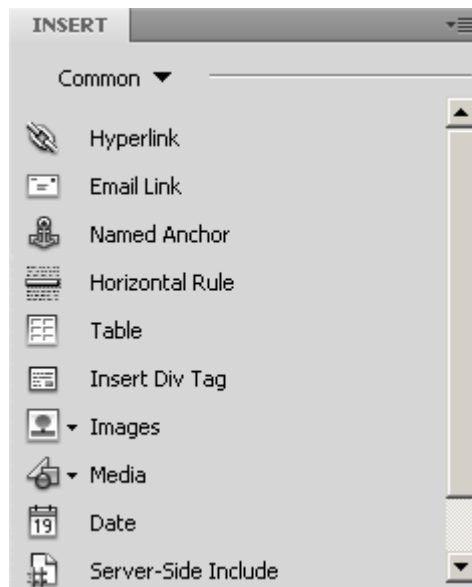


Fig 2.3 Meniul de obiecte Insertbar, așa cum apare în Dreamweaver Stiletto

- **Comenzi** – aceste extensii pot rula orice task, fie că primesc sau nu date de la utilizator. În general comenzile sunt invocate din meniul “Commands”, dar pot fi apelate și din alte extensii. Fișierele de comenzi sunt stocate în directorul Configuration\Commands.

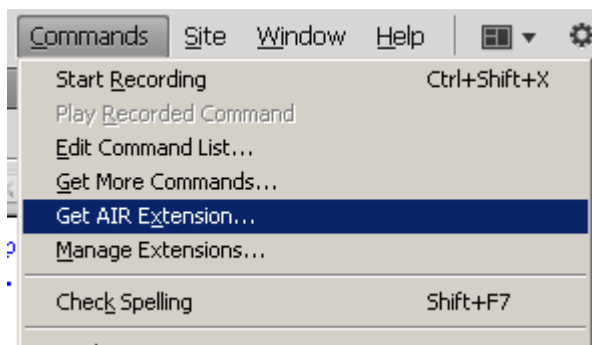


Fig 2.4 Meniul de comenzi

- **Comenzi meniu** – aceste extensii extind API-ul de comenzi pentru a permite rularea de taskuri când anumite elemente din meniu sunt apelate. Comenzile de meniu permit și crearea de submeniuri dinamice.
- **Toolbar (bara de unelte)** – aceste extensii pot adăuga elemente la toolbarurile existente sau pot crea noi toolbars în Dreamweaver. Noile toolbars apar sub cel standard. Uneltele sunt stocate în directorul Configuration\Toolbars.



Fig 2.5 Toolbarul pentru alegerea diferitelor stiluri de rendare a paginii

- **Rapoarte** – aceste extensii pot adăuga rapoarte customizate sau pot modifica modelele deja existente
- **Property Inspector (Inspectorul de proprietăți)** – aceste extensii apar în

panoul de inspectare al proprietăților. Majoritatea inspectorilor din Dreamweaver fac parte din distribuția de bază și nu pot fi modificați, dar inspectorii customizați pot suprascrise interfețele existente în Dreamweaver sau pot crea interfețe noi pentru a analiza taguri definite de utilizator. Inspectorii sunt stocați în folderul Configuration\Inspectors.

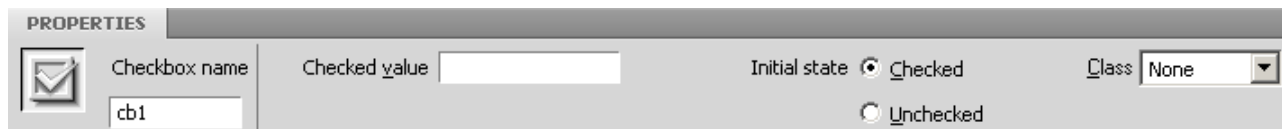


Fig 2.6 Inspectorul pentru un input de tip checkbox.

- **Floating panel (panouri flotante)** – aceste extensii permit adăugarea de panouri flotante la interfața Dreamweaver. Aceste panouri pot interacționa cu textul selectat, cu întreg documentul sau cu taskul curent. Panourile sunt stocate în directorul Configuration\Floaters.
- **Data source** – aceste extensii permit crearea de conexiuni dinamice la informații stocate în baza de date. Extensiile pot fi accesate din meniul Plus (+) al panoului de Legături (Bindings panel). Directorul în care sunt stocate acest tip de extensii este Configuration/Data sources.
- **Data translators (Translatorii de date)** – aceste extensii convertesc cod non-HTML în cod HTML ce poate fi vizualizat în fereastra documentului, când aceasta este setată pe vederea Design (Design View, sau Split). Aceste extensii blochează și codul non-HTML pentru a preveni afișarea lui de către Dreamweaver. Fișierele translator sunt stocate în directorul Configuration\Translators.

2.4. Interfața de programe de aplicații – lucrul cu extensii.

Interfața de programe de aplicații (API⁵) pentru extensiile Dreamweaver creează un schelet pe baza căruia orice extensie poate fi customizată, atât folosind cod Javascript cât și cod C (codul fiind astfel apelabil din fișiere DLL)

Modul în care Dreamweaver procesează extensiile Javascript este următorul: La pornirea programului, acesta verifică directorul Configuration/*tipul_extensiei*⁶ unde *tipul_extensiei* reprezintă una dintre categoriile de extensii descrise mai sus. Dacă întâlnește un fișier al unei extensii, Dreamweaver procesează codul Javascript folosind următorii pași:

- Compilează tot codul găsit între tagurile <SCRIPT>
- Execută orice cod găsit între tagurile <SCRIPT>, care nu face parte din declarația

5 API – definire și descriere <http://en.wikipedia.org/wiki/API>

6 Motivul pentru care Dreamweaver procesează aceste fișiere la pornire este pentru că unele extensii ar putea avea nevoie să inițializeze variabile globale, care vor fi inițializate în acest pas, și astfel pot fi apelate de mai multe extensii care le-ar putea folosi.

unei funcții (declararea de variabile globale, etc)

Pentru fișierele Javascript externe, definite în atributul **src** al tagului `<SCRIPT>` Dreamweaver le va procesa executând următorii pași:

- Citirea fișierului
- Compilarea codului
- Executarea procedurilor

Extensiile terminate și pregătite pentru distribuție pot fi împachetate într-un format recunoscut de aplicația Extension Manager⁷, care vine împreună cu instalația de Dreamweaver, și care este folosită pentru a ușura adăugarea/ștergerea de extensii customizate, altele decât cele standard.

2.5. Modelul Obiectual al Documentului (DOM⁸)

În aplicația Adobe Dreamweaver, Modelul Obiectual al Documentului, numit în continuare DOM, este o structură extrem de importantă pentru dezvoltatorii de extensii. Modelul permite accesul și manipularea elementelor din documentul utilizatorului precum și din documentul extensiei. Prin reprezentarea tagurilor și atributelor ca și obiecte și proprietăți, DOM permite limbajelor de programare să acceseze și să manipuleze documente și conținutul acestora.

Structura unui document HTML poate fi văzută asemeni unui arbore. Rădăcina arborelui este reprezentată de tagul `html`, cu succesorii `head` și `body`. Succesorii ai tagului `head` ar putea fi tagurile: `title`, `meta`, `script`, `link`, etc iar ai tagului `body`: `h1`, `h2`, `div`, `p`, `a`, etc. Această relație continuă apoi pe fiecare nivel pentru fiecare tag ce conține alt tag.

Structura arborescentă a unui DOM este stocată și reprezentată ca o ierarhie de noduri părinte și noduri copil. Rădăcina nu are nici un părinte, iar nodurile frunză nu au noduri copii. La orice nivel în structura documentului HTML, orice element HTML poate fi accesat de Javascript ca și nod. Folosind această structură, putem accesa astfel întreg documentul sau orice componentă a sa.

Accesul obiectelor cu ajutorul Javascript poate fi făcut referind acel obiect fie prin numele său sau prin indexul său. De exemplu, dacă dorim să accesăm un input de tip text, cu numele "prenume", element ce se găsește într-un formular cu numele "formular1", elementul fiind al 2-lea input în formular, iar formularul fiind primul formular din document, atunci ambele referințe sunt valide și vor returna același nod:

- referind numele: `document.formular1.prenume`

7 The extension installation file format – Adobe Press – documentul conține informații legate de modul de împachetare al extensiilor pregătite pentru distribuție.

8 DOM – acronym pentru Document Object Model

- referind indexul: `document.forms[0].elements[1]`⁹

2.6. Care DOM?

Dreamweaver permite accesul la 2 modele obiectuale ale documentului. Primul DOM face referire la documentul curent pe care lucrează utilizatorul, iar al doilea DOM face referire la structura documentului extensiei. Modul de referire al celor 2 modele este diferit.

În Javascript, modul standard de apelare al obiectelor din documentul activ, este următorul: `document.element` (unde *element* este elementul pe care îl referim). În Dreamweaver, `document` face referire la modelul folosit de extensie, astfel, `document.forms[0]` face referire la primul formular întâlnit în interfața extensiei. Referirea obiectelor din documentul utilizatorului se face prin apelul `dw.getDocumentDOM()`, `dw.createDocument()` sau alte funcții care returnează obiectul `document`.

De exemplu, într-o extensie, referirea primei imagini din documentul utilizatorului se face în felul următor:

```
var dom = dw.getDocumentDOM(); // preluare DOM
var primaImagine = dom.images[0]; // preluare prima imagine
primaImagine.src = "altaImagine.gif"; // setare atribut SRC
```

Accesul la aceste 2 tipuri de modele este permis doar dezvoltatorilor de extensii. Utilizatorul are acces doar la documentul curent, iar accesarea obiectelor din documentul curent se face folosind funcțiile și accesorii standard din Javascript, astfel referirea modelului va fi făcută folosind `document`.

9 Numerotarea pentru elementele unui array în Javascript începe de la elementul de pe poziția 0. Astfel, `elements[1]` face referire la al doilea element, iar `forms[0]` face referire la primul formular din document.

3. FRAMEWORKUL SPRY

Frameworkul Spry este un framework Ajax opensource dezvoltat de compania Adobe, folosit pentru dezvoltarea de aplicații RIA (Rich Internet Applications) – aplicații internet cu conținut și interacțiuni multiple. Spre deosebire de celelalte frameworkuri Javascript existente, gen Dojo sau Prototype, Spry este destinat mai ales webdesignerilor, decât dezvoltatorilor web, și asta datorită ușurinței în utilizare.

Frameworkul este integrat direct începând cu Dreamweaver versiunea CS3. Spry poate fi însă folosit cu orice alt software existent sau direct în editorul HTML pe care îl folosim de obicei. Tot ce trebuie știut pentru a integra Spry în paginile noastre e HTML, CSS și Javascript.

Spry este implementat sub forma unui set de librării Javascript. Este constituit din 3 părți: Spry Data, Spry Widgets și Spry Effects. Aceste componente pot fi folosite împreună sau separat

3.1. Spry Effects

Componenta Spry Effects permite adăugarea de efecte oricărui element de pe o pagină de internet. Efectele se adaugă printr-o singură linie de cod, iar aceste efecte pot fi folosite pentru a evidenția anumite informații, crea tranziții animate, sau modifica vizual un element pentru o perioadă determinată de timp. Printre efectele oferite de Spry 1.6.1¹⁰ se numără:

- **Fade** – elementul dispare sau apare
- **Blind** – elementul este închis sau deschis asemenea unei cortine
- **Grow** – elementul își schimbă mărimea
- **Highlight** – elementul este scos în evidență
- **Shake** – elementul este mișcat rapid
- **Slide** – elementul permite crearea de slideuri. Este folosit de obicei în grupe de elemente cu efect Slide.
- **Squish** – elementul este redimensionat
- **Observers** – observatorii permit execuția de cod customizat sau execuția altor efecte după ce anumite schimbări au fost depistate pe unele elemente (ex: schimbare clasă, schimbare efect, etc).
- **Clustering** – permite rularea de efecte, unul după celălalt. Efectele sunt depuse într-un cluster, apoi vor fi executate în ordine.

¹⁰ Referințele din această lucrare de licență se referă la versiunea 1.6.1 a frameworkului Spry

3.2. Spry Data

Componenta Spry Data transformă informații complexe, primite din surse variate, într-un format bazat pe linii și coloane care poate fi apoi folosit oriunde în pagină. Formatele de informații cu care poate lucra includ XML, JSON și HTML. Componenta permite de asemenea și adăugarea de regiuni dinamice în pagina pe care lucrăm, regiuni care controlează modul de afișare și poziționare al informațiilor dorite fără să scriem o linie de cod Javascript.

Clase folosite pentru lucrul și importul de informații:

- **CSV Data Set** – permite importul de informații dintr-un fișier CSV
- **XML Data Set** – permite importul de informații dintr-un fișier XML
- **HTML Data Set** – permite importul de informații dintr-un fișier HTML
- **JSON** – permite folosirea informațiilor venite din arrayuri în formatul JSON

Asupra informațiilor importate se pot executa operații de sortare, căutare filtrată pe Xpath (pentru XML), schimbul de date între diferite containere de informații, paginare, observatori pe containerele de date precum și popularea widgeturilor cu aceste date (de exemplu popularea unui Spry Accordion).

3.3. Spry Widgets

Componenta Spry Widgets conține o serie de componente web avansate, create folosind cod HTML, puțin CSS și o cantitate infimă de cod Javascript. Ca în cazul altor componente, stilizarea se face folosind HTML și CSS, iar widgeturile pot răspunde la apăsări de taste, funcționând chiar și când suportul pentru Javascript este inexistent.

Printre widgeturile populare se numără:

- **Accordion** – creează un acordion folosind cod static HTML sau un data set, din Spry Data.
- **Widgeturi de validare** – form text field, form confirm password, form password validation, form checkbox, form select, form textarea, etc. Sunt asociate în general cu un formular ce urmează a fi postat, scopul lor fiind de a valida datele înainte de a fi trimise.
- **Sliding panels** – efectul de panouri glisante este unul dintre cele mai folosite, pe lângă acordion.
- **Tabbed panels** – Panouri controlate prin taburi
- **Tooltip** – creează note informative, fie din cod static sau dinamic.

- **Rating** – widgetul dezbătut în această lucrare. Permite adăugarea unui control de votare pe pagină sau pentru orice înregistrare de pe pagină.

4. WIDGETUL SPRY RATING

Capitolul dezbate componentele widgetului Spry Rating, opțiunile de configurare ale acestuia, modul de integrare în Dreamweaver Stiletto și extensiile folosite pentru a obține funcționalitățile dorite în Dreamweaver.

4.1. Despre widgetul Spry Rating

În Spry, fiecare widget constă dintr-un fișier Javascript, ce conține logica, și un fișier CSS, ce conține stilizarea acestuia. Integrarea unui widget în pagină constă în inserarea acelor 2 fișiere. Fișierele corespunzătoare sunt numite după widgetul pe care îl reprezintă, deci în cazul nostru vom utiliza fișierele SpryRating.css și SpryRating.js, împreună cu imaginile folosite pentru afișarea steluțelor de votare.

4.2. Anatomia widgetului Spry Rating

Widgetul poate fi folosit în 2 cazuri diferite: ca și componentă statică, ce nu va permite utilizatorilor să voteze ci doar să vizualizeze votul curent, sau ca și componentă dinamică, ce permite utilizatorilor să își exprime propriul vot. Toate widgeturile sunt inițializate apelând constructorul respectiv clasei pe care o reprezintă, după ce codul HTML corespunzător widgetului a fost adăugat. Constructorul primește 2 valori, id-ul elementului HTML care îl conține și un set de opțiuni folosite pentru configurare (opțional).

Exemplu:

```
<span id="spryrating1" class="ratingContainer">
  <span class="ratingButton"></span>
  <span class="ratingButton"></span>
  <span class="ratingButton"></span>
  <span class="ratingButton"></span>
  <span class="ratingButton"></span>
  <!--Adăugare mesaj customizat-->
  <span class="ratingRatedMsg">Mulțumim pentru vot !</span>
</span>

<script type="text/javascript">
  var spryrating1 = new Spry.Widget.Rating("spryrating1");
</script>
```

Fiecare widget are atașat un set de clase ce controlează aparența și modul de interacționare al acestuia. În exemplul de mai sus, întregului widget îi atașăm clasa `ratingContainer`, iar fiecare steluță va fi reprezentată folosind clasa `ratingButton`. Când utilizatorul va alege ce vot va da, alăturat va apare mesajul definit pe elementul cu

clasa `ratingratedMsg`, și anume "Mulțumim pentru vot!".



Fig 4.1 Widgetul de rating, așa cum este afișat în pagină

Poate fi folosit orice tag HTML pentru container sau pentru stelute, singurul lucru important constă în respectarea structurii widgetului: un container principal ce conține în structura sa componentele steluțelor și componentele mesajelor afișare.

```
Container SPAN, DIV, etc
    Stelute SPAN, DIV, etc
    Mesaje SPAN, DIV, etc
```

4.3. Funcționarea widgetului când suportul Javascript lipsește

Pentru a permite funcționarea widgetului când suportul Javascript lipsește trebuie adăugate câteva taguri HTML adiționale. Elementele adăugate în plus vor fi ascunse când suportul Javascript este prezent. Înlocuirea se face folosind un input de tip text, în care vom putea afișa și prelua valoarea votului, precum și un buton de submit, pentru a posta noile valori.

```
<form method="get" name="form1" action="SpryRating.php">
  <span id="spryrating1" class="ratingContainer">
    <span class="ratingButton"></span>
    <span class="ratingButton"></span>
    <span class="ratingButton"></span>
    <span class="ratingButton"></span>
    <span class="ratingButton"></span>

    <input type="text" name="numeElement" value="3.5" />
    <input type="submit" value="Voteaza"/>

  </span>
</form>
```

4.4. Construirea unui widget Spry Rating

În pagina în care dorim să adăugăm votarea, scriem următorul cod în secțiunea head:

```
<script type="text/javascript" src="SpryRating.js"></script>
<link href="SpryRating.css" rel="stylesheet" type="text/css" />
```

Rendarea votării o vom face introducând codul respectiv acesteia (pentru votare cu 5 stele):

```
<span id="spryrating1" class="ratingContainer">
  <span class="ratingButton"></span>
  <span class="ratingButton"></span>
  <span class="ratingButton"></span>
  <span class="ratingButton"></span>
```

```
<span class="ratingButton"></span>
</span>
```

Următorul pas constă în adăugarea constructorului pentru widget:

```
<script type="text/javascript">
    var rating = new Spry.Widget.Rating("spryrating1"); </script>
```

4.5. Configurarea widgetului

Adăugarea de funcționalități opționale se face prin configurarea widgetului, opțiuni care sunt adăugate la constructorul widgetului, după ce specificăm id-ul elementului căruia îi atribuim votarea. Fiind definite într-un array, opțiunile sunt delimitate între acolade { }.

4.5.1. Protejarea widgetului pentru votare:

```
<script type="text/javascript">
    var rate = new Spry.Widget.Rating("spryrating1", {readOnly:true});
</script>
```

4.5.2. Setarea valorii inițiale a widgetului:

```
<script type="text/javascript">
    var rate = new Spry.Widget.Rating("spryrating1", {ratingValue:2.5});
</script>
```

Codul de mai sus va crea o instanță a widgetului de votare, iar valoarea inițială va fi setată la 2.5, deci pentru un sistem de votare de 5 stele, punctajul va fi setat la jumătate.

4.5.3. Trimiterea valorii votate la server:

```
<script type="text/javascript">
    var rate = new Spry.Widget.Rating("spryrating1",
    {saveUrl:'SpryRating.php?id=spryrating5&val=@@rw_Rating@@'});
</script>
```

Widgetul include modalități de comunicare cu serverul, pentru a trimite și primi valoarea unei votări. Trimiterea se poate face prin 2 metode, folosind POST sau GET. Metoda standard de trimitere este GET. Când trimitem datele votării astfel, trebuie să specificăm opțiunea **saveUrl** care primește URL-ul căruia îi trimitem aceste date. În cazul de mai sus trimitem valoarea votată pe widgetul cu id **spryrating5** scriptului **SpryRating.php**

Pentru a trimite datele folosind metoda POST trebuie adăugată opțiunea **postData**, ce va conține URL-ul și datele votării, precum în exemplul de mai jos:

```
<script type="text/javascript">
    var rate = new Spry.Widget.Rating("spryrating1", {postData:
'id=spryrating2&val=@@ratingValue@@', saveUrl:'SpryRating.php'});
```

```
</script>
```

Valoarea votării va fi trimisă în variabila @@ratingValue@@, iar în cazul metodei GET, în variabila @@rw_Rating@@

4.5.4. Modificarea dinamică a valorii widgetului

Folosind opțiunea **afterRating** se poate specifica ce valoare va lua widgetul după votare, care poate să fie valoarea curent votată, sau valoarea returnată de server după ce s-a votat (deci o medie a voturilor totale). Pentru aceasta trebuie adăugat în interiorul codului widgetului un input de tip text, care va stoca noua valoare. Id-ul acestui input va fi transmis ca valoare pentru opțiunea **ratingValueElement**. Această opțiune este folosită întotdeauna împreună cu opțiunea **saveUrl**.

```
<span id="spryrating1" class="ratingContainer">
  <span class="ratingButton"></span>
  <span class="ratingButton"></span>
  <span class="ratingButton"></span>
  <span class="ratingButton"></span>
  <span class="ratingButton"></span>
  <input type="text" id="ratingValue" name="dynamic_rate"
value="2"/>
</span>

<script type="text/javascript">
  var rate = new Spry.Widget.Rating("spryrating1",
    {ratingValueElement:"ratingValue",
      afterRating:'serverValue',
      saveUrl:'SpryRating.php?id=spryrating5&val=@@rw_Rating@@'});
</script>
```

4.5.5. Votare cu ajutorul tastelor

Modificările folosind tastatura permit utilizatorului să controleze widgetul folosind săgețile sau alte taste de acces definite. Navigarea widgetului cu ajutorul tastaturii este suportată în mod standard, însă poate fi oprită setând opțiunea **enableKeyboardNavigation** pe fals. Specificarea altor taste de navigare decât cele standard se poate face folosind opțiunile pentru taste: **moveNextKeyCode** (mărește valoarea votului), **movePrevKeyCode** (scade valoarea votului), **doRatingKeyCode** (realizează votarea)

```
<script type="text/javascript">
  var rate = new Spry.Widget.Rating("spryrating1",
{moveNextKeyCode:34 /* tasta PgUp */, movePrevKeyCode: 33 /* tasta PgDn */,
doRatingKeyCode:32 /*tasta Space*/});
</script>
```

5. EXTENSIA SPRY RATING PENTRU DREAMWEAVER CS4

Capitolul precedent a expus detaliile frameworkului Spry, integrarea widgetului Spry Rating într-o pagină web precum și codul necesar pentru această integrare și opțiunile de configurare a sistemului de votare, odată creat pe pagină.

Capitolul curent dezbate modul de creare a unei astfel de extensii și modul de integrare a unui astfel de widget într-o pagină de internet creată în aplicația Dreamweaver. Integrarea widgetului se face creând o serie de extensii Dreamweaver ce permit integrarea acestora folosind meniul de obiecte sau cel de comenzi și apoi ușoara editare de către utilizator al acestui widget folosind Property Inspector.

Astfel extensia în sine este construită folosind o suită de alte extensii, care integrate, ajută la o utilizare mult mai facilă și mai ușoară a componentelor de care utilizatorul comun are nevoie pentru a scrie pagini de internet funcționale și în pas cu noile tehnologii.

Dacă prima parte ce prezenta widgetul de votare, se adresa în special programatorilor, extensia integrată în Dreamweaver se adresează utilizatorilor de rând, care pot să aibă cunoștințe Javascript de bază, sau chiar deloc, și care pot configura acest widget folosind doar opțiunile oferite de Inspectorul de Proprietăți.

5.1. Componentele extensiei Spry Rating

Am discutat inițial despre tipurile de extensii care pot fi create în Dreamweaver, și am listat câteva dintre cele mai importante și mai folosite dintre ele. Extensia Spry Rating construiește astfel peste acestea și creează o extensie în sine, folosindu-se de următoarele tipuri de extensii Dreamweaver:

- Commands (comenzi)
- Inspectors (inspectori)
- Objects (obiecte)

Definește apoi în fișiere XML configurările pentru colorarea codului (Code Coloring), afișează metodele disponibile și opțiunile de configurare ale widgetului în Code Hints, localizează mesajele în XML-ul pentru Strings, adaugă clasa widgetului la clasele comune în directorul Startup și definește metodele specifice acestora în directorul Shared destinat pachetului Spry.

Toate aceste componente și pașii respectivi vor fi detaliați pe parcursul acestei lucrări.

5.2. Spry.DesignTime.Widget.Base

Toate widgeturile Spry folosite în Dreamweaver extind clasa `Spry.DesignTime.Widget.Base`.

Această clasă are metode comune tuturor widgeturilor de adăugare/ștergere a claselor pe care le are un element, de schimbare a id-ului acestuia, de modificare a opțiunilor sale, de a seta tipul widgetului, de a atașa evenimente la care widgetul să răspundă, de a returna întreg elementul conținut de widget precum și de a verifica dacă widgetul este valid sau nu. Un widget este valid atunci când structura sa este validă (adică este format din elementele necesare widgetului respectiv, fiecare element fiind specificat clar în documentație).

La inițializare, clasa primește ca parametri DOM-ul pe care operează, numele obiectului și elementul care va fi reprezentat de widget.

```
Spry.DesignTime.Widget.Base = function(dom, objName, element)
{
    this.dom = dom;
    this.objName = objName;
    this.element = this.getElement(element);
    this.element_id = (typeof element == "string" || !element) ?
element : element.id;
};
```

5.3. Spry.DesignTime.Widget.Manager

Clasa `Spry.DesignTime.Widget.Manager` este cea care ține evidența tuturor widgeturilor de pe pagină, pentru a ușura modificările aplicate acestora.

Clasa conține doar o referință la DOM-ul pe care operăm, și un array de obiecte de tip `Widgets`.

Metodele expuse de această clasă sunt:

- `getManagerForDocument` – returnează o referință la managerul widgeturilor pentru documentul curent.
- `getWidget(type, id)` – returnează widgetul de tip `type` și id `id`
- `setWidget(type, id, widget)` – adaugă în array un widget de tip `type`, cu id-ul `id`, referit de obiectul `widget`
- `deleteWidget` – șterge widgetul specificat
- `getAllWidgets` – returnează referința la array-ul ce conține toate widgeturile
- `removeUnlistedWidgets` – șterge o serie de widgeturi primite ca parametru într-un array.

5.4. Spry.DesignTime.Editing.Utils

Clasa conține o colecție de funcții generale ce ușurează editarea widgetului.

- `canInsertWidget` - metoda verifică dacă widgetul poate fi inserat
- `findWidgetContainer` - returnează containerul widgetului (nodul care îl conține)
- `getNewJSObjectName` - returnează numele disponibil pentru obiectul nostru javascript. Fiecare widget este inițializat în constructorul său, unde specificăm id-ul widgetului și opțiunile sale. Pentru un widget existent cu numele "SpryRating1", metoda ne va returna pentru un nou widget, stringul "SpryRating2".

5.5. Spry.DesignTime.Widget.Rating

Clasa **Spry.DesignTime.Widget.Rating** este folosită de inspectorul de proprietăți pentru a ușura modificarea diferitelor stări ale widgetului și pentru a efectua modificările necesare în codul HTML al widgetului. Clasa extinde clasa de bază pentru widgeturi **Spry.DesignTime.Widget.Base**¹¹ Pe lângă starea default, inițială, widgetul mai expune 2 stări, `rated` și `readonly`. `Rated` este folosită atunci când votul a fost efectuat iar `readonly` când nu mai este permisă votarea widgetului.

Cele 2 stări ar putea fi reprezentate în felul următor:



Fig 5.1 starea votat



Fig 5.2 starea readonly

Stările pot fi schimbate din căsuța de selecție "Preview states" din Property Inspector:

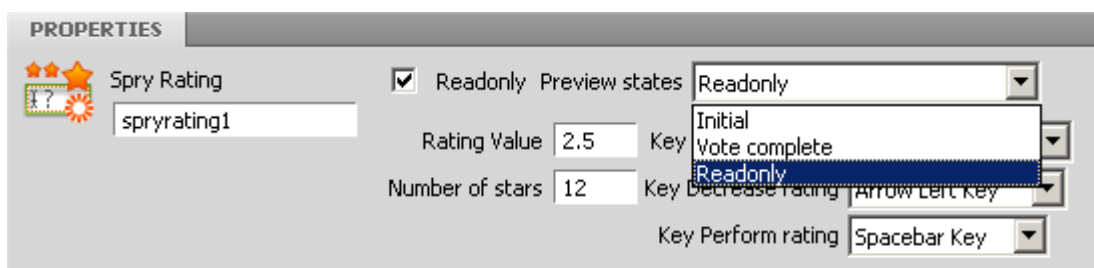


Fig 5.3 Selectbox cu stările widgetului

Starea `readonly` este accesibilă doar dacă a fost bifată în prealabil opțiunea "Readonly".

În cod, stările sunt definite ca elemente ale unui array, și anume arrayul `widgetStates`

```
this.widgetStates = {  
  rated: {
```

¹¹ vezi subcapitolul 5.2

```

        displayName: dw.loadString("spry/widgets/Rating/rated state"),
        stateClass: "ratedClass",
        messageClass: "ratingRatedMsg",
            defaultMessageLocation: dw.loadString("spry/widgets/Rating/rated
message"),
        availability: "rated"
    },
    readOnly: {
        displayName: dw.loadString("spry/widgets/Rating/readonly state"),
        stateClass: "readonlyClass",
        messageClass: "ratingReadOnlyErrMsg",
            defaultMessageLocation: dw.loadString("spry/widgets/Rating/readonly
message"),
        availability: "readOnly"
    }
}

```

DisplayName va fi numele afișat în selectbox, stateClass și messageClass sunt clasele css aplicate elementului în momentul în care o stare este activă (stateClass va fi atașat întregului bloc de cod ce conține widgetul, iar messageClass doar tagului ce conține mesajul afișat), defaultMessageLocation va fi stringul localizat afișat de mesajul stării iar availability va defini în ce condiții este disponibilă starea respectivă.

Arrayul este definit și inițializat în constructorul widgetului, locul în care este apelat și constructorul widgetului de bază:

```
Spry.DesignTime.Widget.Base.call(this, dom, objName, element);
```

Widgetul primește ca parametri referința la DOMul pe care operăm, numele obiectului (widgetului), elementul HTML pe care inserăm widgetul, și argumentele primite de acesta (idul tagului pe care îl inserăm și opțiunile de configurare a widgetului: rating inițial, rating maxim, stare inițială, etc)

Printre metodele de bază ale clasei Spry.DesignTime.Widget.Rating se numără:

- **getAssets** – creează și retunează un array de obiecte de tip Asset¹². Acest array va fi folosit pentru a copia fișierele folosite de widget în directorul de assets. Pe lângă clasa de bază folosită în DesignTime, extensia folosește fișierele de bază incluse în frameworkul Spry. Aceste fișiere pot fi integrate într-o aplicație folosind orice program sau editor text, inclusiv Notepad. Clasa DesignTime și celelalte extensii Dreamweaver sunt însă cele care fac integrarea widgetului în Dreamweaver. Astfel, pentru funcționarea widgetului în browser avem nevoie de Asseturile acestuia.

12 Prin Asset se înțelege un fișier css, javascript sau imagini folosite de widgetul nostru.

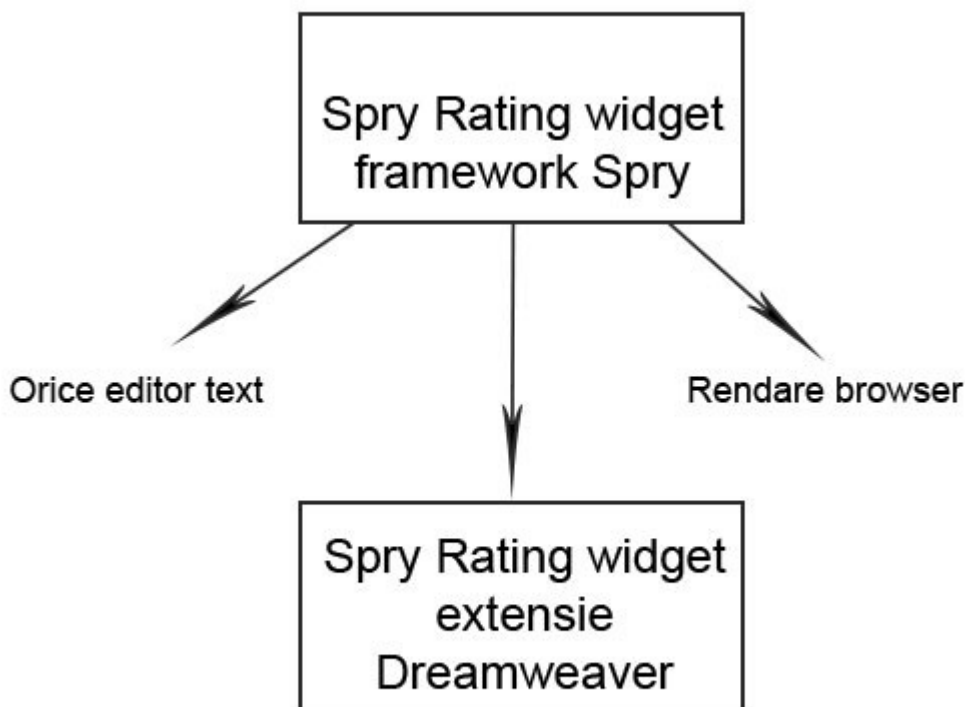


Fig 5.4 relaționare widgetul Spry Rating și extensie Dreamweaver Spry Rating

Widgetul este format dintr-un fișier CSS, un fișier Javascript și fișierele de imagine aferente steluțelor de votare. Modul de populare al arrayului cu elemente de tip asset este următorul:

```

var assets = new Array();
var tempObj;

tempObj = new Object();
tempObj.fullPath = "Shared/Spry/Widgets/Rating/SpryRating.js";
tempObj.file = "SpryRating.js";
tempObj.type = "javascript";
assets.push(tempObj);

tempObj = new Object();
tempObj.fullPath = "Shared/Spry/Widgets/Rating/SpryRating.css";
tempObj.file = "SpryRating.css";
tempObj.type = "link";
assets.push(tempObj);
  
```

Creăm astfel arrayul de asseturi și apoi îl populăm cu obiecte. Fiecare obiect trebuie să aibă definită calea la fișier, numele fișierului și cel mai important, tipul acestuia. Tipul va fi folosit pentru crearea codului HTML corespunzător assetului. De exemplu, pentru un asset de tip link vom genera codul:

```
<link rel="stylesheet" type="text/css" href="SpryRating.css" />
```

- **getTagSnippet** – metodă statică ce returnează un string ce conține codul HTML necesar widgetului. Funcția primește ca parametrii idul ce va fi setat widgetului și nodul pe care este făcută selecția curentă. Acest nod va determina poziția de inserare a noului widget. Va fi inserat și un input de tip text, pentru a permite afișarea/votarea atunci când suportul pentru Javascript lipsește.

- **getNewWidgetConstructorSnippet** – metodă statică ce returnează codul Javascript folosit pentru instanțierea widgetului. Un widget Spry este format din 2 părți. Codul HTML aferent widgetului, și codul HTML pentru inițializarea acestuia. Primește ca parametrii idul widgetului și opțiunile de configurare.

```

Spry.DesignTime.Widget.Rating.getNewWidgetConstructorSnippet           =
function(id, opts)
{
    var strOptions = Spry.DesignTime.Widget.Rating.getObjectAsString(opts);
    var extra = (strOptions != "{}") ? (', ' + strOptions) : "";
    return 'var ' + id + ' = new Spry.Widget.Rating("' + id + '" + extra
+ ');';
};

```

- **addWidgetMessage** – metoda inserează codul HTML aferent stării curente selectate din “Preview states”. De exemplu, pentru mesajul de `readOnly`, va insera codul:

```
<span class="ratingReadOnlyErrMsg">ati votat deja</span>
```

- **getStatesLabels** – metoda returnează textul ce va fi afișat fiecărei stări disponibile, conform configurației widgetului curent. De exemplu, widgetul Spry Rating conține 3 stări: Inițial, Votat și ReadOnly. Starea `readOnly` și mesajul aferent acesteia este disponibil însă doar când este bifată opțiunea “ReadOnly” in Property Inspector.

- **getStatesValues** – dacă metoda de mai sus returna textele afișate de stările disponibile, această metodă returnează valorile disponibile stărilor widgetului.

- **isValidStructure** – verifică dacă structura unui widget este validă. De exemplu, un widget Spry Rating nu ar fi valid în cazul în care nu ar avea nici o steluță de votare, fiecare steluță de votare fiind reprezentată de un tag `span` cu clasa `ratingButton`.

```
<span class="ratingButton"></span>
```

- **getObjectAsString** – metodă statică folosită pentru a serializa un obiect. Primește ca parametru obiectul ce va fi serializat.

- **setOption, getOption, removeOption** – metode folosite pentru a prelua/seta sau șterge opțiunile widgetului din constructorul Javascript.

Exemplu:

```

<script type="text/javascript">
<!--
var spryrating1 = new Spry.Widget.Rating("spryrating1",
{ratingValue:2.5, moveNextKeyCode:39, movePrevKeyCode:37,
doRatingKeyCode:32, readOnly:true});
//-->
</script>

```

În cazul acesta avem setate 5 opțiuni pe widgetul nostru: **ratingValue**: valoarea inițială a votării, **moveNextKeyCode**: tasta folosită pentru a mări votul, **movePrevKeyCode**: tasta folosită pentru a scădea votul, **doRatingKeyCode**: tasta folosită pentru a vota, **readOnly**: dacă widgetul nostru este în starea readOnly sau nu.

- **countStars** – metoda returnează câte stelute poate să conțină maxim un widget, numărând astfel câte spanuri cu clasa "ratingButton" găsește pe widgetul nostru.
- **updateTotalRating** – metoda modifică valoarea inițială a votării cu cea specificată de noi in Property Inspector sau direct în cod.
- **showTabIndex** – pentru a permite votarea cu ajutorul tastaturii, fiecărei stelute îi va trebui atașată proprietatea "tabindex" alături de o valoare număr întreg, ce va defini ordinea de parcurgere a elementelor.

Exemplu de stelută fără taste active:

```
<span class="ratingButton"></span>
```

Stelută cu taste activate:

```
<span class="ratingButton" tabindex="6"></span>
```

6. INTEGRAREA ȘI COMPONENTELE EXTENSIEI

Capitolul prezintă modul de integrare și configurare al extensiei Spry Rating în modul de lucru Dreamweaver Stiletto. În prima parte vom prezenta modul de adăugare al extensiei la o pagină și parametrii pe care îi poate primi widgetul din Property Inspector sau din Code View, în secțiunea de opțiuni. Următoarea parte va prezenta diferitele părți componente ale acestei extensii și vom trece peste secțiunile de cod mai importante.

6.1. Inserarea și modificarea extensiei Spry Rating

Inserarea extensiei se face din meniul de Obiecte, selectând categoria de obiecte Spry.

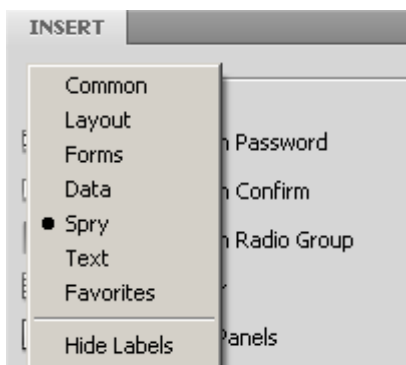


Fig 6.1 Meniul de obiecte, așa cum apare în CS4

Din clasa de obiecte Spry se alege extensia Spry Rating.



Fig 6.2 Inserarea extensiei Spry Rating folosind meniul de obiecte clasic

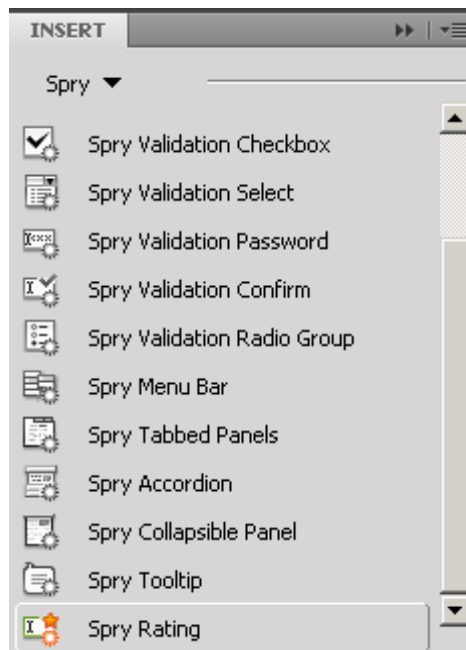


Fig 6.3 Inserarea extensiei Spry Rating folosind noul meniu de obiecte

Prima dată când este inserată o extensie, va fi adăugat codul HTML aferent extensiei:

```
<span id="spryrating1">
  <input type="text" name="text1" id="text1"/>
  <span class="ratingButton"></span>
  <span class="ratingButton"></span>
  <span class="ratingButton"></span>
  <span class="ratingButton"></span>
  <span class="ratingButton"></span>
  <span class="ratingRatedMsg">Thank you for your vote!</span>
</span>
```

codul javascript pentru instanțierea obiectului design time:

```
<script type="text/javascript">
<!--
var spryrating1 = new Spry.Widget.Rating("spryrating1");
//-->
</script>
```

și codul necesar pentru introducerea asseturilor. Inițial, până când documentul nu este salvat, Dreamweaver va folosi niște copii temporare:

```
<script src="file:///C:/Documents and Settings/toshiba/Application Data/
Adobe/Dreamweaver
CS4/en_US/Configuration/Temp/Assets/eam2A.tmp/SpryRating.js"
type="text/javascript"></script>
<link href="file:///C:/Documents and Settings/toshiba/Application
Data/Adobe/Dreamweaver
CS4/en_US/Configuration/Temp/Assets/eam2A.tmp/SpryRating.css"
rel="stylesheet" type="text/css" />
```

Odată ce documentul va fi salvat, vom fi informați de faptul că asseturile corespunzătoare extensiei vor fi copiate în directorul SpryAssets.

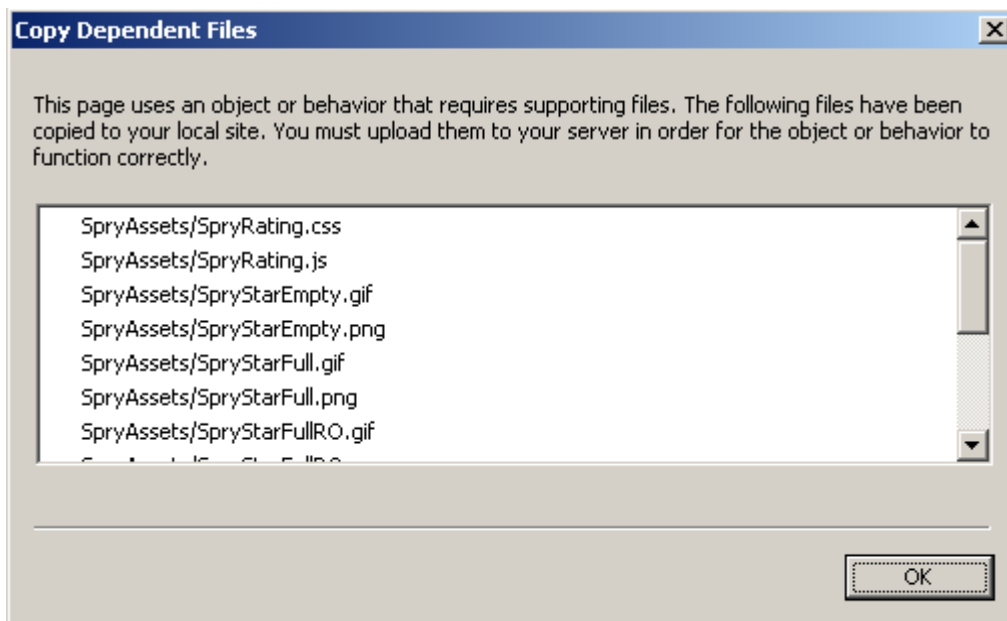


Fig 6.4 Fereastra ne informează despre fișierele ce vor fi copiate

Calea către fișierul Javascript și CSS va fi schimbată către cea relativă:

```
<script src="SpryAssets/SpryRating.js" type="text/javascript"></script>
<link href="SpryAssets/SpryRating.css" rel="stylesheet"
type="text/css" />
```

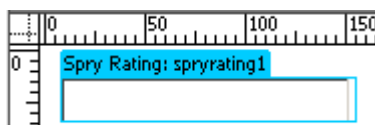


Fig 6.5 Modul în care apare extensia în fereastra Design View.

Editarea atributelor widgetului se face din Property Inspector.

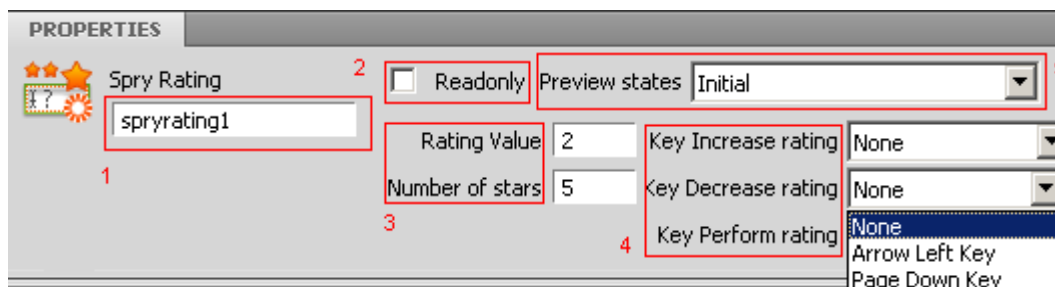


Fig 6.6 Selectarea tastelor pentru votare in Property Inspector

Numeralele reprezintă:

1. Numele widgetului, folosit pentru instanțierea obiectului Spry Rating și pentru idul elementului HTML ce îl va conține.
2. Căsuța Readonly, marchează dacă votarea este permisă sau nu pentru acest widget, setând starea acestuia pe votat, și adăugând în căsuța de message (preview states, nr 5) un nou mesaj ce poate fi definit.

3. În cele 2 căsuțe poate fi definită valoarea inițială a widgetului precum și numărul maxim de steluțe ce pot fi votate.
4. Cele 3 căsuțe de selecție permit selectarea tastelor folosite pentru a efectua votarea cu ajutorul tastaturii.
5. Căsuța de selecție a stărilor permite definirea mesajelor pentru diferitele stări disponibile ale widgetului.

Toate modificările asupra acestor câmpuri vor fi adăugate și în partea de opțiuni a constructorului Javascript definit pe widget, iar unele chiar și pe copii elementului HTML pe care este definită votarea (de exemplu, definirea tastelor de vot va adăuga codul tastelor în constructorul Javascript dar va adăuga și opțiunile de `tabindex`¹³ pe fiecare steluță de votare)

6.2. Componentele extensiei

Extensia Spry Rating este constituită dintr-o serie de extensii Dreamweaver. Acestea au ca scop integrarea widgetului în modul de lucru pentru a ușura modul în care utilizatorul lucrează cu elementele componente ale acestui sistem. De exemplu, un widget Spry poate fi introdus atât din meniul Objects > Spry cât și apelând meniul Commands > Spry. Pentru această integrare sunt necesare scrierea a două extensii, una care să răspundă apelului venit din meniu și cealaltă care să răspundă apelului venit din meniul Objects. Pentru celelalte părți integrante ale unei extensii, cum ar fi Property Inspector, vor fi scrise alte extensii integrante.

În funcție de ce va face extensia/widgetul nostru, vom folosi extensiile Dreamweaver care să răspundă cerințelor noastre.

Principalele extensii folosite de widgetul nostru sunt Comenzile, Obiectele și Inspectorii.

6.2.1. Comenzi

Funcțiile utile expuse de fișierul de comenzi sunt folosite și de Obiectul Spry Rating pentru a returna codul corespunzător constructorului Javascript, elementului HTML ce va conține widgetul și pentru a returna lista de asseturi folosite de acesta.

Comenzile Adobe Dreamweaver pot să execute orice modificare asupra documentului curent folosit de utilizator, asupra oricărui alt document deschis sau asupra oricărui document HTML de pe harddisk. Comenzile pot insera, șterge sau rearanja taguri și atribute HTML, comentarii sau text.

Comenzile sunt fișiere HTML. Secțiunea `body` dintr-un fișier de comandă poate conține

¹³ **tabindex** permite definirea unui index pe fiecare element HTML, permițând astfel navigarea folosind tastele în ordinea acestui index

un formular HTML care să accepte opțiuni pentru comanda noastră. Partea desemnată de `head` conține funcțiile Javascript care procesează datele primite din formular și controlează modificările care sunt aduse documentului.

Fișierele folosite pentru a crea comenzi:

Cale	Fișier	Descriere
Configuration/Commands	numecomanda.htm	specifică interfața
Configuration/Commands	numecomanda.js	conține funcțiile ce vor fi executate

6.2.1.1 Modul de lucru al comenzilor.

Când utilizatorul alege o opțiune din meniul de comenzi, următoarele evenimente au loc:

- Dreamweaver apelează funcția `canAcceptCommand()` pentru a determina dacă opțiunea din meniu poate fi apelată sau nu. Dacă funcția va returna fals, opțiunea din meniu va apărea ca neapelabilă, iar execuția se va opri. În caz adevărat, execuția va continua iar opțiunea din meniu poate fi apelată.
- Utilizatorul selectează o comandă din meniu.
- Dreamweaver apelează funcția `receiveArguments()` pentru a procesa argumentele pe care comanda le-ar putea primi. Apelul funcției este opțional, și va fi executat doar în cazul în care funcția este definită în comanda noastră.
- Dacă e definită, Dreamweaver apelează funcția `commandButtons()` pentru a determina ce butoane vor apărea în partea dreaptă a ferestrei de dialog. Conținutul ferestrei de dialog este definit în formularul din secțiunea `body`, așa cum am explicat mai sus, însă butoanele de control (de ex: Ok, Cancel) sunt definite de această funcție.

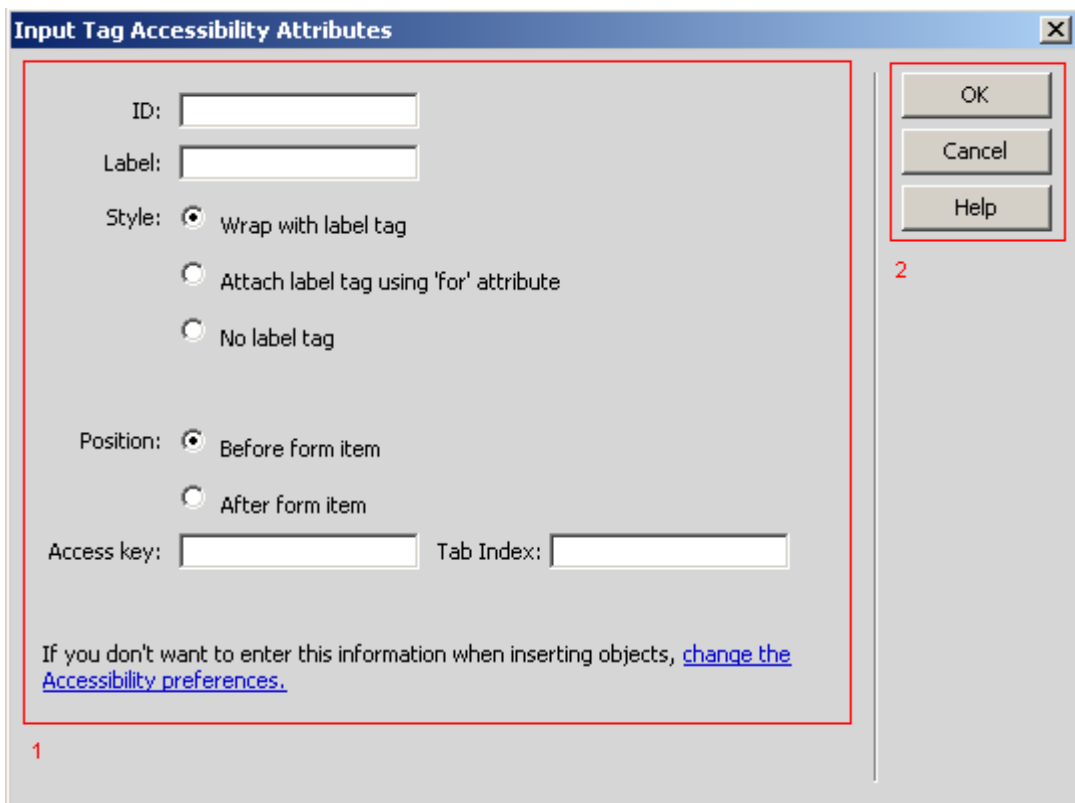


Fig 6.7 formular rendat ca și dialog (1. modul în care va apărea formularul definit în comandă, 2. butoanele definite de funcție pentru această comandă)

- Dreamweaver scanează apoi fișierul html al comenzii după un tag `form`. Dacă e găsit, Dreamweaver apelează funcția `windowDimensions()` pentru a redimensiona această fereastră. Dacă funcția nu este definită, Dreamweaver va redimensiona automat căsuța de dialog.
- Dacă fișierul HTML al comenzii definește handlerul `onLoad` pe tagul `body`, Dreamweaver va executa apoi acest cod.

Următorii pași vor fi executați doar dacă este definit un formular pentru comandă.

- Utilizatorul selectează opțiunile pentru comandă. Dreamweaver apelează handlerele definite pe câmpurile existente.
- Utilizatorul face apoi click pe una dintre opțiunile definite de comanda `commandButtons()`
- Dreamweaver execută codul asociat. Căsuța de dialog rămâne vizibilă până când din comandă este apelată funcția `window.close()`.

Comenzile apar automat în meniul de comenzi. Pentru a preveni apariția unei comenzi în meniu, va trebui adăugată următoarea linie de cod pe prima linie a fișierului HTML.

```
<!-- MENU-LOCATION=NONE -->
```

Comenzile pot fi apelate și din alte extensii folosind funcția `dw.runCommand()`

6.2.1.2 Comanda Spry Rating

Fișierul HTML al comenzii arată în felul următor:

```
<!-- MENU-LOCATION=NONE -->
<html xmlns:mmstring="http://www.macromedia.com/schemes/data/string/">
<head>
<!-- Copyright 2006-2007 Adobe Systems Incorporated. All rights
reserved. -->
<title><MMString:LoadString id="spry/widgets/Rating/title" /></title>
<script type="text/javascript"
src="../../Shared/Spry/DesignTime/WidgetBase.js" ></script>
<script type="text/javascript"
src="../../Shared/Spry/Widgets/Rating/DesignTime/SpryRatingDesignTime.js"
></script>
<script type="text/javascript"
src="../../Shared/Common/Scripts/AssetInfoClass.js"></script>
<script type="text/javascript"
src="../../Shared/Common/Scripts/dwscripts.js"></script>
<script type="text/javascript"
src="../../Shared/MM/Scripts/CMN/TemplateUtils.js"></script>
<script type="text/javascript"
src="../../Shared/Spry/DesignTime/EditingUtils.js"></script>
<script type="text/javascript"
src="../../Objects/Forms/formInsert.js"></script>

<script type="text/javascript" src="SpryRating.js"></script>

</head>
<body onLoad="createWidget()">
</body>
</html>
```

Prima linie de cod nu permite adăugarea automată a acestei comenzi în meniul Commands al aplicației. Este definit apoi namespaceul `mmstring` folosit pentru manipularea stringurilor folosite pentru traducerea elementelor widgetului. Secțiunea `<MMString:LoadString id="spry/widgets/Rating/title" />` încarcă în titlul comenzii textul reprezentat de șirul cu id `spry/widgets/Rating/title`. Fișierul `SpryRatingStrings.xml`¹⁴ definește toate stringurile folosite de acest widget.

Urmează apoi inserarea diferitelor scripturi folosite de extensia noastră. Deoarece vom crea nu doar un widget visual, ci și un obiect care va reprezenta widgetul în DesignTime și care ne va permite o ușoară manipulare a acestuia, vom include scriptul `SpryRatingDesignTime.js`. Din moment ce clasa `Spry.Widget.Rating` extinde clasa de bază a widgeturilor, o vom include și pe aceasta din fișierul `WidgetBase.js`.

¹⁴ De regulă toate stringurile folosite de extensiile Dreamweaver sunt definite în fișiere XML salvate în directorul `Configuration/Strings`

`AssetInfoClass.js` este folosit pentru adăugarea asseturilor folosite de extensie. Pe lângă celelalte scripturi comune folosite pentru a genera formulare de inserare precum și accesul funcțiilor folosite de `des` (`dwscripts.js`), ultimul script inserat va fi cel în care avem logica extensiei, `SpryRating.js`. La încărcarea comenzii, Dreamweaver va apela funcția `createWidget()`, definită în fișierul `SpryRating.js`.

Fișierul Javascript al comenzii arată în felul următor:

```
function createWidget()
{
    // codul comenzii
}

function createWidgetStr()
{
    return RETURN_TAG;
}

function getScriptStr()
{
    return SCRIPT_STR;
}

function getAssetList()
{
    return ASSET_LIST;
}

function getWidgetID() {
    return ID;
}
```

Metodele comenzii nu vor fi expuse detaliat deoarece această comandă nu implementează nici unul dintre apelurile enumerate mai sus, în afară de handlerul `onLoad` pe tagul `body`. Metoda apelată la `onLoad`, `createWidget()`, va returna în câteva variabile specifice comenzii codul pentru constructorul widgetului, codul HTML al widgetului, lista de asseturi și idul widgetului. Aceste date vor fi apelate și de Obiectul extensiei, și pot fi apelate de orice altă extensie.

6.2.2. Obiecte

Scopul obiectelor este de a insera anumite șiruri de cod în documentul utilizatorului. Un obiect apare într-un tab pe bara de Insert sau în meniul Insert când obiectul este stocat în directorul `Configuration/Objects`. Adăugarea unui nou obiect constă în crearea unui subdirector în locația menționată anterior, precum și în editarea fișierului `insertbar.xml` care

definește categoriile ce apar în taburile de Inserare.

Extensiile obiect au 3 componente:

- Fișierul obiect, care definește ce va fi inserat în document.
- O imagine de 18x18 pixeli care va reprezenta vizual obiectul în bara de Inserare.



Fig 6.8 Obiectul Hyperlink pe bara de Inserare, categoria Common

- Fișierul insertbar.xml care definește unde va apărea obiectul pe bara de Inserare.

6.2.2.1 Modul de lucru al obiectelor

Când utilizatorul adaugă un nou obiect, următoarele evenimente au loc:

- Dreamweaver apelează funcția `canInsertObject()` pentru a determina dacă putem introduce obiectul în documentul curent.
- Fișierul HTML al obiectului este scanat după tagul `form`. Dacă un formular este găsit, Dreamweaver apelează funcția `windowDimensions()`, dacă este definită, pentru a redimensiona fereastra. Dacă nu există nici un formular definit, acest pas este sărit.
- Dacă este afișată o căsuță de dialog, utilizatorul poate introduce datele necesare și apoi alege OK.
- Este apelată acum funcția `objectTag()` și valoarea returnată de această funcție este inserată în documentul curent la sfârșitul selecției curente (selecția curentă nu este înlocuită).
- Dacă funcția `objectTag()` nu este găsită, Dreamweaver caută funcția `insertObject()` și o apelează pe aceasta în schimb.

6.2.2.2 Obiectul Spry Rating

Ca orice extensie obiect, obiectul Spry Rating este format din 3 fișiere: o imagine, fișierul HTML corespunzător obiectului și fișierul Javascript corespunzător codului executat de obiect.

Fișierul HTML al obiectului arată în felul următor:

```
<!-- MENU-LOCATION=NONE -->
<html xmlns:MMString="http://www.macromedia.com/schemes/data/string/">
  <head>
    <title><MMString:LoadString id="spry/widgets/Rating/title" /></title>

    <script type="text/javascript"
      src="../../Shared/Common/Scripts/dwscripts.js"></script>
```

```

<script type="text/javascript"
src="../../Shared/MM/Scripts/CMN/TemplateUtils.js"></script>
<script type="text/javascript"
src="../../Shared/Spry/DesignTime/EditingUtils.js"></script>
<script type="text/javascript"
src="../../Shared/Spry/DesignTime/WidgetBase.js"></script>
<script type="text/javascript"
src="../../Shared/Spry/Widgets/Rating/DesignTime/SpryRatingDesignTime.js">
</script>
<script type="text/javascript" src="SpryRating.js"></script>
</head>

<body>
</body>
</html>

```

Practic sunt încărcate aceleași fișiere Javascript folosite pentru Editare și crearea widgeturilor ca și în cazul extensiilor comenzi. La final este încărcat fișierul Javascript corespunzător extensiei obiect:

```

function canInsertObject() {
    var dom = dw.getDocumentDOM();
    var retVal = true;

    retVal = (dom.getIsLibraryDocument() == false);
    if( retVal )
    {
        retVal = Spry.DesignTime.Editing.Utils.canInsertWidget(
            dom, "Spry.Widget.Rating");
    }

    return retVal;
}

function isDOMRequired() {
    return true;
}

function insertObject() {
    var dom = dreamweaver.getDocumentDOM();
    var selNode = dom.getSelectedNode();

    if( !Spry.DesignTime.Editing.Utils.canInsertWidget(dom) )
    {
        return;
    }

    var cmdFile = dreamweaver.getConfigurationPath()
        + "/Commands/SpryRating.htm";
    var cmdDOM = dreamweaver.getDocumentDOM(cmdFile);
    dreamweaver.popupCommand("SpryRating.htm");
}

```

```

var assetList = cmdDOM.parentWindow.getAssetList();
var scriptStr = cmdDOM.parentWindow.getScriptStr();
var retStr = cmdDOM.parentWindow.createWidgetStr();

if( retStr )
{

    if (assetList && assetList.length)
        dom.copyAssets(assetList);

    if (scriptStr)
        dom.addJavaScript(scriptStr, false);

    dom.insertHTML(retStr, false);
}
}

```

Funcția `canInsertObject()` verifică dacă obiectul poate fi inserat în documentul curent. Atâta timp cât documentul curent nu este un document librărie și putem insera un widget de tip `Spry.Widget.Rating`, metoda ne va permite să apelăm celelalte metode implementate în API-ul extensiilor obiect din Dreamweaver. Metoda este apelată când utilizatorul dorește să insereze un obiect din `Insertbar`.

Metoda `isDOMRequired()` specifică dacă obiectul are nevoie de un DOM valid pentru a putea efectua inserarea. Dacă funcția returnează adevărat, Dreamweaver sincronizează vederile de Cod și Design ale documentului înainte de a executa comenzile asociate.

Următoarea metodă API apelată este `insertObject()`. Practic este metodă care execută tot codul necesar widgetului nostru. Prima dată este generată o referință la DOMul documentului pe care lucrăm, apoi luăm nodul curent selectat. Executăm apoi extensia comandă definită la primul pas, și apelate funcțiile utile din fișierul de comandă, funcții ce returnează codul HTML și Javascript necesar widgetului, precum și lista de asseturi necesare acestuia. Din moment ce și extensia obiect execută același cod ca și extensia comandă, este logică alegerea de a folosi codul din comandă în ambele extensii:

```

var assetList = cmdDOM.parentWindow.getAssetList();
var scriptStr = cmdDOM.parentWindow.getScriptStr();
var retStr     = cmdDOM.parentWindow.createWidgetStr();

```

Urmează apoi inserarea codului pentru widget:

```

if( retStr )
{
    if (assetList && assetList.length)
        dom.copyAssets(assetList);
}

```

```

    if (scriptStr)
        dom.addJavaScript(scriptStr, false);

    dom.insertHTML(retStr, false);
}

```

Funcțiile `objectTag()` și `insertObject()` se exclud reciproc. Dacă există una dintre ele, cealaltă nu mai trebuie definită. Care este diferența între ele și când ar trebui folosite? Ambele sunt folosite pentru a introduce cod în documentul utilizatorului. `objectTag()` lucrează însă pe selecția curentă și permite introducerea codului după nodul selectat. `insertObject()` ar trebui folosit însă în următoarele cazuri:

- Când dorim să introducem cod în mai multe porțiuni
- Când dorim să introducem cod în altă zonă decât după cea selectată
- Când dorim să validăm datele introduse înainte de a le afișa

În cazul curent am folosit `insertObject()` tocmai datorită faptului că obiectul introduce cod în mai multe zone, atât cod HTML când creează widgetul, cât și cod Javascript când instanțiem widgetul și când definim legătura la asseturile acestuia.

6.2.3. Translatori

Translatorii traduc cod markup specializat – cod PHP, JSP, CFML, etc – în cod ce poate fi citit și afișat de Dreamweaver. În Dreamweaver pot fi traduse atribute ale tagurilor sau chiar taguri și blocuri de cod întregi. Toți translatorii sunt fișiere HTML.

Tagurile sau blocurile de cod traduse trebuie să fie definite în zone blocate pentru a nu afecta codul original. Atributele traduse nu au nevoie de aceste zone blocate însă, ceea ce ușurează cu mult inspectarea lor.

6.2.3.1 Modul de lucru al translatorilor

Dreamweaver tratează toți translatorii la fel, indiferent dacă translatează taguri întregi sau doar atribute. La pornire, Dreamweaver citește toate fișierele din directorul `Configuration/Translators` și apelează funcția `getTranslatorInfo()` pentru a obține informații despre translator. Dreamweaver ignoră fișierele translator care nu au această funcție definită.

Dreamweaver apelează apoi funcția `translateMarkup()` de fiecare dată când utilizatorul adaugă sau modifică conținutul existent. Cazurile în care este apelată această funcție din translatorii sunt următoarele:

- deschiderea unui fișier în Dreamweaver

- trecere la Design View din Code View
- schimbarea proprietăților unui obiect în documentul curent
- inserarea unui obiect (din Insertbar)
- reîncărcarea documentului curent
- aplicarea unui template documentului
- adaugă conținut folosind clipboardul sau drag and drop
- invocă o comandă, behavior, server behavior, inspector de proprietăți sau altă extensie care setează proprietatea `innerHTML` sau `outerHTML` al oricărui tag din documentul curent.

6.2.3.2 Traducătorul de Spry Widgets

Fișierul HTML folosit pentru traducerea widgeturilor este `SpryWidget.htm` din directorul `Configuration\Translators`. Pentru toate widgeturile Spry este definit un traducător comun, de aceea nu există un fișier `SpryRating` specific acestui widget. Acesta conține în secțiunea `head` și logica extensiei, de aceea nu este nevoie de inserarea unui fișier Javascript extern. Codul care ne interesează este acesta:

```
<script type="text/javascript">
//expresie regulata pentru a gasi toate widgeturile
var    spryWidgetIdRegExp    =    /(?:var)?\s+(\w+)\s*?=\s*?new\s+?
(Spry\.Widget\.?\w+)\s*?\s*(\s*?['"](?:[^\s"]+|['"]\s|.)\s*?)\s*/g;

var spryWidgetConstructors = MM.SPRY_WidgetConstructors;

function translateDOM( dom, sourceStr )
{
    if( typeof dom == 'undefined' )
    {
        return;
    }

    translateWidgetsInDOM( dom,          sourceStr,          spryWidgetIdRegExp,
spryWidgetConstructors);
}

function getTranslatorInfo()
{
    returnArray = new Array(7);

    returnArray[0] = "SPRY_WIDGET"
    returnArray[1] = "Spry Widget"
    returnArray[2] = "0"
    returnArray[3] = "1"
    returnArray[4] = "Spry.Widget."
```

```

returnArray[5] = "byString"
returnArray[6] = "50"

return returnArray;
} // getTranslatorInfo()
</script>

```

Inițial Dreamweaver apelează funcția `getTranslatorInfo()`. Această funcție oferă informații legate de translator și de fișierele pe care operează acesta. Funcția returnează un array în care elementele trebuie să apară în ordinea următoare, iar fiecare poziție reprezintă:

- `translatorClass` - definește în mod unic translatorul.
- `title` – descrie translatorul în maxim 40 de caractere
- `nExtensions` – specifică numărul de extensii de fișiere pe care rulează. Dacă e zero, înseamnă că vom rula translatorul pe orice tip de fișier. În cazul acesta, `nRegExps` este următorul element din array.
- `extension` – specifică o extensie de fișier (ex: "htm", "php") pe care acest translator rulează. Acest element ar trebui să conțină un array egal cu numărul de `nExtensions` definit precedent.
- `nRegExps` – specifică numărul de expresii regulate care vor fi rulate. Dacă valoarea acestei variabile e zero, `runDefault` e următorul element definit în array.
- `regExps` – specifică o expresie regulată ce va fi rulată pe expresia noastră. Arrayul ar trebui să conțină același număr de elemente ca cel specificat de `nRegExps` în pasul precedent.
- `runDefault` – specifică când va fi rulat translatorul. Următoarele valori sunt posibile:
 1. *allFiles* – translatorul este executat întotdeauna
 2. *noFiles* – translatorul nu se execută niciodată
 3. *byExtension* – se execută doar la prezența fișierelor care au una dintre extensiile definite mai sus.
 4. *byExpression* – se execută la întâlnirea unora dintre extensiile regulate definite mai sus
 5. *byString* – se execută dacă translatorul întâlnește un anumit șir definit.
- `priority` – specifică prioritatea cu care acest translator rulează. De exemplu, pe tagul `img`, de reprezentare a unei imagini, pot fi scrise numeroase translatoare, în funcție de întâlnirea anumitor atribute. Prioritatea va determina care dintre translatorii definiți pentru acel tag va executa translatarea. 0 este cea mai mare prioritate, iar 100

cea mai mică.

Pentru widgetul nostru, codul va fi interpretat în felul următor:

```
Clasa translatorului
returnArray[0] = "SPRY_WIDGET"
titlul translatorului
returnArray[1] = "Spry Widget"
numărul de extensii pe care rulăm widgetul (toate)
returnArray[2] = "0"
numărul de expresii regulate verificate de translator
returnArray[3] = "1"
verificăm existența stringului Spry.Widget
returnArray[4] = "Spry.Widget."
verificarea e făcută după existența șirului
returnArray[5] = "byString"
prioritatea translatorului (medie)
returnArray[6] = "50"
```

Un exemplu simplu și practic de creare și reprezentare al unui translator l-ar putea reprezenta o extensie care integrează un anumit gadget Google. Gadgeturile Google sunt în esență asemănătoare widgeturilor Spry sau widgeturilor oferite de alte aplicații/frameworkuri. Diferă numele însă funcționarea în esență este aceeași.

Pentru început, putem lua ca exemplu gadgetul de "Curs valutar".

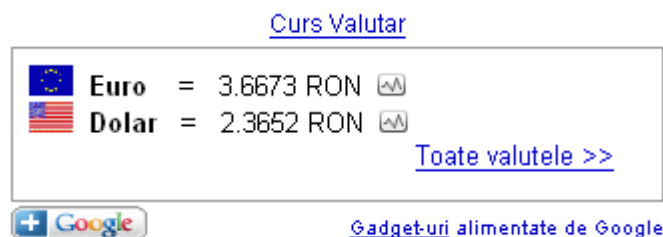


Fig 6.9 Gadgetul Google "Curs Valutar" așa cum arată pe o pagină de internet

Codul HTML folosit pentru a reda acest gadget este următorul:

```
<script src="http://gmodules.com/ig/iframe?url=http://andreiterente.googlepages.com/cursvalutar.xml&synd=open&w=318&h=68&title=Curs+Valutar&border=%23ffffff%7C3px%2C1px+solid+%23999999&output=js"></script>
```

Intervine însă următoarea problemă. Gadgetul este afișat folosind doar cod Javascript extern, iar reprezentarea are loc când pagina este vizualizată în browser. Dreamweaver CS4 aduce o nouă vedere, "Live view" ce permite și executarea codului Javascript din pagină și preluarea rezultatelor în direct. În timpul editării noi folosim însă vederile de Cod sau Design. În vederea de Cod vom vedea codul Javascript. Ce vom vedea însă în vederea de Design?

Prin scrierea unui translator pentru această extensie, putem arăta o imagine care să reprezinte gadgetul în vederea Design. Am putea folosi chiar următoarea imagine:



Fig 6.10 posibilă imagine pentru extensie, în vederea Design

Astfel putem reprezenta vizual extensia noastră, și putem ușura și modul în care aceasta este selectată. Codul translatat este interpretat de Dreamweaver, însă nu este afișat în codul paginii, deoarece el este procesat transparent de aplicație. Este reprezentat doar în mod vizual, și doar în interiorul aplicației, pentru a facilita lucrul cu extensiile oferite de aceasta.

Întreg procesul de traducere este executat de funcția API `translateMarkup()`, care va returna în final stringul translatat, care va fi apoi reprezentat de Dreamweaver. În exemplul de mai sus funcția ar procesa stringul referit de codul Javascript și ar returna următorul cod:

```

```

Codul returnat nu va fi prezent în sursa documentului ci doar va fi interpretat de Dreamweaver.

Schimbarea atributelor gadgetului, cum ar fi titlul, tipul bordurii, lățimea și lungimea vor putea fi schimbate cu ajutorul Inspectorului de Proprietăți. În pașii următori vom prezenta ce este acesta și modul în care este integrat de extensia Spry Rating.

6.2.4. Inspectorul de proprietăți

Inspectorul de proprietăți este cel care permite editarea atributelor diferitelor blocuri de cod sau al tagurilor din documentul curent. Prescurtat de obicei PI (Property Inspector), Inspectorul de proprietăți este una dintre cele mai utilizate extensii în lucrul cu documente.

Dreamweaver vine cu o suită de PI integrați ce permit setarea proprietăților pentru majoritatea tagurilor HTML standard. Acești inspectori sunt parte integrantă a sistemului, și nu vor putea fi găsiți în directorul `Configuration/Inspectors`, unde sunt definiți inspectori customizați, însă pot fi suprascriși dacă definim inspectori pe tagurile standard.

Prima linie întâlnită în inspectori este reprezentată de un comentariu în următorul format:

```
<!-- tag : tagNameOrKeyword, priority : 1to10, selection : exactOrWithin, hline, vline, serverModel-->
```

unde:

- `tagNameOrKeyword` – reprezintă tagul ce va fi inspectat, sau următoarele cuvinte cheie: `*COMMENT*` (pentru comentarii), `*LOCKED*` (pentru regiuni blocate), sau `*ASP*` (pentru taguri ASP)]

- `1to10` – reprezintă prioritatea inspectorului. Mai mulți inspectori pot fi definiți pentru aceleași taguri, așa că trebuie să știm care inspector se va executa în final. 1 e valoarea cea mai mică, iar 10 înseamnă că are prioritate maximă.
- `exactOrWithin` – indică dacă trebuie să fie respectată selecția întocmai sau poate să conțină doar o parte din ea. Exemplu, o căutare exactă pe tagul `IMG` va returna elementul în PI doar dacă avem selectat acest tag. O selecție doar pe `IM` sau `MG` nu va executa inspectorul. O selecție parțială însă ar returna adevărat.
- `hline` (opțional) – generează introducerea unei linii orizontale în PI între partea de sus și cea de jos a acestuia.
- `vline` (opțional) – generează o linie verticală între numele tagului și restul proprietăților din PI.
- `serverModel` (opțional) – indică modelul de server folosit de PI. Dacă acesta nu corespunde cu modelul server folosit de documentul curent, atunci PI nu se execută. De exemplu, dacă lucrăm pe un document `PHP` și `serverModel` e definit ca `ASP`, PI nu se va executa.

Următorul exemplu definește comentariul pentru un inspector pe tagul customizat “`MYSQL`”, cu o verificare exactă pe selecție, de prioritate mare, definită pe server model `PHP`.

```
<!--      tag:MYSQL,      priority:8,      selection:exact,hline,vline,
serverModel:PHP-->
```

6.2.4.1 Modul de funcționare al PI

La pornire, Dreamweaver citește prima linie din fiecare inspector, și anume comentariul ce definește regulile de inspectare. Când utilizatorul face o selecție pe documentul curent, sau mută cursorul pe altă poziție, următoarele evenimente au loc:

- Dreamweaver caută inspectorii care au tipul de selecție `within`.
- Dacă găsește așa ceva, va căuta inspectorii cu tip de selecție `within` care corespund tagului curent selectat. Dacă nu găsește astfel de inspectori, îi va căuta pe cei cu selecția de tip `exact`.
- Pentru primul tag găsit care are mai mulți inspectori, Dreamweaver apelează funcția API `canInspectSelection()`. Dacă apelul returnează fals, Dreamweaver elimină acest inspector dintre candidați.
- Dacă rămân mai mulți candidați și după acest apel, Dreamweaver sortează inspectorii după prioritate.
- Dacă și după această selecție rămân inspectori multipli cu aceeași prioritate, Dreamweaver îi sortează alfabetic.
- Este apelată apoi funcția `inspectSelection()` pentru preluarea informațiilor

corespunzătoare selecției curente.

- Sunt executate apoi evenimentele atașate controalelor definite în PI. De exemplu, pentru schimbarea numelui, este folosit un input de tip text, căruia îi putem atașa o metodă la evenimentul `onChange`.

6.2.4.2 PI pentru Spry Rating

Fișierele Inspectorului se găsesc în Configuration/Inspectors, și sunt reprezentate de:

Cale	Fișier	Descriere
Configuration/Inspectors	spry_rating.htm	formular ce definește PI
Configuration/Inspectors	spry_rating.js	codul folosit de extensia PI
Configuration/Inspectors	spry_rating.png	imaginea folosită în PI

Fișierul HTML conține codul markup ce formează formularul afișat de Property Inspector. Acesta permite modificarea proprietăților extensiei Spry Rating prin interfața sa simplă și intuitivă, dându-ne acces la majoritatea opțiunilor de configurare a extensiei.

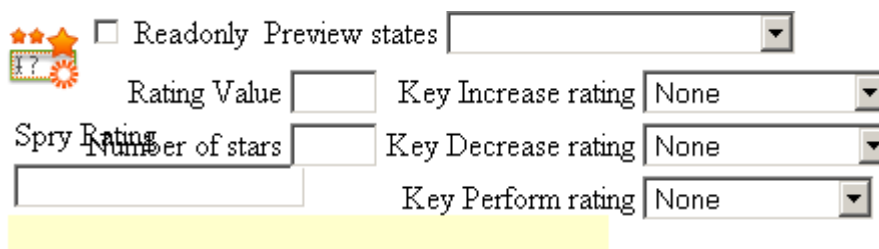


Fig 6.11 PI deschis în browser

Formularul HTML este apoi interpretat și rendat de Dreamweaver. Pentru fiecare element al formularului, este executat codul atașat evenimentului controlului pe care lucrăm. În general se execută funcții la evenimentul `onChange()`, majoritatea controalelor fiind căsuțe de text.

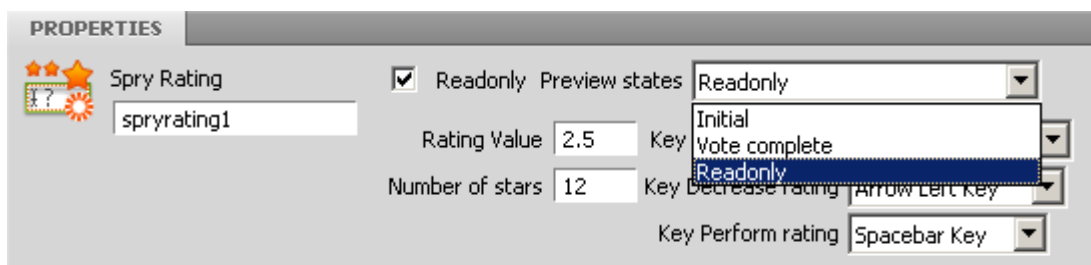


Fig 6.12 extensia PI pentru Spry Rating în Dreamweaver

Pe lângă formularul rendat, fișierul HTML include și referințele Javascript la clasele comune folosite de extensia noastră, inclusiv `WidgetBase.js`, `WidgetManager.js` și

SpryRatingDesignTime.js.

Partea care face toată munca extensiei este însă fișierul Javascript, `spry_rating.js`:

Acesta expune metodele tipice¹⁵ unei astfel de extensii, `canInspectSelection()`, `inspectSelection()`, precum și alte metode customizate necesare extensiei Spry Rating.

Primele linii sunt reprezentate însă de variabilele globale utilizate pentru a prelua informațiile din formularul PI. Are sens din punct de vedere funcțional și al memoriei să definim variabile globale pentru extensie, deoarece la un moment dat va fi selectat și afișat un singur inspector, chiar dacă pe pagina noastră avem de exemplu 10 widgeturi de votare. Tocmai din acest motiv nu vom defini variabile locale pentru orice extensie pentru a prelua aceste date.

```
// ține minte idul widgetului (idul elementului HTML ce conține widgetul)
var WIDGET_ID;
// referință la căsuța de bifare Readonly
var READONLY;
// referință la căsuța text ce conține valoarea curentă a votării
var RATING_VALUE;
// referință la căsuța text ce conține valoarea maximă ce poate fi votată
var NUMBER_OF_STARS;
// referință la căsuța de selecție pentru tastele care execută votarea
var KB_DO_RATING;
// referință la căsuța de selecție pentru tastele care măresc valoarea votării
var KB_RATING_UP;
// referință la căsuța de selecție pentru tastele care micșorează valoarea votării
var KB_RATING_DOWN;
// valoarea numerică a numărului de stele posibile
var numberOfStarsValue;
// valoarea votării
var ratingValue;
// valoarea tastei selectate, folosită pt toate cele 3 căsuțe de taste succesiv
var kbRateVal;
```

Variabilele sunt inițializate în funcția `initializeUI()`:

```
function initializeUI()
{
    WIDGET_ID = dwscripts.findDOMObject("widgetId");
    READONLY = new CheckBox("", "readOnly");
    READONLY.initializeUI();
    WIDGET_STATES = new ListControl("widgetStates", null, true);
    RATING_VALUE = dwscripts.findDOMObject("ratingValue");
    NUMBER_OF_STARS = dwscripts.findDOMObject("numberOfStars");
    KB_DO_RATING = new ListControl("kbDoRating", null, true);
    KB_RATING_UP = new ListControl("kbRatingUp", null, true);
    KB_RATING_DOWN = new ListControl("kbRatingDown", null, true);
    WIDGET_ID.value = "";
```

15 vezi subcapitolul 6.2.4.1 – Modul de funcționare al PI

```
}
```

Funcția `canInspectSelection()` a fost descrisă mai sus. Ea este prima funcție API apelată pe un inspector, fiind apelată de fiecare dată când selecția în documentul curent este schimbată. În funcție de rezultatul returnat de această funcție, vom determina dacă inspectorul curent poate procesa elementul selectat sau nu.

Pentru extensia `Spry Rating`, funcția `canInspectSelection()` arată în felul următor:

```
function canInspectSelection()
{
    var bCanInspectSelection = false;
    var dom = dw.getDocumentDOM();
    var selectedNode = dom.getSelectedNode(true, false, true);
    var attr;
    if( selectedNode && selectedNode.getTranslatedAttribute )
    {
        attr = selectedNode.getTranslatedAttribute('Spry.Widget.Rating');
    }

    if( attr && attr.length > 0 )
    {
        bCanInspectSelection = true;
var widgetMgr = Spry.DesignTime.Widget.Manager.getManagerForDocument(dom);

        if( !widgetMgr.getWidget('Spry.Widget.Rating', selectedNode.id ) )
        {
            dom.runTranslator("Spry Widget");

            if (!widgetMgr.getWidget('Spry.Widget.Rating', selectedNode.id ))
            {
                bCanInspectSelection = false;
            }
        }
    }
    return bCanInspectSelection;
}
```

Inițial inspecția noastră ia nodul curent selectat în document. Dacă acesta prezintă atributul translatat `"Spry.Widget.Rating"` atunci inspecția noastră poate fi validă. Apelăm apoi widget manager pentru documentul curent, și vedem dacă acesta conține în lista sa și widgetul nostru (widget manager conține o listă cu toate widgeturile Spry de pe pagina curentă). Dacă widgetul nu face parte din listă, atunci probabil că widget managerul nu este sincronizat cu noile modificări, iar selecția noastră nu mai este validă, widgetul fiind probabil șters între timp. În funcție de validitatea acestei selecții, funcția returnează dacă poate fi apelată extensia PI pe nodul curent.

Dacă inspecția este validă, este executată apoi funcția `inspectSelection()`.

```

function inspectSelection()
{
    initializeUI();

    var dom = dw.getDocumentDOM();
    var selectedNode = dom.getSelectedNode(true, false, true);

    if (!canInspectSelection())
    {
        return;
    }
    var containerId = selectedNode.id;
    var widgetMgr = Spry.DesignTime.Widget.Manager.getManagerForDocument(dom);
    var widgetObj = widgetMgr.getWidget('Spry.Widget.Rating', containerId);

    WIDGET_ID.value = containerId;

    if( !widgetObj || !widgetObj.isValidStructure() )
    {
        displayTopLayerErrorMessage(dw.loadString("spry/widget/alert/broken
structure"));
        return;
    }
    clearTopLayerErrorMessage();

    widgetObj.refresh();

    if( widgetObj.opts )
    {
        ratingValue = (typeof widgetObj.opts.ratingValue == "undefined") ?
            "" : dwscripts.trim(widgetObj.opts.ratingValue.toString());
        RATING_VALUE.value = ratingValue;

        WIDGET_STATES.setAll(widgetObj.getStatesLabels(),
            widgetObj.getStatesValues());
        WIDGET_STATES.pickValue(widgetObj.getDisplayedState());
        kbRateVal = (typeof widgetObj.opts.doRatingKeyCode == "undefined") ?
            "" : dwscripts.trim(widgetObj.opts.doRatingKeyCode.toString());
        KB_DO_RATING.pickValue(kbRateVal);
        kbRateVal = (typeof widgetObj.opts.moveNextKeyCode == "undefined") ?
            "" : dwscripts.trim(widgetObj.opts.moveNextKeyCode.toString());
        KB_RATING_UP.pickValue(kbRateVal);
        kbRateVal = (typeof widgetObj.opts.movePrevKeyCode == "undefined") ?
            "" : dwscripts.trim(widgetObj.opts.movePrevKeyCode.toString());
        KB_RATING_DOWN.pickValue(kbRateVal);
        KB_DO_RATING.getValue();
    }
    else

```

```

{
    RATING_VALUE.value = "";
}
READONLY.checkBox.checked = (widgetObj.getOption("readOnly", true));

numberOfStarsValue = widgetObj.countStars();
NUMBER_OF_STARS.value = numberOfStarsValue;
}

```

Toate variabilele extensiei sunt initializate în primul pas, la apelul funcției initializeUI(). Urmează apoi selecția nodului curent, și returnarea widgetului corespunzător acestui nod. După ce returnăm widgetul corespunzător, este selectată partea de opțiuni a acestuia, și anume lista de opțiuni definită în constructorul Javascript al widgetului.

```

<script type="text/javascript">
    var rate = new Spry.Widget.Rating("spryrating1", {readOnly:true});
</script>

```

Codul de mai sus expune o singură opțiune, și anume interzice votarea. În acest caz doar opțiunea Readonly va avea valoarea opțiunii, restul câmpurilor fiind populate cu valorile standard. Relația este în ambele sensuri, astfel că orice modificare adusă în PI se va reflecta și în constructorul Javascript al widgetului.

Următorul exemplu grafic va limpezi și mai mult modul de interacțiune între PI și widget:

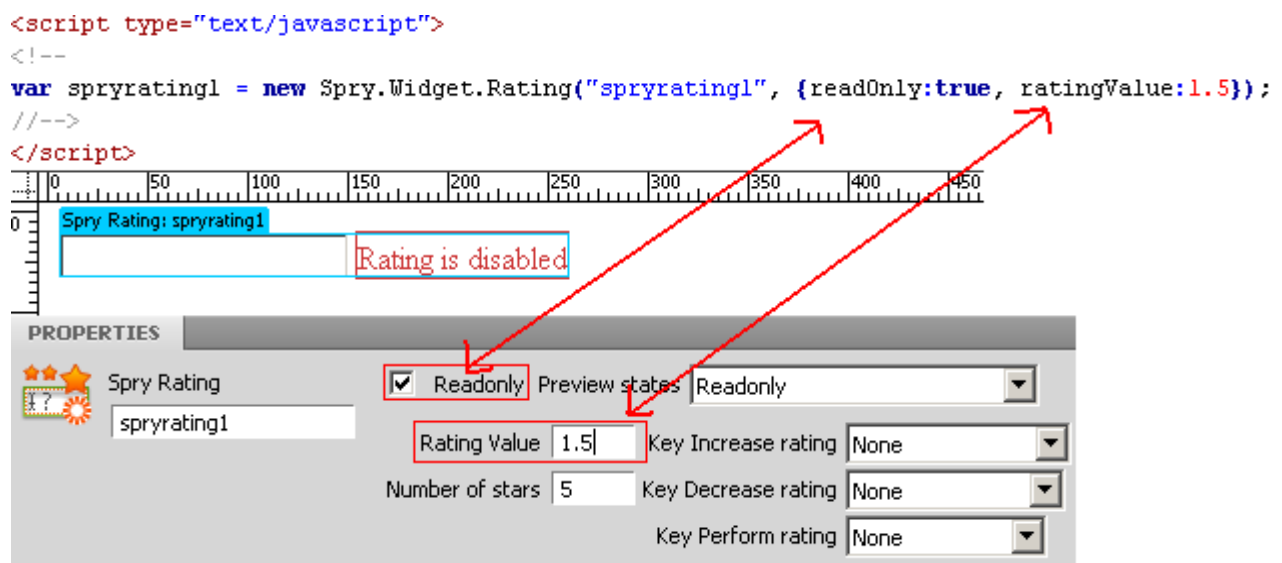


Fig 6.13 relația între PI și cod

După cum se vede în imagine, orice modificare adusă este vizibilă în ambele sensuri. Modificările nu sunt aduse doar codului Javascript, ci și codului HTML de reprezentare al elementului. Bifarea opțiunii Readonly va duce de exemplu și la adăugarea următorului cod HTML în widget:

```
<span class="ratingReadOnlyErrMsg">Rating is disabled</span>
```

cod ce este afișat imediat și în vederea Design.

Cum trimite însă formularul schimbările necesare pentru cod? Prin intermediul evenimentelor atașate controalelor folosite.

```
<input type="text" name="ratingValue" class="min_maxEditBox"
onBlur="updateTag('ratingValue')" id="ratingValue" />
```

Codul de mai sus descrie modul în care PI pentru câmpul **ratingValue** trimite noua valoare Inspectorului, apelând funcția `updateTag`, atunci când controlul pierde focusul (evenimentul `onBlur`). Funcția `updateTag` este cea mai folosită funcție din Inspector, și este apelată de toate controalele acestuia. Fiind cea mai mare funcție din codul Inspectorului, voi lista doar o parte din ea, și un exemplu de procesare a controlului primit.

```
function updateTag(attrib)
{
    // preluare obiect widget in widgetObj
    if (attrib)
    {
        switch (attrib)
        {
            case "widgetId":
            {
                var newId = WIDGET_ID.value;
                if( newId == containerId )
                    return;

                if( newId.length == 0 )
                {
                    alert(dw.loadString("spry/widget/alert/need unique id"));
                    return;
                }

                if( dom.getElementById(newId) )
                {
                    alert(dw.loadString("spry/widget/alert/id already exists"));
                    return;
                }

                if( !dom.isValidIDValue(newId) )
                {
                    alert(dw.loadString("spry/widget/alert/id is invalid"));
                    return;
                }

                widgetObj.updateId(newId);

                widgetMgr.setWidget('Spry.Widget.Rating', newId, widgetObj );
            }
        }
    }
}
```

```
}  
}
```

Exemplul prezintă modul de schimbare al idului pentru widgetul curent. În `widgetObj` va fi stocată o referință la obiectul curent selectat. Codul pentru această operațiune a fost prezentat de câteva ori în lucrare, așa că vom trece peste partea aceasta. Fiecare câmp ce apelează funcția `updateTag`, trimite ca parametru numele câmpului. În funcție de ce câmp apelează funcția vom ști ce proprietate să modificăm.

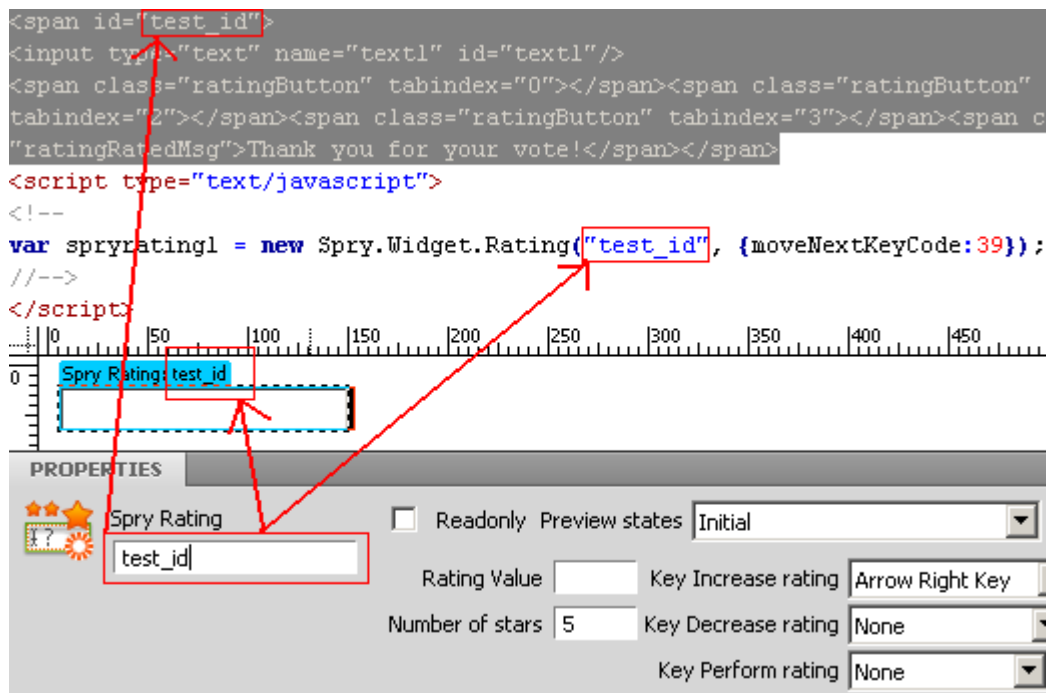


Fig 6.14 apelare funcția `updateTag` pe evenimentul `onBlur`

Figura 6.14 prezintă modul în care apelul acestei funcții realizează schimbarea idului pentru elementul HTML al widgetului și pentru constructorul Javascript al acestuia. Pornind de la exemplul de cod prezentat mai sus, inițial se preia noua valoare a widgetului din câmpul definit pentru id. În cazul în care elementul cu acel id există sau idul nu e valid, vom afișa o eroare și vom ieși din funcție (`dom.getElementById(newId)` și `!dom.isValidIDValue(newId)`). Dacă noul id e în regulă, vom schimba valoarea pentru toate elementele componente ale acestuia, și vom modifica valoarea acestuia și în widget manager:

```
widgetObj.updateId(newId);  
widgetMgr.setWidget('Spry.Widget.Rating', newId, widgetObj );
```

Modificarea valorilor celorlalte controale se face tot în această funcție, în același apel `switch`, însă în cazul celorlalte opțiuni vor fi schimbate nu doar valorile HTML ale widgetului, ci și opțiunile definite în constructorul Javascript.

6.2.5. Alte extensii

Pe lângă extensiile importante dezbătute în rândurile precedente, extensia Spry Rating se mai folosește și de alte extensii Dreamweaver pentru o integrare completă:

Code coloring - Code coloring permite definirea culorilor și a cuvintelor cheie care vor fi evidențiate, atunci când inserăm o extensie.

```
<script type="text/javascript">
<!--
var spryselect1 = new Spry.Widget.ValidationSelect("spryselect1");
var spryselect2 = new Spry.Widget.ValidationSelect("spryselect2");
var sprytextareal = new Spry.Widget.ValidationTextarea("sprytextareal");
//-->
</script>
```

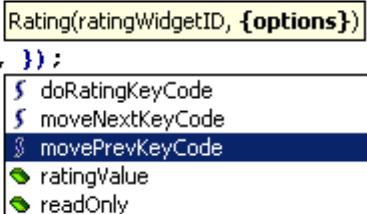
Fig 6.15 code coloring pentru diferite widgeturi

Conținutul extensiei de Code coloring este reprezentat de un fișier XML. În cazul extensiei Spry Rating acesta arată în felul următor:

```
<codeColoring
  xmlns:MMString="http://www.macromedia.com/schemes/data/string/">
  <scheme id="JavaScript">
    <classlibrary name="SpryRatingClasses"
      id="CodeColor_JavascriptSpryClasses">
      <class>Spry.Widget.Rating</class>
      <class>Spry.Widget.Rating.addLoadListener</class>
    </classlibrary>
  </scheme>
</codeColoring>
```

Code Hints – extensia permite definirea de code hints pentru widgeturile/extensiile noastre. Asemeni extensiei de code coloring, este reprezentată de un fișier de configurare XML.

```
<javascript">
<w Spry.Widget.Rating("spryrating1", {readOnly:true, });
```



The image shows a code hints popup window for the `Spry.Widget.Rating` constructor. The popup title is `Rating(ratingWidgetID, {options})`. The list of options includes: `doRatingKeyCode`, `moveNextKeyCode`, `movePrevKeyCode`, `ratingValue`, and `readOnly`. The `movePrevKeyCode` option is currently selected and highlighted in blue.

Fig 6.16 Code hints pt Spry Rating

Strings – permite extinderea șirurilor folosite de extensia noastră cu suport pentru localizare. Ca și celelalte 2 extensii, configurarea se face într-un fișier XML pe care realizăm traducerea șirurilor folosite. Exemplu:

```
<strings>
  <string id="spry/widgets/Rating/title"
    value="Spry Rating"/>
  <string id="spry/widgets/Rating/readonly message"
```

```

        value="Rating is disabled" />
<string id="spry/widgets/Rating/voted message"
        value="Thank you for voting!" />
<string id="spry/widgets/Rating/rated state" value="Vote complete" />
<string id="spry/widgets/Rating/readonly state" value="Readonly" />
</strings>

```

6.3. Împachetarea și distribuția extensiei Spry Rating

Odată ce o extensie este terminată, aceasta trebuie împachetată pentru a putea fi folosită cu aplicația Extension Manager. Fișierul de instalare pentru extensii este un fișier XML cu extensia mxi care servește toate informațiile de care are nevoie Managerul de Extensii.

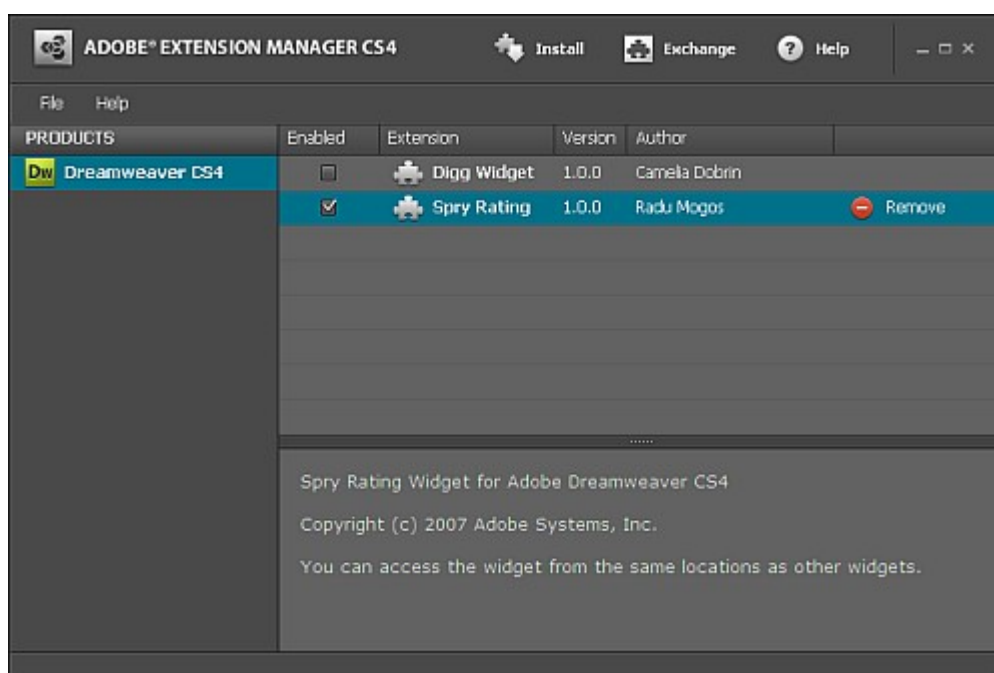


Fig 6.17 Managerul de extensii CS4

Documentul XML folosește taguri XML numite taguri MXI. Acestea descriu informațiile legate de extensia respectivă, pornind de la autor, licența de instalare și până la lista de fișiere ce vor fi incluse în extensie.

În continuare voi prezenta o scurtă listă cu cele mai importante taguri MXI folosite pentru crearea pachetului Spry Rating:

macromedia-extension – principalul tag folosit pentru definirea extensiei, numele acesteia, versiunea, și tipul extensiei. Codul extensiei Spry Rating:

```

<macromedia-extension
  name="Spry Rating" version="1.0.0"
  type="Command" requires-restart="true">
</macromedia-extension>

```

products – definește o listă de produse pentru care poate fi instalată extensia. O extensie poate fi integrată și cu celelalte soluții Adobe existente. Codul extensiei Spry Rating:

```
<products>
  <product name="Dreamweaver" version="10" primary="true" />
</products>
```

description – descrie ce face extensia.

```
<description>
  <![CDATA[
    Spry Rating Widget for Adobe Dreamweaver CS4<br>
    <br>
    Copyright (c) 2008 Adobe Systems, Inc.<br>
  ]]>
</description>
```

ui-access – specifică textul ce va fi afișat în Managerul de Extensii când extensia este instalată.

```
<ui-access>
  <![CDATA[
    You can access the widget from the same locations as other widgets.
  ]]>
</ui-access>
```

files – partea cea mai importantă a unui fișier MXI. descrie lista de fișiere folosită de extensie, listă ce va fi folosită pentru a împacheta extensia. Rezultatul final al acesteia va fi un fișier MXP care va conține toate setările și fișierele referite de acest document XML.

```
<files>
  <file source="CodeColoring/SpryRatingCodeColor.xml"
    destination="$dreamweaver/Configuration/CodeColoring" />
  <file source="CodeHints/SpryRatingCodeHints.xml"
    destination="$dreamweaver/Configuration/CodeHints" />
  <file source="Commands/SpryRating.htm"
    destination="$dreamweaver/Configuration/Commands" />
  <file source="Commands/SpryRating.js"
    destination="$dreamweaver/Configuration/Commands" />
  <file source="Inspectors/spry_rating.htm"
    destination="$dreamweaver/Configuration/Inspectors" />
</files>
```

Crearea pachetului pentru distribuție se face tot din Managerul de extensii. Rezultatul va fi un fișier MXP care conține toate informațiile și fișierele necesare pentru a funcționa în Dreamweaver.

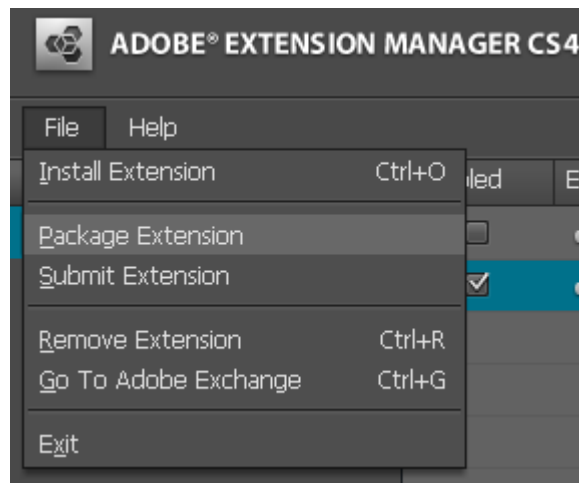


Fig 6.18 Crearea pachetelor

7. CONCLUZIE

8. BIBLIOGRAFIE

Cărți

1. *Adobe Dreamweaver CS3 extending Dreamweaver*, Adobe Press, 2007
2. *Adobe Dreamweaver CS3 API reference*, Adobe Press, 2007
3. *The extension installation file format*, Adobe Press, 2007
4. *Dreamweaver CS3: The Missing Manual*, O'Reilly Media, 2007
5. *Macromedia Dreamweaver 8 API reference*, Macromedia Press, 2005
6. *Developing Extensions for Macromedia Dreamweaver 8*, Macromedia Press, 2005
7. *Extending Dreamweaver MX*, Macromedia Press, 2002

Referințe internet

8. Muller, A., *Starting with Spry*,
<http://www.builderau.com.au/program/ajax/soa/Starting-with-Spry/0,339028327,339278552,00.htm> , accesat pe 4 Februarie 2008
9. *Documentație Adobe Spry Framework*,
<http://labs.adobe.com/technologies/spry/docs.html>, accesat pe 8 Februarie 2008
10. *Build Ajax Components With Spry*,
http://www.webmonkey.com/tutorial/Build_Ajax_Components_With_Spry, accesat pe 10 Martie 2008
11. Best practices with Spry,
http://labs.adobe.com/technologies/spry/articles/best_practices/ , accesat pe 14 Aprilie 2008
12. Using Spry Widgets: Rating Widget Overview,
http://labs.adobe.com/technologies/spry/articles/rating_overview/index.html , accesat pe 22 Aprilie 2008