OpenCms Workplace úpravy

Workplace (především jeho část Explorer) v jeho stávající podobě má z hlediska uživatelské přívětivosti a použitelnosti řadu nedostatků:

- všechny akce nad daným objektem se spouštějí přes kontextové menu. Možná řešení:
  - ke každému objektu přidat panel, ze kterého se budou přímo spouštět potřebné akce
  - ke každému objektu přidat pouze checkbox a mít pouze jeden globální panel s akcemi
  - různé kombinace a modifikace zmíněných řešení
- SLOŽITÁ modifikovatelnost (bude probráno později).
- občasné problémy pod různými prohlížeči (IE a čistý FireFox v pořádku, zjištěné problémy v Mozille a Opeře). Důvodem je nejspíš „zajímavé“ využití Javascriptu.
- zbytečně složité vytváření nových objektů
- …

V dalších částech proberu možnosti úprav Workplace.

Konfigurační soubor

Celou řadu vlastností lze ovlivňovat změnami v souboru *opencms-workplace.xml*.

- výchozí lokále
- automatické zamykání objektů – pokud je tato položka nastavena na false, jsou ovlivněny dostupné funkce kontextového menu. Pokud uživatel nemá daný objekt zamknut, nemůže s ním téměř nic provádět (kromě kopírováni, publikování, prohlížení jeho vlastností a samozřejmě zamknutí)
- u všech typů objektu ve VFS (folder, JSP, …) lze provézt několik změn nastavení
  - nastavit výchozí práva pro standardní skupiny uživatelů (Admin, User, ProjectManager).
  - specifikovat dostupné položky v kontextovém menu a u každé určit jaké JSP se použije
  - ovlivnit jaké vlastnosti budou k dispozici v dialogu, který se otevře při vytváření a editaci objektu
  - určit jsp stránku, která se otevře při vytváření nového objektu
  - ikonku o daného objektu :)
- a několik dalších, méně podstatných nastavení

Upravení vzhledu a funkcí Exploreru

Výchozí startovní JSP stránka vypadá takto:

```
<%@ page import="org.opencms.workplace.explorer.*,org.opencms.jsp.*"%>
<%@ page import="org.opencms.workplace.CmsWorkplace"%>
<%
    CmsJspActionElement cms = new CmsJspActionElement(pageContext, request, response);
    CmsExplorer wp = new CmsExplorer(cms);

%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=<%= wp.getEncoding() %>">
</head>
<frameset rows="24,*" border="0" frameborder="0" framespacing="0">
    <frame name="explorer_head"
           src="<%= CmsWorkplace.getSkinUri() %>commons/empty.html"
           noresize scrolling="no">
    <frame
    <%= wp.getFrameSource("explorer_body", cms.link("explorer_body_fs.jsp")) %>
           noresize scrolling="no">
</frameset>
</html>
```

Prvotní předpoklad byl, že se upraví tato výchozí stránka a vytvoří se varianty tříd generujících Explorer mód Workplace. Tyto třídy však nevytváří výsledný HTML a Javascript kód, ale pouze Javascript, ve kterém jsou uloženy potřebné údaje (pro každý objekt je vytvářen speciální řetězec o 20 podpoložkách: název, url, title, oprávnění,...). Tyto řetězce využijí dvě JS knihovny *explorer.js* a *tree.js* k vygenerování Exploreru!!! Prostě odpornost. Netuším jaký rozumný důvod vedl vývojáře k tomuto řešení (napadá mě snad jen menší objem posílaných dat a „ulehčení“ serveru při generování výstupu). Z těchto poznatků jsem došel k několika možným řešení úpravy Exploreru:

1. Nechat stávající třídy být a upravit pouze Javascriptové knihovny, které generují koncové HTML a radši zapomenout, jak je to implementováno:). Takto by se dalo docílit drobných změn např. nahrazení kontextového menu panelem s tlačítky, změna skinu, atd.. Předpokládám, že časová náročnost implementace takovýchto změn by nebyla velká (1 den).
2. Pokud by se požadavky na změny nepodařilo vyřešit výše uvedeným způsobem, muselo by se přistoupit k razantnějšímu přístupu. Upravit stávající třídy a JSP tak, aby vytvářeli konečný HTML nebo přistoupit na jejich řešení a provést úpravy v dynamicky generovaném Javascriptu a v knihovnách, které s ním pracují. Ani jedno z těchto řešení se mi příliš nezamlouvá. Především velká provázanost jednotlivých částí systémů a celková složitost implementace (víceméně všechen výstup je generován programově a JSP slouží jen ke spouštění potřebných metod) by rozhodně způsobila řadu problémů. Časovou náročnost u těchto postupů nedovedu odhadnout.
3. Dle požadavků vytvořit vlastní Workplace. Pokud totiž bude OpenCms použito nejen pro portál KIVu, ale vybere si ho i tým WebFav, lze očekávat, že v něm budou pracovat „normální“ uživatelé, kterým nejspíš tento stávající přijde složitý a nebudou v něm chtít pracovat. Navrhnutí a implementace vlastního Workplace by tak bylo vhodnou investicí.

Shrnutí

Jak bylo zmíněno na začátku, hlavním neduhem stávajícího Workplace je špatná modifikovatelnost, která plyne z jeho implementace (skoro celá prezentační část ručně kódována ve třídách, zhůvěřilé použití Javascriptu). Nejde jen o výše probíranou úpravu vzhledu Workplace, ale též úprava stávajících formulářů (vytváření nových objektů, nastavování vlastností, upload formulář, ...) je zbytečně pracná, ale proveditelná.