

```

32 <div id='sxX2'>175</div>
33 <div id='sxY2'>100</div>
34 <div id='sxHodnota2'>". $Hodnota."</div>
35
36 <div id='sxDruh3'>3</div><!-- Spinač1 -->
37 <div id='sxFarba3'>". $Farba1."</div>
38 <div id='sxX3'>75</div>
39 <div id='sxY3'>110</div>
40 <div id='sxHodnota3'>". $Spinac1."</div>
41
42 <div id='sxDruh4'>3</div><!-- Spinač2 -->
43 <div id='sxFarba4'>". $Farba2."</div>
44 <div id='sxX4'>285</div>
45 <div id='sxY4'>190</div>
46 <div id='sxHodnota4'>". $Spinac2."</div>
47 ";
48 ?>

```

Treba poznamenať, že uvedený kód slúži ako generátor nového stavu zásobníka. Využíva údaje, ktoré vygeneroval v predošlom volaní a ktoré mu JavaScript poslať v hodnotách parametrov *Spinac1*, *Spinac2* a *Hodnota*. Po ich zistení v riadkoch 3 až 9 nasleduje v riadkoch 10 až 19 vygenerovanie nových hodnôt. Namiesto posielania údajov z „predošlého kola“

a generovania nových hodnôt na tomto mieste by v reálnom systéme bolo získanie údajov z databázy, resp. iného zdroja.

Vlastné vygenerovanie odpovede sa realizuje v podobe kódu HTML, ktorý v prvok div s priradenými identifikátormi obsahuje potrebné údaje. Tie budú zapísané do prvku div s identifikátorom *divObrazky*, ktorého obsah je skrytý (pozri riadok 11 vo výpise 1 a riadky 59, 60 vo výpise 5). Funkcia *ZobrazObrazky* z výpisu 5 z prvkov div použitím pomocnej funkcie *Text* vyberie údaje a odovzdá ich na spracovanie funkcií *SvgVytvorObraz* v časti SVG.

Záver

Vzájomná súčinnosť kódu HTML, SVG, JavaScript a PHP umožňuje získať možnosť monitorovania stavu systému. Je zjavné, že uvedený príklad predstavuje určité zjednodušenie najmä v časti PHP. Veríme však, že môže byť návodom na vytvorenie zaujímavých aplikácií už aj preto, lebo doplnenie prípadných iných druhov obrázkov je pri využití opísaného zdrojového kódu vec rutiny.

■ IMRICH BURANSKÝ

Caché 5.1 – Generátor kódu

V predchádzajúcom príspevku sme hovorili o XML ako o novodobom fenoméne na komunikáciu medzi aplikáciami a o tom, ako tento protokol možno jednoducho použiť pri budovaní aplikácií v Caché.

V tomto príspevku na uvedenú tému čiastočne nadviažeme. Ukážeme si, že použitie XML (ale zďaleka nielen XML) je v Caché jednoduché vďaka jej unikátnej schopnosti – schopnosti generovať kód podľa kritérií zadaných programátorom.

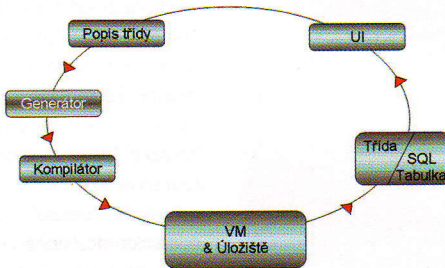
O čo ide? Predstavte si typickú aplikáciu dnešnej doby; pôjde o viacvrstvovú aplikáciu, ktorá veľmi pravdepodobne bude používať niektorú z rozšírených objektových technológií pre prezenčnú vrstvu a pre vrstvu aplikačnej (business) logiky. Ako úschovňa dát posluží (ó, hrôza!) nejaká relačná databáza.

Opomenieme problémy, ktoré vyplývajú z potreby napasovať objekty prvých dvoch vrstiev na ploché tabuľky relačnej databázy (a ktoré v Caché akosi z princípu nikdy nenastanú), a sústredíme sa na aplikačnú vrstvu.

Dobry analytik/programátor pri analýze projektu spravidla pri zostavovaní modelu aplikácie nájde veľa miest, kde bude môcť s výhodou použiť niektorú z veľmi praktických schopností objektov – dedičnosť a polymorfizmus. Takže vo svojom modeli bude definovať rôzne abstraktné triedy a rozhrania, ktoré potom bude implementovať v mnohých triedach, opisujúcich už konkrétne elementy aplikácie. Čo to v praxi predstavuje? V mnohých triedach bude stále znova a znova implementovať kód, ktorý sa bude len čiastočne líšiť od kódu zdedeného z bazových tried. Iste, bude mať k dispozícii rôzne viac či menej dokonalé pomôcky ako refaktoring a intuitívne dopĺňovanie textu (intellisense). Pravda, nutnosti znova písať a modifikovať už skôr napísaný kód sa nevyhne. Netreba zdôrazňovať, že v takto tvorenom kóde môže vyrobiť nové chyby.

A teraz sa dostávame k tomu, čo je generátor kódu v Caché a čím môže pomôcť vývo-

járom urýchliť vývoj ich aplikácií a obmedziť počet chýb. Prv než si na príklade XML ukážeme konkrétne použitie, najskôr na obrázku vysvetlíme princíp.



Obr. č. 1 Schéma implementácie objektového modelu Caché

Definície tried, navrhnuté analytikom či programátorom, treba pred prvým použitím najskôr previesť do kódu zrozumiteľného virtuálneho stroju Caché. To má na starosti prekladač (kompilátor). V prípade perzistentných tried takisto kompilátor musí rozhodnúť, ako a kam sa do viacrozmerných databázových štruktúr Caché budú ukladať hodnoty vlastností tried. Kompiláciou tried vzniknú jednak špeciálne moduly, ktoré opisujú a tým sprístupňujú rozhranie tried programátorom, a jednak moduly, ktoré uskutočňujú vlastný prístup do databázy alebo vykonávanie implementovaných metód tried virtuálnym strojom.

V prípade, že opisovaná trieda obsahuje metódy označené ako metódy generátora kódu, je kód vnútri týchto metód vykonaný virtuálnym strojom v priebehu kompilácie a jeho vykonaním vznikne cieľový kód metódy.

Vďaka tomuto mechanizmu teda možno definovať budúcu podobu kódu metód tried, ktoré ešte vôbec neexistujú a až niekedy v budúcnosti vzniknú, napr. zdedením od triedy majúcej metódy označené ako metódy generátora kódu. A to je presne prípad XML.

Pokiaľ si predstavíte, aké rôznorodé môžu byť dokumenty XML, ako môžu na opis jedného a toho istého používať napr. atribúty alebo

elementy, je zjavné, že napr. v projekte obsahujúcom desiatky či stovky tried by bolo časovo veľmi náročné implementovať metódy na serializáciu a deserializáciu objektov do/z XML. A to je ešte potrebné vziať do úvahy rekurzívne volanie týchto metód pre referenčné atribúty oných tried.

Oveľa efektívnejšie je ísť na to naopak, napísať jednu abstraktnú triedu (v Caché sa volá %XML.Adaptor), v tejto triede implementovať metódy na XML export, import, schémy atď. a prostým podedením od tejto triedy získať zakaždým iný, ale správny kód vykonávajúci zmiernenú serializáciu a deserializáciu. Lebo v tom je čaro generátora kódu – v okamihu návrhu modelu je o budúcich triedach známe len jediné: že budú dediť od práve definovanej triedy; nevieme, aké vlastnosti budú mať, koľko ich budú mať, ale vieme stopercentne, nech už budú vyzeráť akokoľvek, ich metódy na serializáciu a deserializáciu budú správne vystavovať všetky ich vlastnosti ako prvky dokumentov XML.

Mimochodom, všetka perzistencia je v Caché tiež implementovaná pomocou generátora kódu, lebo každá trieda, ktorá je odvodená od triedy %Persistent, sama vie pracovať s hodnotami svojich vlastností, t. j. vie ich ukladať a na požiadanie vrátiť.

Záver: vhodným návrhom modelu a použitím generátora kódu sa dá zefektívniť nielen návrh a implementácia aplikačnej vrstvy, ale i prezenčnej vrstvy v prípade použitia webového rozhrania Caché Server Pages (CSP) a, samozrejme, ho možno aplikovať i v rámci databázovej vrstvy Caché.

Ten, kto pochopí princíp generátora kódu, môže tvoriť svoje aplikácie oveľa rýchlejšie, elegantnejšie a s menej chybami než programátor používajúci klasický prístup k objektovému programovaniu.

■