Semestrální práce z předmětu KIV/WEB

Tomáš Maršálek
marsalet@students.zcu.cz

14. prosince 2011

1 Použité technologie

1.1 html

Použití html je jedním z povinných požadavků práce. Použitá verze je HTML4, některé prvky (pouze několik) jsou podporovány až verzí HTML5.

1.2 css

Vzhled webových stránek je celkem jednoduchý, ale přehledný. Nemá problém se zmenšováním okna nebo při zoomování přímo v prohlížeči. Veškeré pozicování, velikosti písma, barvy, atd. jsou zajištěny výlučně kaskádovými styly. Stylování přímo v html je dnes zastaralé. Kvůli čitelnosti jsou všechny styly ve zvláštním souboru, nikde není použit inline zápis typu `<tag style="...">`.

Obtížnými prvky bylo stylování formulářů a spodního panelu. Naštěstí se stylováním formulářů zabývá poměrně mnoho lidí, proto nebylo těžké vyhledat správný způsob zarovnání pomocí samotného css bez použití tabulek. Tabulkové rozložení stránek je dnes kontroverzní téma, kdy jedna skupina tvrdí, že tabulky neslouží k uspořádání prvků na stránce, když už máme dostatečně silné kaskádové styly, které stejnou věc zvládnou přehledněji, například pomocí *float* vlastnosti. Výhodou *float* oproti tabulkovému rozložení je i lepší přizpůsobivost při manipulaci s velikostí okna nebo přiblížením. Proto mé rozhodnutí nepoužívat tabulky k tomuto účelu.

Další problém s neintuitivním řešením je spodní panel přilepený k dolnímu okraji obrazovky (sticky footer). Opět se naštěstí našlo pár návodů, jak tohoto docílit.

1.3 php

Klíčovou technologií celé práce je právě php. Velké množství kódu je ve stránkách vygenerováno dynamicky, protože obsah je závislý na oprávnění uživatele k přístupu k tomuto obsahu. Například běžný nezaregistrovaný uživatel podle zadání nemá přístup k seznamu členů klubu. Ředitel klubu má nejvyšší oprávnění a má tedy možnost vkládat nové aktuality, odehrané zápasy, přidávat nebo upravovat hráče a má k tomu k dispozici formuláře na příslušné stránce. Tyto prvky musí být zajištěny na straně serveru tak, že každý uživatel uvidí nejvýše tolik, kolik jeho přístupová práva dovolují.

Důležité je zajistit bezpečnost a stabilitu webových stránek. Jsou ošetřeny případy, kdy by se uživatel dostal omylem nebo se zlým úmyslem na některou stránku jinak, než povoleným způsobem. Například kdyby chtěl upravit jiného uživatele jednoduchým přepsáním GET parametru v url. I kdyby se uživateli podařilo odeslat požadavek na změnu údajů, při vyhodnocování požadavku stejně dojde ke zkontrolování povolení k přístupu, proto není možné, aby někdo manipuloval s cizími daty. Samozřejmě za předpokladu, že vše funguje tak jak má, což obecně není jednoduchá záležitost.

Kontrola údajů zadaných pomocí formuláře je kontrolována i na straně klienta, ale hlavní kontrola je samozřejmě ta na straně serveru. Ta však dává upozornění pouze v těch případech, kdy nebylo možné provést kontrolu u klienta. To je například při vkládání uživatele, jestli již takový existuje.

1.4 SQLite 3

Databázový systéme je zde implementace SQL jazyka SQLite 3. Zvolil jsem ji pro svoji jednoduchost oproti známémějšímu a rozšířenějšímu MySQL. Webové stránky velikosti této semestrální práce by s přehledem obsloužil jednoduchý xml soubor, ale zadání vyžaduje použití databáze, proto nejodlehčenější variantou je právě SQLite. Tato implementace je opravdu jednoduchá na používání, což je její největší výhodou. Výhodou je také netřeba databázového serveru, protože celá databáze je po celou dobu uložena v jediném souboru. Přenositelnost databáze je tedy zcela triviální.

1.5 ostatní

Kontrola formulářů je velmi efektivní při použití javascriptu a ještě efektivnější v kombinaci s AJAXem. Rozhodl jsem se ale vyzkoušet nové formulářové prvky HTML5, které poskytují jednoduchou kontrolu a ještě jednodušší obsluhu formulářů. Například při vstupu data nebo čísla se opravdu usnadní práce jak programátora, tak uživatele při používání onoho formuláře. Podpora těchto prvků zdá se být dnes na přijatelné úrovni.

2 Adresářová struktura a architektura

Webové stránky mají poměrně jednoduchou architekturu. Použil jsem takové postupy a technologie, aby komplexita architektury odpovídala komplexitě

stránek tak, jak jsou vnímány uživatelem. Zdrojové kódy jsou v kořenovém adresáři a kaskádové styly v adresáři *css*. Protože nebyl použit javascript ani jiné prvky zajišťující logiku na straně klienta, jiné adresáře nejsou ani potřeba. Databáze je uložena v souboru *data.db* rovněž v kořenovém adresáři.

2.1 architektura

Rozvržení stránek bylo zvoleno tak, aby co nejvíce šetřilo zdrojovým kódem a bylo tedy co nejjednodušší v při jakýchkoliv případných změnách. Uživatel je vždy přítomen na stránce *index.php*, obsah je vždy dynamicky vygenerován podle GET parametru *id*. Toto rozvržení umožňuje ponechat zdrojový kód horního a dolního panelu stejný a pouze měnit obsah.

Každá stránka obsahující obsah má vždy dva účely (s vyjímkou stránky *kontakt.php*, která je víceméně pouze statického charakteru. Prvním účelem je zobrazit obsah tak, jak ho získáme z databáze. Druhý účel zajišťuje logiku celé stránky. Tedy zajišťuje vyhodnocování formuláře té stejné stránky. V případě úspěchu při vyhodnocování formuláře přesměruje uživatele na stejnou stránku, ale tak, aby se pouze zobrazil obsah. Tento styl je poměrně hojně využívaný, protože umožňuje ponechat logiku i obsah jedné stránky v jednom souboru.