Fakulta matematiky, fyziky a informatiky – Univerzita Komenského, Bratislava

Vypracovanie webstránky predmetu

Základy umelej inteligencie 1 a 2

Michal Čertický

školiteľ: RNDr. Mária Markošová, PhD.

Obsah:

1. Úvod

Ako je už z názvu zrejmé, cieľom tejto práce je vytvorenie funkčného a obsažného webu, ktorý má uľahčiť a obohatiť vyučovanie predmetov Základy umelej inteligencie 1 a 2.

Tieto požiadavky som si doplnil o ďalší zámer – vypracovanie webu takým spôsobom, aby bol ľahko a rýchlo modifikovateľný a aktualizovateľný aj bez znalostí jazyka HTML, či všeobecných zásad webdesignu. Za účelom splnenia tohto cieľa som sa rozhodol upustiť od implementačnej jednoduchosti triviálnych HTML dokumentov, a zvolil si za svoj nástroj skriptovací jazyk PHP a jednoduchý databázový jazyk SQL. Tie mi totiž umožňujú vytvoriť komplexnejšiu webovskú aplikáciu. Rozhodol som sa používateľov webu rozdeliť na 2 skupiny – študentov a administrátorov, pričom administrátorom bude povolené meniť obsah stránky.

Ďalším cieľom, ktorý som si pred tvorbou webu zadal je zefektívnenie priebehu cvičení. Toto je docielené obohatením, alebo úplným prepracovaním, či nahradením niektorých úloh pre študentov. Samozrejmosťou sú vzorové riešenia najzložitejších zadaní. K efektivite priebehu cvičení taktiež môže prispieť interface na online uploadovanie vyriešených úloh.

Aby boli možnosti, ktoré web ponúka využiteľné vo vyučovacom procese, je nutné zabezpečiť jednoduchý spôsob zverejňovania obsahu webu – obzvlášť zadaní a vzorových riešení úloh na cvičenia.

Oblasťou vyučovania, do ktorej ale nechcem zasahovať je hodnotenie študentov. K dispozícii bude množstvo zadaní, ktoré majú rozdielnu obtiažnosť. Myslím si, že žiadať od študentov vypracovanie všetkých úloh by bolo príliš zaťažujúce a časovo náročné – a to nie len pre riešiteľov, ale aj pre cvičiacich, ktorý by riešenia opravovali. Preto ponechám bodové ohodnotenie jednotlivých úloh, prípadne výber úloh, ktoré sú povinné, na vyučujúcich, resp. cvičiacich.

2. Webstránka

V tejto časti sa budeme zaoberať implementáciou webu. Preberieme si obsah a tematické členenie webu, vysvetlíme použitie pre obidve skupiny používateľov – študentov a administrátorov, vysvetlíme technologické detaily implementácie a odôvodníme výber programovacích jazykov. Zmienime sa aj o bezpečnostných opatreniach, ktoré sú pri tvorbe webu nevyhnutné.

Ďalej opíšeme, akým spôsobom sa stránka snaží spĺňať štandardy pre použiteľnosť a prístupnosť pre zdravotne postihnutých používateľov.

Ďalej objasníme základné opatrenia, ktoré sme prijali pre lepšiu kompatibilitu s webovskými vyhľadávacímy systémami (ako napríklad Google) a ktoré majú za cieľ zlepšenie vyhľadávacích výsledkov.

Na koniec zhrnieme výhody a nevýhody, ktoré z takto zvolenej implementácie vyplývajú.

2.1. Rozdelenie obsahu

Obsah webu je rozdelený na deväť tematických častí – deväť podstránok:

Aktuality – Aktuálne informácie a oznamy pre študentov. Oznamy sú načítavané z vlastnej tabuľky v databáze. Ide o časť, ktorej obsah bude pravdepodobne najčastejšie modifikovaný administrátormi. Každý oznam sa skladá z krátkeho nadpisu, dátumu a času v ktorom bol zverejnený a samotného textu odkazu. Oznamy majú navyše nepovinnú zložku – url, na ktoré bude čitateľ odkázaný po kliknutí na nadpis. V prípade, že administrátor url nezadá, nadpis nie je odkazom a nedá sa naň kliknúť.

Informačný list predmetu – Podstránka, ktorá poskytuje študentom základné informácie o predmete. Nájde tu krátky popis náplne predmetov ZUI 1 a ZUI 2, sylaby týchto predmetov v niekoľkých bodoch a odkazy na odporúčanú literatúru.

Pravidlá a podmienky – V tejto časti sa nachádzajú podmienky absolvovania oboch spomínaných predmetov.

Hodnotenie – Jednoduchá hodnotiaca stupnica pre obidva predmety (zvlášť). Počty bodov potrebných pre získanie jednotlivých známok.

Prednášky – Podstránka obsahujúca slidy z prednášok oboch predmetov vo forme Powerpointových prezentácií na stiahnutie. Obsah tejto pdstránky je samozrejme dynamický a ľahko modifikvateľný. Po prihlásení sa s administrátorským kontom sa dajú prezentácie mazať a pridávať nové.

Cvičenia – Podstránka s cvičeniami je najkomplexnejšou časťou webu ZUI. Ak užívateľ nie je prihlásený ako administrátor, dovolí mu prezerať si zadania úloh na tie cvičenia, ktoré sú zverejnené a uploadovať na server svoje riešenia. V prípade, že je k zadaniu priložené aj vzorové riešenie a navyše je administrátorom zverejnené, čitateľ si ho môže stiahnuť. Ak sa však prihlási administrátor, môže skrývať, alebo zverejňovať jednotlivé cvičenia, či vzorové riešenia a manipulovať (sťahovať si, alebo vymazávať zo servera) s odovzdanými riešeniami študentov. Pri každom odovzdanom riešení má samozrejme presný dátum a čas odovzdania.

Kontakty – Jednoduchá podstránka , ktorá poskytuje informácie potrebné pri kontaktovaní vyučujúcich, alebo cvičiacich.

Užitočné odkazy – V tejto časti webu sa čitateľ dostane k zoznamu odkazov na iné stránky, ktoré mu môžu pomôcť pri štúdiu predmetu. Odkazy sú dynamicky načítavané z databázy a pre administrátorov je ľahké ich kedykoľvek vymazávať, či pridávať nové. Odkazy sa skladajú z url stránky, na ktorú smerujú a krátkeho popisu.

Hľadanie – Podstránka ktorá umožňuje za pomoci vyhľadávacej služby Google hľadať potrebné informácie vrámci webu ZUI, ale aj vrámci celého internetu. Hľadanie je založené na jednoduchom vyhľadávaní výskytov zadaného reťazca znakov na stránkach.

2.2. Návod na použitie

(a.) Použitie pre študentov:

Použitie stránky pre študentov sa v podstate nelíši od ostatných stránok. Nie je potrebné žiadne prihlasvanie – teda stránka je prístupná pre verejnosť. Prístup na web ZUI teda nie je prirodzene obmedzený iba pre študentov, ktorí majú predmet zapísaný. Informácie, ktoré sa tu dajú nájsť, totiž môžu pomôcť aj študentom, ktorí nie sú rozhodnutí, či si predmet zapísať, alebo nie, prípadne komukoľvek, kto hľadá infrmácie týkajúce sa oblasti umelej inteligencie.

Stránka je vytváraná s dôrazom na čo najmenšie požiadavky na softvérové, či iné vybavenie. Optimalizovaná je síce pre najpoužívanejšie prehliadače (Mozilla Firefox, Internet Explorer, Opera...), no na jej použitie úplne postačia aj menej komplexné systémy, ako napríklad staršie prehliadače bez podpory CSS, alebo Screen-readery pre zrakovo postihnutých. Samozrejmosťou je, že na zobrazenie webu nie je potrebná podpora Flash, alebo Java technológií.

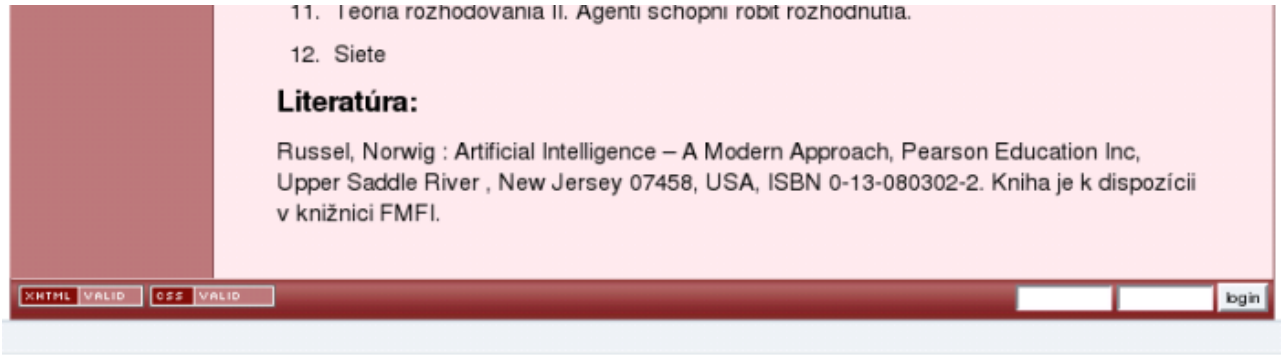
Čo sa týka obsahovej časti webu, zaujímavé sú podstránky s prednáškami a cvičeniami. Pre otvorenie súborov s prednáškami je potrebný akýkoľvek prehliadač .ppt súborov (najpoužívanejšie sú Microsoft Powerpoint, alebo OpenOffice). Treba však upozorniť, že priložené prezentácie sú pôvodne iba pomôckou pre prednášajúceho a nie je možné spoľahnúť sa iba na nich pri príprave na skúšku. Obsahujú iba stručné zhrnutie prednášanej problematiky a pre podrobnejšie štúdium by študenti mali použiť odporúčanú literatúru (uvedenú v časti INFO).

Časť s cvičeniami obsahuje niekoľko predpripravených programov, ktoré je potrebné stiahnuť, rozbaliť a skompilovať. Komprimované sú v bežnom formáte tar. Na ich rozbalenie je potrebné mať niektorý z archivačných programov – napríklad linuxovský Tar, alebo windowsovský Winrar, prípadne 7zip, ktorý je nainštalovaný vo všetkých učebniach na fakulte. Predpripravené programy sú napísané v jazyku Java, teda na ich skompilovanie a spustenie budete potrebovať balík Java Development Kit. Ten je k dispozícii pre počítače na fakulte, prípadne sa dá zadarmo stiahnuť z webu firmy Sun.

Niektoré priložené vzorové riešenia sú naprogramované v programovacom jazyku Delphi, ktorý je blízky hlavne študentom nižších ročníkov. Vývojársky balík Borland Delphi, ktorý je potrebný (nie však jedinou alternatívou) na kompilovanie týchto riešení je k dispozícii na fakulte, prípadne niektoré jeho verzie sú stiahnuteľné zdarma na nekomerčné využitie.

(b.) Použitie pre administrátorov:

Administrátorom v tomto kontexte rozumieme vlastníka administrátorského konta – teda cvičiacich, aleb prednášajúcich predmetu ZUI, prípadne webmastera starajúceho sa o web. Prvým krokom pri modifikovaní stránky administrátorom je prirodzene jeho prihlásenie. K nemu potrebuje prihlasovacie meno a heslo. Tie administrátor zadá do formulára v spodnej časti stránky – pozri obr.1.



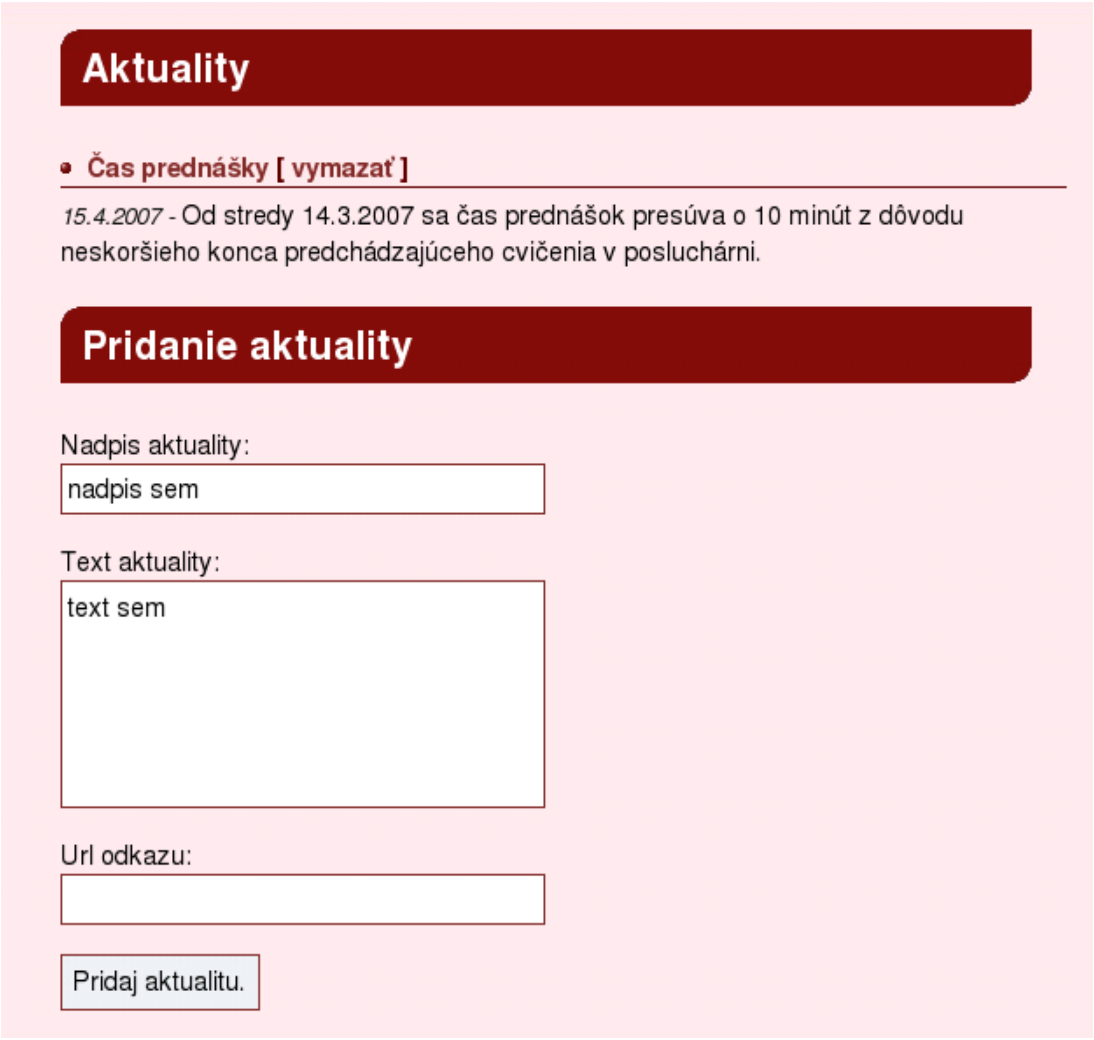
obr.1. - prihlasovací formulár pre administrátorov

Po prihlásení sa na serveri vytvorí session – záznam o administrátorovi, ktorý trvá až do odhlásenia, alebo zanikne po čase nečinnosti určenom konfiguráciou servera (nastavenie sa v prípade webservera Apache nachádza štandardne v súbore httpd.conf). Počas prihlásenia môže administrátor modifikovať web na nasledovných miestach:

Aktuality – Na podstránke s aktuálnymi oznamami má administrátor možnosť jednoducho pridávať ďalšie oznamy, prípadne vymazávať už pridané. Vymazávať sa odkazy dajú jednoduchým kliknutím na slovo “vymazať”, ktoré sa nachádza za nadpisom každého z nich.

Pozn: Pri vymazávaní odkazov buďte opatrní, nakoľko si systém nevyžaduje potvrdenie vášho rozhodnutia. Na vyžiadanie potvrdenia by totiž bolo potrebné použiť ďalšiu technológiu, (JavaScript, prípadne Flash, alebo Java) čo by znížilo kompatibilitu webu s jednoduchšími prehliadačmi.

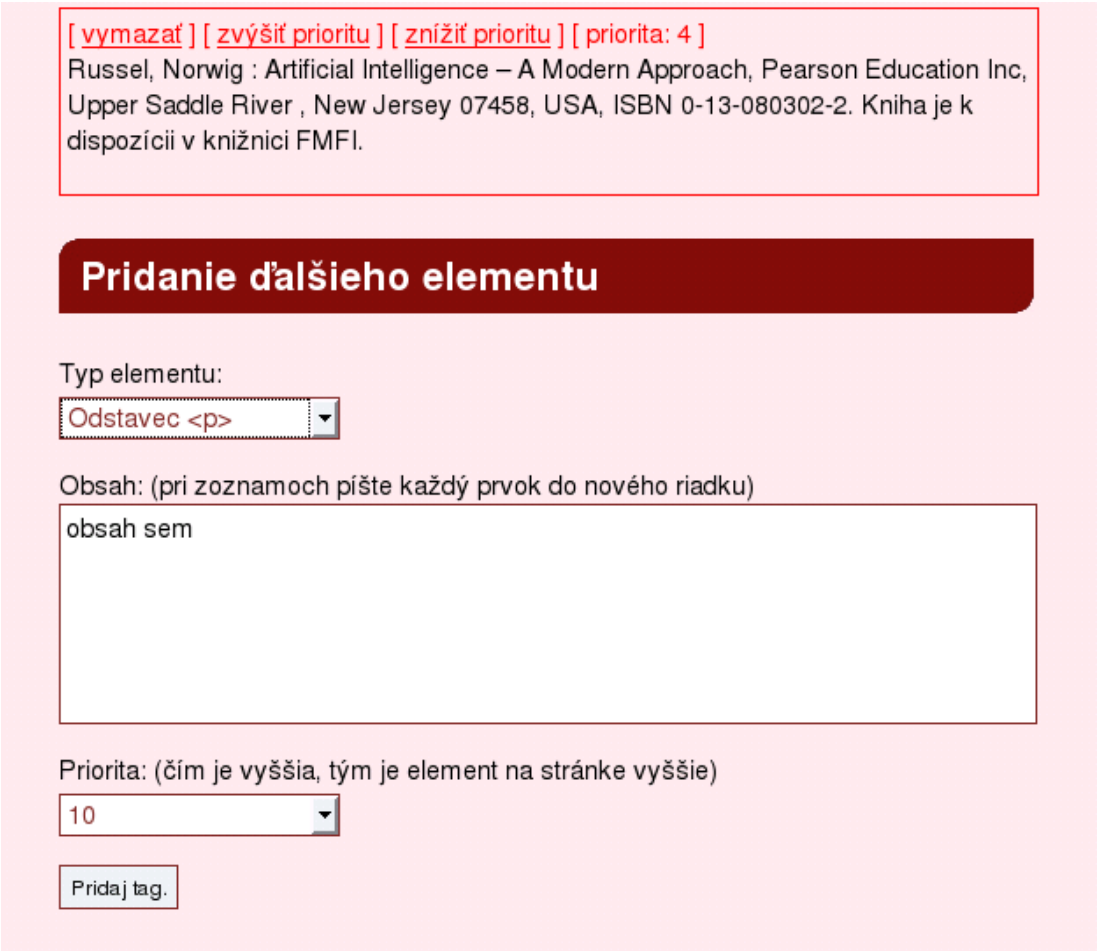
Na pridávanie nových odkazov má administrátor k dispozícii jednoduchý user-friendly formulár. Ten sa skladá z dvoch povinných polí – nadpisu a textu odkazu, a jedného nepovinného poľa – url odkazu, na ktorý oznam odkazuje. Do povinných polí je možné zadať a kýkoľvek jednoduchý text (samozrejme s limitovanou dĺžkou). Nie je však povolené používať niektoré špeciálne znaky. Do nepovinného poľa sa zadáva kompletné url odkazu – tj. v tvare “http://www.domena.sk”, pričom ak sa pole nechá prázdne, oznam nikam neodkazuje. (pozri obr.2.)



obr.2. - modifikácia aktualít

Info, Pravidlá, Hodnotenie, Kontakty – Na týchto podstránkach môže administrátor pridávať, vymazávať, alebo premiestňovať celý obsah. Ten sa skladá z jednotliných elementov. Systém podporuje päť typov elementov, ktoré zodpovedajú najpoužívanejším HTML tagom – odstavce P a DIV, nadpis H3, neusporiadaný a usporiadaný zoznam UL a OL. Poradie zobrazenia elementov na stránke je dané ich prioritou – čím je vyššia, tým je na stránke element vyššie. Mazanie a zmena priority elementov je možné pomocou odkazov pri samotných elementoch.

Samotný formulár na pridávanie elementov pozostáva z troch polí – výberové polia s typom a prioritou elementu a textové pole, kam sa zadáva obsah. V prípade odstavcov a nadpisu nie je tvar zadaného obsahu dôležitý. Ak však pridávate zoznam, každý nový riadok (znak nového riadku, resp. stlačenie enteru) sa chápe ako jeden prvok zoznamu. (pozri obr.3.)



obr.3. - pridávanie elementov

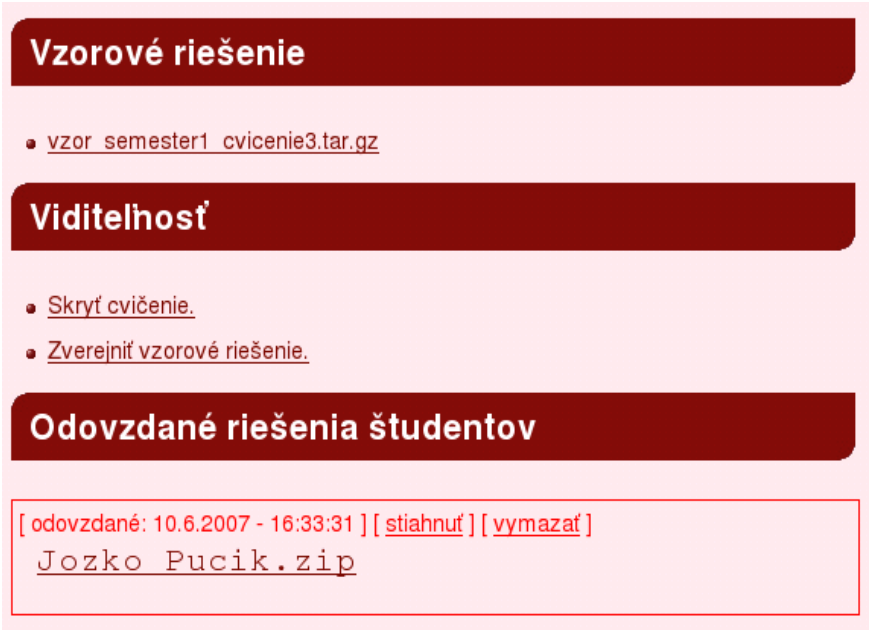
Prednášky – V tejto časti môže administrátor vymazávať, alebo uploadovať na server súbory s prezentáciami z prednášok. Maximálna veľkosť súboru je 5MB. Prednášky sú rozdelené na 2 skupiny – letné a zimné, resp. z predmetu ZUI1 a ZUI2. Utriedené sú abecedne, teda je jednoduché dosiahnuť ich požadované poradie zvolením vhodných názvov súborov.

obr.4. - pridávanie prednášok

Cvičenia – Najčastejšie používanou administrátrskou funkciou v sekcii s cvičeniami bude prevdepodobne skrývanie a zverejňovanie samostatných cvičení. To sa dá vykonať po kliknutí na samostatné cvičenie, v spodnej časti stránky – pod nadpisom "Viditeľnosť".

V prípade, že k cvičeniu je priložené aj vzorové riešenie, v sekcii "Viditeľnosť" sa nachádza taktiež možnosť zverejniť, alebo skryť toto vzorové riešenie. Štandardne sú vzorové riešenia pred študentmi skryté v špeciálnych adresároch so zakódovanými názvami a na stránke sa pre nich nezobrazujú.

Okrem nastavovania viditeľnosti cvičení a vzorových riešení si môže administrátor sťahovať, prípadne vymazávať odoslané riešenia od študentov. V najspodnejšej časti stránky – pod nadpisom "Odovzdané riešenia študentov" sa nachádzajú všetky súbory, ktoré k danému cvičeniu študenti uploadli na server. Mali by to byť archívy obsahujúce riešenia, s veľkosťou nepresahujúcou maximálnu dĺžku riešenia. Túto môže zmeniť webmaster tým, že zmení hodnotu konštanty MAX_UPLOAD_SIZE v konfiguračnom súbore "configuration.php" v kreňovom adresári webu. Predvolená dĺžka je 500kB. Bežný administrátor, bez prístupových práv k súborom na serveri toto zmeniť nemôže. Ku každému odovzdanému riešeniu má samozrejme administrátor k dispozícii aj presný čas a dátum odovzdania. (pozri obr.5.)

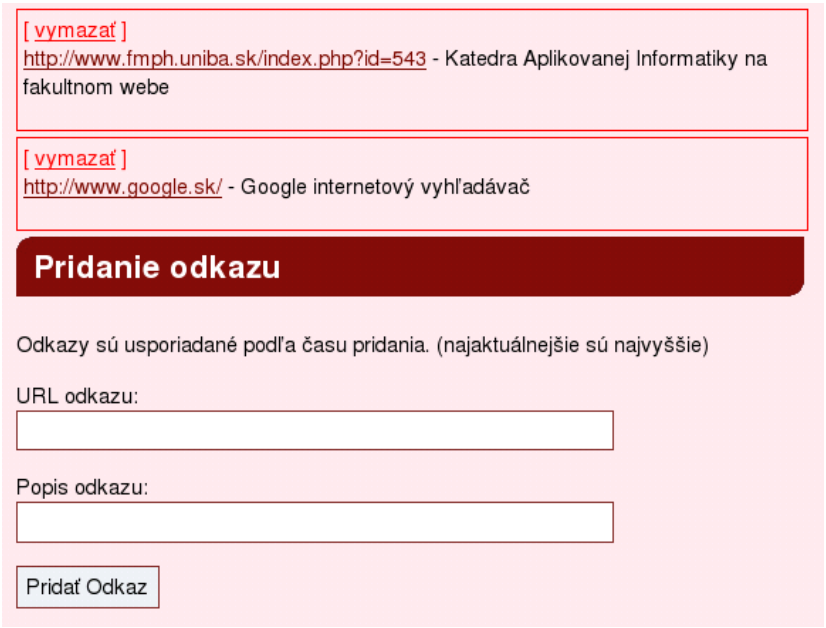


obr.5. - cvičenia a odovzdané riešenia

Linky – V tejto časti webu sa nachádzajú odkazy na rôzne iné užitočné stránky. Administrátor ich môže do systému jednoducho pridávať, alebo ich z neho vymazávať. Odkazy sa vymazávajú kliknutím na slovo "vymazať" bezprostredne nad samotným odkazom.

Na pridávanie odkazov sa v spodnej časti podstránky nachádza krátky formulár s dvoma poľami – URL odkazu a Popis odkazu. Do poľa URL sa zadáva kompletná adresa webu na ktorý odkazujeme (v tvare napr. http://www.domena.sk/). Do poľa popis administrátor napíše krátky, niekoľkoslovný popis odkazovaného webu. (pozri obr.6.)

Pozn: Odkazy sú na stránke usporiadané podľa času pridania tak, že najnovšie sú najvyššie.



obr.6. - pridávanie a mazanie odkazov

2.3. PHP stromový model dedičnosti

Celý web ZUI je tvorený nie celkom typickým spôsobom. Ide o webovskú aplikáciu, ktorá je naprogramovaná objektovo v jazyku PHP. Na štrukturovanie aplikácie som použil svoj vlastný model – akýsi stromový model dedičnosti. Ide o model pripomínajúci často používaný "factory model", avšak s niekoľkými rozdielmi. Pri factory modele existuje jeden skript, ktorý rozhoduje o tom, aká stránka sa zobrazí na základe často zložitého algoritmu. V mojom modeli je toto rozhodovanie distribuované medzi uzly stromovej štruktúry tried, kde každý uzol je dedičom svojho otca a koreň je základná trieda "PAGE". Táto trieda PAGE má definované základné metódy:

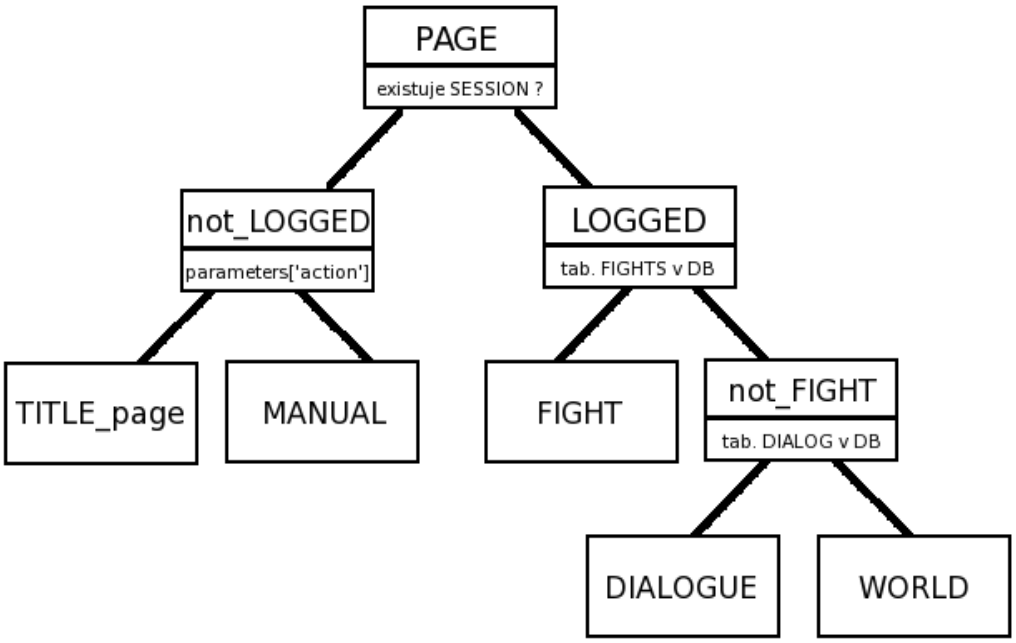
- GetParameters
- Display
- DisplayHead
- DisplayBody
- abstraktná metóda DisplayContent

Platí, že metóda Display zobrazí niektoré nevyhnutné HTML tagy a vo svojom tele zavolá metódy DisplayHead a DisplayBody. DisplayHead potom vypíše hlavičku dokumentu, zatiaľ čo DisplayBody zobrazí tie prvky dokumentu, ktoré sú rovnaké pre všetky podstránky (napr. menu, alebo hlavička dokumentu) a potom zavolá metódu DisplayContent, ktorú majú všetky podstránky (teda listy v strome) predefinovanú tak, aby zobrazila ich vlastný obsah.

Rozhodovanie, aká stránka sa zobrazí funguje nasledovným spôsobom: Pri každej požiadavke užívateľa o zobrazenie stránky sa na webserveri spustí skript index.php. Ten nerobí nič iné, ako že vytvorí objekt PAGE – tj. zavolá jeho konštruktor. Trieda PAGE, čiže koreň stromového modelu, si pri volaní konštruktora zavolá metódu GetParameters, ktorá zistí všetky hodnoty z polí GET, POST, prípadne SESSION (a pridá si ich do poľa "this->parameters"). Potom sa trieda PAGE na základe hodnôt vo svojom poli "parameters" rozhodne, ktorého svojho syna vytvorí. Ak je syn listom stromu, tak má predefinovanú metódu DisplayContent tak, aby zobrazila jeho obsah, a jeho konštruktor volá zdedenú metódu Display. Ak však syn listom nie je, tak sa situácia opakuje – pozrie sa na hodnoty v poli "parameters" a podľa nich sa rozhodne ktorého svojho syna vytvorí.

Tento prístup je výhodný hlavne pri komplexných aplikáciách s množstvom rôzne hierarchicky štrukturovaných podstránok. Nemusíme totiž mať jeden enormne zložitý skript na rozhodovanie, ktorú podstránku zobraziť, ale toto rozhodovanie si rozložíme na viacero jednoduchších podmienok.

Na lepšie pochopenie uvediem jednduchý príklad aplikácie, ktorá využíva stromový model. Majme nejakú online hru, kde sa hráč po prihlásení môže prechádzať po hernom svete, rozprávať sa s fiktívnymi postavami, alebo bojovať. Bez prihlásenia si môže čítať manuál ku hre, alebo prezerať titulnú stránku. Stromová štruktúra tried by mohla vyzerať ako na obrázku 7.



obr.7. - príklad použitiamodelu stromovej dedičnosti na online hre

V tomto prípade sa pri vytvorení objektu PAGE skriptom index.php, konštruktor triedy PAGE pozrie, či existuje vytvorený SESSION pre dané pripojenie. Ak nie, tak vytvorí svojho nasledovníka – objekt triedy not_LOGGED. Ten sa zas pozrie na hodnotu premennej "action" v poli parameters a podľa nej rozhodne, či vytvoriť objekt TITLE_page, alebo MANUAL. Tieto objekty sú listami, takže majú metódu DisplayContent tak, aby zobrazovala ich obsah. Ich konštruktor teda iba zavolá metódu Display. Triedy LOGGED a not_FIGHT sú príkladom faktu, že rozhodovanie, akého syna vytvoriť môže prebiehať aj na základe nejakých hodnôt vybraných z databázy.

Už v tomto jednoduchom príklade je viditeľné, že keby sme použili bežný "factory" model, tak by sme potrebovali pri každom zobrazení stránky volať skript, ktorý by sa rozhodoval na základe jednej podmienky na SESSION, hodnoty "action" (ktorú by hľadal v poliach GET a POST), a dvoch dotazov na databázu. Takýto skript by bol dosť zložitý a zložitosť by rýchlo narastala, keby sme mali rozsiahlejšiu aplikáciu. Nehovoriac o tom, že rozširovanie aplikácie po čase by bolo neprehľadné.

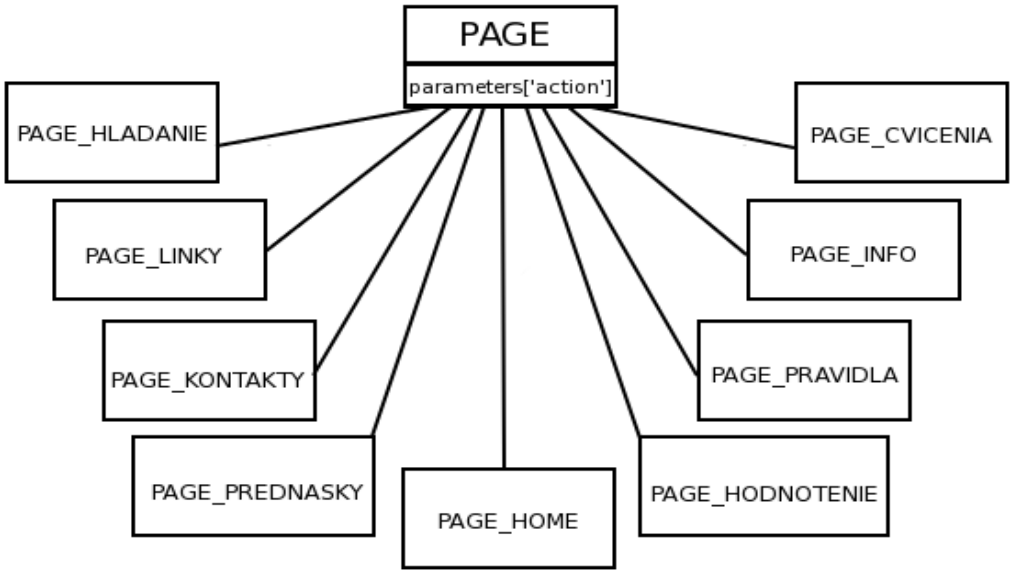
Tu sa dostávame k otázke aktualizácie webu – a to v zmysle pridávania nových častí, resp. podstránok, či odoberania už existujúcich. To je v prípade použitia tohto systému implementačne nenáročné. Pre pridanie ďalších častí webu si len vyberieme vhodné miesto v strome, vytvoríme jedinú zdedenú tiredu a upravíme rozhodovací algoritmus jej otca. To znamená, že pridávanie nových súčastí prebieha výhradne lokálne a neovplyvňuje väčšinu aplikácie.

Ďalšou výhodou je možnosť jednoduchej zmeny celkového designu. Všetky prvky, ktoré sú spoločné pre všetky stránky sú totiž zachytené v metóde DisplayBody, ktorá sa dedí. Stačí teda upraviť túto metódu v triede PAGE (koreni stromu) a úprava sa bude týkať všetkých jej následovníkov – teda všetkých podstránok.

2.4. Použitie stromového modelu na tvorbu webu ZUI

V tejto fáze web ZUI nie je komplexnou aplikáciou, ktorá by sa nedala naprogramovať inak, ako za použitia modelu opísaného v predchádzajúcej kapitole. Každopádne si však treba uvedomiť, že požiadavky na web sa budú časom vyvýjať a zvyšovať. Preto je potrebné zabezpečiť pružnosť a jednoduchosť zmien a aktualizácií webu. Už som vysvetlil, ako k tomuto stromový model dedičnosti prispieva.

V tomto prípade si môžte všimnúť, že pri rozhodovaní v triede PAGE sa nezaujímame o to, či je niekto prihlásený. To, či je prihlásený administrátor riešime až v jednotlivých listoch stromu, nakoľko podstránky sú vždy tie isté, nezávisle na tom, či je niekto prihlásený – pribúdajú iba niektoré funkcie. Takéto riešenie samozrejme nie je nevyhnutné, no zdalo sa mi praktickejšie. (obr.8.)



obr.8. - stromový model webu ZUI

Objekt PAGE teda pri prihlásení načíta hodnoty z polí GET a POST pomocou funkcie GetParameters a potom sa na základe hodnoty “action” z poľa parameters rozhodne, ktorého syna vytvorí. Syn potom zavolá zdedenú metódu Display, ktorá zas zavolá jeho predefinovanú metódu DisplayContent, čím sa zaistí zorbrazenie správneho obsahu.

Metódy triedy PAGE, ktoré jej potomkovia dedia:

GetParameters – Načíta všetky hodnoty z polí GET a POST a skopíruje ich do poľa this->parameters. V prípade, že by sa jedna premenná nachádzala v poli GET a POST zároveň, tak sa pridá iba hodnota z POST. Okrem toho skopíruje hodnotu parameters['action'] aj do zvláštnej premennej this->action.

Display – Zobrazí stránku, pričom používa metódy DisplayHeader, DisplayLeft, DisplayContent a DisplayFooter. Túto metódu nezmenenú dedia všetci potomkovia triedy PAGE.

DisplayHeader – Zobrazuje header dokumentu.

DisplayLeft – Zobrazí menu na ľavej strane stránky.

DisplayFooter – Zobrazí pätičku stránky s prihlasovacím formulárom pre administrátorov.

DisplayContent – Táto metóda triedy PAGE nerobí nič. Slúži iba na to, aby ju mohla volať metóda Display, pričom všetci potomkovia triedy PAGE si ju predefinujú tak, aby zobrazovala ich obsah.

DisplayTags – Túto funkciu zdedia a používajú niektorí potomkovia triedy PAGE. Slúži na načítanie z databázy a zobrazenie elementov, ktoré sú súčasťou obsahu jednotlivých podstránok. Taktiež slúži na pridávanie ďalších elementov v prípade, že je prihlásený administrátor.

Dôležité súbory:

index.php – Skript, ktorý sa volá pri každom načítaní webu. Najskôr spustí skripty "security.php" a "configuration.php" a potom vytvorí inštanciu triedy PAGE.

security.php – Tento skript prejde všetky hodnoty v poliach POST a GET, teda všetky vstupy posielané serveru užívateľom, a vykoná na nich potrebné bezpečnostné opatrenia (odfiltruje nebezpečné znaky, odstráni biele miesta, atď...).

configuration.php – Súbor s nastaveniami. Tento skript definuje potrebné konštanty, ktoré sa neskôr využívajú inde. Konštanty sa buď týkajú prístupu k databáze, alebo obsahujú doplnkové nastavenia webu. Tento súbor je ľahko modifikovateľný aj pre človeka, ktorý nerozumie implementácii webu do hĺbky. Obsahuje však aj časť kódu, ktorá by nemala byť nikým zmenená – a to definícia DB-handlu ako konštanty. To zjednodušuje prístup k databáze na iných miestach v kóde.

Adresáre:

classes – Obsahuje samostatné súbory jednotlivých tried.

css – Súbory s CSS štýlmi.

cvicenia – Adresár obsahuje súbory v tvare semesterX_cvicenieY.inc (X a Y sú čísla semestra a cvičení), z ktorých sa načítavajú zadania úloh na cvičenia. Ak je potrebná modifikácia zadania nejakej úlohy, stačí upraviť tieto súbory. Taktiež je tu podadresár "vzorove_priklady", v ktorom sa nachádzajú ďalšie adresáre. Názov týchto adresárov je md5-hash stringu "semesterXcvicenieY", kde X predstavuje číslo semestra a Y číslo cvičenia. V každom z týchto adresárov sa nachádza jeden súbor s presným názvom "vzor_semesterX_cvicenieY.tar.gz", čo je archív obsahujúci vzorové riešenie úlohy z cvičenia Y zo semestra X. Dôvodom, prečo je názov týchto adresárv hashovaný je, že nechceme aby mali študenti prístup k vzorovým riešeniam skôr, ako ich administrátor zverejní.

dlv – Tu sa nachádza samostatná aplikácia – webovské rozhranie pre DLV solver, ktorá je súčasťou zadania a pomôckou pri riešení cvičenia číslo 9.

files – Obsahuje rôzne súbory, ktoré majú návštevníci webu k dispozícii na stiahnutie.

img – Obrázky, ktoré sú súčasťou designu stránok.

prednasky – Obsahuje powerpointové prezentácie z prednášok rozdelené do podadresárov ZUI1 a ZUI2. Obsah týchto adresárov je dynamicky čítaný skriptom.

upload – Tu sa nachádzajú všetky riešenia cvičení, ktoré študenti na server nahrali. Tie sú rozdelené podľa cvičení do adresárov s názvom "semesterX_cvicenieY", kde X je číslo semestra a Y číslo cvičenia. Riešenia sú uložené v archívoch, ktorých názov je menom študenta, ktorý ho odovzdával. Prístup k týmto súborom má administrátor priamo z podstránky s konkrétnym cvičením. Študent z tejto podstránky môže súbory iba uploadovať, nemôže si ich však prezerať.

2.5. Prístupnosť a použiteľnosť

Ak má byť nejaký web používaný viacerými používateľmi, musíme predpokladať, že na jeho zobrazenie budú použité rôzne zariadenia a softvér. Samozrejme každá skupina užívateľov môže mať na web rozdielne požiadavky. Je preto dôležité zabezpečiť, aby bol web rovnako dobre zobrazovaný na rôznych browseroch, rozlíšeniach, farebných schémach, či celkovo rozličných zariadeniach (napríklad zrakovo postihnutí užívatelia často používajú miesto monitora screen readery). Tieto, a mnohé iné aspekty webu zahŕňa jeho použiteľnosť.

Prvým krokom k optimalizácii webu je zabezpečenie validity dokumentu – tj. syntaktickej správnosti jeho zdrojového kódu. Za typ dokumentu sme zvolili Strict XHTML verziu 1.0. Tento typ dokumentu je ľahko parsovateľný, a dostatočne optimalizovaný, resp. zjednodušený. Je založený na XML, teda má šancu, že sa bude v budúcnosti ešte dlhšiu dobu rozšírene používať. Pri programovaní webu sme preto mali na pamäti syntaktickú správnosť a dokument priebežne validovali – za použitia validátora, ktorý poskytuje organizácia W3C.

Design webu je zabezpečený výhradne kaskádovými štýlmi CSS, ktoré sme taktiež priebežne validovali, aby sme dosiahli zrozumiteľnosť CSS kódu pre všetky browsery s podporou CSS štýlov.

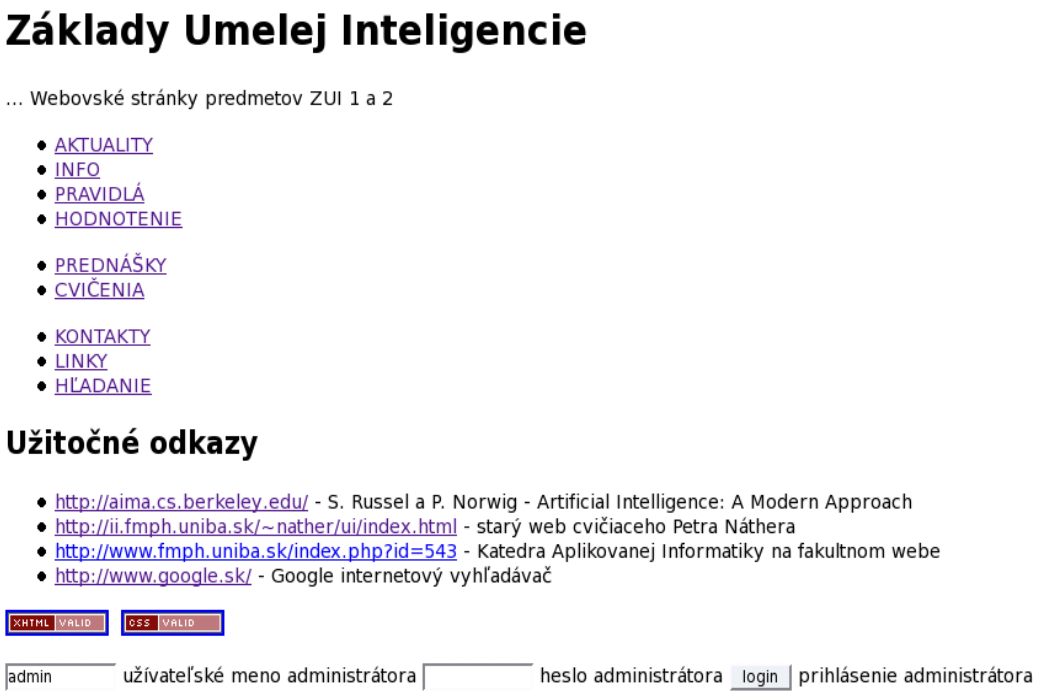
Bohužiaľ rôzne prehliadače zobrazujú CSS design rôznym spôsobom, čím vznikajú designové chyby. Preto nestačilo iba jednoduché validovanie CSS kódu, ale bolo treba manuálne testovať web na jednotlivých prehliadačoch. Rozdiely v zobrazení jednotlivými prehliadačmi sme potom riešili použitím tzv. CSS hackov. Stretli sme sa napr. s tým, že Internet Explorer 6 nesprávne zobrazoval niektoré prvky. Potrebovali sme teda upraviť zobrazenie prvku v IE6, ale nezmeniť pritom zobrazovanie ostatnými prehliadačmi. Predefinovali sme preto niektoré vlastnosti tak, aby ich IE6 zbrazoval správne, ale ponechali sme aj staršie definície zobrazenia vlastností. K tým sme pridali indikátor “!important”, ktorý prikazuje browserom použiť danú vlastnosť aj v prípade, že je niekde inde v kóde predefinovaná. IE6 však ešte nepozná indikátor “!important”, takže sa riadi neskoršou definíciou vlastnosti. Takto sme prinútili IE6, aby zobrazoval design na základe iných definícií vlastností, ako ostatné prehliadače. Toto je len jeden z viacerých rôznych CSS hackov použitých pri vytváraní designu webu ZUI.

Takýmto postupným dolaďovaním sa podarilo dosianuť, že web ZUI je kompatibilný so všetkými bežne používanými prehliadačmi – Mozilla Firefox, Internet Explorer 6 a 7, Opera, atď. Navyše je optimalizovaný aj na zobrazovanie prehliadačmi, ktoré nepodporujú CSS, prípadne textovými prehliadačmi, ktoré nezobrazujú žiadnu grafiku. Taktiež je zabezpečené bezproblémové používanie webu s použitím screen readera.

Tu sa dostávame k otázke prístupnosti webu ZUI. Už spomínaná kompatibilita so screen readermi je zabezpečená popismi jednotlivých polí vo formulároch a alternatívnymi textami ku grafickým prvkom webu. Spomínané popisy sú jednoduchým CSS hackom nastavené tak, aby sa na stránke zobrazovali iba vtedy, keď browser nepodporuje CSS, či grafiku. Preto sú viditeľné iba vtedy, keď sú potrebné a nekazia design stránky ak potrebné nie sú. Okrem toho sme sa samozrejme vyhli použitiu framov, ktoré znemožňujú používanie bez myšky (okrem iných nevýhod týkajúcich sa obzvlášť SEO) a tabuliek, ktoré taktiež často komplikujú

orientáciu na stránkach zrakovo postihnutým, alebo akýmkoľvek používateľom screen readerov. Už spomínanou samozrejmosťou je nepoužitie technológií Flash, alebo Java a zobrazovanie textov pomocou hypertextu a nie obrázkov. K prehľadnosti webu taktiež prispieva rozumné štrukturovanie obsahu vrámci každej stránky – menu vo forme zoznamu odkazov je umiestnené bezprostredne pod nadpisom a pod ním sa nachádza samotný obsah webstránky. Až v poslednej časti stránky je prihlasovací formulár pre administrátorov, ktorý nie je bežným užívateľom vôbec používaný. Teda obsah je štrukturovaný tak, aby boli najčastejšie čítané časti čo najvyššie pod nadpisom, a najmenej čítané časti boli na konci.

Vrámci sprístupňovania webu zdravotne postihnutým užívateľom nesmieme zabudnúť na čitateľov s miernym zrakovým postihnutím, ktorí nepoužívajú screen readery. Týmto užívateľom je zabezpečené bezproblémové resizovanie, resp. zväčšovanie nie len textu, ale celého obsahu stránky. Pri stránkach, ktoré na toto nemyslia spôsobuje často zmena veľkosti textu rozhádzanie designu a stránka sa stáva nečitateľnou. Tomuto sme sa vyhli použitím relatívnych jednotiek (ex, em) pri väčšine CSS vlastností súvisejúcich s veľkosťami. So zväčšujúcim sa písmom sa tak zväčšujú aj všetky ostatné prvky webu a design zostáva neporušený.



obr.9. - zobrazenie webu bez podpory CSS štýlov

2.6. Bezpečnosť

Keď sa bavíme o bezpečnosti na úrovni PHP aplikácie, pravdepodobne máme na mysli hlavne kontrolu užívateľských vstupov. Všetky údaje, ktoré aplikácia dostáva od užívateľov musia byť pozorne skontrolované a odfiltrované. Táto kontrola musí prebiehať na serveri – v žiadnom prípade nie na strane klienta. Akékoľvek kontroly na strane klienta (napr. JavaScriptové podmienky) sa dajú ľahko obísť alebo oklamať.

Dáta posielané klientom sú v našom prípade v poliach GET a POST. Tieto vstupy filtruje naša aplikácia na dvoch úrovniach – samostatným skriptom "security.php" a neskôr v jednotlivých triedach, ktoré sa starajú o zobrazenie všetkých podstránok.

Skript "security.php" prejde všetky hodnoty obsiahnuté v poliach GET a POST a vykoná na nich niekoľko úprav:

- odstráni biele miesta zo začiatku a konca reťazca znakov (zabráni tým, aby užívateľ obchádzal neskoršie kontroly vyplnenia formulárov medzerami)
- nahradí špeciálne znaky ich korektnými HTML alternatívami, čím zachováva validitu webu
- pridá spätné lomítka pred úvodzovky a apostrofy, aby zabránil pokusom užívateľa podsúvať webserveru nechcené príkazy

Jednotlivé triedy potom vykonávajú špecifickejšie kontroly. Napr. trieda PAGE_CVICENIA si pri odovzdaní zadania skontroluje veľkosť uploadovaného súboru a dĺžku jeho názvu pred tým, ako ho skopíruje na server.

2.7. SEO

SEO, alebo Search Engine Optimalization je sada techník, ktorá sa snaží dosiahnuť takú formu dokumentu, že bude vysoko hodnotený webovskými vyhľadávacími službami (napr. Google). Dokumenty sa optimalizujú na určitú frázu - "keyword". Cieľom je, aby vyhľadávač po zadaní daného keywordu zobrazil náš dokument vo výsledkoch čo najvyššie. Štandardne je SEO dôležitou súčasťou tvorby hlavne komerčných stránok. Napriek tomu sa o optimalizáciu pokúsime aj pri tvorbe webu ZUI. Uľahčíme tak študentom hľadanie stránky a zbavíme ich potreby pamätať si jej adresu.

Za keyword sme si v našom prípade zvolili slovné spojenie "Základy Umelej Inteligencie" - čiže celkom prirodzene názov predmetu. Hlavnými krokmi, ktoré sme podnikli pre optimalizáciu webu sú:

- Titulok (title tag) a hlavný nadpis (h1 tag) stránky sú tvorené keywordom. Vyhľadávače hľadajú stránky podľa výskytu keywordov v dokumente, pričom výskyt práve na týchto dvoch miestach hodnotia najvyššie.
- Dokument je validný. Vyhľadávače vyššie oceňujú kvalitné dokumenty bez chýb.
- Meta-informácie dokumentu sú vhodne vyplnené. Hodnota "keywords" mierne prispieva k hodnoteniu dokumentu, preto sme do nej zahrnuli náš zvolený keyword. Momentálne je však hodnotiacimi algoritmami vyhľadávacích systémov hodnotená slabšie ako v minulosti.

- Meta-informácia “description” poskytuje stručný popis webu, ktorý môže vyhľadávač priamo využiť v zozname výsledkov. To uľahčuje čitateľovi nájdenie toho, čo hľadá v kope výsledných odkazov.
- Vyhýbame sa praktikám, ktoré sú vyhľadávačom hodnotené ako neoprávnený pokus o zvýšenie návštevnosti (tzv. black hat SEO). Napríklad zahrnutie veľkého množstva keywordov do textu stránky. Optimálne zastúpenie keywordu v texte stránky je asi 5-8%. Na našej úvodnej stránke som externým počitadlom nameral zastúpenie slov “základy”, “umelej” a “inteligencie” v rovnakej hustote – 4,4%.

Existuje mnoho ďalších vecí, ktorými sa dá dosiahnuť lepšie hodnotenie nášho webu, ale tie sa dajú podniknúť až za používania stránky. Dôležitá je napríklad častá aktualizácia obsahu. Samozrejme taktiež navštevovanosť webu a počet stránok, ktoré obsahujú odkazy na tento web, pričom vyššiu hodnotu majú odkazy z lepšie hodnotených stránok.

V každom prípade, výsledky takejto optimalizácie sa nedajú pozorovať hneď. Vyhľadávače upravujú svoje hodnotenia iba pri návštevách webu svojimi automatizovanými programami, ktoré si web prezrú iba raz za čas. To znamená, že na priaznivé výsledky si budeme musieť počkať pár dní, až týždňov. Urýchliť sa to dá zabezpečením odkazov na množstve iných stránok.

2.8. Výhody a nevýhody

Aké sú teda výhody a nevýhody zvolenej implementácie oproti jednoduchému HTML dokumentu? Čo sa týka výhod, za najdôležitejšiu považujem možnosť jednoduchej aktualizácie, alebo zmien, bez potreby prepisovania množstva dokumentov zvlášť. O to sa stará stromový model dedičnosti (kap. 2.3). Veľkým prínosom je administrátorský systém, ktorý umožňuje všetky potrebné činnosti pre cvičiacich a prednášajúcich bez práv k súborom na serveri, či vedomostí o webdesigne. Priebeh cvičení a kontrolu riešení značne uľahčuje možnosť uploadovania riešení priamo cez webstránku.

Čo sa týka nevýhod, najväčšou je zrejme náročnosť na technológie na strane servera. Okrem bežného webservera potrebujeme podporu PHP a SQL databázy. Úkony, ktoré server vykonáva pri každom zobrazení stránky sú taktiež o niečo výpočtovo náročnejšie, no server napriek tomu nezatažujú natoľko, aby to bol vážny problém. Za poslednú teoretickú nevýhodu môžme považovať bezpečnostné riziko. Zložitejšia aplikácia komunikujúca s užívateľom je vždy náchylnejšia na prieniky, ako statický dokument. Ak sa však dôsledne kontrolujú vstupy, reálne nebezpečenstvo nehrozí.

Aby som to zhrnul, prirodzene vidím zvolenú optimalizáciu ako najoptimálnejšiu. Výhody, ktoré sú s ňou spojené podľa môjho názoru značne prevyšujú vymenované nevýhody.

3. Obsah – cvičenia

Jedným z hlavných cieľov, tejto práce je vytvorenie zadaní úloh, ktoré budú študenti riešiť na cvičeniach zo ZUI. Tieto úlohy by mali pokrývať najdôležitejšie časti učiva preberaného na prednáškach a ich cieľom je naučiť študentov, prakticky využívať vedomosti nadobudnuté na tomto predmete.

Pri tvorbe niektorých zadaní som sa do značnej miery inšpiroval cvičeniami z minulých rokov a ich zadania zdokonalil, prípadne doplnil. Naopak niektoré zadania sú celkom nové a nezahŕňajú nič, čo sa už vyskytlo na cvičeniach ZUI.

Cvičenia som sa snažil oživiť a zatraktívniť pre študentov, no neuberať pritom na ich náročnosti. Naopak si myslím, že náročnosť cvičení je v porovnaní s minulými rokmi o niečo vyššia. Tomuto faktu sa bude zrejme mierne prispôsobovať aj spôsob bodového hodnotenia cvičení.

Atraktívnosť, cvičení je dosahovaná sústredením úloh na problémy, ktorých riešenie by malo začiatočníkov v odbore umelej inteligencie baviť. Podstatná časť úloh je z oblasti agentovo orientovaného programovania (Wumpus, Shakey...) a jedna úloha je zameraná na vytvorenie ťahovej hry človeka proti UI.

Pre lepšiu predstavu riešiteľa o problematike sú niektoré zadania, či predpripravené programy obohatené grafickou zložkou. Pozornosť študenta sa často snažím zamerať na podstatnú časť úlohy tým, že pridávam k zadaniu predprogramovanú časť riešenia, ktorú stačí doplniť o dôležitú časť kódu. Šetrím tak jeho čas, ktorý by strávil riešením podproblémov nesúvisejúcich s UI. K jednej úlohe som pre študenta dokonca pripravil webovské rozhranie DLV solvera, aby ho nemusel inštalovať na svoj počítač, a mohol úlohu riešiť priamo cez prehliadač.

V niektorých úlohách sa využívajú aj služby, ktoré poskytujú externé stránky. Menovite celosemestrálna úloha letného semestra je zameraná na vytvorenie agenta schopného zúčastniť sa verejnej a medzinárodne známej súťaže TAC (Trading Agent Competition). Podobne prvé cvičenie zahŕňa komunikáciu s chatbotom Alice na webe alice.pandorabots.com, ako aj tvorbu vlastného chatbota za pomoci interfacu na webe www.personalityforge.com.

3.1. Cvičenie ZUI1 – 1 – Turingov test, chatboti

V prvom rade treba podotknúť že toto cvičenie je považované za akési motivačné prvé cvičenie. Jeho obtiažnosť je nízka a snaží sa zábavným spôsobom sprostredkovať študentovi prvé kontakty s chatbotmi.

Zadania dvoch úloh prvého cvičenia predchádza zopakovanie podstatných informácií z prednášky. Uvedená tu je Turingova definícia inteligencie z roku 1950 a štyri body, ktoré by mal podľa tejto definície zvládať inteligentný systém.

Vrámci prvej úlohy má študent navštíviť stránku alice.pandorabots.com, kde sa "porozpráva" s chatbotom Alice (víťazom Loebnerovej ceny z roku 2001). Pri rozhovore si má študent všímať ako, a či vôbec Alice zvláda 4 činnosti uvedené vyššie. Jej schopnosti zvládať tieto činnosti by mal riešiteľ oznámkovať a svoje známky zdôvodniť. Prípadné návrhy ako chatbota vylepšiť sú vítané.

obr.10. - rozhovor s chatbotom Alice na webe alice.pandorabots.com

Druhou úlohou pre študenta je vytvorenie vlastného chatbota na externom portáli www.personalityforge.com. PersonalityForge ponúka študentom jednoduchý interface, v ktorom si za krátky čas môžu spraviť vlastného chatbota, nastaviť mu kľúčové výrazy a odpovede, prípadne rôzne pluginy. Samozrejmosťou je možnosť rozprávať sa so svojim botom, ako aj s botmi iných užívateľov.

3.2. Cvičenie ZUI1 – 2 – Jednoduchý reaktívny agent

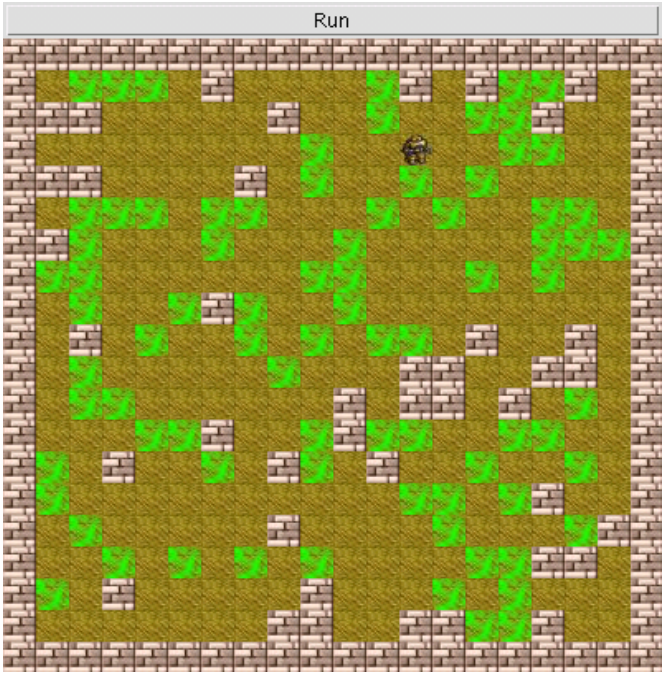
K tomuto cvičeniu máme pre študentov predpripravený program implementujúci agenta a jeho svet. Ide o aplikáciu v jazyku Java, ktorá predstavuje svet pozostávajúci z poľa pozícií, pričom na každej pozícii môže byť prekážka, nečistota, čisto, alebo agent. Úlohou agenta je vyčistiť tento svet od nečistôt. Aktuátormi agenta sú metódy **`moveFW()`** (pohyb agenta o 1 pozíciu dopredu), **`turnLEFT()`** a **`turnRIGHT()`** (otočenie o 90 stupňov vľavo, alebo vpravo) a metóda **`suck()`**, ktorá vyčistí agentovu pozíciu.

Agent má 1 perceptor, ktorý je implementovaný funkciou **`percept()`**. Tá vracia pole integerov rozmeru (2n+1)x(2n+1), kde n je tzv. dohľad agenta (tj. parameter určujúci ako ďaleko agent okolo seba vidí - zadáva sa pri spustení sveta). Toto pole je zaplnené číslami, ktoré reprezentujú pozície okolo agenta.

Súčasťami programu, ktoré sú pre riešiteľa zaujímavé sú súbory Guistarter.java a MyAgent.java. Guistarter je trieda, ktorá vytvorí a spustí svet, pričom dianie v ňom graficky zobrazuje v okne. Súbor MyAgent.java obsahuje zdedenú triedu MyAgent, vrámci ktorej má riešiteľ predefinovať abstraktnú metódu Act(). Táto metóda sa stará o správanie agenta v jednotlivých krokoch. Pravidlom je, že agent v jednom volaní Act() vykoná maximálne raz krok dopredu, alebo čistenie nečistôt.

Presnejšie informácie týkajúce sa spúšťania sveta sú k dizpozícii na stránke.

Úlohou je naprogramovať reaktívneho agenta, ktorý sa pokúsi vyčistit prostredie. Agent si nesmie nič zapamätávať, smie konať len na základe vnemu z daného kroku. Slovo “pokúsi sa” je v zadaní schválne, nakoľko nie je možné naprogramovať agenta, ktorý by zaručene vyčistil náhodné prostredie bez ukladania poznatkov o svete, alebo predchádzajúcich akciách.



obr.11. - grafické zobrazenie sveta s agentom

3.3. Cvičenie ZUI1 – 3 – Goal-oriented agent

V tomto cvičení sa riešitelia vrátia k prostrediu z minulého cvičenia (kapitola 3.2.). Svet zostáva nezmenený, iba úloha je zložitejšia. Tentokrát je cieľom naprogramovať goal-oriented agenta, ktorý úspešne vyčistí svet (ak je to možné). Môže si čokoľvek zapamätávať a vytvárať si model sveta. Pravidlom zostáva, že v každom kroku je povolené len jedno volanie percept() a jedno volanie niektorej z metód moveFW, turnLEFT a turnRIGHT.

K tomuto príkladu máme pre študentov pripravené aj vzorové riešenie. Agent je vyriešený tak, že si postupne vytvára model sveta, v ktorom sa nachádza. Na základe hodnôt, ktoré dostáva ako výstup funkcie percept() si vypĺňa hodnoty vo svojom vlastnom modelovom poli. Jeho konanie je rozdelené do troch fáz: V prvej fáze sa agent snaží dostať čo najďalej vo všetkých štyroch smeroch, čím zisťuje rozmer sveta. Keď toto dokončí, postupne presúma všetky zatiaľ neznáme miesta tým, že sa k nim dostane na dohľad. Nakoniec navštívi všetky znečistené pozície a vyčistí ich. Vykonávanie týchto troch fáz je mierne narušené tým, že agent sa v každom kroku pozrie na políčka bezprostredne okolo seba, a keď má vedľa seba nečistotu, najskôr ju vyčistí, až potom pokračuje vo vykonávaní ostatných činností. Domnievam sa, že tento prístup môže mierne znížiť počet krokov potrebných na vyčistenie sveta.

3.4. Cvičenie ZUI1 – 4 – Search problems

Prostredie pre toto cvičenie je upravenou verziou sveta z predchádzajúcich dvoch cvičení (kapitoly 3.2 a 3.3). Hlavným rozdielom je, že agentovo vnímanie nie je obmedzené dohľadom, ale vidí vždy celý svet (funkcia percept vracia celé pole so svetom). Štartovacia pozícia už nie je náhodná, ale je vždy v ľavom hornom rohu. Svet už navyše neobsahuje žiadne nečistoty. Spúštanie prostredia však zostalo rovnaké.

Úlohou agenta v tomto prípade nie je zbieranie nečistôt, ale nájdenie cesty zo začiatočnej pozície do pravého dolného rohu miestnosti. Na riešenie problému by si študentov agent mal konštruovať cestu pomocou niektorého z prehľadávacích algorotmov z prednášok (najlepšie A*). Úloha sa dá redukovať na prehľadávanie grafu, kde vrcholmi sú CLEAN pozície v poli, ktoré nám vracia percept a hrany sú medzi susednými miestami v poli. Agent by si mal najskôr pomocou percept() funkcie prezriet svet, nájsť v ňom čo najkratšiu cestu k cieľu (prípadne ak neexistuje žiadna cesta, napísať o tom hlásenie na konzolu), uložiť si ju do zoznamu krokov a až potom sa vydať na cestu.

3.5. Cvičenie ZUI1 – 5 – Constraint satisfaction problems

K piatemu cvičeniu máme pre študentov pripravený jednoduchý javovský program, ktorý implementuje klasický CSP problém – zafarbovanie austrálskych teritórií troma farbami (resp, zafarbovanie vrcholov grafu tak, aby susedné vrcholy mali rôzne farby). Prirodzene máme neorientovaný graf, ktorého vrcholy predstavujú jednotlivé teritóriá a hrany sú medzi tými vrcholmi, ktoré predstavujú susedné štáty.

Riešitelia budú modifikovať výhradne súbor MyMapGraph.java, kde si naprogramujú metódu ColorGraph(). Používať pri tom budú hlavne predpripravené funkcie IsBorder(S1,S2), GetName(), GetColor, a metódu Color(State,Color).

Očakáva sa, že na optimalizáciu algoritmu zafarbovania použijú riešitelia jednu z heuristických metód spomínaných na prednáške.

Pripravená aplikácia je konzolová, bez grafického zobrazovania. Spúšťa sa príkazom "java CSP". Po spustení sa vytvorí graf, zbehne procedúra ColorGraph() a vypíšu sa všetky teritóriá s farbami, ktoré ste im priradili. Taktiež program vypíše čas, ktorý potreboval na výpočet. V prípade, že graf nie je zafarbený správne, vypíše všetky chyby.

3.6. Cvičenie ZUI1 – 6 – Hry

Riešitelia tohoto cvičenia majú v podstate voľné ruky – celé riešenie si programujú sami, nakoľko zadanie je príliš všeobecné, aby sa opäť dala predprogramovať nejaká aplikácia. Tak isto je na nich aj voľba programovacieho jazyka. Ten v tomto prípade nie je dôležitý.

Úlohou je naprogramovať hru, kde agent, ktorý ju hrá predvedie použitie techník minimaxu, A-B orezávania, alebo cut-off searchingu. Riešenie je možné vo dvojiciach, alebo samostatne, pričom v riešení dvojíc nehrá proti agentovi človek, ale iný agent. Čas premýšľania agenta by mal byť obmedzený na niekoľko sekúnd. Príkladom hier, kde sa dá predviesť použitie minimaxu sú dáma, alebo šach.

Vzorové riešenie priložené k tomuto cvičeniu je implementáciou hry dáma, kde proti sebe hrajú dvaja agenti. Obaja používajú na heuristické ohodnotenie možných ťahov minimax, pričom používajú rôzne heuristické funkcie na hodnotenie jednotlivých stavov.

Riešenie ja naprogramované v Delphi. Svet vo vzorovom riešení dokáže fungovať s premenlivou veľkosťou, používa thready, každému agentovi prideľuje rovnaký čas, ktorý môže byť užívateľom zmenený v GUI a kontroluje legitímnosť ťahov. Umožňuje dokonca aj hru človeka proti agentovi, alebo dvoch ľudí proti sebe.

3.7. Cvičenie ZUI1 – 7 a 8 – Propositional logic – Wumpus world

Úloha určená na 7. a 8. cvičenie je o niečo náročnejšia ako predchádzajúce úlohy. Hlavne pokiaľ ide o čas strávený riešením. Z toho dôvodu je rozdelená na dve cvičenia. Na cvičení číslo 7 sa cvičiaci oboznámi so zadaním a pripraví si teoretický podklad pre riešenie. Zvolí si vhodný programovací jazyk a spôsob implementácie. Cvičenie čislo 8 slúži na konzultáciu problémov, ktoré sa pri riešení vyskytli, s cvičiacim. Predpokladá sa, že môžu nastať komplikácie s logickým teoretickým základom inferenčného algoritmu. Cvičiaci môže pomôcť s pravidlami použitými pri modifikácii KB.

Úloha spočíva v naprogramovaní agenta žijúceho vo svete – tzv. Wumpus World [1]. Nakoľko je spôsob implementácie a programovací jazyk voliteľný, súčasťou úlohy je taktiež naprogramovanie sveta.

Wumpus world predstavuje jaskyňu, v ktorej sa nachádzajú priepasti, zlato a príšera Wumpus. V jaskyni je tma, takže agent vníma svet iba prostredníctvom zápachu, ktorý vydáva wumpus, keď je na susednej pozícii a vánku, ktorý cíti keď je na susednej pozícii priepasť. Zlato agent vníma, iba keď sa nachádza na posícii, na ktorej stojí. Úlohou agenta je nájsť zlato a vyniesť ho von z jaskyne bez toho, aby spadol do priepasti, alebo by ho zabil wumpus. Agent má navyše možnosť vystreliť na niektoré miesto v jaskyni, pokiaľ si myslí, že je tam wumpus. Má však iba jeden výstrel.

Agent si ukladá informácie o prostredí do knowledge base – buď vo forme logických výrokov, alebo predikátov. V našom prípade budeme ukladať logické výroky. Z uložených výrokov si agent vyvodzuje ďalšie a na základe informácií, ktoré má v KB sa rozhoduje o svojom ďalšom kroku.

Riešiteľ by teda mal naprogramovať agenta takým spôsobom, aby fungoval vo svete s uspokojujúcou úspešnosťou – teda aby dopravil zlato von z jaskyne vo väčšine prípadov. Nemôžme požadovať 100% úspešnosť, pretože pri náhodne generovaných prostrediach môže ľahko vzniknúť situácia, keď agent nemôže jednoznačne vyvodiť dostatok potrebných informácií. Navyše môže nastať také rozloženie prekážok, že je cieľ nedosiahnuteľný.

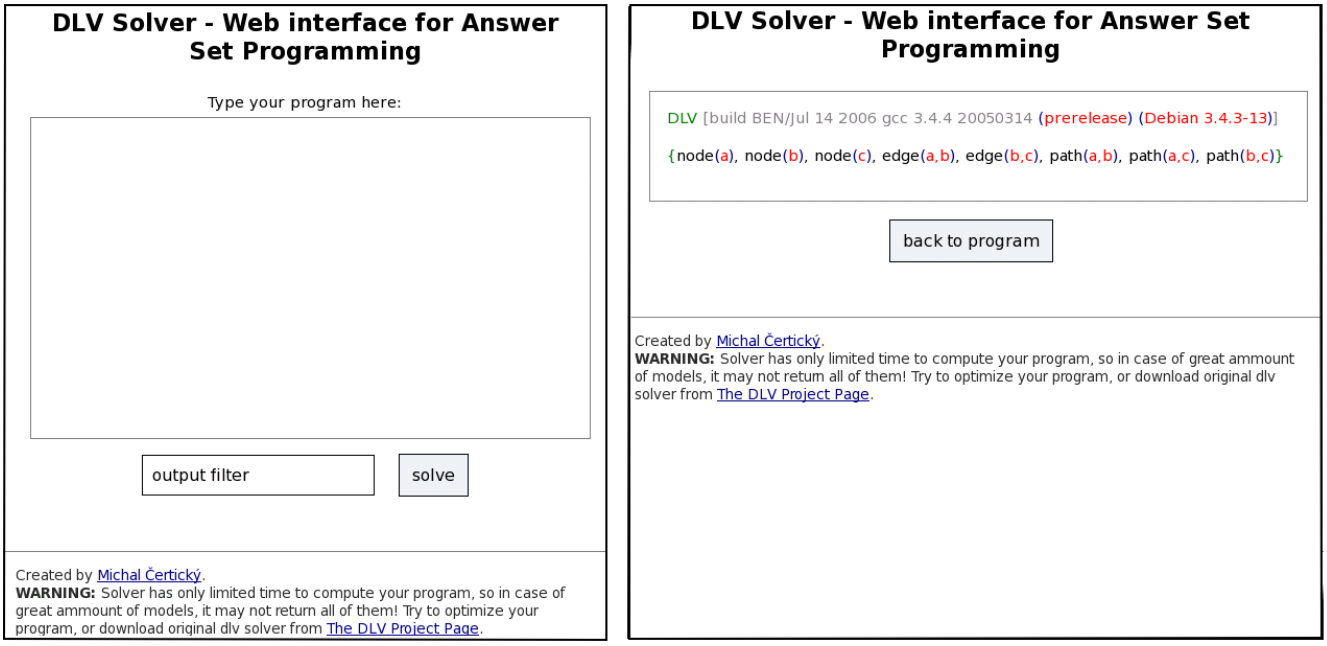
K úlohe prikladáme aj vzorové riešenie naproramované v Delphi. Agent z riešenia má uspokojujúce výsledky, pričom je implementovaný v súlade s nápovedou uvedenou pri zadaní úlohy. Svet zobrazuje jendoduché GUI, v ktorom si užívateľ sám spúšťa jednotlivé kroky. Zobrazený je okrem sveta aj stav Knowledge Base.

3.8. Cvičenie ZUI1 – 9 – First order logic – základy

Toto cvičenie má okrem vybudovania praktických základov predikátovej logiky za cieľ aj ukázanie, aké výhody má logické programovanie pri riešení niektorých problémov. V tomto prípade riešime constraint satisfaction problem z cvičenia 5 (kapitola 3.5.).

Úlohou je zasa vyriešenie problému zafarbenia mapy austrálie, no tentokrát nebude študent programovať imperatívne. Napísať má logický program (v štandardnom logickom jazyku), ktorý daný graf zafarbí. Na interpretáciu logického programu sa bude používať voľne stiahnuteľný DLV solver (link na stránku projektu DLV je zahrnutý v zadaní cvičenia).

Súčasťou cvičenia je aj špeciálne za týmto účelom vytvorený webovský interface DLV solvera. Umožňuje spúšťať logické programy bez inštalácie solvera, priamo cez internetový prehliadač (obr.12). K zadaniu úlohy je taktiež priložené vysvetlenie syntaxe a vzorový jednoduchý program.



obr.12. - webovský interface pre DLV solver pripravený pre zjednodušenie riešenia cvičenia

K tomuto cvičeniu je priložené taktiež vzorové riešenie vo forme krátkeho logického programu. Interpreter ako výstup tohoto programu vráti všetky možnosti zafarbenia teritórií tak, aby susedné teritóriá nemali rovnakú farbu. Výsledky sú popísané binárnym predikátom col(štát,farba).

3.9. Cvičenie ZUI1 – 10 – Knowledge representation

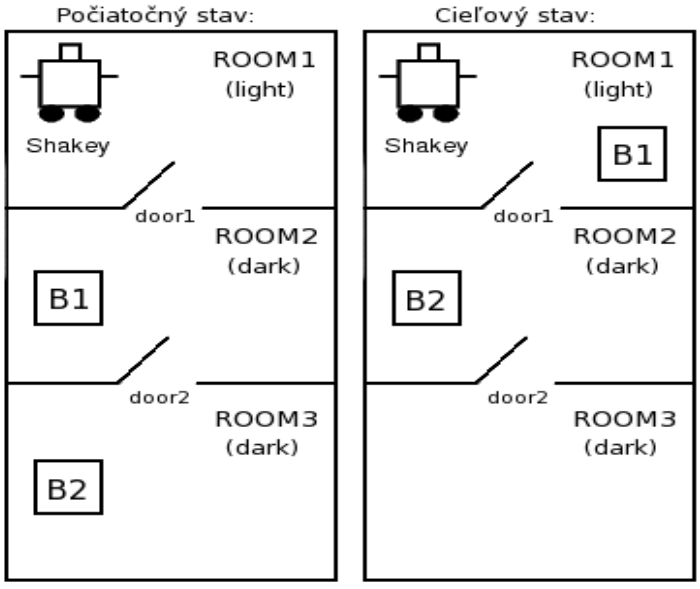
Desiate cvičenie pozostáva z dvoch úloh, ktoré spočívajú v popisovaní či už modelových situácií (wumpus world – úloha 1), alebo situácií z reálneho sveta (nakupovanie v supermarkete – úloha 2) predikátovou logikou. Úlohy sú teoretické a ich cieľom je z veľkej časti diskusia medzi študentami a cvičiacim.

3.10. Cvičenie ZUI1 – 11 – Shakey's world

V jedenástom cvičení sa opäť vraciame k agentovo orientovanému programovaniu. Shakey je agent, ktorý operuje v svete pozostávajúcom z miestností, vypínačov svetla, dverí a krabíc. Shakey sa po tomto svete vie pohybovať, prepínať vypínače a prenášať krabice.

V zadaní cvičenia máme predikátovou logikou popísaný Shakeyho svet a akcie, ktorých je schopný. Tento popis nie je nutné striktne dodržať – riešiteľ si môže predikáty mierne upraviť, keď to potrebuje.

Študent má za úlohu naprogramovať plánovací algoritmus pre shakeyho. Dané sú shakeyho akcie a počiatočný a cieľový stav (pozri obr.13.) popísané konjunkciou predikátov popisujúcich svet. Súčasťou riešenia je implementácia vzťahu medzi akciami a ich efektami na stav sveta. Pravidlá pre aplikovanie akcií sú popísané v zadaní cvičenia. Výstupom plánovacieho algoritmu má byť postupnosť akcií, ktoré vedú z počiatočného do cieľového stavu. Túto postupnosť stačí vypísať, nie je potrebné implementovať samotného agenta, ani s ním nič robiť.



obr.13. - počiatočný a cieľový stav Shakeyho sveta.

Agent má preniesť krabice do prílušných miestností.

Samozrejme nezáleží na výbere programovacieho jazyka. Priložené vzorové riešenie je programované v Jave. Pri tvorbe plánu používa dve metódy, ktorých výsledok následne porovná – prehľadávanie stavového priestoru do hĺbky a prehľadávanie do hĺbky so zvyšujúcou sa hĺbkou. V našom prípade je rýchlejšou metódou úplné prehľadávanie do hĺbky, ktoré vyskúša 1052 akcií, kým sa dostane k cieľovému stavu, zatiaľ ča iteratívne prehľadávanie vyskúša až 10525 akcií.

3.11. Cvičenie ZUI1 – 12 – Plánovanie v reálnom svete

Cvičenie číslo 12 je podobne ako cvičenie 10 (kapitola 3.9.) skôr teoretické. Cieľom je opäť diskusia medzi študentami a cvičiacim a spoločné hľadanie možných riešení. Pozostáva z troch úloh týkajúcich sa plánovania v reálnom svete. Riešenia týchto úloh majú teoretický charakter a nevyžadujú programovanie.

3.12. Cvičenie ZUI2 – 1 – Trading Agent Competition

V letnom semestri nie sú pravidelné každotýždenné cvičenia a hodnotenie štdentov prebieha na základe jedného celosemestrálneho projektu. Celkový počet bodov získaný za cvičenia v letnom semestri získavajú študenti za tento projekt.

Spomínaný projekt rieši jeden z problémov **TAC SCM**, alebo **TAC Classic**. Špecifikácia, softvér, dokumentácia a api sa nachádza na webe www.sics.se/tac. Projekt je možné vypracovať aj vo dvojiciach. Ide o vypracovanie agenta, ktorý sa napája na vzdialený server a zúčastní sa multiagentovej súťaže simulujúcej konkurenčné prostredie v podnikaní s hardvérom. Agent musí rozumným spôsobom počas trvania súťaže nakupovať komponenty od výrobcov, vyrábať počítačové zostavy a včas plniť objednávky zákazníkov. Cieľom je prirodzene maximalizácia zisku.

Vzorové riešenie používa 2 pomocné triedy:

Trieda Deal reprezentuje jednu transakciu od prijatia objednávky od zákazníka až po odoslanie hotových zostáv. Každá takáto transakcia obsahuje (okrem iného) príznak status, ktorý hovorí, v ktorom štádiu sa Deal nachádza. Inštancie triedy Deal sú skladované v prioritnom rade podľa dátumu dodania a podľa priority sa riešia.

Trieda DesiredComponents je vlastne pole všetkých druhov komponentov, v ktorom si pamätáme, koľko komponentov potrebujeme objednať a koľko sme ich objednali. Navyše si v tejto triede pri komponentoch pamätáme aj to, do akého dátumu ich majú dodávatelia poslať atď.

Agent si pomocou štruktúry DesiredComponents objednáva súčiastky. Z prebytočných súčiastok na sklade sa priamo podľa objednávok produkujú dalšie počítače.

Ceny počítame na základe podielu agenta na trhu. Ak máme dostatočný podiel, cenu zvyšujeme a naopak.

4. Záver

Na záver mojej práce si dovolím konštatovať, že sa mi podarilo splniť ciele, ktoré som si v úvode vytýčil. Samozrejme to, nakoľko bude moja práca užitočná pri výučbe predmetu ukáže až čas, no myslím si, že má potenciál zjednodušiť výučbu a priebeh cvičení, ako aj zvýšiť prístupnosť študijných materiálov.

Okrem cieľov, ktoré som si definoval v úvode práce, som počas tvorby webu prišiel s ďalšími nápadmi a myšlienkami a pokúsil som sa ich zrealizovať. Konkrétne som sa rozhodol zaistiť prístupnosť webu zdravotne postihnutým, či užívateľom s neštandardným technickým vybavením (kapitola 2.5. - Prístupnosť a použiteľnosť), a optimalizovať ho pre webovské vyhľadávacie služby (kapitola 2.7. - SEO). Nevyhnutnou súčasťou tejto optimalizácie však je aj čakanie, kým vyhľadávače zaradia web do svojej databázy, takže nemôžem hodnotiť nakoľko sa mi podarilo tieto ciele splniť skôr, ako sa web umiestni na server.

Miernou tematickou odbočkou od zadania práce bolo vysvetlenie fungovania modelu PHP aplikácie, ktorú som použil (kapitola 2.3 – Stromový model dedičnosti). Nakoľko ide o môj originálny model, musel som ho v tomto texte odprezentovať, pretože bez neho by bola špecifikácia webu ako PHP aplikácie neúplná.

Rozhodnutie o použití technológií PHP a MySQL, ktoré som učinil už v úvode práce hodnotím po ukončení ako správne. Možnosti, ktoré mi poskytli som úspešne využil a umožnil administrátorom jednoduché a rýchle modifikovanie obsahu webu, ako aj kontrolu odovzdaných riešení.

Najdôležitejšiu zložku obsahovej časti webu tvoria cvičenia. Zadania úloh som volil tak, aby som dosiahol vyššiu atraktívnosť úloh, no zachoval ich prínos pre vedomosti a zručnosti študentov. Dbal som aj na to, aby boli úlohy splniteľné pre študentov 2., resp. 3. ročníka FMFI. To bolo dôvodom zvolenia programovacích jazykov Java a Delphi pre implementáciu zahrnutých programov, nakoľko obidva jazyky sú vyučované na povinných predmetoch v prvom a druhom ročníku.

Čo sa týka snahy o zefektívnenie priebehu cvičení, ide o ďalšiu vec, ktorú jasne ukáže až čas. Už teraz ale môžeme pozitívne hodnotiť zrýchlenie bežných činností, ktoré cvičiaci často vykonával. Napríklad sprístupnenie zadania cvičenia, či vzorového príkladu, resp. stiahnutie si odovzdaných riešení teraz cvičiacemu zaberú menej ako 20 sekúnd. Tak isto, ak bude efektívna SEO-optimalizácia webu, tak sa zrýchli čas prístupu študentom, ktorí si nepamätajú link. Keďže ide o často opakované činnosti, každá ušetrená sekunda je cenná.

Nakoľko som pri písaní tohoto textu vychádzal prevažne z vlastných bohatých skúseností s webdesignom, prípadne z absolvovaných prednášok ZUI, nečerpal som veľa informácií z konkrétnych externých zdrojov. Zoznam použitej literatúry preto nie je práve obsažný.

Literatúra:

[1] Russel, Norwig : Artificial Intelligence – A Modern Approach

[2] Ing. Radim Smička : Optimalizace pro vyhledávače - SEO

[3] online dokumentácia jazyka PHP na stránkach http://php.net/

[4] online príručka HTML a CSS na stránkach http://www.jakpsatweb.cz/