FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
UNIVERZITY KOMENSKÉHO
BRATISLAVA

Web formulárový framework

Diplomová práca

Marcel Sýkora 2009

Web formulárový framework

DIPLOMOVÁ PRÁCA

Marcel Sýkora

UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

KATEDRA INFORMATIKY

Informatika

Školiteľ záverečnej práce
Ing. Maroš Ivančo

BRATISLAVA 2009

Zadanie diplomovej práce

Cieľom našej práce je preskúmať a analyzovať situáciu, možnosti a prístupy v oblasti tvorby webových formulárov. Z analýzy vyplynie špecifikácia takzvaných ideálnych vlastností formulárového frameworku. Súčasťou práce má byť aj špecifikácia, návrh a implementácia formulárového frameworku, na základe kompromisu niektorých už spomínaných ideálnych vlastností a odôvodnením ich výberu. Východiskom frameworku bude existujúce riešenie dforms.

Čestne vyhlasujem, že som diplomovú prácu vypracoval samostatne, s použitím literatúry a zdrojov uvedených v závere práce.

Bratislava, máj 2009 Marcel Sýkora

Abstrakt

Web svojím stále rozširujúcim sa záberom zasahuje takmer všetkých oblasti každodenného života. Z ďaleka už nie je takou statickou prezentáciou, akou býval. Dynamickosť webu predstavujú najmä rôznorodé webové aplikácie. Našimi opratami k nim sú užívateľské rozhrania – webové formuláre. Táto práca sa zaoberá analýzou samotných webových formulárov ako aj ich tvorbou. Tvorbu predstavujú najmä web formulárové frameworky. V práci prinášame formálnejší pohľad na formulárové frameworky a ich zaujímavé vlastnosti. Za prínos považujeme nevšedný formálnejší pohľad na webový formulár a web formulárový framework. Súčasťou práce je aj implementácia web formulárového frameworku. Vychádza z existujúceho riešenia, ktoré rozširuje o implementáciu vlastností čitateľnosti kódu a prístupnosti používateľom s obmedzeniami. Implementácia ukazuje prístup akým by sa mohol pôvodný framework uberať, čo je zároveň ďalším z prínosov práce.

Kľúčové slová: framework, web formulár, web front-end, webová aplikácia

Obsah

Zoznam obrázkov a tabuliek

Zoznam obrázkov

Zoznam tabuliek

Zoznam skratiek

AJAX (Asynchronous JavaScript and XML) - Je všeobecné označenie pre technológie vývoja interaktívnych webových aplikácii, ktoré menia obsah svojich stránok bez nutnosti ich znovunačítania.

CSS (Cascading Style Sheets) – Štýly rozširujúce možnosti zobrazenia značiek v rámci HTML dokumentu. Umožňujú podrobne definovať farby, pozície, formát, či zarovnanie prezentácie HTML dokumentu.

DOM (Document Object Model) - Spôsob prístupu k XML alebo HTML dokumentu, každá entita v dokumente je reprezentovaná jedným uzlom v stromovej štruktúre.

GUI (Grafic User Interface) – grafické používateľské rozhranie

HTML (HyperText Markup Language) - Značkovací jazyk pre hypertext.

HTTP – (Hypertext Transfer Protocol) - internetový protokol určený pôvodne pre výmenu hypertextových dokumentov vo formáte HTML.

Java - objektovo orientovaný programovací jazyk, ktorý vyvinula firma Sun Microsystems.

JSP (java server pages) – Technológia pre tvorbu interaktívnych webových stránok využívajúca na serverovej strane platformu java. Pred jej prvým použitím ju web server prekladá na servlet.

JVM (Java Virtual Machine) - je množina počítačových programov a dátových štruktúr, ktorá využíva modul virtuálny stroj k spusteniu ďalších počítačových programov a skriptov vytvorených v jazyku Java

MVC (Model-View-Controler) - Architektonický návrhový vzor oddeľujúci dáta, prezentáciu a biznis logiku.

SVG (Scalable Vector Graphics) - Jazyk pre dvojdimenzionálnu grafiku vo formáte XML.

W3C (World Wide Web Consortium) - Konzorcium, ktoré vyvíja spolupracujúce technológie, špecifikácie, príručky, softvér a nástroje, ktoré vedú k zlepšovaniu internetu.

XML (eXtensible Markup Language) - veľmi jednoduchý a flexibilný značkovací jazyk.

XML Schema - Jazyk vyvíjaný W3C konzorciom na popis štruktúry XML dokumentov.

XSD (XML Schema Definition) - inštancia XML Schemy.

XPath (XML Path Language) - XML jazyk obsahujúci výrazy, pomocou ktorých sa dajú adresovať časti XML dokumentu.

XSL (eXtensible StyleSheet Language) - Rodina jazykov, ktoré určujú ako majú byť dokumenty XML formátované alebo transformované. Patria do nej jazyky XSLT, XSL-FO,XPath.

XSLT (XSL Transformation) - XML jazyk pre transformácie XML dokumentov.

Úvod

Za posledných pätnásť rokov internet a web prešli okolo mnohých míľnikov až napokon predstavujú fenomén dneška. Ušli od relatívne úzkeho kruhu vedcov až k širokým masám. Priniesli obrovské množstvo technológii, štandardov a prístupov. Niektoré sa používajú dodnes, ale mnohé z nich zapadli prachom na smetisku internetových dejín. V každom prípade web a internet zasahujú do takmer každej stránky bežného života, či už prostredníctvom poskytovania informácii, služieb alebo komunikačných kanálov a podobne.

Tvár a forma webu prešli tiež mnohými zmenami od jednoduchých statických stránok cez dynamické portály až k dnešným web aplikáciám, ktoré môžeme pokojne porovnávať s klientskymi. Web aplikácie sú založené na rôznych technológiách, štandardoch, architektúrach a prístupoch. Čo však majú spoločné sú používatelia, bez nich by stratili význam. Používateľské rozhranie zabezpečuje prepojenie medzi používateľmi a aplikáciami. Vo svete webových aplikácii ho nazveme webovým formulárom.

Webovým formulárom niekedy nebýva venovaná dostatočná pozornosť a vážnosť. Ale nešťastne navrhnutý formulár môže odradiť používateľa a niekto tým môže stratiť potenciálneho zákazníka. Mnohokrát je toto malé veľké nešťastie spôsobené, nedostatočnou pozornosťou ergonómii a detailom pri vývoji webového formulára, ktorá bola spôsobená prílišným zameraním sa na technickú stránku.

Preto sa v tejto práci venujeme webovým formulárom ako takým, pozeráme sa na ne zblízka, snažíme sa preniknúť hlbšie k ich podstate, dosiahnuť istú formu abstrakcie. Tiež sa zameriavame na oblasť tvorby webových formulárov, rôznym prístupom, technológiám, štandardom. Tieto sú základom pre frameworky, ktoré tvorbu webových formulárov v mnohých prípadoch uľahčujú a obohacujú, čo dáva priestor k zlepšeniu ergonómie a detailov samotných formulárov.

Diplomová práca pozostáva z troch kapitol a jednej cd prílohy. Stručný obsah jednotlivých kapitol popisujú nasledovné odstavce.

Prvá kapitola venovaná rozboru webových formulárov, ich črtám a súčastiam a tiež niektorým technológiám a štandardom, ktoré za nimi stoja.

Druhá kapitola je zameraná na web formulárové frameworky, rozbor ich zaujímavých vlastností, rozdelenie na koncepty a tiež niektorým konkrétnym frameworkom.

V tretej kapitole sa venujeme web formulárovémub frameworku dforms, ktorý vychádza z reálneho riešenia DForms.

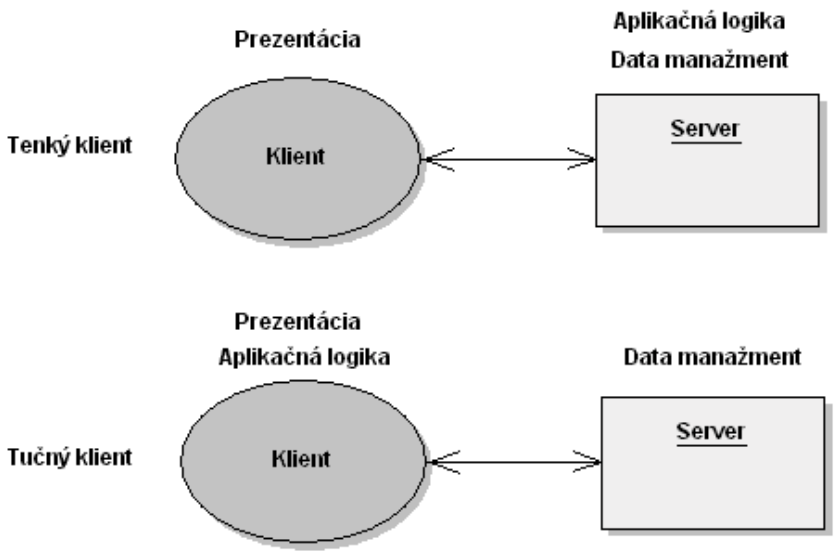
Príloha obsahuje implementáciu frameworku dforms popísaného v tretej kapitole a niekoľko príkladov jeho použitia.

1. Úvod do webových formulárov

Devätnáste storočie bolo nazývané aj „storočím pary“. Dobu v ktorej práve žijeme, môžeme smelo nazvať „storočím formulárov“. V bežnom živote sa s nimi stretávame na každom kroku: rodný list, zápis do základnej školy, vysvedčenie, rôzne úradne žiadosti od žiadosti o vydanie občianskeho preukazu až po daňové priznanie. Skrátka formuláre nás sprevádzajú celým životom.

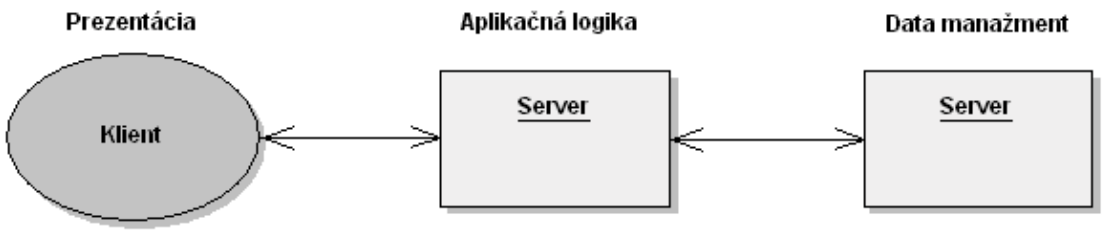
Formulár je formátovaný dokument obsahujúci prázdne polia, ktoré sú určené na vyplnenie dátami. Prínosom formulára je nutnosť vyplniť len prázdne polia, pričom ostatné informácie sú jeho fixnou súčasťou, čo zabezpečuje striktnosť pri jeho vypĺňaní a uniformitu pri spracovaní dát, ktoré pomocou neho získavame.

Niektoré rozhrania používateľských vstupov pokročilých grafických softvérových aplikácií a systémov sú inšpirované tiež papierovými formulármi. Tieto rozhrania môžeme nazvať aj aplikačnými formulármi. Ich rozvoj je úzko spätý s rozvojom grafických používateľských rozhraní, ktoré boli najskôr príznačné pre doménu klientských aplikácií. Softvérový prístup k formulárom rozširuje možnosti formulárov ako takých. Môže priniesť početnejšiu paletu variabilných prvkov formuláru, ktoré zabezpečujú striktnosť vypĺňaných dát. Prvky aplikačného formulára poskytujú používateľovi väčší komfort prostredníctvom ich špecifickej definície alebo tiež možnosťou opakovanej modifikácie, ktorá je príznačná pre softvérové riešenia.



Obrázok 1.1: 2-vrstvová architektúra

Príchod a rozvoj celosvetovej siete Internet priniesol so sebou niekoľko architektúr pri vývoji aplikácií. Nás bude zaujímať najmä architektúra klient-server, pri ktorej je aplikácia modelovaná ako množina služieb poskytovaných servermi klientom. Klient resp. klientska aplikácia je aktívna, posiela žiadosti na služby, čaká a dostáva odpovede od serveru. Server je pasívny, počúva žiadosti autorizovaných klientov a obsluhuje ich. Logická štruktúra aplikácie odráža rozdelenie úloh do viacerých vrstiev – prezentačná, aplikačná a dátová vrstva. Tieto vrstvy sú zase rozdelené medzi klientov a serveri, čo prináša v zásade ďalšie dve architektúry a to 2-vrstvovú a 3-vrstvovú. 2-vrstvová ponúka dva modely. Jeden s tenkým klientom, kde je na klientovi len prezentačná vrstva, aplikačná a dátová sú na serveri. Pri druhom modeli hovoríme o tučnom klientovi, lebo je na ňom sústredená prezentačná aj aplikačná vrstva, pričom na serveri je len dátová. Pri 3-vrstvovej architektúre ostáva prezentačná vrstva na klientovi a aplikačná aj dátová sú priradene každá rôznemu serveru.



Obrázok 1.2: 3-vrstvová architektúra

Aplikačný protokol HTTP Webu je tiež založený na architektúre klient-server. *Webová aplikácia* je aplikácia poskytovaná užívateľom z webového serveru cez sieť Internet, alebo jej vnútropodnikovú obdobu - intranet. V skorších typoch sieťových aplikácií typu klient-server mala každá aplikácia svoj vlastný klientský program, ktorý slúžil ako jej užívateľské rozhranie a musel byť inštalovaný na osobnom počítači každého užívateľa. Aktualizácia serverovej časti typicky vyžadovala aj aktualizáciu klientských programov na každej pracovnej stanici, čo zvyšovalo náklady na podporu a znižovalo efektívnosť zamestnancov.
Webová aplikácia využíva bežne rozšírený webový prehliadač ako klienta. Ten je potom v úlohe tenkého klienta, nakoľko sám o sebe nepozná logiku aplikácie a teda pri jej zmene nie je nutné ho aktualizovať. Koncept webových aplikácií sa stal v niektorých doménach populárnym natoľko, že sa začal uplatňovať pri vývoji aplikácií, ktoré boli dovtedy realizované klientskými aplikáciami. Rozvoj webových aplikácií priniesol nutnosť prispôsobiť aplikačné formuláre webovému prostrediu.

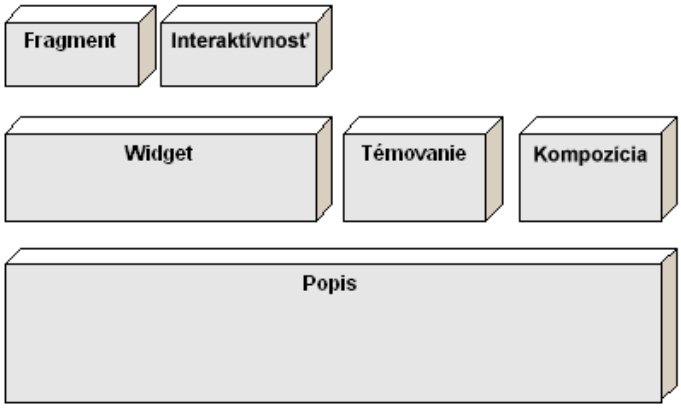
Webový formulár je rozhranie medzi užívateľom a serverom a teda aj webovou aplikáciou. Rozdiel medzi webovým a aplikačným formulárom spočíva v spôsobe a odozve spracovania dát. Ďalší rozdiel predstavuje užšia paleta webového formulára. Webové formuláre neboli súčasťou webu hneď od jeho vzniku, boli pridané do značkovacieho jazyka HTML verzie 3.2 v roku 1992 ako reakcia na rastúci rozvoj Internetu a tlak tvorcov webových prehliadačov, na ktorých tlačil rozvoj obchodovania prostredníctvom Internetu. V tom čase boli spomínané rozdiely výraznejšie ako v súčasnosti, nakoľko ich potierajú nové technológie a prístupy v oblasti webu.

Z pohľadu užívateľa je webový formulár tvorený jedným až niekoľkými ovládacími prvkami. Tieto prvky majú rôznu povahu. Vo väčšine prípadov sú jednoduché a užívateľovi známe z aplikačných formulárov. Ďalšie menej obvyklé ovládacie prvky sú použité pri špecifických požiadavkách na webový formulár, vznikajú napríklad kombináciou známych ovládacích prvkov, ale aj ako úplne nové ovládacie prvky, pripadne špecifické mini aplikácie. Ovládacie prvky budeme označovať ako *widgety*.

Webový formulár tvorí niekoľko aspektov, z ktorých sa skladá alebo ho vytvárajú. Tieto aspekty pri konkrétnom riešení zastupujú rôzne technológie a štandardy. Spôsob akým sú vybrané štandardy a technológie použité môžu zastrešovať frameworky.

1.1. Aspekty webového formulára

Aspekty webového formulára predstavujú abstraktný model akým sa dá pozerať na zloženie webového formulára, sú to časti, ktoré vytvárajú jeho prezentačný a funkčný charakter.



Obrázok 1.3: Aspekty webového formulára

1.1.1. Fragment

Fragment môže byť čiastočným nosičom prezentácie informácií alebo funkcionality. Ak túto jednotku osamostatníme, tak v kontexte webového formulára zvyčajne nepredstavuje ucelený prezentačný alebo funkčný koncept.

1.1.2. Widget

Prvok webového formulára, ktorý poskytuje ucelený prezentačný alebo funkčný koncept. Widget predstavuje atomickú jednotku, pri budovaní webového formulára. Vzniká spojením interaktívnosti a niekoľkých fragmentov.

1.1.3. Kompozícia

Kompozícia predstavuje usporiadanie, rozloženie prvkov formulára – widgetov a fragmentov. Nový widget môže vzniknúť kompozíciou existujúcich widgetov.

1.1.4. Interaktívnosť

Interaktívnosť predstavuje správanie, ktoré je priradené jednému alebo skupine fragmentov, widgetu alebo kompozícii. Interaktívnosť priraďuje správanie na základe udalostí naviazaných na fragmenty, widgety a kompompozície, výsledkom je prezentácia, modifikácia alebo validácia dát. Technicky je implementovaná skriptovacím jazykom v závislosti od použitej technológie pri implementácii webového formulára.

1.1.5. Témovanie

Témovanie zodpovedá za vzhľad prvkov formulára. Konkrétne črty vzhľadu môžu byť priradené jednotlivým fragmentom, widgetom, kompozíciám alebo celému webovému

formuláru. Pri použití niektorých technológii a prístupov môže témovanie ovplyvňovať nielen estetickú ale čiastočne aj funkčnú stránku webového formulára.

.

1.1.6. Popis

Popis predstavuje definíciu webového formulára prostredníctvom definovania ostatných aspektov.

Model webového formulára sa premieta do dvoch rovín, ktoré sú implikované klient-server architektúrou webu.

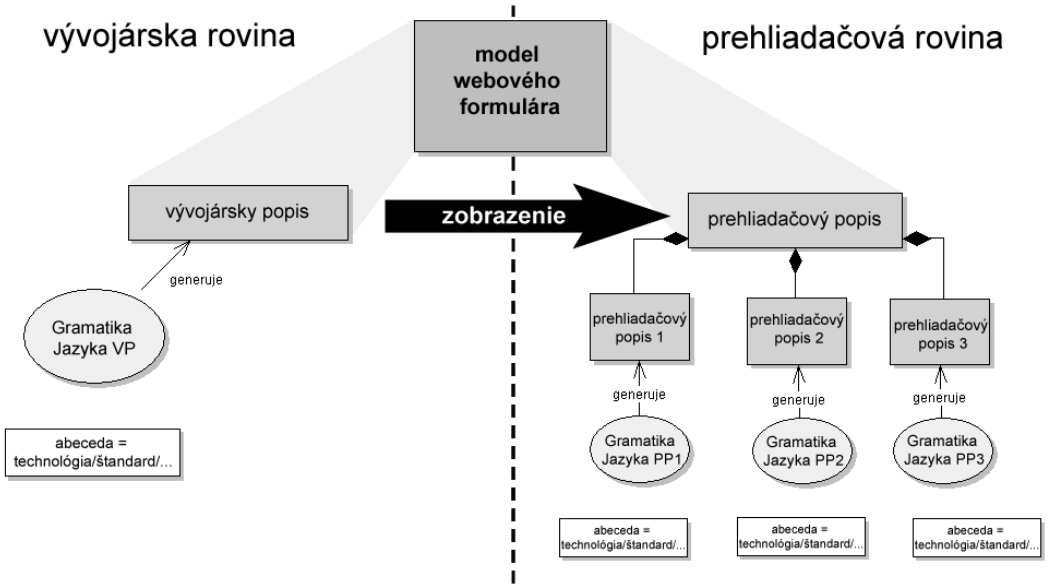
Prvá sa odvíja od servera poskytujúceho webové stránky, ktorých súčasťou môže byť webová aplikácia, ktorá obsahuje webový formulár. Tieto na server museli byť niekým nasadené a ešte pred tým vyvinuté, dizajnované. Tu sa dostávame k vývojárovi a teda prvú rovinu nazveme vývojarska rovina. Obraz priemetu modelu webového formulára do nej predstavuje popis, ktorý nazveme vývojársky popis.

Druhá rovina súvisí s klientom, ktorý je pri webe predstavuje webový prehliadač, ktorý sprostredkováva používateľom web stránky obsahujúce webové aplikácie a s nimi tiež webové formuláre. Preto druhú rovinu budeme nazývať rovinou prehliadača a popis modelu v nej prehliadačovým popisom.

Webový formulár je teda pri svojom vzniku vo vývojárskej rovine a po skončení implementácie sa nasadením alebo až prezentovaním v prehliadači dostáva do roviny prehliadačovej. Pričom aspekty webového formulára sú vo väčšine prípadov v jednotlivých rovinách zastúpené rozdielnymi technológiami, štandardmi alebo jazykmi. To znamená najmä v prehliadačovej rovine, že popis je vlastne kompozitným popisom rozdeleným pre jednotlivé aspekty a technológie, ktoré ich zastupujú.

Pozrime sa na popis v ľubovoľnej rovine cez optiku formálnych jazykov[5], tak konkrétny popis určitého webového formulára predstavuje *slovo* patriace jazyku všetkých formulárov, ktoré generuje určitá *gramatika*. *Abeceda*, nad ktorou sú slová generované danou *gramatikou*, je zastúpená spravidla programovacím alebo značkovacím jazykom. Jednotlivé aspekty formulára si môžeme predstaviť ako *terminálne symboly*. *Pravidlá* tejto *gramatiky* technologicky definuje určitý framework, napríklad prostredníctvom XML Schémy alebo knižnice v programovacom jazyku.

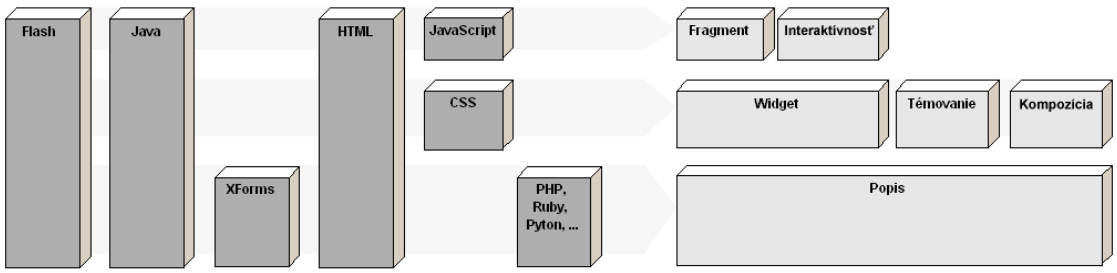
Framework okrem *gramatiky* tiež definuje *zobrazenie* množiny vývojárskych popisov do množiny prehliadačových popisov. Podrobnejšie sa frameworkom budeme venovať v 2. kapitole.



Obrázok 1.4: Vývojárska a prehliadačova rovina

1.2. Technológie

V tejto podkapitole sa budeme venovať niektorým technológiám a spôsobu akým sa spolupodieľajú pri popise aspektov webových formulárov.



Obrázok 1.5: Vzťah aspektov webového formulára a technológii, ktoré ich implementujú

1.2.1. JavaScript

JavaScriptu je skriptovací jazyk, ktorého kód beží lokálne v majoritných prehliadačoch. Je teda vhodný na implementáciu interakcií na strane klienta. V prípade nášho modelu aspektov webového formulára to znamená, že môže implementovať interaktívnosť v prehliadačovej rovine. Pri mnohých riešeniach tomu tak je, napriek tomu, že existujú menšie odchýlky v jeho interpretácii v jednotlivých prehliadačoch a väčšie odchýlky spôsobené rôznou implementáciou DOM, pomocou, ktorého pristupuje k elementom.

1.2.2. CSS

CSS je štýlovací jazyk, ktorým zvykneme popisovať prezentáciu značkovacieho jazyka. Je navrhnutý, tak aby primárne umožnil oddeliť obsah od prezentácie. Je teda vhodný pri témovaní prvkov webového formulára za predpokladu, že tie sú implementované niektorým značkovacím jazykom.

1.2.3. HTML

HTML je hypertextový značkovací jazyk, v ktorom vieme definovať a zároveň implementovať webové formuláre so základnými widgetmi. Znamená to, že HTML

pokrýva všetky aspekty webového formulára, dokonca v oboch rovinách súčasne. Konkrétnejšie napríklad fragment, widget a funkcionalita sú zastúpené v HTML elementmi pre formuláre. Témovanie a kompozíciu tiež vieme dosiahnuť elementmi HTML alebo vhodným nastavením ich atribútov.

1.2.4. Java

Java môže pokrývať rôzne aspekty webového formulára. Ale v zásade má dva prístupy.

V prvom prístupe je Java vývojárskym popisom webového formulára prostredníctvom určitej objektovej štruktúry na strane serveru. Pričom smerom ku klientovi väčšinou dochádza k transformácii na technológie bližšie prehliadaču ako napríklad HTML, JavaScript a iné.

Pri druhej možnosti je Java vývojárskym aj prehliadačovým popisom a implementuje všetky aspekty webového formulára prostredníctvom Java appletu cez jednotlivé balíčky a rozhrania jazyka Java. Java applet je distribuovaný prehliadaču vo forme byte kódu ten ho interpretuje ak má k dispozícii JVM. Java platforma zabezpečuje kompatibilitu medzi operačnými systémami.

1.2.5. PHP, Ruby, Python, ...

Objektovo orientované jazyky, ktoré sú pri vývoji webových aplikácii a formulárov využívané ako skriptovacie jazyky, podobne ako Java v jednom z jej prístupov k webovým formulárom pokrývajú len vývojársky popis, ktorým modelujú webový formulár. Pričom v objektovej štruktúre formulára môže byt zachytená aj kompozícia a témovanie. Pri transformácii vývojárskeho popisu na prehliadačový je objektová štruktúra transformovaná do štandardov, technológií, ktoré sú prirodzené pre prehliadač. V mnohých prípadoch tejto transformácii napomáhajú šablóny, ktoré predstavujú parciálny prehliadačový popis niektorých aspektov.

1.2.6. XForms

XForms je odporúčanie špecifikované W3C. Je to XML formát predstavujúci novú generáciu značkovacieho jazyka pre webové formuláre. Je navrhnutý tak, aby oddeľoval prezentáciu od obsahu, znížil potrebu skriptovania, aby umožňoval znovupoužiteľnosť, rozšíriteľnosť, redukoval volania medzi klientom a serverom. Predstavuje posun k deklaratívnemu programovaniu, ktoré umožňuje jednoduchšie zvládnutie, údržbu a tiež ladenie. Je kompatibilný s ostanými XML štandardami ako CSS, XML Schéma, XPath. Nie je samostatným typom dokumentu, ale je určený na integráciu do značkovacích jazykov ako XHTML alebo SVG.

Slabinou XForms je zatiaľ slabá priama podpora majoritnými prehliadačmi. Existuje viacero prístupov ako je implementovaný a to buď inštaláciou programových rozšírení do prehliadačov. Alebo jeho transformáciou na strane servera alebo na strane klienta do formátov a štandardov, ktoré dnešné prehliadače priamo podporujú. Z tejto slabiny vyplýva, že XForms pri súčasnej podpore prehliadačov pokrýva len vývojársky popis aspektov webového formulára.

1.2.7. Flash

Technológia Flash je založená na troch kľúčových prvkoch. Prvým je Flash animácia, je zároveň zdrojovým kódom a nosičom prezentácie, užívateľského rozhrania a interakcie, ktorú technológia Flash poskytuje. Druhým je Flash prehrávač, ktorý môže byť ako samostatná aplikácia na klientovi alebo ako programové rozšírenie webového prehliadača. Umožňuje používateľovi spúšťať Flash animácie. Tretím prvkom je vývojové prostredie pre tvorbu Flash animácií. Prvky užívateľského rozhrania – widgety, grafické prvky a ich rozloženie sa pomocou neho modelujú vizuálne. Funkcionalita je zastúpená objektovo orientovaným programovacím jazykom ActionScript, ktorý je súčasťou Flash technológie. Výsledná Flash animácia môže implementovať webový formulár a prestavuje jeho popis. Flash je teda pre webový formulár vývojárskym aj prehliadačovým popisom.

Flash technológia pri svojom vzniku slúžila najmä na prezentáciu rôznych animácii. Jej presadeniu pomohlo, že na rozdiel od ostatných animačných formátov disponovala širšími možnosťami pri nižších kapacitných nárokoch, čo bolo dosiahnuté vektorovým prístupom k animácii a grafickým prvkom. Ďalšou výhodou Flash technológie je zachovanie identickej prezentácie Flash animácie v rôznych prehliadačoch a platformách, čo zabezpečuje jej interpretácia Flash prehrávačom.

Kritikou Flash technológie pri jej skorších verziách bolo narušenie konvencií zaužívaných pri normálnych HTML stránkach. Napríklad odlišné správanie sa pri označovaní textu, pri práci s widgetmi a pri iných interakciách. Ďalšou kritikou bolo, že Flash predstavoval bariéru pre používateľov s obmedzeniami. Tá bola spôsobená nemožnosťou zväčšenia textu, zvýšenia kontrastu, neprítomnosťou alternatívnych popisov, nedostatočnou podporou interakcie pri exkluzívnom používaní klávesnice a podobne. Niektoré z týchto nevýhod boli odstránené, iné pretrvávajú dodnes, prípadne neboli odstránene na všetkých platformách.

Inú vlnu kritiky predstavujú výhrady voči niektorým uzavretým konceptom, ktoré Flash technológia prináša, ako napríklad výsledná Flash animácia neumožňuje prístup k jej zdrojovému kódu. Inou kritikou je výkonnosť Flashu, tá nie je rovnako optimalizovaná na rôznych platformách. S výkonom súvisí aj neúmerné čerpanie systémových prostriedkov pri niektorých druhoch aplikácií.

2. Formulárové frameworky

Podľa[1] framework je abstraktným modelom konkrétnej domény alebo dôležitého aspektu v nej. Framework môže modelovať ľubovoľnú doménu, či už je to technická doména, koncept ako komunikácia, používateľské rozhranie, kompilátor alebo aplikačná doména ako bankovníctvo, poisťovníctvo. Framework poskytuje používateľovi opakovane použiteľný dizajn a opakovane použiteľné implementácie.

Framework je charakteristický obráteným riadením (inversion of control). Znamená to, že framework volá špecifický kód aplikácie, tým sa odlišuje od knižnice, ktorá je volaná z kódu špecifického pre aplikáciu.

Pri objektovo-orientovaných jazykoch klasifikujeme frameworky podľa metód rozširovania funkčnosti na *white-box*, *black-box* a *gray-box.*

White-box frameworky poskytujú triedy, ktoré sú dedené a môžu byť predefinované triedami konkrétnej aplikácie a teda prispôsobené jej potrebám a špecifikám. Výsledná aplikácia, systém je tesnejšie zviazaný s detailmi implementácie daného white-box frameworku.

Black-box frameworky poskytujú triedy na inštacovanie a použite tak ako sú bez ďalšej zmeny. Majú definované rozhrania ktorými sa môžu pripojiť špecifické komponenty samotnej aplikácie, pomocou kompozície objektov.

Gray-box frameworky vznikajú kombináciou white-box a black-box. Predstavujú väčšinu reálnych frameworkov. [2]

Aplikačné frameworky pozostávajú z „pripravených-na-použite“ a „polo-dokončených“ stavebných blokov. Celková architektúra, kompozícia a interakcie stavebných blokov, sú tiež preddefinované. Časti aplikačného frameworku, ktoré túto architektúru definujú ostávajú zvyčajne nemenné a bývajú označované ako zmrznuté miesta (*frozen spots)*.

Produkovanie špecifickej aplikácie, systému obyčajne znamená prispôsobiť stavebné bloky špecifickým potrebám. Vo všeobecnosti aplikačný framework štandardizuje, zovšeobecňuje aplikácie pre špecifickú doménu. Ale zároveň niektoré aspekty nie sú predvídateľné. A teda im zodpovedajúce časti aplikačného frameworku musia zostať generické, aby sa mohli jednoducho prispôsobiť špecifickým potrebám. Tieto časti sú takzvanými horúcimi miestami (*hot spots*). Framework, ktorý poskytuje adekvátne adaptácie horúcich miest, považujeme z hľadiska kvality za dobre navrhnutý. [2]

Ďalší pohľad na zloženie aplikačného frameworku podľa[1] hovorí, že aplikačný framework je framework, ktorý viaže spolu množinu existujúcich frameworkov, aby pokryl väčšinu aspektov určitej domény. Presnejšie povedané, aplikačný framework je len ďalším frameworkom podobnej veľkosti ako sú frameworky ktoré používa. Teda na aplikačný framework sa môžme pozerať aj ako na niečo ako kompozitný framework.
Aplikácie, systémy, ktoré vzniknú softvérovým vývojom založeným na aplikačných frameworkoch, sa stavajú rozšíreniami aplikačného frameworku.

Použitie frameworkov vo všeobecnosti prináša výhody aj nevýhody.

Hlavným prínosom je uplatnenie znovupoužiteľnosti, čo prináša zníženie nákladov, skrátenie času dodania a zvýšenie kvality výsledného produktu. Pri frameworkoch dochádza k znovupoužitiu dizajnu aj kódu, často bez nutnosti použitia nových technológií, nakoľko je framework založený na technológiách známych. Znovupoužitie kódu, ktorý bol už napísaný, testovaný a používaný, zvyšuje spoľahlivosť a znižuje

celkový čas vývoja. Ako sme spomínali framework samotný môže rozdeľovať aplikáciu na relatívne samostatné časti, ktoré tiež môžu byť frameworkami. Rozdelenie umožňuje distribúciu úloh v rámci vývojového tímu a teda zefektívnenie jeho činnosti. Frameworky prenášajú určité pozitívne vlastnosti na výslednú aplikáciu priamo, tým že ich implementujú ako napríklad bezpečnosť. Alebo nepriamo tak, že spôsob, akým sú postavené, respektíve akým s nimi vývojár pracuje, si vynucuje dodržiavanie doménovo špecifických alebo všeobecných najlepších praktik.

Nevýhod použitia frameworkov je viacero, pričom v niektorých prípadoch sa skôr jedná o riziká, ktoré prináša ich použitie. Znížený výkonu výslednej aplikácie v dôsledku použitia frameworku, môže spôsobovať všeobecnosť jeho kódu, ktorý nemusí byť optimalizovaný pre určité špecifické situácie. Na naučenie sa frameworku sa často vyžaduje značné úsilie, aby ho vývojári vedeli efektívne a správne použiť. Preto je vhodné ak sa daný framework používa opakovane a teda náklady na jeho naučenie sa rozložia na viac projektov. Jedným z rizík je výber frameworku. Vhodnosť pre doménu, na ktorej má byť použitý, môže ovplyvniť flexibilita, čo znamená, že framework sa nemusí podariť prispôsobiť doméne. Rizikový je aj výber z hľadiska možných kolízií s ostatnými použitými frameworkami, systémami alebo technológiami a teda zlyhanie interoperability, ak je v konkrétnom riešení požadovaná. Chyby a bezpečnostné problémy frameworku môžu nepriaznivo ovplyvniť výslednú aplikáciu. Pri testovaní a ladení výslednej aplikácie, môže byť zdrojom komplikácií inverzia riadenia, pretože nie je jednoduché prechádzať aplikáciou po krokoch. Niekoľko nevýhod, ktoré ale nesúvisia priamo s použitím frameworkov, prináša ich vývoj. Vývoj frameworkov je nákladný a náročný. Dobre navrhnutý framework musí spĺňať určité kľúčové vlastnosti, v opačnom prípade sa môže ľahko stať nepoužiteľným. Jedná sa o jednoduchosť, rýchlu nasaditeľnosť, flexibilitu a tiež musí zahŕňať teóriu domény, ktorú pokrýva. Testovanie a hľadanie chýb vo frameworku môže byť tiež problematické, nakoľko je abstraktným riešením.

Formulárové frameworky a frameworky podporujúce prácu s ovládacími prvkami môžeme rozdeliť do dvoch skupín:

1. Skupina sú samostatné frameworky podporujúce ovládacie prvky, niekedy označované ako *widget toolkit*, *widget library* alebo *GUI toolkit*. Niektoré sú poskytované s operačným systémom, s grafickou nadstavbou operačného systému alebo existujú samostatne a sú alternatívou k štandardným frameworkom, *widget toolkitom*. Niektoré sú naviazané na určitú platformu, iné sú naopak platformovo nezávislé.

2. Skupina sú frameworky, ktoré sú súčasťou väčších ucelených aplikačných frameworkov, pričom v kontexte web formulárových frameworkov hovoríme zvyčajne o web aplikačných frameworkoch. Tieto sú zvyčajne užšie späté s doménou a platformou, pre ktorú je daný web aplikačný framework určený. Niektoré z týchto frameworkov sa ešte ďalej delia na rôzne časti, pričom tieto môžu využívať rôzne knižnice, frameworky na funkcionalitu ako je validácia, animácie, efekty a iné.

Pri vývoji používateľského rozhrania, webových formulárov sa určité prvky často opakujú či už so žiadnymi alebo väčšími či menšími obmenami, ale ak sa na to pozrieme z väčšieho odstupu, určité vzory sa stále opakujú. Doména webových aplikácií, čo sa týka vývoja, je pomerne frekventovaná, a teda vo všeobecnosti je s ňou aj dostatok skúseností, čo znižuje riziko výberu nesprávneho frameworku. Pri použití frameworkov v doméne webových aplikácii a webových formulárov pravdepodobne prevažujú výhody tohto prístupu nad nevýhodami.

Túto hypotézu podporuje aj prax, kde ak vývojári nepoužívajú nejaký framework tretej strany, tak po určitom čaše vývoja webových aplikácii dospejú k vlastnému frameworku. Takýto framework je väčšinou pevnejšie zviazaný s platformou, nad ktorou bol vyvíjaný, ale nie je to nutné pravidlo a vzniknutý framework môže byť použitý aj na inej platforme. Po istom čase môže byť vymenený iným treťostranným frameworkom, ktorý

funguje na podobných princípoch, ale napríklad ponúka širšie možnosti. Alebo môže nastať opačný prípad a framework je uvoľnený tretím stranám, ktoré ho začnú používať.

2.1. Vlastnosti web formulárových frameworkov

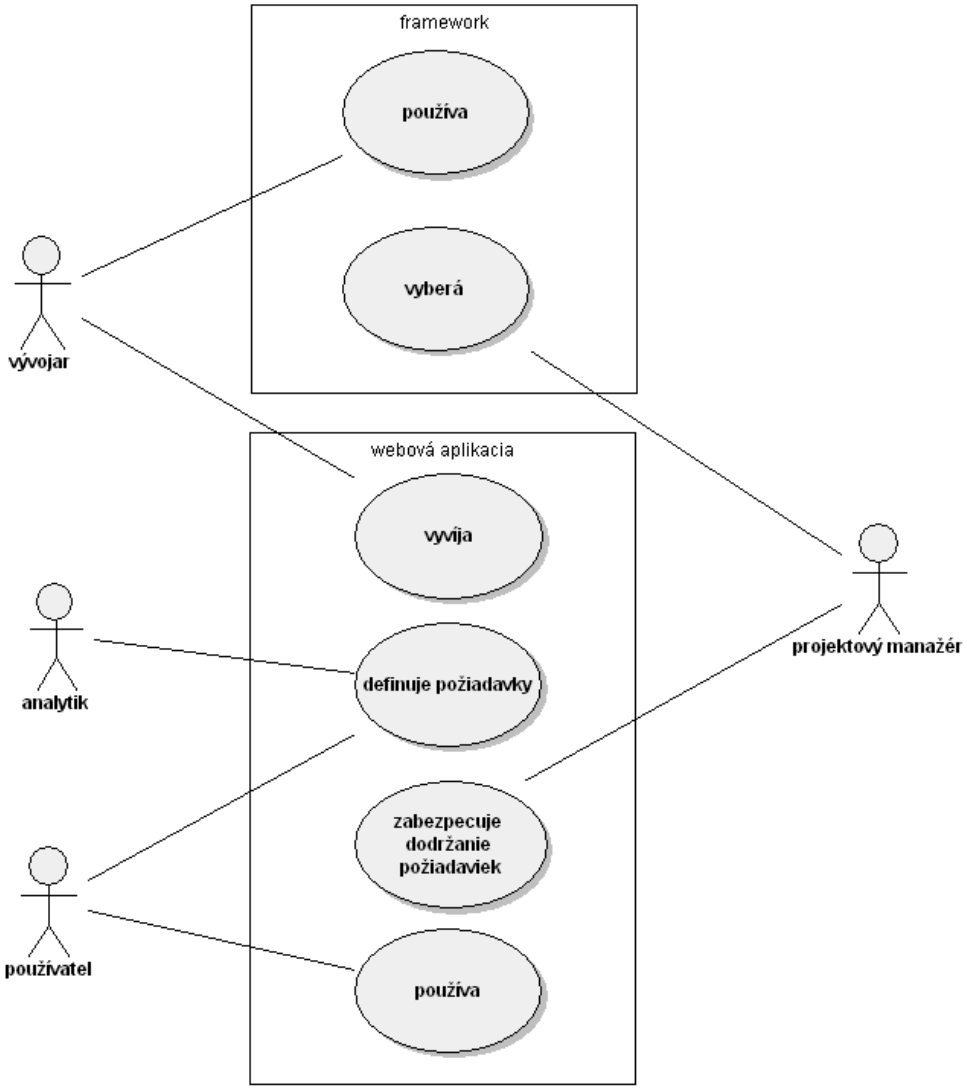
Vlastnosti web formulárových frameworkov by sa dali kategorizovať rôzne. Nás však z praktického hľadiska zaujalo kritérium, ktoré závisí od role, v ktorej je pozorovateľ vzhľadom na framework.

Prvá je rola vývojára, ktorého zaujíma ako bude s frameworkom pracovať a teda vlastnosti ako učiaca krivka, nástroje na podporu vývoja, platforma, na ktorej je daný framework založený, znalosti, ktoré musí pre prácu s frameworkom mať.

Druhá predstavuje pohľad role používateľa, ktorý bude výslednú aplikáciu používať, resp. role analytika, ktorý pomáha používateľovi, klientovi definovať požadované vlastnosti výslednej aplikácie, ktoré táto aplikácia získa aj vďaka použitiu konkrétneho frameworku.

Tretiu rolu predstavuje projektový manažér projektu, pri ktorom vzniká webová aplikácia. Vlastnosti zaujímavé pre túto rolu sú zväčša premostením medzi vlastnosťami zaujímavými pre prvú a druhú rolu. Napríklad projektový manažér môže na základe vlastností prvej role plánovať ako, za aký čas a ako efektívne budú naplnené vlastnosti zaujímavé pre druhú rolu.

Ako charakter tretej role naznačil, toto rozdelenie nie je striktné a teda niektoré vlastnosti sú na rôznej úrovni zaujímavé pre viaceré role.



Obrázok 2.1: Role prichádzajúce do styku web formulárovým frameworkom

2.1.1. Rola vývojára

Väčšinu vlastností, ktoré sú zaujímavé najmä z pohľadu vývojára na framework, priamo aj nepriamo ovplyvňuje povaha frameworku. To znamená na akom jazyku je založený, aké používa rozhrania a štandardy a zároveň do akej miery je s nimi vývojár oboznámený.

2.1.1.1. Jazyková závislosť

Pre vývojára, ktorý bude potenciálne s frameworkom pracovať, je dôležité na akom programovacom jazyku je založený, ak takáto väzba medzi frameworkom a programovacím jazykom existuje. Ak takáto väzba neexistuje alebo je veľmi slabá, to znamená, že framework síce využíva nejaký programovací jazyk, ale vývojár ho nutne nemusí ovládať na to, aby mohol pomocou daného frameworku vyvíjať. V mnohých prípadoch je lepšie, ak daný programovací jazyk aj napriek slabej väzbe ovláda.

Je tiež zaujímavé, aké rozhrania používa framework, či sa jedná štandardy, pri ktorých je väčšia pravdepodobnosť, že sa s nimi vývojár už stretol. Alebo je rozhranie neštandardné, čo ale neznamená je aj náročné na zvládnutie.

Podobne ako rozhrania sú dôležité technológie, ktoré sú súčasťou frameworku a miera akou ich vývojár ovláda.

2.1.1.2. Čitateľnosť kódu

Za čitateľnosť kódu je v prvom rade zodpovedný vývojár, ale tiež môže byť ovplyvnená programovacím jazykom, na ktorom je framework postavený, v ďalšom rade rozhraniami, ktoré používa a tiež štandardami, ktoré sú v ňom zakomponované. Napríklad ak sú rozhrania obšírne, potom je aj pre vývojára náročnejšie písať dobre čitateľný kód. Iný prípad kde framework ovplyvňuje čitateľnosť môže nastať, ak je kód príliš fragmentovaný po rôznych fyzických štruktúrach.

2.1.1.3. Nástroje pre podporu vývoja

V závislosti od frameworku, technológií a štandardov v ňom použitých, môže byť pre vývojára prínosné, ak existuje podpora daného frameworku alebo niektorej jeho súčasti

pre multiplatformové vývojové prostredia, naproti tomu niektoré riešenia naopak nútia vývojára použiť s daným frameworkom aj konkrétne vývojové prostredie. Ďalšou možnosťou je, že framework vývojára vôbec neovplyvňuje z hľadiska vývojového prostredia, ak je napríklad založený na dobre známych štandardoch a teda má širokú podporu.

2.1.1.4. Krivka učenia

Ďalšou vlastnosťou frameworku, ktorú ovplyvňujú jeho črty ako použite technológie, rozhrania alebo programovací jazyk, je krivka učenia. Predstavuje závislosť zvládnutia frameworku jeho používateľom od času. Môže byť ovplyvnená v pozitívnom zmysle ak vývojár pozná a ovláda technológie a rozhrania použité vo frameworku.

Pozitívny vplyv na rýchlejšie naučenie sa, zvládnutie má aj úroveň a hĺbka dokumentácie frameworku. Napríklad ak sa dokumentácia venuje nielen rozhraniam a postupom ako s frameworkom pracovať ale aj samotnému frameworku, čiže na akých princípoch funguje on sám. Náhľad na framework môžu poskytnúť aj rýchle príručky a názorné príklady. Prínosom pri riešení zložitejších a špecifickejších problémov je komunita používajúca konkrétny framework. Čím je väčšia, tým väčší je počet diskusných fór, blogov, mailových konferencií a podobne, kde môže vývojár nájsť riešenie problémov alebo im prislúchajúce pokročilejšie príklady.

Na počiatočnú fázu učenia sa frameworku vplývajú aj inicializačné technické a konfiguračné faktory. To znamená systémové a programové prostriedky, ktoré potrebuje vývojár, aby mohol začať vyvíjať pomocou frameworku.

2.1.1.5. Spätná kompatibilita

Spätná kompatibilita je zaujímavá pri dlhodobom používaní frameworku pri jednom komplexnom riešení. Tu je zaujímavé, či najbližšie prichádzajúce ohlásené verzie budú aspoň čiastočne kompatibilné s aktuálnou používanou prípadne vybranou verziou, z hľadiska rozhraní alebo objektových štruktúr a podobne.

Podobný prípad nastáva, keď je framework v pracovnej verzii, a teda sa často a aj zásadne mení, ale napriek tomu je použiteľný pri implementácii. Autori, pokiaľ nie je framework stabilizovaný, nevynakladajú úsilie na zachovanie spätnej kompatibility. Potom môže byť zaujímavým kompromisom, ak je taká situácia a potreba, keď framework podporuje súbežnosť viacerých svojich verzií.

2.1.2. Rola používateľa, analytika

2.1.2.1. Medzi prehliadačova kompatibilita

Jedna z najzaujímavejších vlastností ako pre používateľa a analytika, ale do istej miery aj pre vývojára. Ak framework potiera rozdiely medzi zobrazovaním a funkcionalitou výslednej prezentácie formulárov v rôznych prehliadačoch, tak to znamená prínos pre každú rolu, ktorá prichádza do styku s prezentáciou. K prezentácii môže pristupovať väčší počet používateľov, projektový manažér môže naplánovať kratší čas venovaný na vývoj prezentácie, vývojár sa nemuší venovať optimalizácii formulárov pre jednotlivé prehliadače.

Medzi prehliadačova kompatibilita je zároveň jedným z hlavných aspektov motivácie vývoja frameworkov špeciálne zameraných na webové formuláre.

2.1.2.2. Internacionalizácia a lokalizácia

Nakoľko web formulárové frameworky napomáhajú vývojárom pri tvorbe prezentačnej vrstvy webových aplikácií, je prínosné ak majú priamo podporu internacionalizácie a lokalizácie, aj keď pre konkrétne riešenie nemusí byť aktuálne žiadaná. Čo neznamená, že táto požiadavka časom nepribudne alebo nebude aktuálna pri riešení iného projektu. Tiež je zaujímavé, či je pri internacionalizácii a lokalizácii myslené aj rozširujúce časti, prípadne na exotickejšie jazyky v závislosti od lokality, smer písma, rôzny systém číslic a podobne.

2.1.2.3. Prístupnosť pre ľudí s obmedzeniami

Existujú frameworky generujúce výstup, ktorý podporuje prístupnosť na rôznych stupňoch definovaných W3C[3]. Prístupnosť pre ľudí s rôznymi obmedzeniami má prinajmenšom humánnu pridanú hodnotu, v prípade že doména, v ktorej je aplikovaný framework, priamo nevyžaduje podporu prístupnosti.

2.1.2.4. Dostupnosť pri nepripojení

Niektoré frameworky respektíve technologické prístupy, na ktorých sú založené, umožňujú v rôznej miere obmedzenú funkcionalitu na klientskej časti webového formulára, aj keď je používateľ offline. Podobná nemenej zaujímavá črta niektorých riešení spočíva vo vyňatí funkcionality webového formuláru respektíve webovej aplikácie mimo prostredia webového prehliadača, čo môže byť potenciálne prínosné pre niektoré domény.

2.1.3. Rola projektového manažéra

2.1.3.1. Licencovanie

Licencie jazykov, technológií a štandardov, ako aj licencia samotného frameworku ovplyvňujú doménu, v ktorej bude framework použitý, ale taktiež schopnosť vývojárov stotožniť sa s frameworkom, možnosť rozšíriteľnosti, prispôsobiteľnosti a tiež rozsah a spôsob podpory pri používaní frameworku.

2.1.3.2. Závislosti pri nasadení a používaní

Použitie frameworku môže priniesť určitú väzbu na softvérové prostriedky, v závislosti od technológií, na ktorých je framework postavený. Napríklad framework pri nasadení v produkčnom prostredí môže požadovať konkrétne softvérové prostriedky, s ktorými spolupracuje, ktoré využíva alebo v ktorých beží. Iným príkladom závislosti frameworku od softvérových prvkov je, keď používateľ musí disponovať nadštandardnými resp. rozširujúcimi softvérovými prostriedkami, aby využil plnú funkčnosť prezentácie webových formulárov.

V zásade pri vývoji frameworku existuje snaha väzby a závislosti minimalizovať na najnižšiu možnú mieru, prípadne sa ich úplne zbaviť a to najmä na strane klienta. Ale existujú domény, kde tieto závislosti nehrajú rolu resp. sa s nimi počíta výmenou za lepší výkon, rozšírenú funkcionalitu alebo pridanú prezentačnú hodnotu.

2.1.3.3. Výkonnosť

Framework obsahuje nadbytočný kód oproti konkrétnemu špecifickému riešeniu, ale vďaka tomu je abstraktnejší a všeobecnejší a teda môže byť viacnásobne použitý. Zovšeobecnenie môže priniesť s kódom navyše aj procesnú réžiu. Tieto faktory znižujú

výkonnosť. Preto pri výbere frameworku treba na túto skutočnosť prihliadať. Pri vývoji frameworku ak sú ostatne aspekty ako rozhrania, používané vzory, technológie ustálené, je snaha o optimalizáciu. Zníženie výkonnosti do určitej miery prichádza do úvahy za predpokladu iných prínosov ako napríklad rozšírená funkcionalita, širšia doména použitia alebo zlepšenie rozhraní.

2.1.3.4. Rozšíriteľnosť, prispôsobiteľnosť

Rozšíriteľnosť a prispôsobiteľnosť frameworku môže priniesť v konkrétnych riešeniach rozšírenú funkcionalitu alebo zvýšenie výkonnosti výsledného riešenia. V mnohých prípadoch sa jedná o špeciálne požiadavky používateľa, klienta na výsledný produkt, ako napríklad využitie treťostranných služieb, pridanie špecializovaných ovládacích alebo vizualizačných prvkov, špeciálne požiadavky na bezpečnosť a podobne.

Akým spôsobom je v konkrétnom frameworku realizovateľná rozšíriteľnosť a prispôsobiteľnosť sa odráža na úsilí vývojára pri implementácii špeciálnych požiadaviek. Do akej miery je možné framework prispôsobiť a rozšíriť ovplyvňuje výber frameworku na doméne, kde hrozia časté a rôznorodé zmeny.

2.1.3.5. Paleta možností

Paleta možností predstavuje zhrnutie tých vlastnosti frameworku, ktoré vymedzujú oblasti v ktorých sa dá použiť, vymedzujú jeho záber respektíve požiadavky na výslednú aplikáciu, ktoré priamo pokrýva alebo ich vieme s minimálnym úsilím pokryť pri implementácii s jeho použitím. Sú to možnosti, ktorými framework disponuje, bez nutnosti intenzívne využiť jeho rozšíriteľnosť.

vlastnosť / rola	**vývojár**	**používateľ, analytik**	**projektový manažér**
platformová a jazyková závislosť	**		*
licencovanie	*		**
čitateľnosť kódu	**		
nástroje pre podporu vývoja	**		*
krivka učenia	***		*
spätná kompatibilita	*		*
medzi prehliadačova kompatibilita	**	**	*
internacionalizácia a lokalizácia	*	**	
prístupnosť pre ľudí s obmedzeniami	*	**	
dostupnosť pri nepripojení		*	
závislosti pri nasadení a používaní		*	**
výkonnosť		*	**
rozšíriteľnosť, prispôsobiteľnosť	**	*	*
paleta možností	**		**

Tabuľka 2.1: Prehľad zaujímavosti vlastnosti pre role

Na záver tejto podkapitoly uvádzame prehľad uvedených vlastností, počet značiek (*) predstavuje mieru zaujímavosti vlastnosti pre rolu v prislúchajúcom stĺpci v tabuľke 2.1. Tabuľka naznačuje, že medzi kľúčové vlastnosti formulárových frameworkov patria krivka učenia, medzi prehliadačova kompatibilita, rozšíriteľnosť a prispôsobiteľnosť a tiež paleta možností. Framework, ktorý napĺňa tieto vlastnosti v čo najväčšej miere v rámci domény, pre ktorú je určený, konceptu, na ktorom je založený a možností technológií, na ktorých je postavený, má dobré šance sa stať takzvaným ideálnym frameworkom. Preto tieto vlastnosti môžeme označiť ako takzvané ideálne a pri rozbore konkrétnych frameworkov sa zameriame aj na ne.

2.2. Web formulárové frameworky

V tejto podkapitole popíšeme niektoré frameworky, ktoré sa zaoberajú webovými formulármi. Výber frameworkov bude zameraný skôr na priblíženie rôznych prístupov v tejto problematike ako na „merateľné“ porovnanie frameworkov rovnakého charakteru.

2.2.1. Zend framework

Zend framework je open-source objektovo orientovaný web aplikačný framework, implementovaný v PHP5.

Framework svojou architektúrou striktne nenúti používateľov, aby sa držali jednej konkrétnej vývojovej paradigmy, respektíve vzoru, napriek tomu implementuje viaceré dizajnové vzory ako Table Data Gateway, and Row Data Gateway [4] a tiež MVC.

Zend je kompozitný framework skladajúci sa z viacerých komponentov, ktoré implementujú rôznu funkcionalitu, technológie a vzory. Nás bude zaujímať najmä komponent, ktorý slúži na implementáciu formulárov – Zend_Form.

Zend_Form komponent umožňuje:
vytvoriť

- formulár (pomocou triedy Zend_Form)
- elementy formulára (trieda Zend_Form_Element)
- grupy elementov (trieda Zend_Form_DisplayGroup) za účelom špecifického prístupu k nim a ich špecifickej prezentácie
- pod formuláre na logické rozdelenie formulára, ktoré budú vo výslednej prezentácii samostatné formuláre

modifikovať

- atribúty a metadáta hore spomenutým objektom, a teda zmeniť ich povahu pri prezentácii, validácii a manipulácii
- spôsob prezentácie elementov formuláru pomocou dekorátorov (trieda Zend_Form_Decorator)

- atribúty a metadáta môžu byť modifikované aj pomocou konfigurátorov (trieda Zend_Config), ktoré nesú ich nastavenia a môžu byť inicializované z fyzických súborov

Zend obsahuje a môžu byť používané:

- sada formulárových elementov, ktorá je ale o niečo širšia ako štandardné HTML formulárové elementy
- preddefinované dekorátory
- preddefinované validátory(trieda Zend_Validate)
- preddefinované filtre(trieda Zend_Filter)

Väčšina doteraz spomenutých tried môže byť zdedená, rozšírená a prispôsobená špeciálnym potrebám pri konkrétnom riešení. Prispôsobené validátory, filtre a dekorátory môžme formuláru sprístupniť nadstavením inštancie triedy Zend_Loader_PluginLoader.

.

Pri internacionalizácii Zend framework používa komponent Zend_Translate, ktorý spolupracujes s komponentmi Zend_Form, Zend_Validate a Zend_View_Helper_Translate.

Zend nakoľko je objektovo orientovaný a implementovaný v PHP, tak týmto konceptom sa s nim aj pracuje. To znamená na úrovni PHP sa vytvoria objekty slúžiace pre formulár, k nim sa nastavia objekty dátového modelu, aby bola prezentácia previazaná s dátami. Ďalej sa podľa potreby vytvoria aj ďalšie pomocné objekty. Objektom sa ponastavujú atribúty tak, aby vyhovovali špecifickému riešeniu a výsledné značkovanie vyzerá jednoducho a prehľadne, nakoľko tu dochádza väčšinou len k zavolaniu vykresľovacích metód formulárového objektu.

2.2.2. Google web toolkit

Google web toolkit (ďalej len GWT) je kompaktný web aplikačný framework. Zameraný je na webové aplikácie s komplexným užívateľským rozhraním, ktorého logiku na strane klienta zabezpečuje JavaScript a asynchrónne volanej serverovej logiky pomocou technológie AJAX. Pričom všetko je vyvíjané na JAVA platforme.

Hlavné komponenty GWT:

- Kompilátor Javy do JavaScriptu – kľúčový komponent, v momente keď je Java kód pripravený na nasadenie prekompiluje sa do JavaScriptu, dva módy kompilácie. V prvom prípade je výsledný JavaScript čitateľný pre človeka, vhodné pri testovaní. Druhý prípad, keď je kód pripravený na nasadenie do produkcie, výsledný je obfuskovaný, optimalizovaný na výkon, veľkosť, špeciálne pre jednotlivé prehliadače a pre jazykové mutácie.

- Emulátor Java platformy – podpora syntax, konštrukcii jazyka a balicky základnej časti API sú reimplementované v JavaScripte, vďaka tomu kód z Javy preložený kompilátorom do JavaScriptu môže bežať v prostredí prehliadača

- Knižnica komponentov používateľského rozhrania – komponent, ktorý nás bude bližšie zaujímať.

- GWT Hosted Mode – ladiaco-testovací koncept, pri ktorom sa Java kód neprekladá do JavaScriptu, ale JVM spúšťa kód aplikácie ako byte kód v ladiacom móde v špeciálne upravenom prehliadači, štandardne na zabudovanej inštancii Jetty web servera. Prehliadač pozostáva z dvoch okien. V prvom hlavnom okne je výpis ladenia a výnimky, možnosť reštartovania web servera a spustenia hosted prehliadaču. V druhom okne – v hosted prehliadači beží samotná webová aplikácia, ktorú takto môžeme testovať.

Hlavným prínosom knižnice komponentov používateľského rozhrania je medzi prehliadačova kompatibilita. Tú zabezpečuje jednak optimalizovaný preklad javy do JavaScriptu na úrovni logiky, v druhom rade prezentácia komponentov pomocou HTML.

Komponenty môžeme rozdeliť na dve skupiny, jedna tvorí paletu widgetov, v druhej sú panely, komponenty, ktoré zabezpečujú rozloženie prvkov používateľského rozhrania.
K obom skupinám pristupujeme pomocou inštancií ich tried a vzájomnou kombináciou modelujeme používateľské rozhranie.

Rozšíriť množinu prvkov používateľského rozhrania môžeme viacerými spôsobmi. Jedným z jednoduchších je použiť treťostrannú knižnicu komponentov kompatibilnú s GWT, ktorých je dostupných niekoľko. Ďalšou z jednoduchších možností je vytvoriť kompozitný komponent zo štandardných GWT komponentov používateľského rozhrania.
Ostatné dva spôsoby už majú nevýhodu v tom, že pri nich musí vývojár sám myslieť na medzi prehliadačovu kompatibilitu. Ide o vývoj buď v jave, pričom inšpiráciu môže vývojár hľadať práve v spôsobe implementácie štandardných komponentov GWT, alebo použije pri tvorbe JavaScript.

Akýkoľvek spôsob vývoja vlastného komponentu používateľského rozhrania si vývojár vyberie, ak potrebuje pristupovať k prezentačným elementom pomocou DOM, GWT mu na to poskytuje komfortnú triedu, ktorá opäť potiera rozdiely medzi jednotlivými prehliadačmi na úrovni práce s DOM.

Pre internacionalizáciu GWT ponúka tiež viacero možností. Najjednoduchšou a z hľadiska kódu navyše aj najefektívnejšou možnosťou je internacionalizácia statickými reťazcami, pri ktorých sa používajú Java *property* súbory. Internacionalizácia dynamickými reťazcami je pomalšia, ale za to flexibilnejšia, aplikácie pri nej vyhľadávajú lokalizovane reťazce v module na hosťovskej stránke. Tato možnosť stojí za zváženie, ak integrujeme GWT web aplikáciu s existujúcim lokalizačným systémom na strane servera.

GWT taktiež myslí na prístupnosť a má na jej podporu konkrétne balíčky. V dokumentácii poskytuje odporúčania a postupy ako spraviť aplikáciu založenú na GWT prístupnejšou.

2.2.3. JQuery

JQuery je čisto JavaScriptový framework, knižnica. Zjednodušuje prácu s JavaScriptom, predovšetkým už viackrát spomínaným zrušením rozdielov pri použití rôznych prehliadačov.

JavaScriptom ako takým väčšinou neriešime celé používateľské rozhranie resp. webový formulár ale len jednotlivé prvky, *widgety*, ich logiku, prezentáciu, animácie. Samotný JQuery obsahuje úzku paletu prvkov používateľského rozhrania, ale poskytuje efektívne rozhranie na vývoj špecifických.

Zaujímavým je prehliadačovo nezávislý prístup k objektom DOM a to na základe štandardu CSS1 až CSS3. JQuery ma jednoduchú a prehľadnú syntax aj keď nie najbežnejšiu v porovnaní s ostanými generickými JavaScriptovými frameworkami.

Ďalšími prínosnými vlastnosťami JQuery je jeho veľkosť a uzavretosť. Veľkosť prispieva k menšiemu vyťaženiu sieťového spojenia a uzavretosť znamená, že pri jeho použití s iným JavaScriptom alebo JavaScriptovým frameworkom nedochádza k ich ovplyvňovaniu.

Koncept, ktorý predstavuje JQuery resp. JavaScriptový framework, je jednak použiteľný pri menších projektoch, či už pre špeciálne ovládacie prvky typu “editor” alebo ovládacie prvky pre rozloženie. Druhá možnosť využitia sa ponúka v kombinácii s niektorým konceptom web aplikačného frameworku.

Pri JavaScriptovom frameworku je možno otázne, či je to framework v pravom slova zmysle, keďže väčšinou nedochádza k inverzii riadenia a teda aplikácia volá JavaScriptový kód a preto je možno priliehavejšie hovoriť o JavaScriptovej knižnici. Každopádne JavaScriptový framework alebo knižnica má svoje miesto v problematike webových formulárov a preto tu bola aj uvedená.

2.2.4. XUL

XUL – XML User-Interface Language – XML značkovací jazyk pre používateľské rozhranie. Bol vyvinutý Mozillou a beží najmä v medzi platformových aplikáciách Mozilly. Mozilla Gecko mechanizmus poskytuje implementáciu XUL, ktorá bohato pokrýva jeho črty.

XUL pri vývoji aplikácií používa rôzne dobre známe webové štandardy a technológie. Samotný je dialektom XML 1.0 ďalej využíva HTML, CSS, DOM, JavaScript.

Vývojári obyčajne definujú používateľské rozhranie založené na XUL ako tri oddelené množiny komponentov. A to obsah ako XUL dokument, ktorý zastupuje v ponímaní modelu aspektov webového formuláru popis, výber widgetov a rozloženie. Ďalšiu množinu predstavuje témovanie prostredníctvom CSS a súborov so statickou grafikou. A napokon lokalizáciou, ktorú predstavujú fyzické súbory obsahujúce priamočiare lokalizované reťazce.

XUL ma definovanú širokú paletu značiek, ktoré môžeme rozdeliť na niekoľko typov. Top-level a Box-model značky predstavujú rozloženie, Wigets widgety, Events and Scripts značky predstavujú funkcionalitu.

XUL primárne slúži na vytváranie aplikácií Mozilly a ich rozšírení. Ale tiež môže byť využitý pri tvorbe webových aplikácií. Jednou z možností je využiť ho ako popisný jazyk pre nejaký webový formulár, pričom pred stykom s prehliadačom bude transformovaný na iný jazyk priamo podporovaný prehliadačom a ten môžeme považovať za implementačný jazyk. Druhá možnosť nastane ak zúžime doménu prehliadačov na prehliadače založené na Gecko mechanizme potom sa XUL značkovanie stáva natívnym a môžeme využiť jeho rôzne črty. Ďalším prínosom pri takomto kroku je vyššia výkonnosť pri použití widgetov, ktoré sú skrz XUL natívne prehliadaču. Iným pozitívom tohto prístupu je, že môžeme využiť multi platformovosť Mozilla prehliadača a vytvoriť multiplatformovú aplikáciu, ktorá využíva široké možnosti jazyka XUL.

Na zaver tejto kapitoly uvádzame tabuľku 2.2 ktorá zahŕňa porovnanie vyššie spomenutých frameworkov na základe miery splnenia ideálnych vlastností z podkapitoly 2.1.

	medzi prehliadačová kompatibilita
Zend framework	zabezpečená v majoritných prehliadačoch pri použití štandardných wigetov a kompozícii.
Google web toolkit	zabezpečená na aj s optimalizáciou pre jednotlivé prehliadače, dokonca aj pri rozširovaní a prispôsobovaní widgetov a kompozícií
JQuery	zabezpečená čo sa týka pokrytia frameworku číže najmä interaktívnosť, ale aj štandardizuje prístup k DOM v jednotlivých browseroch na základe syntaxe CSS
XUL	nezabezpečená v Gecko nepozitívnych prehliadačoch, číže slabá až nesplnená vlastnosť
	rozšíriteľnosť a prisposobiteľnosť
Zend framework	je rozšíriteľný o nové widgety aj kompozície formou vlastných tried a šablón
Google web toolkit	je rozšíriteľný formou tried a konštrukcií samotného frameworku pričom zabezpečuje aj medziprehliadačovú kompatibilitu
JQuery	prostredníctvom rozširujúcich skriptov využívajúcich JQuery
XUL	osobitný flexibilný prístup k rozšíreniam a prispôsobeniam
	paleta možností
Zend framework	základná paleta widgetov
Google web toolkit	širšia paleta základných widgetov, k dispozícii veľa externých widgetovych knižníc na báze gwt pomerne kvalitne spracovaných

JQuery	základná paleta widgetov, k dispozícii externe widgety nekonzistentnej kvality
XUL	široká paleta widgetov a kompozicií
	krivka učenia
bZend framework	závisí od skúsenosti s jazykom PHP, okrem toho podpora prostredníctvom podrobne spracovanej programátorskej príručky
Google web toolkit	plus skúsenosti s jazykom Java, dobre spracovaný vývojársky manuál, rýchly štart aj prehľad technológie, mnohé existujúce projekty ako pokročilé príklady
JQuery	slušne spracovaný vývojársky manuál, mnohé komunitné weby slúžiace ako príručky
XUL	založený na XML, pomerne intuitívna syntax, tiež prepracovaný vývojársky manuál

Obrázok 2.2: Porovnanie web formulárových frameworkov

2.3. Zhrnutie formulárových frameworkov.

Zovšeobecnením poznatkov o technológiách a frameworkoch, z ktorých niektoré boli spomenuté v podkapitole 1.2 a 2.2 tejto práce, používaných v súčasnej dobe pri tvorbe webových formulárov dostávame niekoľko konceptov. Tieto koncepty odrážajú charakter vývojárskeho popisu a *zobrazenia* do prehliadačového popisu.

Prvý koncept.

Abecedu týchto frameworkov zastupujú programovacie jazyky, ktoré priamo podporujú *zobrazenie* do prehliadačového popisu. Preto zväčša aj *gramatika* vývojárskeho popisu svojimi *pravidlami* a *terminálmi* odzrkadľuje priamu väzbu na prehliadačový popis. A teda samotné *zobrazenie* z vývojárskeho popisu na prehliadačový je v podstate priamočiare a vo väčšine prípadov sa odohráva v oblasti servera, kde majú použité programovacie jazyky v roli *abecedy* podporu.

Tento koncept má pri vývoji webových aplikácií v súčasnej dobe pomerne široké zastúpenie. V roli *abecied* sa jedná o technológie a programovacie jazyky ako PHP,

Python, Ruby, Perl, JSP a iné. Prislúchajúce frameworky predstavujú Zend Framework, Symfony, CakePHP, Ruby on Rails, Django, Apache Struts a mnohé ďalšie.

Druhý koncept.

Vývojársky popis nezachytáva zvyčajne celý webový formulár, ale len widget a jeho interaktívnosť. *Abeceda* je reprezentovaná programovacím jazykom, ktorý je spoločný pre vývojársku aj prehliadačovu rovinu. *Gramatiku* predstavujú abstraktné konštrukcie programovacieho jazyka. *Zobrazenie* do prehliadačovej roviny predstavuje aplikovanie abstraktných konštrukcií vývojárskeho popisu v konkrétnej prehliadačovej rovine, a spravidla sa odohráva až na strane klienta čiže v prehliadači.

Abecedu predstavuje programovací jazyk JavaScript a *gramatiku* frameworky ako JQuery, Ext JS, mootools, mochikit, prototype, dojo, qooXdoo, Rico a iné.

Tretí koncept.

Vývojársky popis zachytáva aspektový model v abstraktnejšej forme ako prvý koncept, to znamená, že nie je na prvý pohľad zrejmé akým prehliadačovým popisom budú jednotlivé aspekty reprezentované v prehliadačovej rovine. Framework predstavuje väčšinou sofistikované *zobrazenie* z vývojárskeho popisu na prehliadačový, tento je zväčša realizovaný na strane servera.

Abecedu v tomto prístupe môžu reprezentovať programovacie jazyky, ale napríklad aj rôzne dialekty XML. Za frameworky predstavujúce *gramatiku* a v tomto prípade najmä *zobrazenie* môžeme považovať Chiba, Orbeon Forms, GWT, Pyjamas, RubJS a iné.

Štvrtý koncept.

Vývojársky popis je zväčša kompozitný, to znamená, že rôzne aspekty formulára sú popísané rozdielnymi vývojárskymi popismi. Tieto popisy sú zväčša generované rôznymi *gramatikami* dokonca nad rôznymi *abecedami*. Framework resp. technológia zabezpečujúca *zobrazenie* z vývojárskej roviny do prehliadačovej ma komplexnejší charakter. Požaduje softvérovú podporu prehliadača na samotné zobrazenie alebo je táto podpora nutná pri prezentácii prehliadačového popisu v prehliadačovej rovine.

Na tomto koncepte sú založené frameworky ako Flex, Silverlight, JavaFx, OpenLaszlo, Xul a iné.

3. Web formulárový framework

V tejto kapitole sa budeme venovať frameworku dforms, ktorý je modifikáciou web formulárového frameworku DForms.

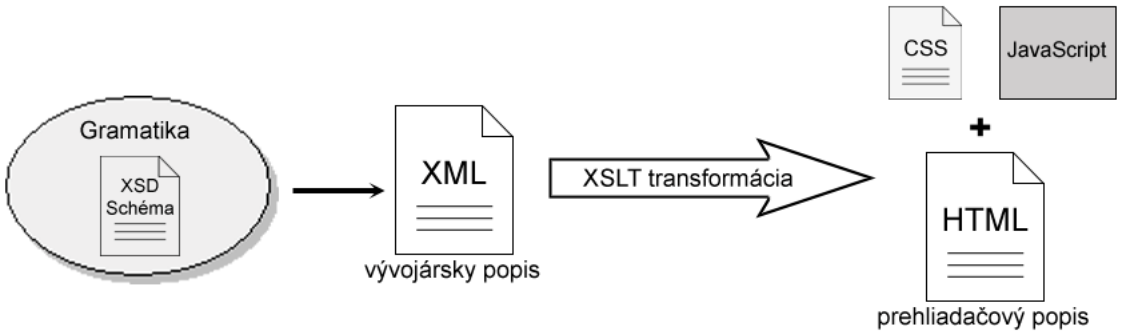
V podkapitole 3.1 rozoberieme architektúru, princípy, štandardy a technológie, na ktorých je výsledný framework založený.

V podkapitole 3.2 špecifikujeme framework prostredníctvom vlastností z kapitoly 2.1

3.1. Návrh frameworku

Framework dforms je web formulárovým frameworkom patriacim do prvej skupiny, ktorá sa spomína v úvode 2. kapitoly tejto práce. Znamená to, že je priamo zameraný iba na problematiku webových formulárov a teda nie je web aplikačným frameworkom, čo neznamená, že nemôže byť využívaný nejakým iným web aplikačným frameworkom.

Z pohľadu paradigmy MVC na webové aplikácie tento framework zastrešuje najmä časť view, teda prezentáciu.



Obrázok 3.1: princíp fungovania frameworku dforms

Ako obrázok 3.1 naznačuje, dforms *zobrazenie* z množiny vývojárskych popisov do množiny prehliadačových popisov realizuje XSL transformáciou. Transformácie zachytené XSL štýlmi tvoria nosnú časť frameworku. Framework teda transformuje XML dokument na HTML dokument. Z rozdelenia frameworkov v podkapitole 2.3 je v dforms uplatnený 3. koncept.

XML dokument z aspektov formulára zachytáva widgety a kompozíciu. XSD schéma definuje pomocné značky, značky pre widgety, kompozíciu a vzťahy medzi nimi.

CSS kaskádové štýly popisujú aspekt témovania ako vo vývojárskej rovine tak aj v rovine prehliadača a teda *zobrazenie* respektíve transformácia ich nemodifikuje. Z rovnakých dôvodov nepodlieha transformácii ani JavaScript, ktorý ale definuje aspekt interaktívnosti.

XSL transformácia je štandardne podporovaná majoritnými prehliadačmi, webový formulár môže byť distribuovaný aj v XML formáte až na klienta. Z praktického hľadiska je tato vlastnosť frameworku využívaná najmä pri vývoji. Ďalšou možnosťou je transformovať XML popis formuláru už na serveri a následne distribuovať ako kombináciu HTML, JavaScriptu a CSS.

Fyzická štruktúra dforms určuje na aké komponenty sa framework delí a je spoločná pre transformačné XSL štýly, témovacie kaskádové štýly a súbory kódu JavaScriptu.

Fyzická štruktúra:

- dforms – koreňový adresár frameworku
    - doap – JavaScript pre validáciu, udalosti, prácu s DOM, ...
    - pickers – JavaScript pre vizuálnu interakciu
    - renderKit – JavaScript, CSS, XSL zabezpečujúce výslednú prezentáciu formuláru
        - editors – widgety pre vstup a editáciu

        - viewers – prezentačné widgety
        - actions – navigačné widgety
        - panels – JS, CSS, XSL pre kompozíciu a kompozičné widgety
        - decorators – nadstavbové JS, CSS, XSL pre widgety a kompozíciu
        - tools – pomocné XSL
        - controls pomocné JS, CSS, XSL
    - xsd – *gramatika* pre vývojársky popis formulárov a *gramatika* pre validačné značky

3.2. Špecifikácia frameworku

Jazyková závislosť

Pri hodnotení tejto vlastnosti treba rozdeliť používateľov-vývojárov frameworku dforms do dvoch skupín. Prvú skupinu tvoria tí, ktorí používajú framework tak ako je, bez využitia jeho rozšíriteľnosti a len čiastočného využitia prispôsobiteľnosti. Naopak druhú skupinu predstavujú vývojári rozširujúci framework o vlastné widgety a prispôsobujúci widgety veľmi špecifickým požiadavkám.

Pri prvej skupine môžeme povedať že framework je jazykovo nezávislý, lebo na prácu s ním stačí vedieť pracovať s XML dokumentmi.

Druhá skupina, ak chce prispôsobovať a rozširovať framework, by mala ovládať CSS, XSLT a JavaScript.

Licencovanie

Čo sa týka technológií, na ktorých je dforms postavený, tak sa jedná prevažne o otvorený charakter. Licencia samotného DForms, z ktorého dforms vychádza, je LGPL.

Čitateľnosť kódu

O čitateľnosti XML dokumentov by sa dalo polemizovať, podobne aj o tom, či má jazyk vplyv na čitateľnosť, ktorá v prvom rade záleží od autora XML dokumentu. Podporiť čitateľnosť môže aj existencia XSD schémy, prípadne náhľad na dokument vo forme grafu.

Napriek týmto skutočnostiam dforms je od jeho predlohy o niečo čitateľnejší, možno pochopiteľnejší vďaka menšej zmene značiek pre widgety a kompozície, z ktorých je hneď zrejmé, o ktorú konkrétnu inštanciu sa jedná a to ako na začiatočnej tak aj na konečnej značke.

Nástroje pre podporu vývoja

Dforms neponúka špeciálny nástroj ani plugin do generického prostredia nakoľko je založený na pomerné rozšírených technológiách, ktoré samé väčšinou podporu majú.

Krivka učenia

Krivka učenia súvisí s jazykovou závislosťou, preto ju opäť budeme hodnotiť z hľadiska dvoch používateľských skupín, o ktorých sme hovorili.

Pre prvú je naučenie sa frameworku viac menej o spoznaní syntaxe, možnosti a teda widgetov, kompozícií, čo je podporené aj názornými príkladmi formulárov.

Druhá skupina je do istej miery ovplyvnená znalosťami o XSLT,CSS a JavaScripte. Tu má framework rezervy, čo sa týka určitého detailnejšieho, ale schematického popisu vnútorného fungovania, prípadne absencia názornej príručky ako vytvoriť vlastný widget, či už nový alebo prispôsobením existujúceho.

Spätná kompatibilita

Nakoľko je aj zdrojový framework ešte pomerne neustálený, tak nemá asi zmysel uvažovať o spätnej kompatibilite. I keď pri zmene rozhraní by sa v súvislosti s povahou vývojárskeho popisu dalo v niektorých prípadoch uvažovať o transformácii, ktorá by menila staré rozhrania na nové.

Medzi prehliadačova kompatibilita

Framework ponúka ekvivalentnú prezentáciu a funkcionalitu pre štandardnú paletu widgetov a kompozícií na majoritných prehliadačoch a to nielen pri ich posledných verziách.

Internacionalizácia a lokalizácia

Dforms ponúka najmä podporu viacjazyčných mutácií a to ako na úrovni widgetov tak aj pre validáciu.

Prístupnosť pre ľudí s obmedzeniami

Je podporená jednak používaním popisov pre jednotlivé widgety a kompozície. Ďalej úpravou wigetov tak, aby boli použiteľné a dosiahnuteľné pomocou klávesnice. K lepšej prístupnosti prispieva aj samotný koncept frameworku, nakoľko jeho výstupom je HTML, ktoré je vo všeobecnosti asi najlepšie prístupné oproti jeho alternatívam.

Dostupnosť pri nepripojení

Ako sme spomínali transformácia z vývojárskeho popisu na prehliadačový môže prebehnúť až v samotnom prehliadači a teda ak máme lokálne XML dokument s popisom formulára a aj samotný framework tak vieme dosiahnuť transformáciu a dostaneme výsledný formulár, ale či vieme zmysluplne využiť jeho vyplnenie už nezávisí od frameworku. Tato črta frameworku je ideálne využiteľná pri vyvíjaní resp. dizajnovaní

formulára nakoľko nemusíme popis nasadzovať do nejakého vzdialeného prostredia, kde je pri väčšine reálnych projektov vo finálnej podobe nasadzovaný.

Závislosti pri nasadení a používaní

Z predchádzajúcej vlastnosti a z technológií použitých v jadre frameworku vyplýva, že framework nemá nadštandardné nároky na prehliadač, pri serveri potrebuje zabezpečenie XSL transformácie, ostatné závislosti väčšinou vyplývajú zo spôsobu, akým je framework využitý v konkrétnom riešení.

Rozšíriteľnosť, prispôsobiteľnosť

Rozšíriteľnosť a prispôsobiteľnosť má vo frameworku široké možnosti, za predpokladu ovládania technológií, na ktorých je založený. Pri rozširovaní však framework nezabezpečuje automatickú detailnú podporu prístupnosti, nakoľko je pri špecifických widgetoch neodhadnuteľná. S podporou internacionalizácie je to o niečo lepšie, ale pri tvorbe úplne nových widgetov netreba na túto problematiku zabúdať.

Paleta možností

Paleta možností je v frameworku dforms oproti DForms chudobnejšia, nakoľko sme pri preberaní widgerov a kompozícií kládli dôraz na dostupnosť, pričom jej zabezpečenie nie je väčšinou triviálne.

Záver

Jedným z cieľov prace bolo preskúmať a analyzovať situáciu, možnosti a prístupy v oblasti tvorby webových formulárov. Tento prieskum nás priviedol k podrobnej analýze samotných webových formulárov. Výsledkom tejto analýzy je pomerne všeobecný abstraktný model webového formulára, ktorý nám odkryl spôsob akým formálnejšie zachytiť podstatu ľubovoľného web formulárového frameworku. A teda sa nám podarilo definovať framework pomocou gramatiky generujúcej inštancie modelu webového formulára a zobrazenia týchto abstraktných modelov do modelu tvoreného konkrétnymi technológiami použitými pri nasadení webového formuláru. Tento nie častý pohľad na web formulárové frameworky nám umožnil tiež určiť niekoľko základných konceptuálnych tried, do ktorých môžeme frameworky rozdeliť.

Z analýzy mnohých web formulárových frameworkov sme tiež dostali množinu všeobecných dôležitých vlastností web formulárových frameworkov. Túto množinu je dôležité preskúmať pri výbere, použití alebo tvorbe web formulárového frameworku. Analýzou jednotlivých vlastnosti z hľadiska rôznych rolí, ktoré sa môžu dostať do styku s web formulárovým frameworkom, sme z týchto vlastnosti vybrali takzvané ideálne vlastnosti. Miera naplnenia týchto ideálnych vlastnosti určuje celkovú použiteľnosť framworku.

Ďalším z cieľov diplomovej práce bolo implementovať framework na základe čo najlepšieho zastúpenia ideálnych vlastnosti. Východiskom bol existujúci framework DForms. Tento framework však v pomerne vysokej miere napĺňa ideálne vlastnosti, a ak by sme ho chceli vylepšiť tam, kde má medzery tak by sme v ňom museli spraviť pomerne radikálne zmeny, čo by prinieslo riziko. Riziko spočíva v tom, že by sa nemusela zachovať miera naplnenia takzvaných ideálnych vlastnosti a teda by sme prišli o využitie niekoľkoročných skúsenosti pri vývoju tohto frameworku. A teda by sme boli pravdepodobne nútení začať od nuly. Odhliadnuc od toho, že by sme boli aj tak

ovplyvnení vývojom stráveným s frameworkom DForms. Vývoj frameworku, ktorý je časovo náročný, by pravdepodobne vyústil do menej kvalitného riešenia s menšou mierou naplnenia ideálnych vlastnosti. A teda sme sa rozhodli vylepšiť niektoré z vlastností, ktoré sú tiež zaujímavé pre web formulárový framework, ale nespadajú do množiny takzvaných ideálnych vlastností. Konkrétne sa jedna o vylepšenie čitateľnosti frameworku a dostupnosti pre používateľov s obmedzeniami, čo určite predstavuje prínos, aj keď nie v smere, ktorý by spravil framework komerčne konkurencieschopnejším.

Zoznam bibliografických odkazov

[1] Riehle, Dirk (2000), Framework Design: A Role Modeling Approach, Swiss Federal Institute of Technology

[2] Pree, W (1994), "Meta Patterns-A Means For Capturing the Essentials of Reusable Object-Oriented Design"

[3] W3C, Web Content Accessibility Guidelines (http://www.w3.org/TR/WCAG20/)

[4] Fowler, Rice, Foemmel, Hieatt, Mee (2002), Patterns of Enterprise Application Architecture, Addison Wesley

[5] John E. Hopcroft, Jeffrey D. Ullman, Formálne jazyky a automaty, Alfa SNTL 1978

[6] W3C, XSL Transformations (XSLT) Version 1.0. (http://www.w3.org/TR/xslt)

[7] Sun Micro Systems, JavaFX (http://www.sun.com/software/javafx/)

[8] W3C, XForms 1.0 (http://www.w3.org/TR/xforms/)

[9] Jack Harrington and Emily Kim (2008), Getting Started with Flex 3, O'Reilly Media

[10] Zend Technologies Inc.(2005-2009), Programmer's Reference Guide
Zend Framework (http://framework.zend.com/manual/en/)

[11] Google, Google web toolkit (http://code.google.com/webtoolkit)

[12] Mozilla, XUL - XML User Interface Language
(https://developer.mozilla.org/En/XUL)

[13] Ian Sommerville, Software Engineering

[14] Don Roberts, Ralph Johnson, Evolving Frameworks: A Pattern Language for Developing Object-Oriented Frameworks