STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST
2007/2008
OBOR 18 – INFORMATIKA

Smart Framework

Řešení pro nové webové projekty

Autoři:
Vojtěch Vít a Štěpán Šindelář

Verze 1.0

2007 / 2008
Gymnázium, Praha 6, Arabská 14
Arabská 14, 160 00, Praha 6

Hlavní město Praha

Čestné prohlášení autorů

Prohlašujeme tímto, že jsme soutěžní práci vypracovali zcela samostatně a uvedli v seznamu literatury veškerou použitou literaturu a další informační zdroje včetně internetu.

V Praze dne 17. března 2008

Vojtěch Vít	Štěpán Šindelář
_____	_____
vlastnoruční podpis autora	*vlastnoruční podpis autora*

Informační stránky projektu

Za účelem prezentace soutěžní práce byla vytvořena informační stránka o projektu, která mimo jiné obsahuje:

- Všechny připojené obrázky, schémata, ukázky apod. v plné kvalitě
- HTML Api-dokumentaci knihovny Smart Library
- **Funkční verzi ukázkové aplikace** (jednoduchý internetový obchod)
- **Videodokumentaci zobrazující průběh tvorby ukázkové prezentace**

Adresa informačních stránek:

http://smart-framework.highnet.eu

Poděkování

Děkujeme našemu spolužákovi Pavlu Volekovi za jeho pomoc při vytváření grafického návrhu a kódu ukázkové prezentace Smart Frameworku a projektové informační stránky, za vytvoření loga projektu, poskytnutí technického zázemí při prezentaci a v neposlední řadě i za duševní podporu. Zároveň se mu omlouváme za hodiny spánku, o které kvůli nám přišel.

Děkujeme také Ing. Boženě Mannové za oponentský posudek a nakonec i všem IT profesionálům, z jejichž článků jsme čerpali cenné informace o nejnovějších technologiích, trendech, knihovnách a jiných možnostech či událostech.

Anotace

Smart Framework je soubor knihoven, standardů a nástrojů zrychlujících vývoj webových projektů, především střední a větší velikosti. Obsahuje několik již existujících technologií a vlastních rozšíření; vše propojené v jednotném a standardizovaném celku. Tato dokumentace vám vysvětlí, proč bylo nutné podobný projekt vytvořit, jaké jsou současné možnosti řešící stejný problém, v čem je projekt výjimečný, co vedlo k výběru použitých technologií, jaké bylo zvoleno řešení a nakonec i jakých výsledků bylo jeho vývojem dosaženo.

Annotation

Smart Framework is a package of libraries, standards and tools accelerating web project development, especially middle to large sized. It contains a few already existing technologies and its own extensions; all interconnected in one unite and standardized way. This documentation will explain to you why it was necessary to develop such a project, what are current options solving the same problem, why is the project unique, what led us to choose used technologies, what solution was created and finally even what results were achieved by its development.

Klíčová slova

Framework, vývoj webů, programování webů, základ webu, webové technologie, knihovna, PHP, MVC, ORM, konfigurační systém

Keywords

Framework, web development, web programming, web fundamentals, web technologies, library, PHP, MVC, ORM, configuration system

Obsah

1. Úvod ... 8
   1.1. Hierarchie technologií ... 8
2. Specifikace požadavků ... 9
3. Analýza ... 10
   3.1. Výběr platformy ... 10
      3.1.1. PHP ... 10
      3.1.2. ASP.NET ... 10
      3.1.3. Java ... 10
   3.2. Nabízené frameworky pro řízení běhu aplikace ... 11
      3.2.1. Zend Framework ... 11
      3.2.2. Symfony ... 11
   3.3. Nabízené implementace ORM ... 12
      3.3.1. PHP Doctrine ... 12
      3.3.2. EZPDO ... 12
      3.3.3. Propel ... 12
   3.4. Autentizační, autorizační a konfigurační systémy ... 12
4. Návrh ... 13
   4.1. Vybrané základní knihovny ... 13
   4.2. Obsah Smart Frameworku ... 13
      4.2.1. Knihovny ... 13
      4.2.2. Nástroje ... 13
      4.2.3. Standardy kódu (coding standards) ... 13
   4.3. Knihovna Smart Library ... 13
      4.3.1. Součást SmartL_Acl ... 14
      4.3.2. Součást SmartL_Application ... 14
      4.3.3. Součást SmartL_Config ... 14
      4.3.4. Součást SmartL_DataBinder ... 14
      4.3.5. Součást SmartL_Profiler ... 14
      4.3.6. Součást SmartL_Version ... 14
      4.3.7. Ostatní součásti ... 14
   4.4. Adresářová struktura aplikace ... 14
      4.4.1. Struktura složky Application ... 15
   4.5. Systém konfigurace ... 16
   4.6. Běh aplikace ... 16
5. Řešení ... 17
   5.1. Jednotkové testy ... 17
   5.2. Api-dokumentace ... 17
   5.3. Koordinace v týmu ... 17
6. Výsledky ... 18
7. Závěr ... 18
   7.1. Shrnutí ... 18
   7.2. Použití ... 18
   7.3. Možné problémy a další kroky ... 18
8. Resumé ... 19
   8.1. Summary ... 19
9. Použitá literatura ... 20
   9.1. PHP ... 20
   9.2. C#, ASP.NET, .NET ... 20
   9.3. Java ... 20
   9.4. ORM ... 20
   9.5. Databáze ... 21
   9.6. HTML, Javascript, Ajax ... 21
   9.7. SEO ... 21
   9.8. Dokumentace ... 21
   9.9. Řízení projektu ... 21
   9.10. Obecné ... 21
10. Přílohy ... 22
   10.1. Class model diagram ... 22
   10.2. Runtime diagram ... 23

Použité zkratky a termíny

Následuje přehled zkratek a termínů použitých v textu dokumentu i s jejich krátkým vysvětlením.

.NET	Zkratka pro „Microsoft .NET Framework“, softwarovou složku, která je součástí OS MS Windows a umožňuje vývoj pro všechny verze tohoto systému (včetně „kapesních“, serverových apod.).
Apache	Zkratka pro „Apache HTTP Server”, webový server s otevřeným kódem pro Linux, BSD, Microsoft Windows a další platformy.
API	Zkratka pro „Application Programming Interface“, tj. „rozhraní pro programování aplikací“. Jde o sbírku procedur, funkcí či tříd nějaké knihovny, které může využívat programátor, který danou knihovnu využívá.
ASP.NET	Framework pro webové aplikace postavený na platformě .NET společnosti Microsoft Corp.
C#	Objektový programovací jazyk, jeden z hlavních jazyků platformy .NET.
Controller	Součást MVC; reaguje na události (typicky pocházející od uživatele) a zajišťuje změny v modelu nebo v pohledu.
Framework	Framework je softwarová struktura, která slouží jako podpora při programování a vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovnu API, návrhové vzory nebo doporučené postupy při vývoji.
GUI	Zkratka pro “Graphical User Interface”, tedy “Grafické uživatelské rozhraní”
Java	Objektový programovací jazyk od společnosti Sun Microsystems Inc. Zároveň se jedná o platformu pro vývoj programů, která je nezávislá na cílovém OS.
Knihovna	Funkční logický celek, který poskytuje služby pro programy. Většinou se jedná o sbírku procedur, funkcí a datových typů, či při objektově orientovaném přístupu o sadu tříd, uložených v jednom diskovém souboru.
Model	Součást MVC; jde o doménově specifickou reprezentaci informací, s nimiž aplikace pracuje.
MVC	Zkratka „Model-View-Controller“, softwarové architektury, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má minimální vliv na ostatní.
ORM	Zkratka „Object-Relational-Mapping“, programovací techniky pro převod dat mezi nekompatibilními typovými systémy v relačních databázích a objektově-orientovaných programovacích jazycích. Výsledným efektem je vytvoření „virtuální objektové databáze“, která může být použita v programovacím jazyce.
PHP	Zkratka pro „Hypertext PreProcessor“, programovací jazyk od spol. Zend Technologies Ltd., určený především pro programování dynamických internetových stránek.
Platforma	Označení pro nějaký typ HW architektury či SW frameworku, který umožňuje běh SW. Typickými platformami jsou např. počítačová architektura, operační systém, programovací jazyk a jeho integrované knihovny či GUI.
Smart Library	Knihovna, která je součástí a tvoří klíčovou část Smart Frameworku.
SmartL	Zkratka pro „Smart Library“.
SVN	Zkratka pro „Subversion“, nejrozšířenější implementaci VCS od společnosti CollabNet inc.

VCS	Zkratka pro „Version Control System“, systém správy více verzí jednoho balíku informací. Používá se v SW inženýrství a vývoji pro správu vytvářených dokumentů a programů, na kterých může pracovat zároveň více lidí.
View	Součást MVC; převádí data reprezentovaná modelem do podoby vhodné k interaktivní prezentaci uživateli.
Web	Označení pro aplikace internetového protokolu HTTP. Je tím myšlena soustava propojených hypertextových dokumentů. V češtině se slovo web často používá i pro označení jednotlivé soustavy dokumentů dostupných na tomtéž webovém serveru nebo na téže internetové doméně nejnižšího stupně (internetové stránce).
Zend	Zkratka názvu společnosti „Zend Technologies Ltd.“.

1. Úvod

Pokud dnes chceme vytvořit nový webový projekt většího rozměru (řekněme od rozsáhlé prezentace firmy či menšího internetového obchodu výše), musíme vytvořit jakýsi „základ“, na kterém celý projekt poběží. Knihovny nabízené spolu s dostupnými platformami (např. programovacími jazyky) sice často nabízí nějaké základní nástroje, ale ty buď nejsou vhodné pro projekty větší velikosti, nebo je stejně nutné udělat poměrně mnoho dodatečné práce, aby byly tyto nástroje použitelné ve výsledné aplikaci. Problém totiž spočívá v tom, že potřebné technologie sice existují, ale je jich mnoho a každá se soustředí na jiný „obor působnosti“. Existují frameworky a samostatné knihovny pro řízení běhu aplikace, ORM, pro autentizaci, konfiguraci apod. Všechny tyto technologie (a mnohé další) větší projekt potřebuje. Neexistuje však volně dostupný balík, který by je nabídl všechny najednou, a zůstal přitom použitelný v jakémkoli vyvíjeném projektu.

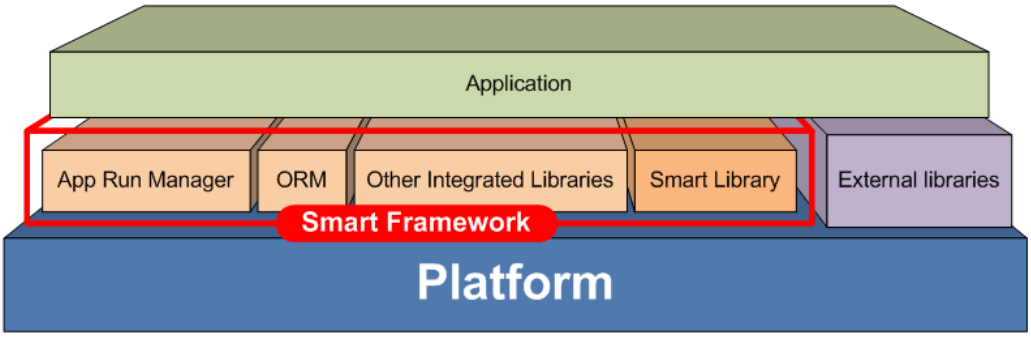
Na začátku vývoje téměř každé větší aplikace si proto její vývojový tým nejdříve vytváří jakýsi svůj „základ“, na kterém pak dále staví. Před tento úkol jsme byli postaveni i my, když jsme v polovině roku 2007 začali vyvíjet jeden takový projekt. Vzhledem k jeho povaze, která vyžadovala maximální rozšiřitelnost a tedy i zobecnění, jsme došli postupně k závěru, že nejlepším řešením tohoto problému bude vytvoření zcela samostatného, obecného balíku základních knihoven. Začali jsme proto této fázi věnovat mnohem více času, než bývá obvyklé.

Tento „základní balík“ by měl obsahovat několik již existujících knihoven a frameworků, navzájem co nejlépe provázaných a spolupracujících, jako by se jednalo o jednotný celek. Sestava těchto technologií by měla být taková, aby umožnila nízkonákladový, ale přesto kvalitní vývoj. Chtěli jsme také, aby tento balík byl co nejvíce rozšiřitelný a přizpůsobitelný. Přitom by ale stále měl obsahovat co nejvíce zabudovaných technologií, aby je později nebylo třeba dodělávat.

Za základní knihovny, které jsme do balíku chtěli zapojit, jsme považovali SW pro správu běhu aplikace, autentizaci, autorizaci, ORM, konfiguraci a také podporu pro vícejazyčnost a grafické šablony. Existuje mnoho SW, který tyto funkce nabízí. Každý více či méně kvalitní, každý v něčem specifický. V první řadě tedy pro nás bylo důležité mezi těmito technologiemi vybrat ty (pro nás) nejvhodnější, čímž se zabývá analýza a návrh.

1.1. Hierarchie technologií

Pro lepší představu o tom, jakou „vrstvu“ v hierarchii technologií použitých při vytváření webových aplikací tento projekt představuje, uvádíme následující názorné schéma:



Ze schématu je jasně vidět, že „Smart Framework“ je balík několika knihoven, a to včetně „Smart Library“, která je jakýmsi spojovníkem těch ostatních a která také obsahuje veškerou námi přidanou funkcionalitu.

Upozorňujeme, že Smart Framework není implementací CMS (Content Management System), jak by se mohlo zdát, ale leží o úroveň níže. To znamená, že aplikace typu CMS na něm může být postavena (v místě bloku „Application“), stejně jako jakákoli jiná, více specializovaná webová aplikace.

2. Specifikace požadavků

1. **Možnost rychlého nasazení**
    - Systém by mělo být možné použít bez větších dodatečných úprav.
2. **Nízké náklady na vývoj i údržbu**
    - Vývoj zahrnuje náklady na vývojové nástroje, prostředky i zdroje.
    - Údržba zahrnuje náklady na provoz a možné rozšiřování.
3. **Rozšiřitelnost, přizpůsobitelnost**
    - Jednotlivé součásti systému by měly být rozšiřitelné a nahraditelné.
    - Systém by měl být použitelný u jakéhokoli typu projektu.
4. **Dobrá podpora ze strany autorů integrovaných knihoven a frameworků**
    - Autoři integrovaných knihoven a frameworků se stanou partnery systému. Je důležité, aby svůj SW i dále rozvíjeli a poskytovali mu podporu.
5. **Integrovaný ORM (Object-Relational-Mapper)**
    - Zabudovaný ORM umožní snadnou údržbu databázových dat.
    - Systém by na jeho konkrétní implementaci však neměl být zcela závislý.
6. **Integrovaný systém autentizace a autorizace**
    - Systém by měl zprostředkovat autentizaci a autorizaci uživatele.
    - Procedury, které k tomu použije, mu budou ale dány konkrétní aplikací.
7. **Konfigurační systém umožňující nastavení aplikace na všech úrovních**
    - V zájmu rychlejšího nasazení systému je vhodné, aby co nejvíce úprav potřebných k přizpůsobení systému konkrétnímu projektu bylo možné provést pouhou změnou konfigurace.
8. **Systém ošetřování výjimek, špatných url apod.**
    - Systém by měl počítat s možností vzniku chyb v programu a umět provést příslušné kroky k jejich ošetření.
9. **Zabudovaná podpora vícejazyčnosti, nadnárodnosti**
    - Všechny součásti systému by měly implicitně umožňovat vývoj nadnárodního webu.
10. **Oddělená prezentační logika od programu**
    - Oddělení prezentační logiky umožňuje změnu grafické podoby stránek bez nutnosti úpravy programu.
11. **Podpora všech multimediálních a jiných soudobých technologií (Web 2.0)**
    - Podpora webových technologií se server-side účastí. Např. AJAX, optimalizace pro vyhledávače, RSS apod.
12. **Dobrá podpora ze strany hostingů, uživatelských komunit a dalších možných partnerů**
    - Použité technologie a platforma by měly být dostatečně rozšířeny, aby bylo zajištěno jejich snadné nasazení.

3. Analýza

Při hledání řešení našeho problému jsme byli postaveni před několik základních otázek. První z nich byla, jakou platformu (programovací jazyk) pro vývoj zvolit. Od té se totiž odvíjí nejen vývoj onoho „základního frameworku“, ale i všech na něm postavených webových aplikací. Museli jsme tedy provést jakýsi průzkum současných technologií, abychom zjistili, která z nabízených možností vyhovuje nejlépe našim požadavkům.

Před podobné rozhodování jsme byli postaveni i v dalších fázích analýzy, kdy jsme porovnávali jednotlivé frameworky a knihovny zvolené platformy. Zajímaly nás především implementace ORM, autorizační, autentizační a konfigurační systémy.

3.1. Výběr platformy

Webové projekty lze vyvíjet téměř v libovolném programovacím jazyce. Pouze některé ale mají vlastní uzpůsobené řešení k těmto účelům a jen pár z nich je skutečně ve větší míře používáno. Mezi ně patří především jazyky (či technologie) PHP, Java, ASP.NET, Ruby on Rails a Python. Vzhledem k našemu požadavku na podporu hostingů a komunit jsme svůj zájem omezili pouze na tyto tři platformy:

3.1.1. PHP

PHP je nejpoužívanějším programovacím jazykem pro vytváření webů. Od své 5. verze podporuje objektově orientované programování, a to především díky jeho převzetí spol. Zend Technologies Ltd., která pro něj vyvinula nové jádro zvané „Zend Engine“. Vzhledem k faktu, že PHP nebylo pro objekty původně navrženo, jeho syntaxe je poněkud nesystematická a mix objektů s procedurálním kódem mívá více záporů než pozitiv.

Do budoucna má ale PHP svou „objektovost“ prohlubovat. Zend to dokazuje i vydáním vlastního, plně objektového frameworku, který by měly weby PHP „nové generace“ využívat. Tento framework používá architekturu MVC, která je dobře rozšiřitelná a testovatelná.

PHP je interpretovaný jazyk, a tak je vůči svým konkurentům teoreticky o něco pomalejší. Na druhou stranu ale existují různé optimalizátory, které tento problém celkem značně tlumí. Příkladem je např. komerční Zend Optimizer.

3.1.2. ASP.NET

ASP.NET je framework od Microsoftu určený pro vývoj webů, postavený na platformě .NET. Nespornou výhodou této platformy je použití pokročilých technologií a plná podpora komerční společnosti. Jazyk C# patří v současné době ke špičce objektových jazyků.

Bohužel samotné ASP.NET využívá systému WebForms, připomínajícímu spíše programování desktopových aplikací než webových projektů (MS tomuto návrhu říká MVP, tedy Model-View-Presenter). Tento systém není dobře rozšiřitelný, testovatelný a ani optimalizovatelný. Bez patřičných úprav se proto hodí spíše jen pro menší webové projekty.

Pokud na druhou stranu ale přihlédneme k chystaným novým technologiím Microsoftu, jako je ASP.NET MVC, Silverlight 2 či LINQ to Entities, tak ASP.NET v budoucnu pravděpodobně získá ve většině oblastí dokonce náskok před svými rivaly. V současné době ale spíše ztrácí.

Nevýhodou ASP.NET jsou také nepřímé náklady na jeho vývoj, způsobené podporou jediného serverového OS, existencí jediného kvalitního nástroje pro vývoj a relativně malé nabídky hostingů.

3.1.3. Java

Java patří k nejrozšířenějším programovacím jazykům na světě. I u webů je hojně využívána, avšak v celkovém počtu vyvíjených projektů leží hluboko pod PHP. Frameworky pro weby, které pro ni existují (např. Spring Framework) nabízí MVC strukturu (podobně jako Zend Framework u PHP).

Java je objektový jazyk, stejně jako její největší konkurent C#. Vznikla ale dříve, díky čemuž se stala tak rozšířenou. Zároveň ale postrádá některá drobná vylepšení, kterými byl obohacen později vzniklý C#. Její vývoj probíhá také mnohem pomaleji než např. u .NET frameworku, a to přesto že je zaštítěna společností Sun Microsystems Inc.

Nespornou výhodou Javy je, že patří k nízkonákladovým řešením. Pro její vývoj i nasazení je možné najít kvalitní open-source řešení, díky čemuž se v této oblasti může srovnávat např. s již zmíněným PHP. To samé pak platí pro její podporu u hostingů apod.

3.2. Nabízené frameworky pro řízení běhu aplikace

Po přijetí požadavku uživatele na zobrazení nějaké stránky musí aplikace vyvolat všechny své součásti, kterých se tento požadavek týká. Způsob, jakým tyto součásti hledá a jak jimi prochází, je otázkou architektury. Frameworky pro řízení běhu aplikace představují implementaci těchto architektur.

V současné době nejpopulárnější architekturou tohoto typu je MVC (zkratka Mode-View-Controller), která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má minimální vliv na ostatní.

Poznámka: Z důvodu výběru platformy PHP (vysvětleno v návrhu) již frameworky a knihovny ostatních platforem v analýze neuvádíme.

3.2.1. Zend Framework

Framework od společnosti Zend Technologies Ltd. s otevřeným kódem. Je vyvíjen od roku 2005 a jeho první „release“ verze byla vydána 30. června 2007. Obsahuje mnoho na sobě nezávislých knihoven pro pomoc při vývoji, které do jisté míry nahrazují a rozšiřují zabudované knihovny jazyka PHP. Ty totiž, na rozdíl od Zend Frameworku, nejsou ve většině případů programovány objektově.

Součástí Zend Frameworku je i část „Zend_Controller“, která se stará o řízení běhu aplikace. Implementuje přitom architekrutu MVC.

Zend Framework je v současnosti velice rychle se rozšiřujícím a rozvíjejícím frameworkem pro platformu PHP, a to zejména díky záštitě spol. Zend Technologies, která zaručuje jistou podporu do budoucna. Zend mimo tento framework navíc spravuje a vyvíjí i samotný jazyk PHP, což v důsledku znamená, že jej může přizpůsobovat potřebám svého frameworku.

3.2.2. Symfony

Framework Symfony byl vytvořen společností Sensio (dřívější název byl Sensio Framework). Jako open source byl uvolněn později, v roce 2005. Vznikl tedy o několik let dříve, než výše zmiňovaný Zend Framework. Tento framework byl od základů postaven pro rozsáhlé webové aplikace. Jeho důležitým specifikem je velice pokročilý systém konfigurace, který umožňuje nastavitelnost na všech úrovních aplikace, a to do té míry, že s ní lze někdy nahradit i psaný program. Někteří kritici dokonce tvrdí, že „vytvářet projekt v Symfony není o programování, ale o psaní konfigurace“. Tento aspekt je ale nutno považovat spíše za velké pozitivum, protože svědčí o dobře navržené struktuře frameworku.

Symfony, stejně jako Zend Framework, implementuje architekturu MVC. Na rozdíl od ZF ale poskytuje lepší zázemí pro vytváření modelů. Pro Symfony jsou totiž speciálně uzpůsobeny mnohé PHP ORM knihovny (např. Propel, který Symfony ve své základní verzi dokonce obsahuje), což o něco zrychluje vývoj.

Nevýhodou Symfony je její zastaralost a nyní již nepříliš jasná podpora. V současnosti je tedy pomalu vytlačován Zend Frameworkem.

3.3. Nabízené implementace ORM

Aby se zjednodušilo vytváření „modelů“, je dobré do Smart Frameworku integrovat některý ORM systém, který bude implicitně podporován (podobně jako Propel u Symfony, viz níže). V PHP ale neexistuje mnoho kvalitních implementací tohoto SW, což je dáno zejména poměrně malým zájmem. Níže tedy uvádíme aspoň tři, které jsou podle nás nejrozšířenější či nejpropracovanější.

Poznámka: Z důvodu výběru platformy PHP (vysvětleno v návrhu) již frameworky a knihovny ostatních platforem v analýze neuvádíme.

3.3.1. PHP Doctrine

Doctrine je open-source projekt vytvářený komunitou od roku 2006. Jedná se o komplexní ORM knihovnu, jejíž cílem je přiblížit se možnostem Javovského Hibernate či .NET NHibernate. V současnosti již obsahuje poměrně mnoho pokročilých funkcí, jako je vlastní dotazovací jazyk DQL či „šablony pro chování modelů“. Díky systému tzv. „listeners“ (posluchačů) je struktura Doctrine navíc hluboce rozšiřitelná, a to i přes svou komplexnost.

Hlavní nevýhodou této knihovny je, že je stále ve vývoji. Současná verze 0.10 je sice považována za stabilní, ale přesto zatím obsahuje chyby. Nejasná je také podpora autorů do budoucna, protože tento projekt nezaštiťuje žádná komerční společnost. Na druhou stranu jsou autoři Doctrine zatím velice aktivní a jeho komunita se rozrůstá, což je způsobeno i tím, že jiné tak propracované ORM řešení v PHP zatím neexistuje.

3.3.2. EZPDO

EZPDO je malá ORM knihovna vzniklá v roce 2007. Její předností a zároveň negativem je jednoduchost. Protože obsahuje pouze základní ORM funkcionalitu, potenciál ORM tak zůstává svým způsobem nevyužit.

3.3.3. Propel

Propel je ORM řešení vyvíjené od roku 2003. I díky tomu je v současné době asi nejrozšířenějším PHP ORM řešením. Vzhledem k malému zájmu o tyto systémy v PHP komunitě se ale nejedná o výraznou podporu. Příznivce nalézá především v komunitě kolem frameworku Symfony, protože pro něj je Propel v podstatě vyvíjen (je dokonce jeho součástí).

Jedná se o poměrně propracovaný ORM systém, ovládající většinu pokročilejších funkcí, jako je ošetření vztahů mezi záznamy 1:N, M:N či dědění. Pro svou funkci hojně využívá specifik objektového programování PHP, jako jsou například „magické metody“. Ty umožňují zavolat neexistující metodu objektu, která se pak zpracuje podle jejího názvu ve speciální metodě „__call()“. Je tak možné volat metody modelu jako např.: „getBookReaderRefsJoinReader()“.

Nevýhodou Propelu je podobně jako u Symfony jeho zastaralost. Používání magických metod jako způsobu získávání dat je navíc velice kontroverzní. Na druhou stranu se ale jedná o léty prověřenou, stabilní knihovnu.

3.4. Autentizační, autorizační a konfigurační systémy

Autentizačních a autorizačních řešení existuje mnoho. Většinou jsou součástí přímo dané platformy nebo jejích rozšíření. Smart Framework by měl podporovat jakýkoli systém, který na něm vyvíjená aplikace zrovna používá (tedy nikoli on sám). Jeho úkolem pouze je, aby tento proces měl zajištěné místo v běhu aplikace a aby zapojení nového systému bylo co nejjednodušší.

Konfigurační systém by měl umožňovat nastavitelnost na všech úrovních. To znamená jak nastavení např. databázových připojení na úrovni celé aplikace, tak definování generovaných webových formulářů na úrovni controlleru (viz MVC). Důležitým aspektem je i možnost „dědění“ konfigurace a tedy vytváření jakýchsi „konfiguračních stromů“.

4. Návrh

4.1. Vybrané základní knihovny

Na základě poznatků z analýzy dostupných technologií jsme se rozhodli použít jako základ Smart Frameworku platformu PHP, a to především kvůli její rozšířenosti, dostupnosti, nízkonákladovosti a široké podpoře.

Jako framework pro řízení běhu aplikace jsme vybrali Zend Framework. Důvodem je oficiální podpora tvůrce jazyka PHP, tedy spol. Zend Technologies Ltd., a také jeho rychlý vývoj a celková perspektivita.

Jako podporované ORM řešení byl vybrán Doctrine. Zde byla volba nejproblematičtější, ale přesvědčila nás jeho komplexnost a především stále aktivní a rychlý vývoj (na rozdíl od dalších dvou řešení). Navíc má Doctrine stejné standardy psaní kódu jako Zend Framework, což přispívá k celkové konzistenci Smart Frameworku.

4.2. Obsah Smart Frameworku

4.2.1. Knihovny

- Zend Framework
- Doctrine
- Smart Library

4.2.2. Nástroje

Součástí Smart Frameworku jsou také tyto vlastní nástroje:

- Nástroj pro převod yml zápisu Doctrine modelů na jejich php variantu

4.2.3. Standardy kódu (coding standards)

Standardy psaní kódu jsou veskrze přejaté od Zend Frameworku (a vlastně i Doctrine). Jedním z důležitých aspektů tohoto standardu je způsob zápisu jmen tříd. Název všech tříd se totiž dá jednoduše přeložit na cestu k jejich umístění, což výrazně pomáhá při jejich načítání (které PHP vyžaduje). Např. třída s názvem „SmartL_Application_Config“ musí být k nalezení pod cestou „SmartL/Application/Config.php“.

4.3. Knihovna Smart Library

Knihovna Smart Library obsahuje veškerá naše vlastní rozšíření, jež jsou součástí Smart Frameworku. Mimo to ale obsahuje také všechny „spojky“, které nutí ostatní knihovny frameworku k jednotě a spolupráci.

Struktura knihovny vychází z konvencí Zend Frameworku (viz. „Standardy kódu“ výše). Následuje výpis všech hlavních komponent Smart Library:

- **SmartL_Acl**
- **SmartL_Application**
  - Config
  - Login
  - Plugin
- **SmartL_Config**
  - Storage
- **SmartL_DataBinder**
- **SmartL_Doctrine**
  - Template
- **SmartL_Profiler**
- **SmartL_Version**
- **SmartL_Zend**
  - Auth
  - Controller
  - Form
  - Validate
  - View

4.3.1. Součást SmartL_Acl

Obsahuje pouze pomocné funkce pro autorizaci uživatele.

4.3.2. Součást SmartL_Application

Mnoho součástí Smart Frameworku tvoří pouze samostatné celky. Jeho účelem je ale takové celky spojovat. K tomu mu slouží především součást knihovny zvaná „SmartL_Application“, která se stará o celkový běh aplikace. Tato třída je v různých fázích běhu aplikace volána, aby provedla své „správcovské“ úkony. Mezi ně patří např. načtení konfigurace a její správa, načtení databázových připojení, autentizace a autorizace uživatele, správa nastavení aktuální šablony, jazyka aj. To vše by měla zprostředkovat tak, aby nenutila zbytku aplikace své „návyky“, k čemuž jí slouží adaptéry.

4.3.3. Součást SmartL_Config

Samostatný konfigurační systém se schopnostmi, které Smart Framework vyžaduje, jsme v PHP komunitě bohužel nenašli, a tak je nutné jej implementovat jako jedno z vlastních rozšíření.

Tato součást Smart Library obsahuje v podstatě autonomní SW pro načtení konfigurace. Jeho hlavní výhodou je zabudovaný systém „dědění“, který umožňuje vytvořit dědičné konfigurační struktury. Více je o tomto pojednáno v sekci „Systém konfigurace“.

4.3.4. Součást SmartL_DataBinder

Zprostředkovává jednoduché provázání modelů s prvky formulářů. Svou funkcí je trochu podobný „DataBindingu“ známému z ASP.NET.

4.3.5. Součást SmartL_Profiler

Umožňuje sledování (a logování) výkonu aplikace, prováděných databázových dotazů a jiných náročných operací. Jedná se o užitečný nástroj při optimalizaci aplikace.

4.3.6. Součást SmartL_Version

Obsahuje informace o verzi Smart Library a verzích integrovaných knihoven, pro které je použití Smart Library zaručeně stabilní.

4.3.7. Ostatní součásti

Smart Library obsahuje i další, pomocné součásti, např. na zabudování podpory vícejazyčnosti do základních knihoven (tedy Zend Frameworku a Doctrine), pro podporu šablonového systému apod. Díky nim je program ještě více provázaný a není přitom narušena konzistence žádné jednotlivé součásti.

4.4. Adresářová struktura aplikace

Po výběru základních knihoven jsme hledali návrh ideální adresářové struktury, která by svým uspořádáním pokryla potřeby všech myslitelných scénářů. Dospěli jsme k závěru, že nejlepší cestou je vytvořit tři základní adresáře:

1) ***Application*** Uchovává veškerou logiku týkající se „konkrétní“ aplikace.
2) ***Library*** Uchovává veškeré externí knihovny, jež aplikace využívá (včetně Smart Library).
3) ***Public*** Jediná složka, která je přístupná http požadavkem přímo (ostatní jsou chráněné).

4.4.1. Struktura složky Application

Vychází z architektury MVC a dělí se do několika „větví“:

- Větev controllerů
- Větev modelů
- Větev pohledů
- Větev konfigurace
- Větev překladů
- Větev dočasných souborů
- Větev úložišť

Každá větev představuje víceúrovňový adresářový strom, který se člení podle své vnitřní logiky (např. controllery a pohledy podle modulu, modely zase podle databází). Doporučené rozdělení je následující:

```
Application/
   Db/                  // Větev modelů
      {dbName}/
         Record/
            {dbRecords}
         Table/
            {dbTables}
   Config/              // Větev konfigurace
      Config.xml
      {moduleName}/
         Config.xml
         Controller/
            {controllerConfigs}
            {templateName}/
               {controllerConfigs}
   {moduleName}/
      Temporary/        // Větev dočasných souborů
      Storage/          // Větev úložišť
      Controller/       // Větev controllerů
         {controllers}
      View/             // Větev pohledů
         {templateName}
         Script/
            {layoutsEtc}
            {controllerName}/
               {viewScript}
   Translations/        // Větev překladů
      {appTranslations}
      {moduleName}/
         {moduleTranslations}
         {controllerName}/
            {controllerTranslations}
```

4.5. Systém konfigurace

Dalším nutným návrhem je struktura konfiguračního systému. Pro opravdu konfigurovatelnou aplikaci je důležitá možnost dědění nastavení z vyšších úrovní na nižší tak, jak se postupně během běhu aplikace upřesňuje uživatelův požadavek.

Hlavní charakteristikou tohoto systému by měla být nastavitelnost způsobu „dědění“, možnost vytvářet dynamické seznamy (např. seznam podporovaných jazyků, který je možné na nižších úrovních upravit odebráním či přidáním dalšího jazyka) a zápis ve formátu XML.

Vzhledem k povaze MVC struktury, vnitřní struktury Zend Frameworku a podpoře systému šablon, je konfigurace dělena do 2-4 stupňů:

1. ***Konfigurace Aplikace***
   Obsahuje globální konfiguraci pro celou aplikaci (napříč moduly).
2. ***Konfigurace Modulu***
   Osahuje konfiguraci logického celku aplikace zvaného „modul“. Tato konfigurace dědí nastavení od „Konfigurace Aplikace“.
3. ***Konfigurace Controlleru a Akcí***
   Obsahuje konfiguraci nějakého controlleru a jeho akcí.
4. ***Konfigurace Controlleru a Akcí specifická pro šablonu***
   Obsahuje upřesnění konfigurace controlleru při zobrazení jistou šablonou (dědí od „Konfigurace Controlleru a Akcí“).

Pro lepší představu uvádíme názorné schéma:

```
Application config
  |
  |_  Module 1 config
  |
  |_  Module 2 config
```

```
Controller config
  |
  |_ Template 1 specific controller config
  |
  |_ Template 2 specific controller config
```

4.6. Běh aplikace

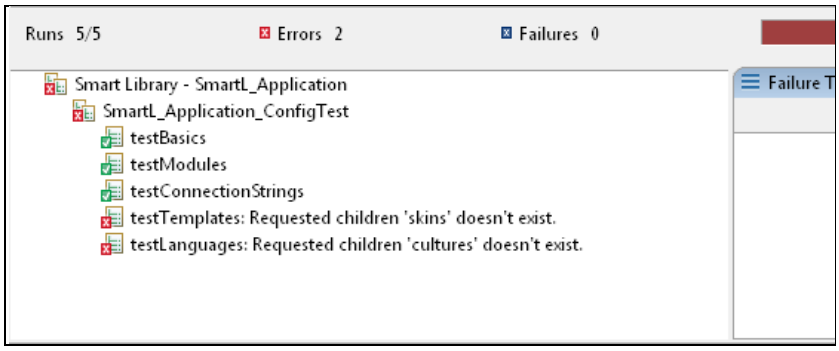
Běh aplikace se dělí do 10 hlavních fází a 18 podfází. Tyto fáze jsou dány jednak použitím architektury MVC a Zend Frameworku, ale i etapami zpracování požadavku třídou SmartL_Application (viz součásti knihovny Smart Library).

Běh aplikace a fáze, kterými prochází, je znázorněn na schématu v příloze číslo 2.

5. Řešení

5.1. Jednotkové testy

Při vytváření knihovny Smart Library jsme se řídili metodou Test Driven Development (TDD) a vytvářeli tak ke každé vyvíjené součásti „jednotkové testy“ za použití testovacího frameworku PHPUnit. Díky nim jsme našli mnoho skrytých závad aplikace a problémů při vytváření nových funkcí.



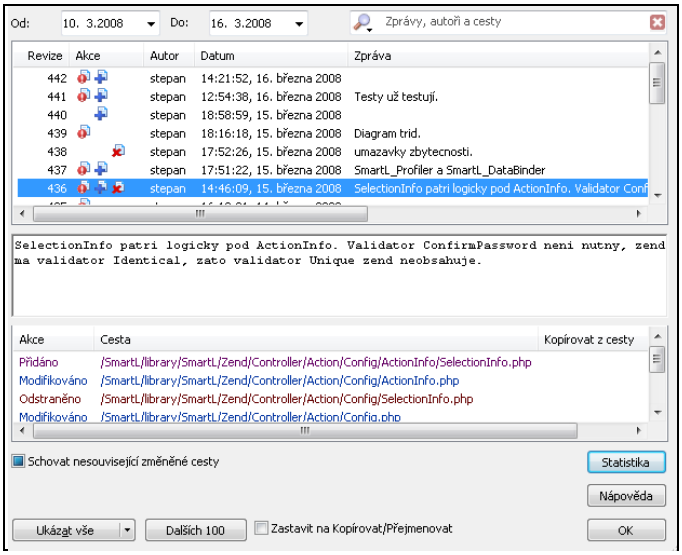
5.2. Api-dokumentace

Veškerý vytvářený program byl průběžně dokumentován pomocí PHP anotací. Na jejich základě je možné kdykoli vygenerovat kompletní api-dokumentaci celého programu. Tyto anotace navíc používají některé vyspělé PHP editory jako určitou náhradu za „silné typování“ proměnných. Nabízejí tak správné nabídky tam, kde by z programu nemohly zjistit pravý typ proměnné.

```
254     /**
255      * Initializes application configuration, front controller, paths etc.
256      *
257      * @param string $basePath   Path to application base dir
258      * @param string $configPath Relative path to configurations dir from
259      *                           application base dir
260      * @param SmartL_Application_Login_Interface $login
261      * @return SmartL_Application
262      */
263     public function init ($basePath, $configPath, SmartL_Application_Login_In
264     {
265         $this->_basePath    = $basePath;
266         $this->_configPath = $configPath;
267         $this->_login       = $login;
```

5.3. Koordinace v týmu

Pro nekonfliktní vývoj v rámci týmu byla použita technologie SVN (SubVersioN), která patří mezi současnou špičku VCS (version control system) řešení nabízených zdarma.



6. Výsledky

Smart Framework jsme poprvé použili k vytvoření ukázkových webových stránek pro SOČ. Jedná se o malý internetový obchod s podporou vícejazyčnosti, grafických šablon a skinů. Demonstruje většinu funkcí obsažených v tomto frameworku, ale přitom se jedná o skutečně funkční aplikaci.

Vytvoření tohoto obchodu trvalo dvěma programátorům a jednomu grafikovi zhruba 5 dní.

7. Závěr

7.1. Shrnutí

Původním záměrem Smart Frameworku bylo vytvořit jakýsi „základní balíček“ knihoven a frameworků, který by všechny obsažené technologie propojil a výrazně tak zjednodušil začátky vývoje webových projektů (především těch větších). Chtěli jsme přitom zvolit takové technologie, které by umožnily nízkonákladový, ale přesto kvalitní vývoj.

Tento záměr se podle nás vydařil. Kombinace jazyka PHP, Zend Frameworku a Doctrine ORM tvoří velmi dobrý základ pro tvorbu jakéhokoli rozsáhlejšího webového projektu. Jejich propojení pomocí knihovny Smart Library umožňuje rychlé nasazení celého balíku do praxe. Není přitom třeba provádět téměř žádné další úpravy. A když už, většina potřebných změn se dá provést pouhou úpravou konfigurace.

Vytvoření ukázkové aplikace pro SOČ naše zdání pouze upevnilo. Internetový obchod, byť velmi primitivní, byl od základu, bez jakýchkoli předpřipravených „polotovarů“, vytvořen za 5 dní, což považujeme za skvělý výsledek.

7.2. Použití

Smart Framework může pomoci všem těm, kteří chtějí za málo času a málo peněz vytvořit velké projekty. Jsou to typicky začínající malé firmy a také podnikající studenti.

7.3. Možné problémy a další kroky

Současná verze Smart Frameworku se zatím nedá považovat za zcela dokončenou. Hlavní proklamované funkce sice již obsahuje, ale zatím nebyl dostatek času na odladění veškerých chyb a nedostatků v návrhu. Pro použití v produkční sféře ho tedy zatím nedoporučujeme.

Všechny tyto problémy ale průběžně evidujeme a pracujeme na jejich nápravě, protože právě tento framework chceme použít v některých svých budoucích projektech. Jeho vývoj proto bude i nadále pokračovat a bude spočívat především ve sledování chování navržené struktury a jejího postupného doladění.

8. Resumé

Při vytváření webových projektů se jejich autoři setkávají s celou řadou opakujících se problémů. Jedním z nich je výběr vhodné platformy a základních technologií. Dalším je ale i jejich samotné nasazení – jednotlivé vybrané technologie sice mohou být ve svém oboru vynikající, avšak zásadním nedostatkem bývá jejich (někdy i vnitřní) nepropojenost. Autoři projektu tak musí ještě před začátkem vývoje samotného webu vynaložit nemalé úsilí, aby donutili tyto technologie ke spolupráci. Pokud by existoval nějaký hotový balík SW, který by umožnil tuto práci co nejvíce usnadnit či úplně přeskočit, autoři by docílili nezanedbatelné úspory na čase.

Takovým řešením je „Smart Framework“. Jedná se o soubor několika základních technologií, které jsou mezi sebou hluboce provázány v jednotném systému. K tomu navíc obsahuje vlastní, nadstandardní funkcionalitu, vhodnou zejména pro projekty středních a větších rozměrů. Celý systém je navržen tak, aby umožnil co největší nastavitelnost, a tedy i co nejsnazší a nejrychlejší nasazení.

Oproti některým jiným řešením má Smart Framework provázanou vnitřní strukturu při zachování velké míry přizpůsobitelnosti a rozšiřitelnosti. Jednotlivé součásti Smart Frameworku byly navíc vybrány tak, aby byl vývoj co možná nejlevnější; přitom ale používá aktuální technologie a je postaven na objektově orientovaném programování.

Použitím Smart Frameworku je možné významně omezit ztrátu času vzniklou vyvíjením „základu projektu“. Nabízí hotové řešení, připravené k rozšíření a přizpůsobení konkrétním potřebám. Vzhledem k výběru nízkonákladových technologií tak otevírá dveře především takovým novým webovým projektům, jejichž rozpočet vyžaduje co největší úspory. To je typické např. u začínajících společností a podnikajících studentů.

8.1. Summary

When creating new web projects, their authors meet with many regularly repeating problems. One of them is choosing a suitable platform and fundamental technologies. Another one is their usage – even if the chosen technologies are excellent in their category, the problem is their minimal interconnectivity (sometimes even within one technology). Project authors must then spend significant amount of their effort to make them cooperate. If there was some ready-to-use SW package which would allow maximally simplifying or completely shifting this work out, the authors would reach considerable time savings.

The solution is “Smart Framework”. It is a package of a few fundamental technologies that are deeply interconnected. Additionally, it contains its own above standards functionality, helpful especially for middle to large sized projects. The whole system is designed to be widely configurable and thus even easily and quickly usable.

In comparison to some other solutions, Smart Framework has interconnected internal structure while preserving good adaptability and extensibility. In addition, all parts of Smart Framework were chosen to do the development as cheap as possible; but it still uses up-to-date technologies and it is based on object-oriented programming.

Using Smart Framework may significantly reduce the time waste that arises from developing the “project fundamentals”. It offers a complete solution prepared for extending and adapting to particular needs. The selection of low-cost technologies opens doors especially to those new web projects that need to keep development cost at the bottom. That is typical, for example, for small starting companies and students being in business.

9. Použitá literatura

Uvedená literatura se nemusí přímo shodovat se zaměřením projektu, avšak znalost technologií a postupů, které popisuje, nám přispěla k utvoření lepšího přehledu o aktuální nabídce na trhu. Díky její znalosti jsme mohli objektivněji porovnat konkurenční technologie při výběru použité platformy, knihoven apod.

Nejčastějším zdrojem informací byl internet. Důvod je nasnadě – pro nejaktuálnější technologie neexistují žádné knižní publikace. Bohužel ale není možné zachytit všechny zdroje, ze kterých jsme čerpali, a tak uvádíme aspoň ty nejpoužívanější.

9.1. PHP

CASTAGNETTO, Jesus. *PHP Programujeme profesionálně*. 1. vyd. Praha: Computer Press, 2002. 656 s. ISBN 80-7226-310-2.

KOSEK, Jiří. *PHP – Tvorba interaktivních webových aplikací: Podrobný průvodce*. 1. vyd. Praha: Grada, 1998. 492 s. ISBN 80-7169-373-1.

MROŽEK, Jakub. *Weblog o Zend Frameworku* [online]. c2005. URL: <http://weblog.ronnieweb.net>.

Symfony Web PHP Framework [online]. c2005. URL: <http://www.symfony-project.org>.

VÁVRA, Vlasta. *Vlastův blog* [online]. c2008. URL: <http://www.vavru.cz>.

Zend Technologies Ltd**.** *Zend Framework Official Site* [online]. c2006. URL: <http://framework.zend.com>.

9.2. C#, ASP.NET, .NET

VIRIUS, Miroslav. *C# pro zelenáče*. Praha: Neocortex, 2002. 254 s. ISBN 80-86330-11-7.

TROELSEN, Andrew. *C# a .NET 2.0 profesionálně*. 1. vyd. Brno: ZONER software, 2006. 1197 s. ISBN 80-86815-42-0.

MACDONALD, Matthew. *ASP.NET 2.0 a C# profesionálně – tvorba dynamických stránek PROFESIONÁLNĚ*. 1. vyd. Brno: ZONER software, 2006. 1376 s. ISBN 80-86815-38-2.

CodePlex – Open Source Project Community [online]. c2006. URL: <http://www.codeplex.com>.

GUTHRIE, Scott. *ScottGu's blog* [online]. c2003. URL: <http://weblogs.asp.net/scottgu>.

Microsoft Developer Network [online]. c2008. URL: <http://msdn.microsoft.com>.

The Official Microsoft ASP.NET Site [online]. c2007. URL: <http://www.asp.net>.

VALÁŠEK, Michal. *ASPNET.CZ – in technology we trust* [online]. c2000. URL: <http://www.aspnet.cz>.

9.3. Java

Java portál [online]. c2008. URL: <http://www.java.cz>.

VÁVRA, Vlasta. *Vlastův blog* [online]. c2008. URL: <http://www.vavru.cz>.

9.4. ORM

Doctrine – Open Source PHP ORM [online]. c2007. URL: <http://www.phpdoctrine.org>.

EZPDO Official Site [online]. c2004. URL: <http://www.ezpdo.net>.

Hibernate – Relational Persistence for Java and .NET [online]. c2006. URL: <http://www.hibernate.org>.

LINQ to Entities [online]. c2008. URL: <http://msdn2.microsoft.com/en-us/library/bb386964.aspx>.

Propel – smart, easy object persistence [online]. c2006. URL: <http://propel.phpdb.org>.

9.5. Databáze

FOX, Dan. *Naučte se ADO.NET za 21 dní*. Praha: Computer Press, 2002. 258 s. ISBN 80-7226-772-8.

MySQL – The world's most popular open source database [online]. c1995. URL: <http://www.mysql.com>.

PostgreSQL – The world's most advanced open source database [online]. c1996. URL: <http://www.postgresql.org>.

9.6. HTML, Javascript, Ajax

ASLESON, Ryan. *AJAX - Vytváříme vysoce interaktivní webové aplikace*. 1. vyd. Brno: Computer Press, 2006. 269 s. ISBN 80-251-1285-3.

BROŽA, Petr. *Programování WWW stránek pro úplné začátečníky*. 1. vyd. Praha: Computer Press, 2000. 153 s. ISBN 80-7226-421-4.

PÍSEK, Slavoj. *Javascript – efektivní nástroj oživení WWW stránek*. Praha: Grada, 2001. 232 s. ISBN 80-247-0014-X.

9.7. SEO

SMIČKA, Radim. *Optimalizace pro vyhledávače*. Dubany: Knihkupectví Jasmínka. 123 s.

SearchEngines.com [online]. c2007. URL: <http://www.searchengines.com>.

9.8. Dokumentace

PETRUŽELKA, Jiří. *Jak psát bakalářskou práci*. Ostrava, 2006. 33 s. Ročníkový projekt na Vysoké škole Báňské – Technická univerzita.

9.9. Řízení projektu

GUCKENHEIMER, Sam. *Efektivní softwarové projekty*. 1. vyd. Brno: Zoner Press, 2007. 255 s. ISBN 978-80-86815-32-6.

Subversion Official Site [online]. c2006. URL: <http://subversion.tigris.org>.

9.10. Obecné

Zdroje uvedené v této kategorii byly použity pro zjištění informací týkajících se více než jednoho určitého oboru. Proto jsou zde uvedeny zvlášť.

Interval – Webdesign a e-komerce denně [online]. c1999. URL: <http://www.interval.cz>.

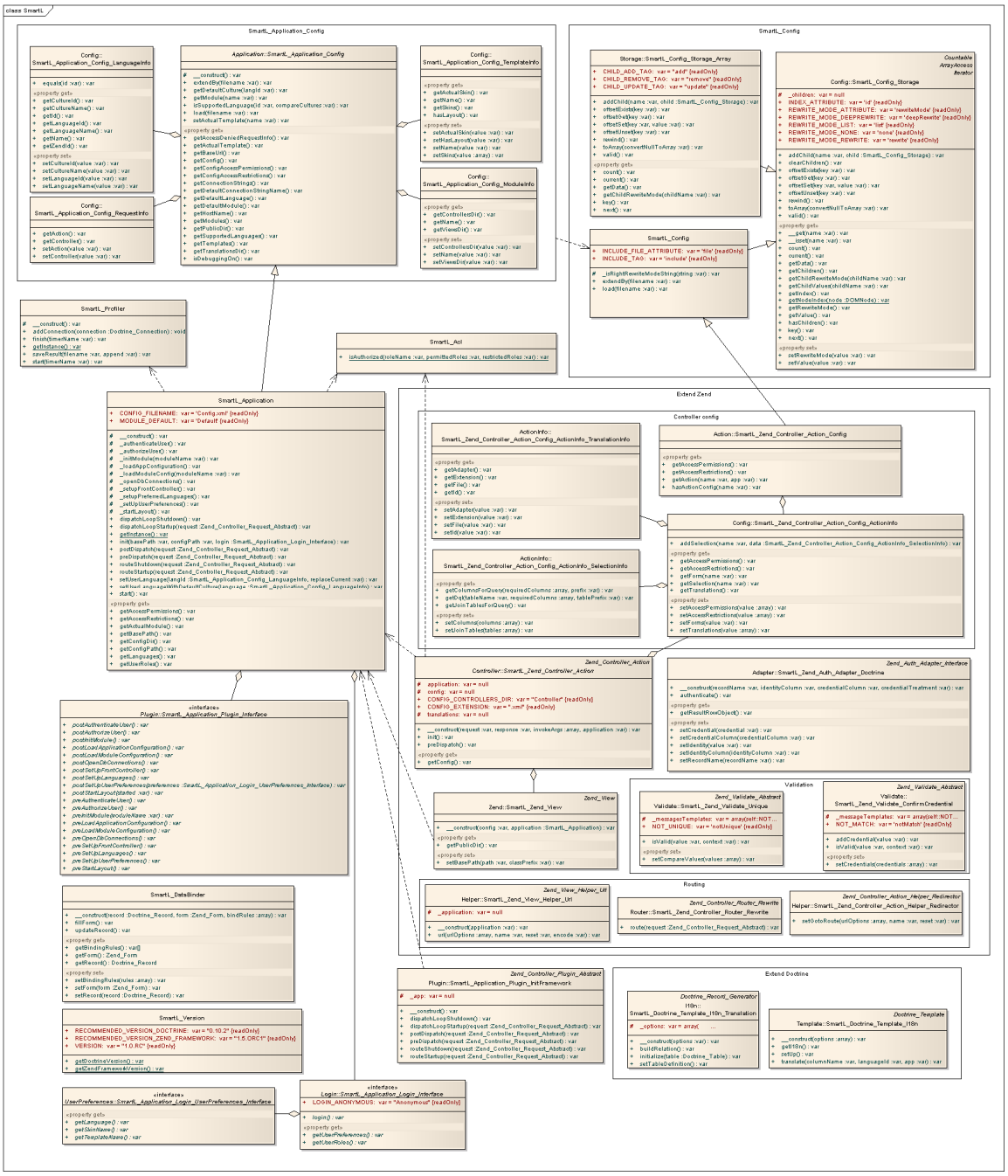
Wikipedia – Otevřená encyklopedie [online]. c2002. URL: <http://cs.wikipedia.org>.

Wikipedia – The free encyclopedia [online]. c2001. URL: <http://en.wikipedia.org>.

10. Přílohy

10.1. Class model diagram

Toto schéma znázorňuje vztahy mezi jednotlivými třídami knihovny Smart Library a ukazuje veřejné metody, pomocí kterých se s nimi dá komunikovat (API).



10.2. Runtime diagram

Toto schéma znázorňuje stavy a fáze zpracování, kterými aplikace při svém běhu prochází (od požadavku až po odpověď). Modré rámečky označují základní fáze vycházející ze struktury Zend Frameworku, oranžové zase fáze vycházející ze Smart Frameworku (především SmartL_Application).