Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Diplomová práce

Testování aplikace Škola OnLine s využitím Visual Studia

Plzeň, 2012 Václav Javorský

Zadání

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne …… Podpis: ……

Poděkování

Rád bych poděkoval panu Ing. Romanu Moučkovi, Ph.D. za vedení práce. Dále děkuji zaměstnancům Plzeňské pobočky společnosti CCA Group a.s. za jejich cenné rady a ochotu, se kterou mi vždy vyšli vstříc. V neposlední řadě bych rád poděkoval své rodině za úžasnou podporu v průběhu studií.

Abstrakt

Práce je koncipována do několika částí. Kapitoly 2 a 3 jsou především teoretické a mají za cíl čtenáře v rychlosti zasvětit do vybraných pojmů v oblasti testování software a seznámit ho také s prostředím Microsoft Visual Studio 2010 a jeho možnostmi. Kapitola 4 pojednává o aplikaci Škola OnLine a o jejích klíčových charakteristikách z pohledu testování a o některých technických detailech. Další kapitoly jsou již zaměřeny prakticky. Pátá kapitola zahrnuje návrh testů vzhledem k možnostem prostředí VS2010 i samotné aplikace. Šestá kapitola popisuje implementaci navržených testů a sedmá pak jejich vyhodnocení. Osmá kapitola zmiňuje testovací software, který by mohl posloužit jako alternativa k VS2010.

This presented study is structured into eight different sections. Following the introduction, the chapters 2 and 3 contain mostly of theoretical contents and are aimed at inducting the reader quickly into the environment and features of Microsoft Visual Studio 2010. The fourth chapter deals with the application of Skola OnLine, its key characteristics in terms of testing, and some other technical details. The remaining chapters are rather practical. The fifth part includes a proposal of tests for both the environment VS2010 and the application itself. Chapter 6 describes the implementation of suggested tests, only for the seventh section to assess them. The eighth and final chapter concludes by listing some testing software that could be used an alternative to VS2010.

Klíčová slova:

Visual Studio 2012;Microsoft;web applications;ASP.NET;software testing;load testing;web performance test;stress test;data driven testing, Team Foundation Server,SQL Server,IIS, Coded UI test

Obsah

1 Úvod

Testování softwaru je činnost, která má na rozdíl od jeho vývoje zcela opačný, tedy destruktivní charakter. Tester musí jako pracovník k danému objektu přistupovat s cílem jej narušit a objevit negativní vlastnosti. To může být leckdy obtížné. A podobně nesnadné je následně zjistit, zda je nalezené pochybení opravdu chybou v testované aplikaci, nebo v běhovém prostředí; či je špatně navržen a proveden samotný test, ba dokonce je-li chyba v testovacím nástroji. Vyšetřování a rozhodování s tím spojené je věčným bojem každého testera, přičemž největšími nepřáteli jsou vlastní neznalost a nedostatek zkušeností.

Předložená práce sice kompletně souvisí s webovou aplikací Škola OnLine, avšak velkou pozornost věnuje i prostředí Visual Studia 2010, jehož některé funkce do detailu rozebírá. Dobře tedy poslouží každému zájemci o testování webových aplikací v tomto prostředí. Šestá kapitola, a obzvláště pak podkapitola *6.9*, provedou čtenáře vybranými problémy, které na stránkách MSDN, případně příslušných diskusních fórech, nebyly kvalitně zdokumentovány. Ušetří mu tak jistě mnoho času i energie.

Pro velké množství v práci uvedených pojmů existují vedle anglických i české ekvivalenty. Přestože je čeština bezesporu tím nejkrásnějším jazykem, jejich využití je v této práci zcela uzpůsobeno terminologii zažité pro prostředí Visual Studia. Cílem je usnadnit čtenáři orientaci v samotném vývojovém prostředí, v jeho dokumentaci i další odborné literatuře.

2 Přístupy k testování webových aplikací

Proces testování je pevně spjatý s vývojem. O nezbytnosti testování a o významu správného přístupu a dodržování určitých pravidel již bylo napsáno mnoho knih a článků. Rozdělení testů do kategorií je k nalezení snad v každé z těchto publikací, které se v této oblasti v něčem shodují a v něčem rozcházejí.

2.1 Přístupy k testování software

Obecně jsou uznávané 3 základní přístupy k testování software. Pojmy *black-box* a *white-box* definuje již Ron Patton [1]. Pojem *grey-box* je pak určitou kombinací prvních dvou zmíněných přístupů.

Black-box

Black-box testing (testování černé skříňky, funkční testování) je způsob, kdy testerovy není známa struktura programu. Návrh testu vychází z požadavků a popisu správné funkce uvedených ve specifikaci.

White-box

White-box testing (testování bílé skříňky, strukturální testování) je obecně technika testování, při které se vychází ze znalosti programu, tedy ze znalosti struktury a logiky jeho kódu [2].

Grey-box

Grey-box testing (testování šedé skříňky) je přístup, kombinující oba předchozí. Testerovi je při návrhu a provádění testů vnitřní struktura programu známa jen částečně [3].

2.2 Statické a dynamické testování

Rozdíl mezi statickým a dynamickým testováním je ten, že statické testování není založeno na funkčním a spustitelném programu. Představuje jej například analýza zdrojových kódů, nebo kontrola a ověřování specifikace.

Dynamické testování ověřuje, zda program běží správně. Testuje jeho chování a porovnává vstupy s očekávanými výstupy

2.3 Míra automatizace testování

Manuální i automatické testování mají svoje výhody. Přesto, že jsou k dispozici stále sofistikovanější testovací nástroje umožňující velmi snadnou automatizaci, lidský úsudek je nenahraditelný. Pro testování software je klíčové právě nalezení jejich správné kombinace a využití vhodných nástrojů.

2.4 Úrovně testování

Úrovně testování souvisí s fázemi vývoje aplikace. V různých fázích jsou doporučeny různé testy [4] a [5] a cílem je zajištění optimální zajištění vývoje software v rámci užité metodiky a brzké odhalení chyb. Rozdělení úrovní testování:

Testy komponent

Testy komponent nebo také *unit testy* mají za cíl izolovat a otestovat nejmenší možný samostatně funkční celek programu, například třídu nebo nějakou metodu. Unit testy vytváří často sám programátor. Jsou to obvykle první testy aplikace a měly by ověřit správnost napsaného kódu na té nejnižší úrovni.

Integrační testování

Při integračním testování se testuje jak správnost spolupráce jednotlivých modulů uvnitř aplikace, tak i komunikace modulů s okolím, tedy s operačním systémem, s jiným software nebo třeba se vstupním a výstupním hardware.

Systémové testování

Systémové testování prověřuje aplikaci jako celek. Zda odpovídá požadavkům a specifikaci zákazníka. Testovací scénáře jsou rozsáhlejší a simulují práci uživatele.

Alfa-testování

Alfa testování je simulace nasazení a používání aplikace nezávislým týmem testerů na straně developera [6].

Beta testování

Beta testování navazuje na alfa testování a je zpravidla prováděno koncovým zákazníkem těsně před akceptačními testy. V ideálním případě by mělo potvrdit, že je software připraven k předání [1].

Akceptační testování

Testování aplikace z hlediska dodání finálního produktu zákazníkovi, který se na základě výsledků rozhodne, zda s převzetím produktu souhlasí.

2.5 Typy testů

Testy software jsou rozděleny do mnoha kategorií a typů. Samotnému dělení se věnuje mnoho článků a knih a každý autor se na problematiku dívá odlišně. Následuje výčet několika základních a obecně zažitých typů, částečně definovaných Ronem Pattonem [1] a uznávaných také například odbornými servery [7] a [6].

Toto rozdělení je spíše teoretické. V praxi se jedná skoro vždy různé kombinace těchto typů. Jejich uvedení je nicméně vhodné pro lepší představu o rozmanitosti softwarového testování a množství pohledů na jeho problematiku.

Regresní testy

Regresní testy mají za cíl zjistit, zda po přidaní nové funkcionality nebo opravě chyby je software jako celek stále funkční. Tedy ověřit, zda funguje to, co fungovalo a bylo ověřeno již před přidáním.

Přírůstkové testy

Přírůstkové testy, nebo někdy označované také jako progresní, se zaměřují na testování nových funkcí nebo modulů, které byly k projektu přidány.

Smoke testy

Základní jednoduchý test, který zjišťuje, zda aplikace běží, tedy zda je dostupná. V případě Školy OnLine tento test představuje spuštění okna prohlížeče a zadání adresy lokálního serveru, na kterém by měla běžet. Zobrazení úvodního přihlašovacího formuláře pak reprezentuje úspěšné nastavení IIS serveru a rozmístění aplikace. Pro ověření přístupu Školy OnLine k databázi je nutné se do aplikace přihlásit jako jakýkoliv uživatel.

Smoke test je asi nejčastěji prováděný test. Vždy je vhodné ověřit tyto dvě základní funkce před spuštěním složitějších automatizovaných testů. Obzvláště inicializace load testu trvá velmi dlouho a jednoduchý smoke test tak dokáže ušetřit spoustu času. O spouštění load testů a s ním spojených komplikací dále pojednává samostatná část v kapitole 6.9.

Load testy (zátěžové testy)

Load testy jsou zaměřeny na sledování chování aplikace v dlouhodobé pracovní zátěži, která odpovídá běžnému provozu. U webových aplikací je simulováno připojení určitého množství uživatelů k serveru a jejich obvyklá činnost. Sledují se parametry celého systému i výkon aplikace. Cílem je odhalit chyby, které se objeví až po několika hodinách nepřetržitého běhu.

Stress testy

Během stress testu je aplikace vystavena extrémní zátěži, ke které za běžného provozu obvykle nedochází. Sleduje se její chování a také k jakým chybám dochází. Jako základ stress testu v praxi obvykle dobře poslouží více či méně modifikovaný load test.

Zotavovací testy

Zotavovacím testem, neboli recovery testem se zjišťuje, zda vůbec a za jak dlouho aplikace obnoví svou činnost po pádu způsobeném přetížením nebo nějakou jinou chybou (selhání operačního systému, hardwaru apod.). Zotavovací test má velmi blízko ke stress testu a někteří autoři odborné literatury považují sledování zotavení za součást stress testu [8].

V případě testování Školy OnLine je zotavovací test součástí stress testu. Přetížení vede k pádu aplikace a z grafů load test analyzeru je pak velmi dobře patrné, za jak dlouho po pádu je celý systém opět schopný odbavovat požadavky uživatelů.

Bezpečnostní testy

Zjišťuje úroveň zranitelnosti aplikace, tzn. jak dobře je chráněna proti neoprávněnému přístupu, jak jsou zabezpečeny citlivé údaje apod.

Výkonnostní testy

Výkonnostní testování neboli performance testing sleduje jak je aplikace výkonná v obsluze požadavků. Měří se doba odezvy a sleduje se, kde dochází během zpracování k nadměrné spotřebě výpočetního výkonu. Kromě klasických testů (v případě Visual Studia 2010 nazývaných Web Performance Testy) jsou ještě k dispozici nástroje pro profilování (performance profiling), tj. pro dynamickou analýzu kódu, která umožňuje vyhledávání částí vhodných k optimalizaci.

Testy použitelnosti

Testování použitelnosti (usability testing) má za cíl ověřit uživatelskou přívětivost aplikace, tedy uživatelské rozhraní. Sleduje se snadnost ovládání, intuitivnost prostředí a srozumitelnost výstupu. Je důležité jej provádět manuálně a soustředit se na doporučení a zažité konvence.

Testování kompatibility

Ron Patton [1] popisuje jako testování kompatibility software schopnost komunikovat nebo předávat data mezi dvěma různými programy a jako možnost běhu v určitém prostředí (v určitém operačním systému)

Funkční testování

Funkční testování testuje, zda aplikace správně provádí funkce, které odpovídají specifikaci. Funkční testování je často realizováno testy grafického uživatelského rozhraní.

Srovnávací testy

Srovnávací test může tvořit jakýkoliv z výše uvedených testů, nebo jejich kombinace, která je opakovaně spuštěna za různých podmínek. Tento typ slouží k porovnání například výkonu dvou různých verzí aplikace, nebo stejné verze s různým nastavením. Je také vhodné porovnat různé konfigurace hardware, které máme k dispozici, popřípadě si tak otestovat změny v nastavení IIS nebo SQL serveru. Je také vhodné provádět srovnávací testy s konkurenčním software.

2.6 Testovací dokumentace

Normy a doporučení

Testování software je věnován standard IEEE 829. Konkrétně specifikace 829-1983 IEEE, později doplněná a vydaná jako 829-1998, nebo jako zatím poslední verze IEEE 829-2008 [9], která specifikuje soubor dokumentů pro testování software.

Testovací plán

Testovací plán je dokument, který popisuje, co a jak bude testováno. Zahrnuje seznam testů, technik a nástrojů, harmonogram testování a specifikuje také požadavky na kvalitu software [4].

Testovací plán by měl na základě zmíněného doporučení IEEE 829 [9] obsahovat následující položky:

- Jednoznačný identifikátor testovacího plánu
- Reference na všechny související dokumenty
- Úvod, kde je uveden účel plánu a souhrn jeho obsahu

- Testované funkce z pohledu struktury programu
- Testované funkce z pohledu uživatele a jeho práce
- Netestované funkce z pohledu uživatele a jeho práce
- Rizika, která jsou s funkcí programu spjata, tzn. vlastnosti a okolnosti potenciálně ohrožující jeho správné fungování.
- Přístup k testování a jeho úroveň
- Kritéria vyhodnocení testu, tedy za jakých podmínek test projde nebo naopak selže
- Co je součástí dodávky testování (co je výstupem plánu)
- Kritéria, kdy je testování možné přerušit a kdy je možné v něm pokračovat
- Požadavky na testovací prostředí (hardwarové a softwarové nároky apod.)
- Požadavky na kvalifikaci testerů
- Kdo je za konkrétní části procesu testování zodpovědný
- Harmonogram testování a možnosti jeho úprav (rezervy a závislosti jednotlivých procesů)
- Potenciální rizika ohrožující průběh testování
- Souhlas vedoucí osoby s testováním
- Seznam zkratek a vysvětlivek

Testovací případ

Testovací případ (test case) popisuje postup pro otestování konkrétního požadavku uvedeného ve specifikaci. Obsahuje také informace o tom, co je vstupem a co je očekávaným výstupem zmíněného postupu. Návrhu testovacího případu je věnována kapitola 2.7.

Testovací scénář

Testovací scénář je soubor testovacích případů seřazených v určitém pořadí, které představuje posloupnost činností uživatele. Jako celek popisuje nějakou hypotetickou situaci a testováním je pak ověřeno, jak program tuto situaci zvládá. Ideální scénář je věrohodný, motivující, snadno ověřitelný a také komplexní [10].

Test suite

Je soubor testovacích případů se stejným cílem, testujících jednu konkrétní část nebo modul.

Testovací skript

Testovací skript je kód napsaný v určitém jazyce, který přesně definuje kroky prováděné v rámci testu uživatelem. V závislosti na programovacím jazyku a nástroji umožňuje testování automatizovat.

Test log

Test log je záznam o průběhu testu a jeho výsledku. Na jeho základě je sestaven report.

Test report

Dokument obsahuje souhrnné informace, zda výsledek testu odpovídá testovacímu případu a pokud ne, popisuje případnou odchylku skutečného výstupu od výstupu očekávaného a okolnosti, za kterých k ní došlo. Jeho součástí mohou být i doporučená vylepšení [11].

2.7 Návrh testovacích případů

Návrh testovacího případu je podle Cem Kanera [10] komplexní činnost, obecně vycházející ze tří bodů:

1) Testovací případy pomáhají získávat informace. Různé druhy testovacích případů jsou vhodné pro získání různých druhů informací.
2) Dobrý testovací případ může být dobrý v několika způsobech. Žádný ale nemůže být dobrý ve všech těchto způsobech.
3) Lidé mají sklony vytvářet testovací případy podle určitých vzorů. Dobrý test, například pro testování oboru vstupních hodnot, je odlišný od testu bezpečnostních rizik.

Testovací případ by měl podle doporučení IEEE 829 [9] obsahovat následující informace:

- Jednoznačný identifikátor testovacího případu
- Priorita testovacího případu, tj. jak vážný dopad by měla případná chyba na výsledek projektu
- Testovaná položka – popis rozsahu testu a souvislostí, případně odkaz na konkrétní informace ve specifikaci programu
- Podmínky pro zahájení testu a zvláštní požadavky
- Co je vstupem (vstupní data)
- Popis jednotlivých kroků
- Očekávaný výstup
- Požadavky na prostředí, ve kterém má být test prováděn
- Vysvětlivky použitých pojmů a zkratek
- Historii změn dokumentu

Výše uvedený výčet je však doporučení. Návrh a tvorba obsahu každého testovacího případu je velmi individuální a závisí na mnoha dalších aspektech. Samotný dokument by také neměl obsahovat informace, které jsou irelevantní a tvoří jej nepřehledným.

Ron Patton [1] ve svých zásadách návrhu uvádí, že testovací případ by měl být navržen s maximálním ohledem na dobrou opakovatelnost a sledovatelnost každého testu, na prokazatelnost testování a také na snadnou organizaci průběhu testu.

2.8 Realizace testů

Realizace testů by měla vždy vycházet z příslušných, výše popsaných, dokumentů. Konečný průběh samotných testů však závisí na testerovi, na podmínkách a nástrojích, které k práci má.

Další faktory, které ovlivňují testování, jsou dány vlivem prostředí. Tzn. metodikou vývoje software, kterou společnost používá, rozsahem a důležitostí projektu, specifickými nároky zákazníka a délkou vyhrazeného časového úseku.

2.9 Vyhodnocování výsledků

Při vyhodnocování testů je nezbytné se soustředit nejen na jasně definované testovací případy, jejich jednotlivé kroky a následné očekávané výstupy, ale i na účel celého testování. Jinými slovy, při sledování konkrétního testovacího případu a jeho výsledku si zachovat i globální pohled na řešení jako celek a neustále si uvědomovat co je primárním cílem práce.

Pokud například výsledek nějakého testu nedává smysl, je třeba jít zpět k návrhu testu. Položit si znovu základní otázky, celý test přezkoumat a snažit se k němu najít alternativy, protože špatný výsledek může také znamenat špatně vytvořený test.

2.10 Webové aplikace

Webové aplikace jsou specifickým druhem software, který ke svému testování vyžaduje i specifický přístup. Jsou vystavovány připojení a požadavkům velkého množství klientů z různých míst ve stejném okamžiku. Je tedy nezbytné u nich kromě funkčnosti testovat také to, zda jsou schopny v rozumném čase odbavit dané množství těchto klientů a zda jsou stabilní i v případě dlouhodobé kontinuální zátěže.

3 Testování s Microsoft Visual Studio 2010 Ultimate

Visual Studio společnosti Microsoft je integrované vývojové prostředí vybavené množstvím nástrojů a schopné velmi dobře spolupracovat s ostatními produkty společnosti, jako například Team Foundation Serverem, Microsoft SQL Serverem atd. Soustředí se na pokrytí celého procesu vývoje - od návrhu architektury až po nasazení finální verze.

Konkrétně verze Visual Studio 2010 Ultimate (dále jen VS2010) je k dostání v několika verzích. Pro testování software je určena především verze Test Professional 2010. Práce na testech Školy OnLine jsou prováděny ve verzi Ultimate, která podporuje všechny funkce tohoto produktu obsažené v ostatních variantách.

Většina softwarového vybavení od společnosti Microsoft použitá při testování aplikace Škola OnLine byla získána v rámci programu *Microsoft Developer Network Academic Alliance* (*MSDNAA*) známé i jako program *DreamSpark Premium*, ke kterému autor práce získal přístup prostřednictvím Katedry informatiky a výpočetní techniky Západočeské univerzity v Plzni. Výjimkou jsou jen operační systémy Windows 7 Ultimate, Windows 7 Professional a Office 2010, které jsou v soukromém vlastnictví autora práce a byly pořízeny jako součást použitých počítačů při jejich koupi.

3.1 Přehled možností testování ve VS2010 Ultimate

VS2010 Ultimate umožňuje testovat software mnoha způsoby a vlastní vývojové prostředí je ještě doplněno o samostatný nástroj Microsoft Test Manager 2010 (dále jen MTM). Všechny nástroje a funkce spojené s manuálním testováním jsou soustředěny právě v MTM, kterému se podrobně věnuje kapitola 0.

3.1.1 Coded UI testy

Coded UI testy jsou testy grafického uživatelského rozhraní. Podporují jak formulářové, tak webové aplikace. Jejich provádění je automatické a je definováno kódem, který je generován nástrojem *Coded UI Test Builder*. Během nahrávání je možné navíc vytvořit tzv. Assert statements, které umožňují ověřit během testování aktuální hodnoty s očekávanými. Na jejich vyhodnocení je pak založen celkový výsledek testu.

Coded UI Test Buider ukládá během nahrávání testu informace o uživatelském rozhraní, na základě kterých v něm pak dokáže vyhledat a identifikovat cílový prvek. Tyto informace mají podobu párů název vlastnosti – hodnota a jsou uloženy v souboru UIMap.Designer.cs. Pro případ, že během coded UI testu nelze z nějakého důvodu určitý prvek najít, je VS2010 vybaveno heuristickým algoritmem pro jeho vyhledání [12].

Struktura testu a UI Mapy

Každý Coded UI Test je tvořen čtyřmi soubory [13]:

- **UIMap.Designer.cs** je automaticky generovaný a není editovatelný. V případě jakýchkoli změn v testu je celý sobor znovu generován. Obsahuje deklarace, UI Map Partial class, metody a další nastavení testu.

- **UIMap.cs** s UI Map Partial class na rozdíl od UIMap.Designeru editovat lze.
- **CodedUITest1.cs** s třídami testu (defaultně jen s jedou), metodami a dalším nastavením je také možné editovat.
- **UIMap.uitest** je XML soubor obsahující strukturu nahrávky Coded UI Testu. Není možné jej přímo editovat a pro změny v něm je nutné použít Coded UI Builder.

3.1.2 Web testy

Web testy se ve VS2010 jmenují celým názvem *Web Performance Testy* a je možné je použít mnoha způsoby. Lze s nimi ověřovat správnost vrácených výsledků, měřit odezvu stránek, zjišťovat jejich kompatibilitu s různými prohlížeči na úrovni HTTP nebo je použít jako základní stavební kámen pro zátěžové testy.

Během instalace VS2010 jsou mimo jiné přidány i dva doplňky (add-ons) do Internet Exploreru. Jsou to *Web Test Recorder 10.0* a *Microsoft Web Test Recorder 10.0 Helper*, které umožňují nahrávání Web Testů v IE. Nahráváním je zde myšleno zaznamenání HTTP požadavků, které IE během procházení testovaných stránek generuje.

Vytvořený Web Test je pak ve VS2010 zobrazen jako stromová struktura, ve které jsou přehledně zobrazeny všechny HTTP požadavky, validační pravidla, pravidla pro extrakci, context parametry, spojení s databází apod. Stromovou strukturu testu je možné editovat, měnit pořadí jednotlivých požadavků, seskupovat je do tzv. transakcí nebo vytvářet smyčky.

Jak bylo uvedeno, Web test se vytváří nahráváním uživatelských akcí v Internet Exploreru. Ten však při následném běhu testu není vůbec využíván. HTTP požadavky a odpovědi jsou zpracovávány čistě Web Test Enginem na úrovni protokolu a obraz stránky, který je na jako součást výsledku testu, je sestaven z log souboru HTTP požadavků a odpovědí [14] .

Web Test Engine dále nezpracovává javascript (včetně AJAX) ani ActiveX a nemůže tak testovat chování stránek na straně klienta. Je však možné simulovat HTTP požadavky, které jsou tímto kódem vytvářeny a ověřovat správnost odpovědi serveru.

Coded Web Testy

VS2010 umožňuje u web testů generovat z jejich záznamu kód stejně, jako je tomu u nahrávek manuálních testů. Test pak není zobrazen ve stromové struktuře, ale jako kód .NET třídy, která generuje požadovanou HTTP komunikaci. Kód Coded Web testu je možné libovolně upravovat.

3.1.3 Load testy

Load test ve VS2010 představuje test, který umožňuje simulovat činnost více uživatelů najednou. Load Test je tvořen:

- Souborem jiných testů a jejich vlastním nastavením
- Souborem counter setů (čítačů výkonu; viz kapitola 3.2.2)
- Vlastním nastavením load testu a parametry

Jako základní stavební element load testu se asi nejčastěji používají web testy. Díky web test enginu lze simulovat velké množství současně aktivních konkurenčních uživatelů jen s minimálními nároky na výpočetní výkon. Zdroj generující takovou zátěž nemusí spouštět

prohlížeč ani provádět skripty. Jeho práce je tedy generování požadavků a vyhodnocování odpovědí.

Scenario

Scenario je ve VS2010 pojem, který zahrnuje definici několika klíčových vlastností load testu, jako jsou:

- Load Pattern, umožňující nastavení, zda bude počet uživatelů konstantní, nebo zda se bude v průběhu testu stupňovat
- Test Mix Model (model zátěže), udávající způsob přiřazení testů uživatelům
- Test Mix, definující soubor použitých testů v load testu a jejich procentuální četnost výskytu v jeho průběhu
- Network Mix a Browser Mix, určující které síťové technologie a které prohlížeče budou během testu emulovány a v jakém poměru

V jednom load testu je možné najednou použít více různých Scenarios a současně při jeho běhu využívat všechny možné kombinace nastavení.

Omezení licencí

V rámci základní licence verze VS2010 Ultimate je v jeho nastavení možnost běhu load testu současně pouze s dvě stě padasáti virtuálními uživateli. Distribuovaný běh testu na více počítačích bohužel také není možný (zde je myšleno spuštění zátěže současně z více zdrojů). Pro odstranění tohoto omezení je nutné získání klíče jednoho z rozšíření: *Visual Studio 2010 Load Test Feature Pack* anebo *Visual Studio Load Test Virtual User Pack 2010* [15]. Tato rozšíření však nejsou v rámci programu MSDNAA k dispozici a testování Školy OnLine tak bylo nutno provést se zmíněným omezením.

3.1.4 Unit testy

Unit test lze ve VS2010 vytvořit buď napsáním kódu, nebo jeho vygenerováním prostřednictvím kontextového menu. A to bez ohledu na to, zda je testovaná metoda public nebo private. Testovací třídy jsou označeny atributy TestClass() a testovací metody TestMethod(). V okamžiku vytvoření unit testu jsou do projektu automaticky přidány reference na jmenný prostor Microsoft.VisualStudio.TestTools.UnitTesting [16]. Ten poskytuje třídy pro:

- Ošetření inicializace a ukončení testu
- Ověření podmínek, které by během testu měly nastat
- Atribute ExpectedException, ověřující, zda nastala požadovaná výjimka
- Třídu TestContext pro ukládání informací například o připojení ke zdroji testovacích dat nebo informací nezbytných k běhu testu webových služeb ASP.NET

Kompletní seznam a popis tříd obsažených v Microsoft.VisualStudio.TestTools.UnitTesting je k dispozici na www stránce s dokumentací [17].

3.1.5 Ordered a generic testy

Ordered test je jen definice množiny libovolných jiných testů, které v něm mají přesně určené pořadí. Ordered test je ještě doplněn podmínkou, zda se má při chybě jednoho testu pokračovat v pořadí dalším testem, nebo zda se má celý proces přerušit.

Generic test představuje možnost do projektu ve VS2010 přidat již vytvořený test nástrojem třetí strany. Podmínkou je, že je test spustitelný z příkazové řádky a že výstup tohoto cizího testu musí být typu Pass nebo Fail. Volitelně může tento test jako součást výsledku poskytovat další informace o svém průběhu [18]. Po vytvoření generic testu je s ním ve VS2010 možné pracovat jako s jakýmkoli jiným testem.

3.1.6 Data driven testy

Test řízený daty může být ve VS2010 libovolný test, kterému jsou předávány informace z externího zdroje dat. Tímto zdrojem může být buď CSV, XML soubor anebo SQL databáze. Funkce je ještě doplněna o následující možnosti nastavení:

- **Access Method**: U každé přidané tabulky lze v Data Sources nastavit konkrétní přístupovou metodu. Na výběr je ze Sequential, Random a Unique.
- **Advanced data cursor**: Při vytvoření smyčky (v provádění web testu) je k dispozici volba Advanced data cursor, kterou v případě jejího nastavení na hodnotu True docílíme načítání nových dat z externího zdroje při každé její iteraci. Načítání je pak prováděno podle výše uvedené přístupové metody.

3.1.7 Data and Diagnostics

VS2010 a MTM umožňuje široké využití data adaptérů pro sběr informací během manuálních i automatických testů. Jejich aktivace a nastavení je buď v souboru s koncovkou .testsettings (například local.testsettings u VS2010 projektu), nebo ve vlastnostech testovacího plánu (u Testing Center v MTM). Defaultně je na výběr z následujících adaptérů [19].

ASP.NET Client Proxy for IntelliTrace and Test Impact je proxy, která umožňuje sběr informací o HTTP voláních klienta na server pro IntelliTrace a Test Impact diagnostic data adaptery.

Event log je nastavení, které k výsledku testu připojuje log událostí.

IntelliTrace data adaptér sbírá specifické trasovací informace pro snazší izolaci chyb, které jsou špatně reprodukovatelné. Nastavení IntelliTrace vytvoří soubor s koncovkou .iTrace, do kterého jsou tyto informace ukládány. Pokud je test neúspěšný, zaznamená se chyba a k ní je na zmíněný soubor vytvořena reference.

Network emulation je funkce, která v testu napodobuje vliv různých síťových adaptérů dle nastavení.

System information představuje možnost uložit do výsledků testu informace o systému, na kterém test proběhl.

Test impact data adapter umožňuje sledovat, které konkrétní metody testované aplikace byly volány během testu.

Video Recorder nahrává obrazovku během testu. K jeho správné funkci je nezbytné mít nainstalovaný *Microsoft Expression Encoder 4* [20].

ASP.NET profiler je určen pro sběr dat při provádění load testů a jako jediný adaptér není dostupný v MTM, ale pouze ve VS2010. Jeho součástí je i Tier Interaction Profiler. Ten

poskytuje informace o tom, kolik času je stráveno obsluhou jednotlivých požadavků stránek na data v databázi.

Zde uvedené data adaptéry je možné nastavit a použít i pro všechny role v testu (tzn. na controlleru[1] a všech agentech[2]) a specifikovat tak sledování průběhu testu v celém testovacím prostředí.

3.2 Testovací prostředí

Jak bylo zmíněno, VS2010 umožňuje vytvoření a správu celého testovacího prostředí. Například prostředí tvořeného serverem a několika klienty.

3.2.1 Role

Role určuje kde (myšleno na jakém počítači nebo počítačích) test poběží a kde budou sbírána data. V každém testu je proto nezbytné tyto role obsadit. V případě lokálního testu je počítač zastává všechny. Pokud je test distribuovaný a běží na více počítačích současně, je možné příslušné role vytvořit a přiřadit v menu Test – Edit Test Settings – výběr názvu souboru s nastavením – Roles.

Samo přiřazení je prováděno automaticky na základě atributů. V menu Test – Manage Test Controller jsou konkrétním agentům přiřazeny atributy a jejich hodnoty. Název atributu i jeho hodnota může být libovolná. Důležité je jen to, aby odpovídal atributu a hodnotě, která je stejným způsobem vytvořena u role v souboru .testsettings, kterou by měl agent zastávat.

3.2.2 Performance Counters a Counter sets

Pro vyhodnocení výsledků testu je klíčové právě získání správných dat z těch správných strojů. Kromě využití data adaptérů popsaných v kapitole 3.1.7 lze sledovat ještě výkon celého systému i jeho jednotlivých částí cílového stroje. To umožňuje nastavení takzvaných *Performance Counters*, neboli čítačů výkonu. Tyto countery periodicky sbírají vzorky stavu měřených hodnot na cílovém stroji. Vzorky jsou následně ukládány jako součást výsledků a v podobě grafu poskytují přehled o průběhu testu a chování celého rigu[3].

Nutno podotknout, že volba counterů se provádí až v nastavení konkrétního load testu, na rozdíl od volby data adaptérů, jejichž výběr a nastavení je uloženo v globálním nastavení testu (v souboru s koncovkou testsettings).

Counter set je množina performance counterů, která slouží především pro jejich přehlednější organizaci. Několik counter setů je ve VS2010 již předpřipravených, vždy pro specifickou technologii, například Application, ASP.NET, .NET Application, IIS, nebo SQL. Krom zmíněných se dají vytvořit i vlastní tzv. *custom counter sety* a do nich přidat libovolné countery z cílového systému.

---

[1] **Controller** je počítač, který spravuje agenty a sbírá od nich výsledky testu. Popisu jeho činnosti, nastavení a možnostem využití jsou věnovány například kapitoly Architektura rigu a Konfigurace testovacího rigu.

[2] **Agent** je počítač, na kterém běží test a na kterém jsou sbírána požadovaná data.

[3] **Testovací rig** (angl. test rig) je název pro soubor počítačů, na kterých běží test [57].

3.2.3 Remote execution a remote collection

Ve VS2010 je možno provádět test třemi různými způsoby:

- Local execution je určeno pro běh a vyhodnocování testů na lokálním stroji.
- Local execution with remote collection umožňuje spustit a vyhodnocovat test na lokálním počítači. Na rozdíl od předchozí možnosti se sbírají data také na ostatních strojích, na kterých jsou instalovaní agenti (například na web serveru s testovanou aplikací).
- Remote execution představuje plně distribuovaný test, který běží a je vyhodnocován na více agentech najednou. Na remote execution se vztahuje omezení licencí, které je popsáno na konci kapitoly 3.1.3.

Výběr metody provádění testu je možný v souboru .testsettings.

3.2.4 Architektura rigu

Testovací rig je složen z počítačů plnících následující funkce:

Klient

Klienta ve smyslu zapojení a konfigurace testovacího rigu představuje počítač s nainstalovaným VS2010, na kterém jsou vyvíjeny a spouštěny tesy a prohlíženy jejich výsledky.

Controller

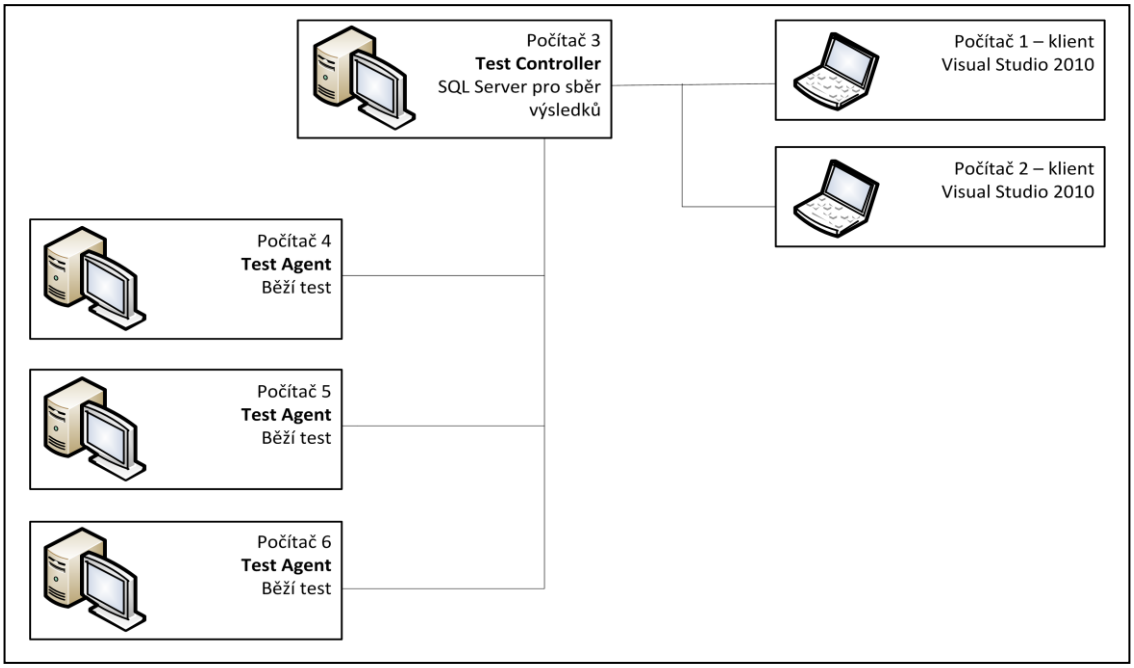
Controller je počítač s nainstalovaným *Test Controller 2010*[4], který běží jako služba, a který spravuje agenty a sbírá výsledky. Spolu s těmito agenty tvoří takzvaný testovací rig.

Agent

Agenti jsou stroje s nainstalovaným *Test Agent 2010*, na kterých běží test a na kterých jsou sbírána data o jeho průběhu. Test Agent může běžet buď jako služba, nebo jako interaktivní proces. V případě volby interaktivního procesu se dá agent využít i k běhu coded UI testů.

V případě lokálního testu mohou být klient, controller i agenti nainstalovány na jednom stroji. Výhody testování ve VS2010 však vyniknou až při distribuovaných testech, kdy lze, díky detailnímu nastavení, věrně napodobit produkční prostředí a získat tak relevantní výsledky. Následující obrázek je příkladem možného zapojení rigu pro distribuovaný test [21].

[4] **Test Controller 2010** i **Test Agent 2010** jsou k dispozici jako součásti balíku *Visual Studio Agents 2010* [58].



Příklad zapojení testovacího rigu

3.3 Team Foundation Server

Microsoft Team Foundation Server 2010 (dále jen TFS) je platforma pro provázání různých služeb a funkcí a podporující spolupráci členů týmu. Tvoří jádro správy životního cyklu aplikace, umožňuje přehledné zobrazení stavu jednotlivých částí projektu i jeho celkového průběhu. Podporuje verzování (nástroj *Source Control*), správu testovacích případů, reportů, chyb, úkolů apod.

Team project collections

Každý týmový projekt je možné v TFS přiřadit do tzv. *Project Collection*, tedy do kolekce projektů. Project Collection usnadňují správu více projektů s podobnými požadavky nebo se společnými zdroji.

Lab management v TFS

Modul *Lab Management* umožňuje správu testovacích prostředí. Pro jeho provoz je však třeba nainstalovat *System Center Virtual Machine Manager Administrator Console* a tou být připojen k *System Center Virtual Machine Manager Serveru* (dále jen SCVMM). Problémy související se zprovozněním Lab Managementu jsou popsány v kapitole 6.1.7.

TFS Team Web Access

TFS Team Web Access je webové uživatelské rozhraní pro TFS, díky kterému je zpřístupněna velká část jeho funkcionality. Web Access je automaticky nainstalován a nastaven při instalaci samotného TFS.

3.4 Microsoft Test Manager 2010

Nástroj Microsoft Test Manager 2010 (MTM) je samostatná aplikace dodávaná s verzemi VS2010 Ultimate a Test Professional. Je složena ze dvou modulů:

Testing Center

V prostředí je integrována kompletní podpora pro manuální testování. Patří sem vytváření testovacích plánů, testovacích případů (test case management) a jejich správa. Dále prostředí pro běh, zaznamenávání, přehrávání testů Test Runner, umožňující také vytvořit chybu (bug) a přidat ji do bug-tracking systému. Přidávání dotazů (Queries) na úkoly a chyby v tomto systému. Základní složkou je testovací plán, ten je rozdělen do *test suits*, které obsahují testovací případy. K jednotlivým testovacím případům je možné přiřadit parametry obsahující údaje, které mají být vyplněny do formulářů aplikace. Samotné testování je tak velmi urychleno a vyplňování formulářů lze nastavit i jako automatické.

Úplná automatizace manuálního testování je v *Testing Center* umožněna funkcí pro záznam a následné přehrávání manuálních testů. Jejich nahrávání je provedeno během prvního testování a při opakování testu stačí pouze po jeho spuštění zvolit možnost přehrát vše. Testing Center MTM umožňuje z těchto nahrávek i generování coded UI testů. S nimi se ale již dále pracuje ve VS2010. Pro sledování chyb a tvorbu reportů je pro každou z rolí k dispozici úplná sada nástrojů Data and Diagnostics. Jejich nastavení je umožněno pro každý test plán.

Lab Center

Lab Center je modul zaměřený na vytváření a správu virtuálních i fyzických strojů a testovacích prostředí. Tester v něm může spouštět, zaznamenávat a následně přehrávat testy distribuované na více strojích. K těmto funkcím je však nezbytné připojení k patřičně nakonfigurovanému TFS serveru.

Tools

Jako třetí doplňující modul, který se v nabídce objeví pod názvem *Tools*, je možno stáhnout Test Scribe Tools [22], který umožňuje export reportů jako dokumentů ve formátu MS Word. Je to alternativa k výstupu poskytovaného webovým rozhraním Team Foundation Serveru.

3.5 Microsoft Visual Studio 2010 Feature Pack 2

Microsoft Visual Studio 2010 Feature Pack 2 je doplněk rozšiřující funkce prostředí VS2010 mimo jiné i v oblasti testování software [23].

- Přehrávání UI testů v prohlížeči Mozilla Firefox
- Editování Coded UI Testů ve zvláštním editoru
- Vytváření UI testů nebo jejich nahrávek pro Silverlight aplikace

3.6 SQL Server 2008 R2

Databázový server tvoří nezbytnou součást celého řešení. SQL Server 2008 R2 poskytuje databázovou vrstvu především pro samotnou aplikaci Škola OnLine. Dále spravuje databáze pro TFS, databázi se zdroji testovacích dat a v neposlední řadě také databázi s výsledky load testů (Load Test Results Repository).

Pokud není nastaveno jinak, VS2010 při instalaci automaticky vytvoří instanci SQL Server Express a v ní databázi nazvanou LoadTest2010, do které se následně defaultně ukládají výsledky load testů. Pro podrobnosti viz kapitolu 6.7.4.

3.7 IIS 7.5

Pro testování webové aplikace je nezbytnou součástí i webový server. Internet Information Services 7.5 (dále jen IIS) je web server dodávaný jako součást operačních systémů Windows Server 2008 R2 a Windows 7 a plně podporuje platformu .NET Framework 2.0 až 4.0 a ASP.NET.

Během testů je pro sledování jejich dopadu na chování aplikace důležité nastavit nejen příslušné systémové performance countery, ale i countery sledující práci IIS a ASP.NET. VS2010 již má pro tento účel defaultně vytvořené counter sety, nicméně je možné je libovolně modifikovat nebo si dle potřeby vytvořit vlastní.

IIS je také třeba pro zpřístupnění webového uživatelského rozhraní TFS.

3.8 Podpora ze strany společnosti Microsoft

VS2010 i ostatní, zde zmíněné produkty, mají velmi dobrou podporu. Zájemcům o vyzkoušení a porozumění vybraným technologiím jsou kromě klasických 60-ti nebo 90-ti denních trial verzí software [24] k dispozici ještě následující dvě možnosti:

Virtual Labs

Microsoft TechNet Virtual Labs (někdy také nazývané jako Hands-On Labs) je možnost vyzkoušet si zvolenou technologii nebo produkt v již předpřipraveném virtuálním prostředí, ke kterému je zajištěn přístup prostřednictvím vzdálené plochy [25]. Předpřipravené prostředí znamená již nainstalované a nakonfigurované potřebné produkty, vzorová data podporující nějaký scénář (např. vzorová ASP.NET aplikace připravená pro manuální a Coded UI testy) a podrobný návod, který zájemce tímto scénářem provede. V případě produktu VS2010 je k dispozici hned několik desítek takových řešení včetně několika přímo zaměřených na testování software [26].

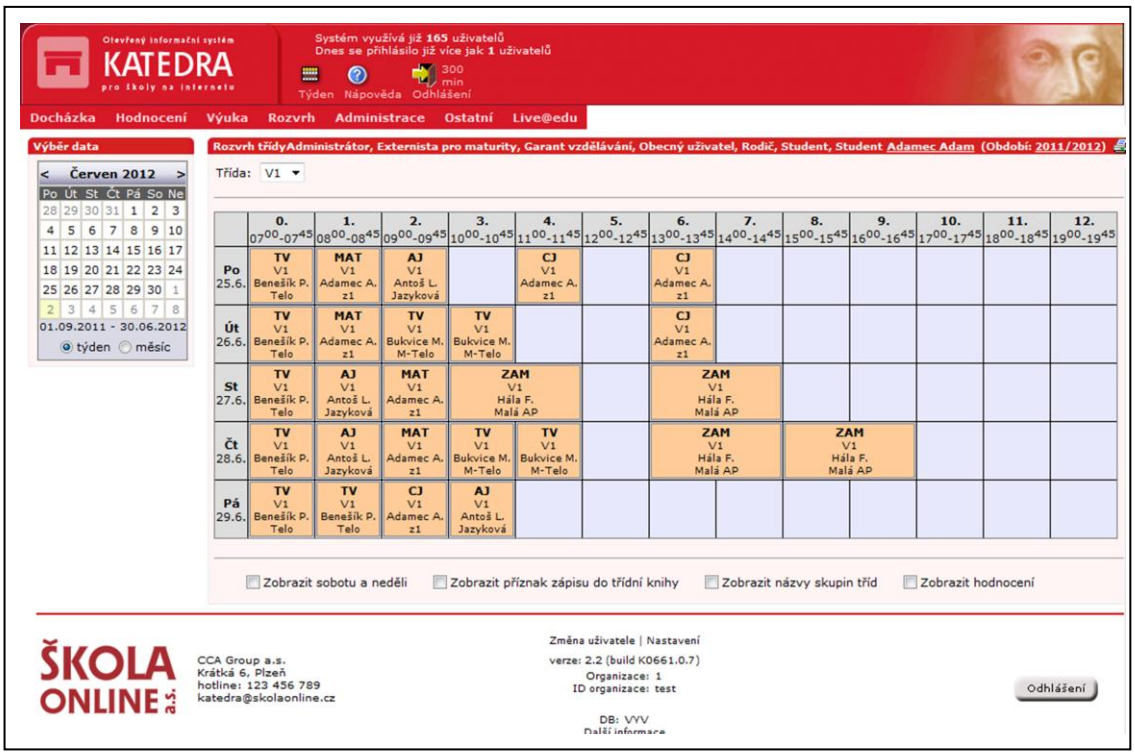
VHDs

Druhou možností je VHD Test Drive Program [27], který představuje sérii virtuálních disků VHD s opět již nainstalovanými a předpřipravenými systémy, aplikacemi a dokumentací, jako je tomu u Virtual Labs. Rozdíl je v tom, že zájemce si stahuje celý soubor VHD a tento systém pak

běží na jeho virtuálním stroji. VHDs tak poskytují mnohem více prostoru pro vlastní experimenty.

4 Aplikace Škola OnLine

Aplikace Škola OnLine je velmi specifická webová aplikace vyvíjená společností CCA Group a.s. a představující komplexní řešení pro školní agendu. Zahrnuje například správu školní matriky, docházky, studijních výsledků, třídní knihy, učebních plánů, rozvrhů, knihovny, přijímacích řízení, vysvědčení a maturit. Tato skutečnost na ni klade zvláštní nároky z hlediska dostupnosti a použitelnosti.



Snímek obrazovky aplikace Škola OnLine

4.1 Technické informace

Aplikace je nasazena na třech serverech, z nichž každý má k dispozici vlastní databázi. Zátěž je mezi ně rozložena pomocí reverzní proxy (viz obrázek).

Ve špičce se počet připojených aktivních klientů pohybuje okolo devíti set. Nároky na výkon celého řešení jsou tedy poměrně vysoké.

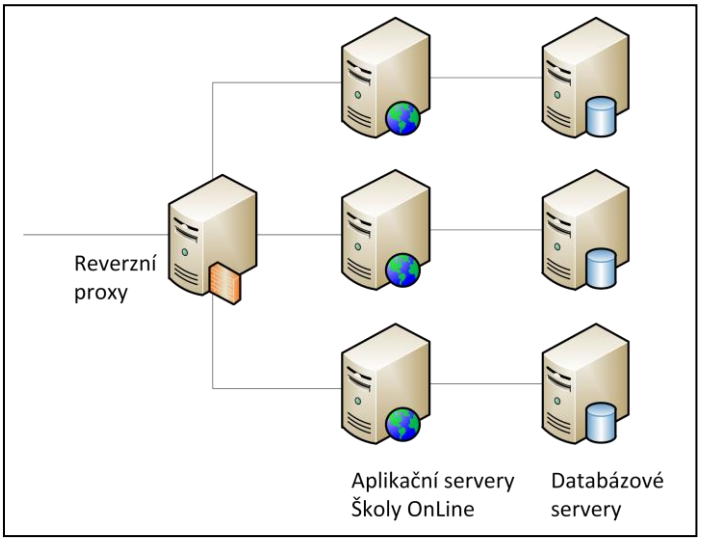


Schéma produkčního nasazení Školy OnLine

V současné době jsou jako aplikační i databázové servery použity stroje IBM xSeries 3550 se čtyřjádrovými procesory Xeon na frekvenci 3GHz, operační pamětí 8GB v aplikačních a 4GB v databázových serverech. Pevné disky pracují při 10000 otáčkách za minutu.

Škola OnLine je založená na .NET Frameworku 3.5 a využívá technologií ASP.NET, AJAX a JavaScript. Z internetových prohlížečů podporuje především Internet Explorer a Mozilla Firefox, ovšem velmi dobře funguje například i v Safari. Uživatelské rozhraní je kromě českého jazyka také k dispozici v angličtině a němčině. Ověření uživatele při přihlášení je možné buď klasicky prostřednictvím formuláře, nebo přes Windows Live ID.

Testovaná verze Školy OnLine je 2.2, build K0661.0.7. Pro testování byla spolu se zdrojovými kódy dodána i databáze VYV s testovacími dat.

4.2 Databáze

Databáze VYV

Databáze VYV slouží pro vývoj aplikace a obsahuje testovací data a účty zhruba 160 uživatelů. Jak je z určení aplikace patrné, aktuálnost dat v databázi hraje důležitou roli. Pro testování byl vytvořen nový školní rok 2011–2012 s několika třídami žáků, úvazky učitelů a odpovídajícím rozvrhem. Pro zajištění větší variability virtuálních uživatelů, byla většině učitelů přiřazena dodatečná uživatelská práva v rámci aplikace.

Testovací databáze VYV_TEST

Databáze VYV_TEST je speciálně vytvořená databáze určená jako zdroj testovacích dat. Je velmi jednoduchá, bez relací a obsahuje jen několik tabulek, které slouží především jako alternativa k původně používaným CSV a XML souborům. Její použití poskytuje větší flexibilitu a lepší dostupnost v případě testování v distribuovaném prostředí.

5 Návrh testovacích případů

K testování Školy OnLine bylo přistupováno dle mnoha kritérií. Rozhodující byla také skutečnost, že aplikace je k dispozici jako hotový produkt. Společnost CCA vyvíjí a provozuje Školu OnLine již řadu let. Jak bylo zmíněno, testování by mělo být součástí celého průběhu vývoje. Některé typy testů, jako například unit testy, v tomto případě částečně ztrácí smysl, protože jednotlivé třídy a metody jsou již funkční a delší dobu používané. Na druhou stranu, hotová aplikace poskytuje příležitost pro vytvoření jednak výkonnostních a zátěžových testů a také automatických funkčních testů uživatelského rozhraní.

Rozdělení testů Školy OnLine v textu práce odpovídá rozdělení z hlediska VS2010. Jejich využití je následující:

- Manuální a coded UI testy pro funkční testování
- Web performance testy pro výkonnostní testování (sledování doby odezvy stránek)
- Ordered testy pro seskupení jednotlivých web testů
- Load testy pro zátěžové testování, stress testy a testy zotavení
- Unit a generic testy byly prováděny experimentálně. Tedy jen jako vzorové, v rámci osvojení práce s VS2010.

Během návrhu a testování nebyla k dispozici technická dokumentace. Požadavky na funkčnost aplikace však do velké míry vyplývají z detailní a dobře zpracované uživatelské příručky [28].

Při posuzování odezvy jednotlivých stránek Školy OnLine je vycházeno z obecně platných limitů, které ve své knize Usability Engineering uvádí Jakob Nielsen [29].

Doba odezvy	Uživatel
0,1s	Má pocit, že aplikace reaguje okamžitě.
1s	Registruje zpoždění, plynulost jeho práce však není narušena.
>10s	Ztrácí soustředění na práci a během čekání se věnuje jiné činnosti.

5.1 Manuální testy a coded UI testy – návrh

Funkční testování bylo rozděleno do dvou částí.

Dostupnost nejčastěji používaných stránek obsahuje jednoduché a krátké testy pro ověření dostupnosti a správného zobrazení stránek ze sekcí docházka, výuka, hodnocení, administrace a rozvrh.

Ověření nejčastěji používaných funkcí obsahuje testy pro přihlášení uživatele, výpis docházky, vyhledávání ve školní matrice a vypůjčení a vrácení knihy z knihovny.

V souvislosti s návrhem manuálních testů jsou k dispozici testovací případy z MTM a TFS. Vzorové ukázky jsou v příloze:

- Příloha 1: Ukázka z dokumentu Test Plan Details, vygenerovaného v MTM prostřednictvím nástroje Tools

- Příloha 2: Příklad testovacího případu vytvořeného v MTM a exportovaného z webového rozhraní TFS

5.2 Web performance testy – návrh

Web testy byly navrženy s ohledem na jejich následné vložení do load testů. Vždy soustředěny na činnost reprezentovanou konkrétní položkou v menu. Seznam testovaných sekcí je následující:

Docházka

- Ředitelské výpisy (průměrná docházka a nezapsané hodiny)
- Absence ve třídách
- Absence v předmětu
- Výpis třídní knihy
- Zadávání docházky žákům
- Zápis do třídní knihy

Hodnocení

- Úprava hodnocení
- Výpis hodnocení žáka

Rozvrh

- Dozory učitelů
- Rozdělení studentů
- Výměna hodin
- Výpisy rozvrhů
- Výpis školních akcí

Administrace

- Číselníky tříd a místností
- Výkazy
- Výpis seznamu zákonných zástupců
- Výpis seznamu učitelů
- Vyhledání žáka ve školní matrice a přidání poznámky
- Vyhledání skupin žáků ve školní matrice dle různých kritérií

Ostatní

- Inventář
- Knihovna (přehled výpůjček)
- Statistiky přístupů
- Přihlášení a odhlášení uživatele
- Změna období

Testovací případy, které posloužily jako základ k vytváření web testů, jsou k dispozici na DVD v souboru *TestovaciPripadySOL.pdf* nebo *TestovaciPripadySOL.docx*. Ukázku jednoho u nich je možné nalézt v Příloze 3: Testovací případ.

Ordered testy

Plán využití ordered testů měl především 2 cíle:

1) Vložit web testy pro určitou skupinu stránek do jednoho ordered testu pro dosažení rychlejšího a snazšího spuštění a následného vyhodnocení.
2) Seskupení web testů do ordered testů a následné vložení těchto ordered testů do load testů pro vytvoření propracovanější konfigurace. Nakonec se však tento postup ukázal jako nefunkční (viz kapitola 6.6).

5.3 Load testy a stress testy – návrh

Cílem load testů bylo:

- Zjištění výkonnostních možností aplikace v daném prostředí (viz kapitoly 6.1.1 a 6.1.2).
- Zjištění kritické úrovně zatížení, kdy už aplikace není schopna odbavovat požadavky klientů.
- Zjištění vlivu intenzivní a dlouho trvající zátěže a odhalení případných chyb.

Klíč ke zmíněným třem bodům představovalo především přiblížit se vlastním modelem zátěže ke skutečnému nasazení aplikace. Činnost uživatelů v aplikaci s tolika funkcemi může být velmi různorodá a pro její správnou simulaci je nezbytné celý proces rozložit. Vytvořit tedy typy scénářů odpovídajících jednotlivým typům uživatelů, jako jsou například ředitel, učitel a administrátor. Každému typu uživatelů přiřadit četnost jeho výskytu vůči ostatním a testovací případy, které jsou pro něj charakteristické. Dále zvolit intenzitu práce uživatele. To znamená, kolik iterací každého testu uživatel zhruba vykoná během pracovní doby.

V případě Školy OnLine byli virtuální uživatelé rozděleni do čtyř skupin. První skupina *Scenario_VedeniSkoly* představovala vedoucí pracovníky, kteří pracují s aplikací celý den a provádějí složitější úlohy jako úpravy rozvrhu, výpisy suplování apod. Další tři skupiny *Scenario_Ucitele_A*, *B* a *C* byli učitelé, využívající Školu OnLine méně a provádějící úlohy jako jsou zápisy do třídnice, správa docházky, výpisy rozvrhu a školní matriky. Celkem je uživatelům rozděleno 38 různých web testů s různou četností. V průměru zhruba 3 web testy za hodinu u učitelů a 7 u vedoucích pracovníků. Stejné složení bylo využito pro stress test *StressTestSimulaceSOL.loadtest*, u kterého byl jen nastaveno stupňované zatížení.

Kromě popsaného modelu byla ještě použita jeho varianta s vyšší intenzitou zátěže, upravená pro dlouhotrvající load testy. Virtuální uživatelé ze skupin učitelů jsou sloučeni do jedné *Scenario_Ucitele_ABC*. Složení bylo mírně zredukováno a intervaly mezi testy každého uživatele zkráceny na dvě až tři minuty.

Přesné rozložení testů je spolu s testovacími případy v příloze číslo 4: Složení load testů.

Optimální zátěž

Za optimální zátěž bylo považováno vytížení procesoru na 75%. Na této hodnotě byly nastaveny prahové hodnoty performance counterů na všech strojích.

Pro zjištění počtu současně připojených uživatelů při průměrném vytížení procesoru serveru na 75% bylo plánováno využít *Goal-Based load pattern* (viz kapitola 6.7.1), který během testu přidává nebo ubírá virtuální uživatele na základě stavu sledované veličiny.

Mezní zatížení

Pro definování mezního zatížení byl navržen Step load pattern, který postupně, bez nastavené horní hranice, přidává uživatele. Hodnota je pak vyčtena z grafů zatížení procesoru, počtu uživatelů a průměrné doby odezvy.

Dlouhotrvající intenzivní zátěž

Po zjištění počtu uživatelů, které je aplikace schopna obsloužit při optimální zátěži, byl navržen load test, který po několik hodin tuto zátěž udržuje.

Počet konkurenčních uživatelů

Jak bylo uvedeno v kapitole 4, počet současně připojených uživatelů Školy OnLine ve špičce se pohybuje kolem devíti set. Vzhledem k zapojení serverů a použití reverzní proxy připadá na jeden server tři sta klientů. VS2010 umožňuje bez dodatečné licence maximální počet 250 virtuálních uživatelů. Během plánování zátěžových testů bylo dosažení této hodnoty považováno za ideální. Rozdíl mezi produkčním a laboratorním prostředím je značný a nelze odhadovat počet uživatelů jen na základě informací o hardware.

6 Implementace testů

6.1 Konfigurace testovacího rigu

Kapitola *Konfigurace testovacího rigu* je velmi úzce spjata s kapitolou 6.9, popisující vybrané problémy, ke kterým může během konfigurace docházet.

6.1.1 Použitý hardware a software

Pro testování Školy OnLine byly využity následující tři stroje. Funkce obou serverů byla dedikována a jejich systém byl optimalizován přímo pro danou činnost v rámci testování. Operační systémy tedy nebyly zatíženy jinými běžícími programy nebo dalším software.

Klient VS2010, controller, agent

Název stroje:	X220_4286-CTO
Systém:	Windows 7 Professional 64-bit (6.1, Build 7601), Service Pack 1
Procesor:	Intel Core i5-2520M CPU @ 2.50GHz
RAM:	12GB
HDD:	HITACHI HTS723232A7A364, 320 GB, 7200 RPM, 16MB cache

Web Server

Název stroje:	SERVERX-PC
Systém:	Windows 7 Ultimate 64-bit (6.1, Build 7601), Service Pack 1
Procesor:	AMD Athlon II X4 640 @ 3.0GHz
RAM:	8GB
HDD:	Western Digital WD3200AAKS 320GB, 7200 RPM, 16MB cache

SQL Server

Název stroje:	DOBRA-PC
Systém:	Windows 7 Ultimate 64-bit (6.1, Build 7601), Service Pack 1
Procesor:	Intel Core 2 Duo E8400 @ 3.00GHz
RAM:	4GB
HDD:	Samsung SpinPoint P-HD160JJ 160GB, 7200 RPM, 8MB cache

6.1.2 Použitý software

X220_4286-CTO

Název	Verze
Microsoft Visual Studio 2010 Ultimate	10.0.40219.1
Microsoft Test Manager 2010	10.0.40219.1
Visual Studio 2010 Feature Pack 2	2.0
Microsoft Expression Encoder 4 Pro	4.0.1639.0
Internet Explorer 9	9.0.8112.16421
Microsoft .NET Framework 4 Extended	4.0.30319
SQL Server 2008 R2 Enterprise Service Pack 1	10.50.2500
Internet Information Services	7.5.7600.16385
Team Foundation Server 2010 SP1	10.0.40219.339
Visual Studio Test Controller 2010	10.0.30319
Visual Studio Test Agent 2010	10.0.30319

ServerX-PC

Název	Verze
Internet Information Services	7.5.7600.16385
Microsoft .NET Framework 4 Extended	4.0.30319
Visual Studio Test Agent 2010	10.0.30319

Dobra-PC

Název	Verze
SQL Server 2008 R2 Enterprise	10.50.1600.1
Microsoft .NET Framework 4 Extended	4.0.30319
Visual Studio Test Agent 2010	10.0.30319

6.1.3 Uživatelské účty

Na každém počítači tvořícím testovací rig[5] (tj. na každém stroji s instalovaným software pro agenta nebo controller) je třeba mít vytvořený zvláštní uživatelský účet, který splňuje následující podmínky [30]:

- Má stejný název a stejné nenulové heslo jako tyto účty na ostatních počítačích.
- Je členem skupiny Administrators na tomto počítači.
- Je dále členem skupin:
  - Performance Log Users
  - Performance Monitor Users
  - TeamTestAgentService
  - TeamTestControllerAdmins
  - TeamTestControllerUsers

Při přidávání uživatele do skupin nesmí být jako součást jeho jména uváděno i jméno počítače.

User account control

Pro správnou funkci controlleru je nezbytné buď kompletní vypnutí technologie user account control (UAC), nebo alespoň její částečné omezení v registrech systému Windows [30]:

V `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System`

nastavit klíč LocalAccountTokenFilterPolicy na hodnotu 1. Tím je vypnuto omezení pro vzdálené řízení uživatelských účtů.

6.1.4 Firewall

Nastavení firewallu pro správnou funkci celého rigu je klíčovou záležitostí a v některých případech je nutné jej dokonce vypnout. Během testování Školy OnLine běžel jen na počítači s controllerem. Firewally na obou serverech byly vypnuté.

Kromě pravidel umožňujících provoz vlastní aplikace (např. porty 1433 a 1434 pro SQL Server, 443 pro HTTPS apod.), je nutné zajistit ještě následující:

[5] Zde je myšlen testovací rig složený ze systémů, které nejsou vytvořeny prostřednictvím Visual Studia Lab Managementu.

Nastavení výjimek pro Team Foundation Server

V případě využití TFS pro testovací rig jsou to porty 8080, 8081, popřípadě 9191 [31] a porty pro příslušný SQL Server.

Nastavení výjimek pro agenty a controller

Porty využívané controllerem a agenty jsou uvedeny části `<appSettings>` v příslušných konfiguračních souborech [32]:

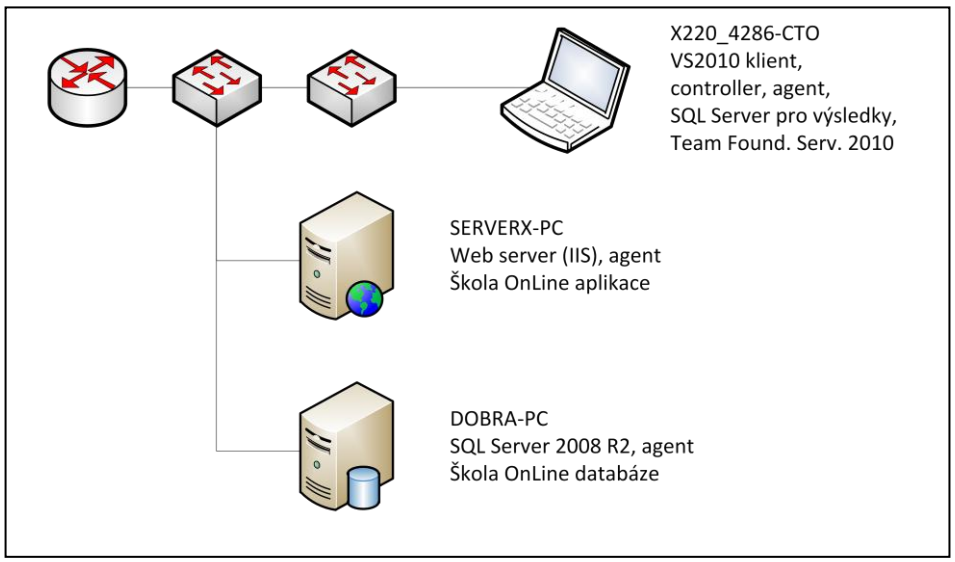
- V souboru `QTCcontroller.exe.config` pro controller. Defaultně jako
  `<add key="ControllerServicePort" value="6901"/>`
- V souboru `QTAgentService.exe.config` pro agenty. Defaultně jako
  `<add key="AgentServicePort" value="6910"/>`

V adresáři `C:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\IDE\`.

Na počítači s agentem tedy příchozí spojení na port 6910 a na controlleru příchozí spojení na port 6901.

6.1.5 Zapojení rigu – topologie sítě

Fyzická topologie zapojení rigu pro testování Školy OnLine je k dispozici na následujícím obrázku.



Zapojení rigu pro testování Školy OnLine

Sestavení testovacího rigu a samotné testování bylo uskutečněno na lokální síti Ethernet typu 1000Base-T (Gigabit Ethernet). Síť tedy poskytovala dostatečnou odezvu pro účely testování webové aplikace.

K dispozici byl skoro celý rozsah privátních IP adres 192.168.1.0/24. Velká jeho část byla využívána při experimentech s funkcí IP switching[6]. V síti bylo sice připojeno ještě několik

[6] Agent využívá při generování zátěže více různých IP adres a tímto způsobem simuluje více klientů testované aplikace.

dalších počítačů, ale jejich provoz byl během testování omezen na minimum a nemohlo tedy dojít ke zkreslení výsledků.

6.1.6 Konfigurace agentů a controlleru

Konfigurace a spuštění agentů a controlleru musí být provedeno pod zvláštním uživatelským účtem, který je na všech počítačích stejný (pro podrobnosti viz kapitola 6.1.3). Konfigurací je zde myšleno nastavení této služby v rámci grafického rozhraní aplikací Visual Studio Test Controller 2010 a Visual Studio Test Agent 2010. Služby samotné, tedy *VSTTController* a *VSTTAgent*, lze nastavit na automatické spouštění pod zmíněným jednotným uživatelským účtem.

Nastavení vztahující se přímo k průběhu testování je proveditelné z grafického prostředí VS2010.

Weighting

Nastavení weighting určuje poměr rozložení váhy generované zátěže mezi agenty. Vzhledem k omezení licencí a testování Školy OnLine jako *local execution with remote collection* nemá jeho přiřazená hodnota význam.

V případě testování způsobem *remote execution* by bylo vhodné nastavit na obou serverech hodnoty 0, aby negenerovaly zátěž sami sobě a nedocházelo tím ke zkreslení výsledků. Váhu na ostatních agentech následně rozložit libovolně dle vlastní potřeby.

IP switching

IP switching je funkce, díky které agent používá k odesílání požadavků na cílový server IP adresy z předem definovaného rozsahu a tak simuluje více klientů (více zdrojů zátěže).

Během testování Školy OnLine bylo provedeno několik experimentů s cílem zjistit vliv nastavení přepínání IP adres. Prokázalo se, že výkon strojů nebyl nijak ovlivněn a nebylo pozorováno ani žádné zpoždění.

IP switching má přínos především při testování zapojení s reverzní proxy s vyvažováním zátěže (load balancingem) mezi více servery [33]. Je také možné jej zapínat a vypínat přímo v nastavení load testu u každého scenario. Tato hodnota je defaultně nastavena na true, tedy zapnuta.

Atributy

Atributy byly na controlleru i v souboru .testsettings nastaveny následujícím způsobem:

Agent Name	Attribute Name	Attribute Value
SERVERX-PC	Web Server	True
DOBRA-PC	SQL Server	True
X220_4286-CTO	Agent	True

Jak již bylo v předchozím textu zmíněno, testování Školy OnLine nebylo možné provádět jako *remote execution*. Využití funkce generování zátěže více agenty najednou není nezbytné v případě simulace malého množství uživatelů. Vzhledem k přenosové kapacitě lokální sítě, výkonu web serveru a počítače použitého jako klient, controller i agent je pro účely testování

Školy OnLine jeden zdroj zátěže dostatečný. Nutno ještě dodat, že VS2010 během testování jako *local execution with remote collection* vůbec nevyužívá agenta instalovaného na X220_4286-CTO. K testům je využíván jen VS2010 klient, controller a pro sběr dat ještě agenti na obou serverech.

Emulace sítě

Vzhledem k charakteru Školy OnLine, tedy aplikace běžící v drtivé většině případů na lokální síti školy, nebo jiné instituce připojené k páteřní síti, nebylo během testů využíváno možnosti emulace pomalejších sítí, které VS2010 nabízí (DSL, 3G, interkontinentální WAN apod.).

6.1.7 Správa prostředí v MTM Lab Center a TFS Lab Management

Přestože MTM Lab Center a TFS Lab Management poskytují možnost snadno spravovat virtuální i fyzické prostředí, nebyla tato funkce během testování Školy OnLine nakonec využita. Vedly k tomu hned dva důvody:

1) Pro správu virtuálního prostředí je potřeba připojení TFS k SCVMM serveru, který vyžaduje pro svůj běh operační systém Windows Server a pro správu virtuálních strojů pak i Hyper-V server. Již na začátku prací bylo provedeno několik pokusů zprovoznit Windows Server 2008 R2 SP1 společně s SCVMM 2008 R2, bohužel však neúspěšně. V té době ještě nebyly k dispozici výše uvedené desktopové počítače jako servery a tato konfigurace vykazovala na stroji ThinkPad X220 určité známky nekompatibility, které se nepodařilo blíže identifikovat ani odstranit. V pozdější fázi, když se podařilo pro projekt získat dva zmíněné servery, byla časová náročnost, náklady i rizika spojená s kompletním přepracováním konfigurace celého testovacího prostředí příliš vysoká.

2) Pro správu fyzického prostředí potřeba přidání test controlleru k TFS serveru. Správa celého testovacího rigu je pak prováděna pouze prostřednictvím TFS. Během práce jevil TFS známky nestability a byl často nedostupný. Tyto potíže se bohužel nepodařilo zcela odstranit a přenechat řízení testů jen na tomto serveru by ohrozilo průběh celého projektu.

6.2 Postup prací

VS2010 umožňuje při implementaci testů velké množství nastavení a použití mnoha doplňků. Ať již jde o různé pluginy, validační pravidla, nebo pravidla pro extrakci informací, vždy jejich provádění a vyhodnocování zabere procesoru určitý čas. U web testů, kdy v jednom okamžiku běží jen jeden test a zátěž systému je minimální, může být čas odezvy sice nepatrně ovlivněn, ale na výsledek testu z hlediska správnosti vliv nemá. Pokud tedy neprovádíme výkonnostní testování a neměříme čas odpovědi na požadavek stránky, jejich užití nic nebrání. Odlišná situace nastává v případě, že tyto web testy skládáme do load testů. Zde může v důsledku používání podobných funkcí dojít k ovlivnění výkonu a zkreslení výsledků. Jejich aplikace byla ve všech případech důkladně zvážena a byla provedena jen pro nezbytně nutné ověření správnosti vráceného výsledku.

TestProject a .NET Framework 4.0

Na rozdíl od Školy OnLine, jež je v .NET Framework 3.5, byl testovací projekt založen na .NET 4.0, který je ve VS2010 pro testovací projekty defaultní. Navíc, pokud je testovací projekt převeden z VS2008 do VS2010, je automaticky změněn i jeho framework z 3.5 na 4.0. K převedení testovacího projektu zpět na 3.5 je nezbytné mít ve VS2010 doinstalovaný Service Pack 1 [34].

6.3 Manuální testování

Práce na manuálních testech byly kompletně prováděny v aplikaci MTM.

Při tvorbě testů Školy OnLine bylo využíváno jak parametrů pro automatické vyplňování formulářů, tak i tzv. *shared steps*, tedy sdílených kroků. Jejich využití se velmi osvědčilo při často opakovaných činnostech, jako je například přihlašování do aplikace nebo nastavení období v kalendáři. Při tvorbě nového testu nemusí být kroky vytvářeny znovu a jsou do postupu vloženy i s popisem očekávaných výsledků, případně pořízeným záznamem.

Nevýhodou zde popsaného testování je skutečnost, že i přes pokročilou úroveň automatizace vyžaduje průběh testu kontrolu nad kurzorem myši a focus. Tester tedy nemůže během testu vykonávat jinou práci a může jen sledovat. Vlastní přehrávání je navíc poměrně pomalé.

Přes velmi dobrou použitelnost a ovládání aplikace MTM bylo v rámci testování zkompletováno jen asi dvacet vzorových testů. Několik z nich bylo dále převedeno na coded UI testy, které poskytují při testování grafického rozhraní mnohem více možností a jejichž kód jde podle potřeby upravovat. Pro tvorbu coded UI testů je však mnohem vhodnější a efektivnější využít nástroj Coded UI Test Builder, který je ale již součástí VS2010.

Při generování kódu z nahrávek manuálních testů použije vždy VS2010 názvy kroků v test plánu jako názvy objektů a metod ve zdrojovém kódu, a to včetně české diakritiky. V případě vytváření testovacích plánů v MTM je tedy nutné s touto vlastností počítat.

Vytvořené manuální testy byly v rámci testovacího plánu *SOL_test_plan_1* rozděleny do dvou *test suits* dle popisu v kapitole 5.1.

6.4 Coded UI Testy – automatické testování uživatelského rozhraní

Doporučení pro tvorbu coded UI testů

Během prací na coded UI testech pro Školu OnLine se osvědčilo dodržování několika zásad:

- Vytvářet coded UI testy ve VS2010 nahráváním pomocí aplikace *Coded UI Test Buider*, tedy nevyužívat nahrávky pořízené v MTM.
- Využívat pro testy více UIMap, vždy jednu mapu pro podobné testy zaměřené na jednu oblast (viz kapitola 0).
- Při nahrávání vytvářet více krátkých částí. To je výhodné při zpětných úpravách a změnách webových stránek.

- Ověření (tzv. assert) vždy vytvořit k ID daného prvku stránky. Testy jsou tak více odolné vůči změnám stránek a jsou funkční i v případě spuštění testu například v jiné jazykové lokalizaci stránek.

Použití více UI Map v projektu

Coded UI Test Builder používá defaultně při vytváření testů jen jednu UIMapu, která je pro všechny testy společná. Pro práci na rozsáhlejších projektech je vhodnější použít více různých UI Map [35]. V projektu se potom lépe orientuje a navíc je z takového uspořádání možné vytěžit další výhody. Jeden test (jeden soubor CodedUITest1.cs) může pro svůj běh využívat více UIMap. Lze si tak ušetřit práci a při tvorbě více testů využívat ve všech například jednu UIMapu pro přihlášení uživatele. Do zdrojového kódu se pak vždy jen přidá příslušná část kódu a potřebná reference.

Následující ukázka je z testu tvorby rozvrhu. Uvedený příklad také využívá souboru Prihlaseni.csv jako externího zdroje dat (viz kapitola 3.1.6). V případě Coded UI testů musí být však heslo v takovém .csv souboru uloženo již přímo jako jeho otisk. Naopak u web testů může být v externím zdroji uloženo ve své původní textové formě.

…

```
using TestProjectSOL.CodedUITestyZakladni.UIMapyZakladni.UIMapPrihlaseniDoSOLClasses;
…

[DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV", "D:\\3kola\\favka\\DIP
\\CCA\\SOL K0661.0\\TestDataSource\\Prihlaseni.csv", "Prihlaseni#csv",DataAccessMetho
d.Sequential), TestMethod]
public void CodedUITestMethodTvorbaRozvrhu()
    {
// Spusteni IE a nacteni uvodni stranky
    this.UIMapPrihlaseniDoSOL.OtevriIEaSOL();
// Overeni nacteni uvodni stranky pro prihlaseni do SOL
    this.UIMapPrihlaseniDoSOL.OverUvodniStranu();
// Prihlaseni do systemu SOL s vyuzitim CSV souboru
    this.UIMapPrihlaseniDoSOL.VyplnJmenoHesloParams.UIUwtctl00JmenoUzivateEditText =
TestContext.DataRow["Jmeno"].ToString();
    this.UIMapPrihlaseniDoSOL.VyplnJmenoHesloParams.UIUwtctl00HesloUzivateEditPasswor
d = TestContext.DataRow["Heslo"].ToString();
    this.UIMapPrihlaseniDoSOL.VyplnJmenoHeslo();
    this.UIMapPrihlaseniDoSOL.KlikniTlacitkoPrihlasit();
// Overeni prihlaseneho uzivatele s vyuzitim CSV souboru
    this.UIMapPrihlaseniDoSOL.OverPrihlasenehoUzivExpectedValues.UIAdamecAdamHyperlin
kInnerText = TestContext.DataRow["Overeni"].ToString();
    this.UIMapPrihlaseniDoSOL.OverPrihlasenehoUziv();
// Otevreni Menu Rozvrh - Tvorba rozvrhu - Aprobace ucitele (test jiz dale vyuziva
UIMapTvorbaRozvrhu)
    this.UIMapTvorbaRozvrhu.OtevriTvorbaAprobace();
…
```

Na konci souboru je nezbytné přidat ještě následující kód:

…

```
public UIMapPrihlaseniDoSOL UIMapPrihlaseniDoSOL
{
   get
   {
      if ((this.map == null))
```

```
            {
                this.map = new UIMapPrihlaseniDoSOL();
            }
        return this.map;
    }
}
private UIMapPrihlaseniDoSOL map;
…
```

Automatizace coded UI testů

Coded UI testy je možné spouštět i automaticky prostřednictvím agenta jako součást load testu. Agent musí být nastaven, aby neběžel jako služba, nýbrž jako interaktivní proces. V rámci testování Školy OnLine nebylo možné do zátěžového testování coded UI testy zahrnout. Musejí mít totiž při běhu přístup k pracovní ploše (tj. kontrolu nad kurzorem myši a focus) a tudíž vyžadují pro jednoho virtuálního uživatele i jednoho fyzického agenta.

6.5 Web testy – testování na úrovni HTTP

Web testy byly rozděleny do kategorií odpovídajícím jednotlivým položkám menu.

K web testům bylo v rámci projektu testování Školy OnLine přistupováno již od počátku jako k základním stavebním kamenům pro load testy. Možnost s testy dále pracovat byla zásadním kritériem při jejich tvorbě a důraz byl kladen především na variabilitu a univerzálnost. Těchto vlastností lze dosáhnout dodržením několika doporučení:

- Spojovat požadavky, které spolu nějakým způsobem souvisí, do transakcí. Ty sice nemají vliv na nastavení testu, ale slouží k usnadnění práce a lepší orientaci v jeho struktuře.
- Použít jen nezbytná pravidla pro validaci a extrakci.
- Validační pravidla vždy cílit na prvek, který vypovídá o správnosti odpovědi a zároveň je co možná nejmenší měrou závislý na jazykové verzi stránky.
- Vždy parametrizovat web server (viz kapitola 6.5.4).

6.5.1 Přihlášení do Školy OnLine

Vzhledem k charakteru Školy OnLine je vhodné začít každý jednotlivý web test přihlášením a změnou období, respektive jeho potvrzením na stránce pro změnu období. Vedou k tomu hned dva důvody:

- Škola OnLine se vždy po přihlášení načte v příslušném školním roce a dni. V případě spuštění testů po skončení aktuálního školního roku nemusí některé z nich správně fungovat, jelikož mohou být závislé na konkrétních datech z databáze, platných jen pro daný rok (například rozvrh, složení tříd apod.).

V případě potřeby je u web testu, vytvořeného zmíněným způsobem, možné snadno období změnit automatickým vložením *Form Post Parametru* `ddlSkolniRok`, odpovídajícímu požadovanému období.

6.5.2 Data driven web test

Z možnosti řízení web testů daty z databáze plyne řada výhod a v případě jejich efektivního využití pro load test jde do jisté míry i o nutnost. Největší benefit poskytují vlastnosti *Access Method* a *Advanced data cursor* zmíněné v kapitole 3.1.6.

Jako dobrý příklad poslouží jednoduchý web test *WebTestVypisRozvrhuRandomTridy.webtest*, jehož funkcí je otestovat stránku pro výpis rozvrhu třídy. Záznam testu byl pořízen s třídou V1, jednoznačně v aplikaci identifikovatelnou řetězcem A3383 (v databázi VYV je údaj uložen v tabulce CCAK_STUDIJNI_SKUPINA, ve sloupci SKUPINA_ID). Tento údaj je uložen v požadavku

```
{{TestWebServer}}/vyvoj/Katedra/App/Rozvrh/KRO003_VypisTridy.aspx
```

ve form post parametru DDLTrida. Pokud je nahrazen o konkrétní třídě odkazem na tabulku Tridy20112012 v databázi VYV_TEST

```
DDLTrida={{SolDboDataSourceVYV_TEST.Tridy20112012.SKUPINA_ID}}
```

a access method u SolDboDataSourceVYV_TEST nastavena na random, bude při dalším spuštění testu vybrána třída, jejíž rozvrh se následně zobrazí, náhodně.

6.5.3 Využívání smyček

Předchozí příklad lze dále rozšířit o přidání smyčky. Pokud jsou požadavky na stránku pro výběr třídy `http://localhost/vyvoj/Katedra/App/Rozvrh/KRO003_VypisTridy.aspx` vloženy do smyčky a je nastaven *advanced data cursor* na true, bude při každém jejím opakováním v rámci jednoho testu načten rozvrh jiné třídy. Test tak simuluje činnost uživatele, který postupně prochází rozvrhy všech tříd uvedených v tabulce Tridy20112012.

Využití smyček nemusí být vždy vhodné. Pokud běží web test sám o sobě a ne jako součást load testu, je celý jeho výsledek udržován v paměti. Spouštění testu s velkým množstvím opakování může způsobit problémy s nedostatkem paměti [36]. Dalším důležitým faktem je, že web testy, ať už sami o sobě, nebo jako součást load testu, by měly sloužit k pokud možno reálné simulaci práce uživatele.

V průběhu testování Školy OnLine nebyly používány smyčky s více jak deseti opakováními. Obvykle byly použity k opakování činností typu zapsání docházky, výpis absence nebo rozvrhu, vyhledávání ve školní matrice apod.

6.5.4 Parametrizace web serveru

Parametrizace serveru je náhrada jeho názvu proměnnou, jejíž hodnota je následně přiřazena v context parametru. Tento krok usnadňuje spuštění testu, pokud testovaná aplikace běží na jiném serveru, než na kterém byl test vytvořen. Všechny testy Školy OnLine mají stejné jméno parametru *TestWebServer*. Context parametr je pak následující:

```
TestWebServer=http://localhost
```

6.5.5 Úprava web testu pro load testy

Vzhledem k nutnosti přihlašování do Školy OnLine popsané v kapitole 6.5.1 je většina vytvořených web testů ve dvou verzích.

- V samostatně funkční verzi obsahující na počátku testu přihlášení uživatele do aplikace a nastavení období. Na jejich konci je uživatel opět odhlášen.
- Ve verzi určené pro load testy, která neobsahuje požadavky na přihlášení a nastavení období. Takto upravený web test má název na konci doplněný o písmena *LT*.

6.5.6 Coded web testy

Stejně jako u coded UI testů, disponuje VS2010 funkcí pro generování kódu i z existujících web testů. Jakmile je kód takto vygenerován, není možné jej měnit prostřednictvím Coded UI Test Builderu. Všechny změny je nutné provádět buď ručně, nebo kód znovu vygenerovat. Proto je vhodné mít před samotným procesem generování nastaveno v testu maximum vlastností.

6.6 Ordered testy

Plán pro využití Ordered testů pro vytváření pokročilé konfigurace load testů nebylo možné uskutečnit. Ordered testy se v load testu nechovají podle očekávání. Ordered test pracuje sám o sobě správně. Pokud je však vložen do load testu, skončí chybou a není k dispozici ani žádný log. Výsledek web testu, jak je k dispozici například při samostatném běhu web testu nebo ordered testu, je prázdný bez jakýchkoli informací o průběhu nebo chybě.

Jejich využití pro seskupení požadovaných web testů nebylo také možné podle původního plánu. V ordered testu totiž není vytvořen virtuální uživatel, který by měl jednu svoji session a přiřazenou cache. Testy jsou jednoduše pouštěny jeden za druhým, jako při samostatném spuštění. Ordered testy tedy nebyly složeny z LT variant pro load testy, ale z variant obsahující přihlášení i odhlášení uživatele.

Pro každý oddíl web testů byl vytvořen jeden ordered test. Testy v něm jsou prováděny v abecedním pořadí.

6.7 Load testy – zátěžové testování

6.7.1 Nastavení load testů

Nastavení load testů bylo uzpůsobeno dosažení cílů stanovených v kapitole 5.3. Následuje popis několika důležitých funkcí.

Load pattern

Pro zátěžové testování Školy OnLine byly využívány všechny modely zatížení:

- *Constant load pattern* s postupným nárůstem zátěže, u něhož bylo díky vhodně nastavenému warm-up time předejito nereálnému přetížení na počátku testu.

- *Step load pattern* pro stress testy. Postupné přidávání v delších intervalech umožní velmi dobře určit hranici, kdy je ještě počet uživatelů pro aplikaci únosný, a kdy již nikoliv.
- *Goal-Based load pattern* pro ověření nalezeného úzkého místa. Aktivita virtuálních uživatelů je regulována automaticky vzhledem k určenému ukazateli (například času procesoru). Tomuto modelu je dále věnována podkapitola 6.7.3.

Initialize a terminate testy

U každého test mixu lze nastavit dva zvláštní druhy testů. *Initialize* a *terminate* poskytují možnost definovat první a poslední test, který každý uživatel vykoná. V případě Školy OnLine je jejich funkce vhodná pro přihlášení a odhlášení uživatele. Pro přihlášení byl vytvořen test *WebTestPrihlaseniTridniUcitel.webtest*, využívající jako zdroj dat tabulku z databáze VYV_TEST *PrihlaseniTridniUcitelOK*. Třídní učitelé jsou z ní vybíráni náhodně a to v kombinaci například s výše zmíněným testem *WebTestVypisRozvrhuRandomTridy.webtest* poskytuje dobrou variabilitu v chování každého virtuálního uživatele. Aby test neselhal kvůli nedostatečnému oprávnění učitele, je každý třídní učitel vždy přiřazen u ostatních tříd jako zástupce.

Test mix model

Ze čtyř možných variant test mix modelů byly využívány především následující dva:

- *Test mix percentage based on the number of tests started*, určený původně pro simulaci kontinuální intenzivní zátěže, později pro všechny testy. Virtuální uživatel je v tomto případě přihlášen do Školy OnLine buď na začátku testu (v rámci initialize testu) a pracuje po celou dobu jeho trvání, nebo se chová jako nový uživatel (viz *Procento nových uživatelů*).
- *Test mix based on user pace*, který byl určen pro věrnější napodobení činnosti skutečných uživatelů. Virtuální uživatel není přihlášen na začátku testu a odhlášen na jeho konci využitím initialize a terminate testů, nýbrž vystupuje jako nepřihlášený uživatel, který provádí web testy, jejichž součástí je přihlášení i odhlášení. V rámci load testu je web test volán jednou za hodinu či za dvě. Pokud by byl uživatel přihlášen jen jednou na začátku testu, aplikace by jej během dlouhé nečinnosti automaticky odhlásila.

U load testů nebylo nakonec požadovaných výsledků dosaženo plánovanými postupy a nastavením VS2010. Konkrétně aplikace modelu zátěže založené na počtu testů za hodinu na uživatele (test mix based on user pace), která se z počátku zdála jako ideální, se ve výsledku ukázala jako nevhodná. Při běhu testu s tímto modelem zátěže vyhodnocovalo VS2010 velké množství testů jako neúspěšných, aniž by byl procesor vytížen. Průměrně se hodnota jeho využití pohybovala mezi 40 a 50 %. Pokud byla však definována stejná zátěž modelem, založeném na procentu četnosti výskytu jednotlivých testů (*test mix percentage based on the number of tests started*), test probíhal podle očekávání. Zmíněná konfigurace byla uložena jako test *LoadTestSimulaceSOL_II.loadtest*.

Procento nových uživatelů

Procento nových uživatelů udává, kolik virtuálních uživatelů se bude neustále připojovat jako nových. Nový uživatel, ve zde uvedeném významu, provede jen jeden web test a odpojí se (simulace návštěvníka). V případě testů Školy OnLine je typicky proveden initialize test

přihlášení, jeden test vybraný z test mixu a následné odhlášení v rámci terminate testu. Virtuální uživatel se během celého průběhu load testu již do aplikace nepřihlásí.

Zbytek uživatelů je přihlášen na začátku testu a odhlášen na konci, napodobujíc tak činnost zaměstnanců.

Procento nových uživatelů Školy OnLine je v porovnání například s elektronickými obchody malé. Pro účely testování byla stanovena hodnota mezi 0 až 5%, v závislosti na způsobu provádění testů. Pokud byly testy prováděny ve variantě určené pro load testy, tedy bez přihlášení a odhlášení obsaženém v každém testu, byl dodatečný výskyt přihlašujících se a uživatelů zajištěn právě tímto parametrem.

Browser mix

Mix prohlížečů byl nastaven pro všechny load testy stejně, na IE 8.0. Jejich výběr je ve VS2010 poměrně omezený. IE 9 je bohužel dostupný až v novém VS2012; a například simulace prohlížečů Firefox a Safari, které by vzhledem k reálné kompatibilitě připadaly v úvahu, jsou k dispozici pouze pro verzi 3 (aktuální verze je 13.0.1 u Firefoxu a 5.1.2 u Safari).

Context parameters

V každém load testu je možné vytvořit libovolný context parametr. Tím lze hromadně nastavit vlastnosti jednotlivým dílčím testům, ze kterých je load test složen. V testech Školy OnLine je u všech využit parametr pro specifikaci web serveru:

```
TestWebServer=http://192.168.1.12
```

Místo IP adresy může být zadán i přímo název stroje (malými písmeny)

`TestWebServer=serverx-pc`, nebo jen `localhost`

Vlastnost Parse Dependent Requests

Vlastnost P*arse Dependent Requests* určuje, zda jsou v rámci parsování daného požadavku na stránku odesílány na server i druhotné požadavky, například na obrázky, CSS apod.[7] [37]. Defaultně je vlastnost na hodnotě true, tedy zapnutá, a druhotné požadavky jsou na server odesílány. Takové chování testu je vhodné v případě snahy o co nejrealističtější napodobení zátěže web serveru. Pokud je ovšem cílem testu zaměřit se na výkon aplikace a na práci s dynamickým obsahem, tedy v případech kdy není nutné zatěžovat server statickými obrázky, může být vhodné funkci vypnout [38].

6.7.2 Nastavení sledování výkonu

Potíže s jejich konfigurací jsou podrobně popsány v kapitole 6.9.4. Během testování bylo jejich využití následující:

Web server

Byl sledován s využitím předpřipravených counter setů ASP.NET a IIS.

[7] Rozdíl mezi *Parse Dependent Requests* a *Cache Control* je, že cache control vypíná v testu druhotné požadavky až poté, co na ně poprvé obdrží odpověď, zatímco parse dependent requests vypíná tyto požadavky zcela.

SQL server

Od počátku testování Školy OnLine byly se sledováním SQL serveru problémy. Defaultní countery se jen málokdy načetly a nepomohly ani jejich úpravy. Relativně uspokojivých výsledků bylo dosaženo až při vytvoření vlastních [39] a [40]. Sledovány byly především následující ukazatele:

- MSSQLServerSOL: SQL Statistics: Batch Requests/Sec
- MSSQLServerSOL: SQL Statistics: SQL Compilations/Sec
- MSSQLServerSOL: SQL Statistics: SQL Re-Compilations/Sec
- MSSQLServerSOL: SQL Statistics: General Statistics
- MSSQLServerSOL: SQL Errors: Errors /Sec
- MSSQLServerSOL: Databases: Log Flushes/sec
- MSSQLServerSOL: Databases: Log Truncations
- MSSQLServerSOL: Databases: Transaction/Sec
- MSSQLServerSOL: Databases: Active Transaction
- MSSQLServerSOL: Buffer Manager: Buffer Cache Hit Ratio
- MSSQLServerSOL: Buffer Manager: Lazy Writes/Sec
- MSSQLServerSOL: Buffer Manager: Checkpoint Pages/Sec
- MSSQLServerSOL: Buffer Manager: Page Life Expectancy

Controller

V případě controlleru nebylo třeba nic měnit. Defaultní countery poskytovaly dostatečné množství informací o stavu systému.

Interval vzorkování

Pro testování Školy OnLine byly ve většině případů dodrženy obecně doporučované intervaly vzorkování [41]:

Délka load testu	Doporučený interval vzorkování
< 1 hodina	5 s
1-8 hodin	15 s
8-24 hodin	30 s
> 24 hodin	60 s

Pro objektivní sledování výkonu celého rigu je důležité omezit množství aktivních data adaptérů na nezbytné minimum. Konkrétně *Test Impact* nebo *IntelliTrace* diagnostika mohou způsobit zkreslení výsledků.

6.7.3 Nastavení Goal-Based

Pro nastavení Goal-Based load pattern je důležité zvolit delší dobu pro vzorkování. Přidávání a odebírání uživatelů během testu je prováděno vždy v okamžiku zjištění stavu, tedy pořízení vzorku. Pokud jsou vzorky příliš časté a vytížení procesoru se rychle mění, může dojít k tomu, že je výsledný load test složen pouze z web testů přihlášení uživatele, jako tomu bylo v případě Školy OnLine. Nastavením dlouhých intervalů vzorkování a omezení navýšení jen na jednoho uživatele v jednom okamžiku bylo dosaženo o něco lepšího průběhu. Pro zjištění počtu uživatelů při optimálním zatížení Školy OnLine se však Goal-Based pattern neosvědčil.

6.7.4 Load test results repository

SQL Server Expres databáze automaticky vytvořená VS2010 má maximální velikost omezenou na 4 096 MB dat. Proto bylo vhodné vytvořit ukládání výsledků novou databázi v SQL Server 2008 R2 Enterprise edici. Součástí instalace VS2010 je i SQL skript, který je možné si pro své vlastní potřeby upravit a použít. Tento skript je defaultně uložen ve složce s instalací:

```
c:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\IDE
```

Samotný příkaz pro vytvoření může být pak následující:

```
SQLCMD /S localhost\MSSQLSERVER2008 -U sa -P skola –i
loadtestresultsrepository.sql
```

Mazání výsledků z repositáře trvá (v závislosti na velikosti dat) velmi dlouho, řádově až desítky minut. Během této doby je VS2010 nepoužitelné a není přístup ani k jeho nastavení apod.

6.7.5 Load testy a ASP.NET profiler diagnostic adapter

Během load testu s tímto adaptérem jsou získávána a následně ukládána data přímo na serveru, na kterém webová aplikace běží. Tyto .vsp soubory jsou defaultně umístěny do složky `C:\Users\TestAgentUserAccount\AppData\Local\Temp` a můžou nabývat nezanedbatelných rozměrů. Například šestnáctihodinový zátěžový test Školy OnLine s šedesáti uživateli a 30s vzorkováním běžící na ServerX-PC nashromáždil 32 GB dat uložených v jediném souboru s názvem *ServerX-PC-w3wp-20120624-115620.vsp*.

Takto dlouhý test však není pro profilování ASP.NET nezbytný. Je doporučeno adaptér nepoužívat pro testy delší než jednu hodinu [42].

6.8 Unit testy

Unit testování Školy OnLine bylo čistě experimentálního charakteru. Jako vhodný objekt pro unit test byla vybrána metoda *KontrolaPristupuNaUlohu* ze třídy Security, umístěná v `\Common\BusinessCommon\Security.cs`. Metoda provádí kontrolu, zda má daný uživatel přístup k požadované úloze.

Metoda Security.KontrolaPristupuNaUlohu:

…

```
public static SecPristup PristupNaUlohu(string sDatabaseID, string roleOsoby, string
ulohaID) {
  if (WebConfigurationManager.AppSettings.Get("Instance") == "VYVOJ")
    return SecPristup.ALLOW;

  DataSet dsUlohy = new DataSet();
  dsUlohy = SOL.DBCommon.Common.GetUlohyRole(sDatabaseID, roleOsoby, ulohaID);

  if (dsUlohy.Tables[0].Rows.Count > 0)
    return SecPristup.ALLOW;

  return SecPristup.DENYAUTH;
}
```

…

Vytvořený unit test (SecurityTest.cs):

```
…

public void KontrolaPristupuNaUlohuTest()
{
    User user = new User("ada"); //
    string ulohaID = "KSK009"; //
    bool overDleOsobaIDOriginal = true; //
    SecPristup expected = new SecPristup(); //
    SecPristup actual;
    actual = Security.KontrolaPristupuNaUlohu(user, ulohaID, overDleOsobaIDOriginal);
    Assert.AreEqual(expected, actual);
    Assert.Inconclusive("Verify the correctness of this test method.");
}
…
```

6.9 Troubleshooting – Problémy při implementaci

Podkapitola Troubleshooting – Problémy při implementaci představuje vybrané problémy řešené během testování Školy OnLine.

6.9.1 Problém s parametrem _LASTFOCUS

Při vytváření web testu ve VS2010 se vždy objeví problém s parametrem __LASTFOCUS. Po ukončení nahrání jednotlivých kroků se IE zavře, VS2010 provede build nového testu, spustí jej a začne automaticky detekovat dynamické parametry. Následně oznámí, že nebyly detekovány žádné parametry a přehrávaný test skončí neúspěšně.

Chyba se vyskytuje nejčastěji hned na stránce http://localhost/vyvoj/katedra/Prihlaseni.aspx při odesílání požadavku na přihlášení a znemožní tak pokračování v jakémkoli dalším testování aplikace.

Informace v okně Test Result:

```
Error  WebTestZadavaniDochazkyM1  TestProjectSOL

The Web test WebTestZadavaniDochazkyM1 was stopped because it encountered an
excessive number of exceptions while attempting to submit requests
```

V okně vyhodnocení testu se místo vykreslení požadované stránky objeví zpráva:

```
Request failed: Context parameter '$HIDDEN1.__LASTFOCUS' not found in test
context
```

Detail výjimky je pak následující:

```
Microsoft.VisualStudio.TestTools.WebTesting.WebTestException: Context
parameter '$HIDDEN1.__LASTFOCUS' not found in test context
   at
Microsoft.VisualStudio.TestTools.WebStress.HelperMethods.UpdateBindingSites(We
bTestCaseContext testCaseContext, String preBoundString)
   at
Microsoft.VisualStudio.TestTools.WebStress.WebTestInstrumentedTransaction.Appl
yCorrelationBindingToParameter(Parameter param)
   at
Microsoft.VisualStudio.TestTools.WebStress.WebTestInstrumentedTransaction.Perf
ormRequestDataBinding()
```

```
   at
Microsoft.VisualStudio.TestTools.WebStress.WebTestInstrumentedTransaction.Exec
ute(WebTestCaseContext testCaseContext, AsyncCallback completionCallback,
Object callerState)
```

Parametr __LASTFOCUS ukládá jméno prvku, který měl na příslušné stránce focus jako poslední, a po následném provedení postback je tomuto prvku navrácen.

Jako nejjednodušší a nejrychlejší řešení se osvědčilo odstranění parametru __LASTFOCUS={{$HIDDEN1.__LASTFOCUS}} z příslušného požadavku. Tedy jeho smazání z větve *Form Post Parameters* a následné znovuspuštění testu.

V případě nahrávání stejných web testů v novém VS2012 RC se problém s parametrem LASTFOCUS již neobjevuje.

6.9.2 KKA001_KalendarTyden,Katedra.ashx a common.ashx

Problém s modulem kalendáře se ve VS2010 objevuje, stejně jako předchozí problém s parametrem LASTFOCUS, u všech nahraných web testů a dochází k němu pokaždé při načítání hlavní stránky s kalendářem

```
http://localhost/vyvoj/katedra/App/Kalendar/KKA001_KalendarTyden.aspx.
```

Web test enginu je sice serverem tato stránka vrácena se stavovým kódem 200 OK a všechny prvky se zobrazí včetně kalendáře tak jak mají, druhotné požadavky na soubory *KKA001_KalendarTyden,Katedra.ashx* a *common.ashx* však skončí s následující chybou a celý test je vyhodnocen jako nezdařený.

```
Server Error in '/vyvoj/katedra' Application.

The resource cannot be found.

Description: HTTP 404. The resource you are looking for (or one of its
dependencies) could have been removed, had its name changed, or is temporarily
unavailable. Please review the following URL and make sure that it is spelled
correctly.

Requested URL:
/vyvoj/katedra/ajax/cz.cca.Katedra.App.Kalendar.KKA001_KalendarTyden,Katedra.a
shx
```

nebo

```
Requested URL: /vyvoj/katedra/ajax/common.ashx
```

Této chybě se lze vyhnout změnou nastavení funkce pro parsování druhotných požadavků. Ve vlastnostech požadavku, ve kterém se chyba modulu kalendáře objeví, je proto nutno nastavit *Parse Dependent Requests* na hodnotu False. Během testu pak není po serveru problematický modul vyžadován.

Přes zde uvedené nastavení je hlavní stránka funkční a v průběhu prací nebylo zjištěno žádné negativní ovlivnění průběhu testů upravených podobným způsobem. V případě potřeby je možné přidat některé z takto globálně zakázaných požadavků do web testu ručně.

6.9.3 Spouštění load testu

Pokud se krátce po spuštění load testu objeví ve výsledcích, tedy v okně Test Results, u testu jako výsledek "NOT EXECUTED", není obvykle doplněn o žádné další informace. Během testování Školy OnLine vždy pomohl restart controllerů a všech agentů.

Po dlouhotrvajícím load testu je vhodné provést restart controlleru i příslušných agentů. Například na web serveru se po restartu procesu Visual Studio Test Agent skokově uvolní velká část obsazené operační paměti, řádově GB[8].

Pokud se po spuštění load testu nenačtou všechny performance countery, viz následující kapitola.

6.9.4 Remote performance monitoring

K problému s nenačítáním performance counterů docházelo při testování Školy OnLine velmi často. Při odstraňování potíží se osvědčil následující postup:

1) Zkontrolovat, zda v systému běží služby *Remote Registry* a *Remote Procedure Call (RPC)*.
2) Přiřadit controller správné síťové kartě. Pro podrobnosti viz kapitola 6.9.7.
3) Vypnout firewall, nebo změnit jeho nastavení tak, aby performance countery neblokoval (viz 6.1.4).
4) Přiřadit počítače s agenty a controllerem do stejné pracovní skupiny.
5) Zkontrolovat, zda komunikace v síti běží na protokolu IPv4 nebo IPv6, nebo jestli je povoleno obojí. Například při spuštění programu *ping* s parametrem cílového počítače zadaným ve formátu doménového jména počítače (tzn. ServerX-PC nebo X220_4286-CTO) byly zprávy echo request a echo reply doručeny s využitím IPv6. Ve VS2010 se na sebe agenti a controller odkazují jak na základě doménového jména počítače, tak i uživatelem definovanou IP adresou. V případě Školy OnLine šlo právě o IPv4 adresu přiřazenou controlleru. Problém byl vyřešen zakázáním IPv6 na síťových kartách příslušných počítačů.
6) Zkontrolovat, zda jsou dostupné perfomance countery ze sledovaného systému v Performance monitoru systému Windows, kde je instalovaný controller (v Administrative tools – Performance Monitor).
7) Ve Visual Studiu přidat v Server Exploreru všechny zúčastněné servery (tj. stroje na kterých beží agenti a controller) a zadat k nim uživatele a hesla (bylo nutné při každém spuštění VS2010).
8) Nastavit *Role* v aktuálním souboru *.testsettings* na *Local execution with remote collection* nebo na *Remote execution*).
9) Nastavit performance counter sety a countery v příslušném load testu (přidat je jako custom přímo z cílového počítače).
10) Ověřit připojení agentů ke controlleru (v menu Test – Manage Test Controller).

[8] Během prováděných testů se množství takto okamžitě uvolněné paměti pohybovalo v rozmezí i 5 až 6 GB.

Pokud se load test spustí a nenačtou se některé performance countery, je třeba test zrušit a restartovat controller i agenty, popřípadě zkontrolovat, zda jsou příslušné countery stále dostupné skrze formulář Manage Counter Sets v okně load testu.

6.9.5 Web performance testy a spouštění v local.testsettings

Při tvorbě a spouštění web testů mohou nastat následující chyby:

```
Failed to queue tests for test run 'Vaclav Javorsky@X220_4286-CTO 2012-04-23
13:07:29' on agent 'SERVERX-PC': Operation is not valid due to the current
state of the object.; The collection-only agent 'SERVERX-PC' failed to
initialize. The agent will not be used for the test run.
```

nebo

```
The process that collects data and diagnostics on agent 'X220_4286-CTO' exited
unexpectedly; no further data and diagnostics collection will occur on this
agent. Test will continue to run. The event log on the agent may contain an
entry with details on the cause of the process death.; Timed out waiting for
agent plugins to initialize.; The test run was stopped or aborted while
waiting for diagnostic data adapters to initialize.
```

Test neproběhne a v okně *Test Result* se pak objeví "NOT EXECUTED" bez dalšího vysvětlení

Chyba byla vždy způsobena nesprávným nastavením v souboru *Test Settings* aktuálně nastaveného jako *Active* (Menu – Test – Select Active Test Settings – výběr například local.testsettings). Ve zmíněném souboru musí být v *Roles – Test execution methods* nastaveno pouze *Local execution* a ne *Remote execution*, ani *Local execution with remote collection*.

6.9.6 Performance profiling web testů

Při profilování výkonu jednotlivých web testů se může objevit následující chyba:

```
PRF0010: Launch Aborted - Unable to start vsperfmon.exe

Profiling started.

Error VSP1737: File could not be opened due to sharing violation:
D:\favka\DIP\CCA\SOL-
K0661.0\SOL\WebTestPrihlaseniAdminRok2011a2012120617.vspPRF0025: No data was
collected.
```

Problém se podařilo vždy vyřešit následujícím postupem:

1) Nastavením v menu Build – Configuration Manager – nastavit konfiguraci příslušného Test Projectu z *Debug* na *Release*, plus následný rebuild.
2) Nastavení Active Test Settings na *Local.testsettings*, nebo jiný .testsettings soubor se způsobem provádění jako local execution (bez remote collection).

6.9.7 Přiřazení controlleru k síťovému adaptéru

Během instalace nemusí být vždy správně přiřazen controller k síťovému rozhraní. Stává se to především v případech, kdy je na počítači více fyzických nebo virtuálních síťových adaptérů. Problém nemusí být hned z počátku patrný. Důsledkem špatného přiřazení pak může být nemožnost načtení performance counterů, popřípadě spuštění samotného testu.

Kterou síťovou kartu bude controller používat, je definováno v souboru QTController.exe.config, defaultně umístěném v adresáři

```
C:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\IDE
```

Změnu je nutné provést jako administrátor počítače. Je složena z následujících kroků [43]:

1) Zastavení služby VSTTController (Visual Studio Test Controller service) z příkazového řádku zadáním příkazu: `net stop vsttcontroller`
2) Otevřením zmíněného souboru QTController.exe.config
3) Přidáním `<add key="BindTo" value="192.168.1.4" />` do sekce `<appSettings>`, kde 192.168.1.4 je konkrétní IP adresa adaptéru, který má controller využívat.
4) Znovu spuštěním služby VSTTController příkazem: `net start vsttcontroller`

Důležité je ještě poznamenat, že na rozdíl od výše uvedeného postupu, který je nezbytné provádět jako administrátor nebo uživatel s administrátorskými právy, pod kterým byl controller nainstalován, jeho spuštění, stejně tak jako spuštění agentů, musí být provedeno pod speciálním uživatelským účtem vytvořeným pro testování (viz 6.1.3).

6.9.8 Chybně nebo vůbec nainstalovaný LAN Emulation Driver

LAN Emulation Driver slouží v kombinaci s VS2010, controllerem a jeho agenty k emulaci různých síťových technologií. Pokud dojde při instalaci driveru v grafickém uživatelském rozhraní VS2010 k chybě, nebo pokud se instalace vůbec nespustí, je k dispozici nástroj pro příkazovou řádku. Jeho umístění je ve složce `C:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\IDE`.

Spuštěním příkazu `VSTestConfig NETWORKEMULATION /DisplayInfo` lze získat informace o stavu ovladače. Při odstraňování problémů se osvědčil postup nejprve ovladač odebrat bez ohledu na to, zda se jeví jako nainstalovaný nebo poškozený, a následně provést opětovnou instalaci.

1) Zadání příkazu `VSTestConfig.exe NetworkEmulation /Uninstall`
2) Zadání příkazu `VSTestConfig.exe NetworkEmulation /Install`
3) Ovladač se následně buď řádně nainstaluje, nebo vypíše konkrétní chybu:

```
Network Emulation Configuration:
                Operating System Check: Microsoft Windows NT 6.1.7601 Service
Pack 1 (64 Bit)
                Installation will occur from a 64 bit process.
                UAC is Disabled.
                Check for existing driver: Done.
                Driver installation files check: Done.
                Network test passed.
                Installing the Network Emulation driver...
                Driver installation: Failed (ERROR=-2147180503)
```

Kód chyby 2147180503 znamená překročený limit počtu aplikací sledujících síťové rozhraní [44]. Windows 7 mají defaultně omezený počet aplikací, které mohou sledovat síťové rozhraní. V případě, že je v systému více takových programů, může při instalaci LAN Emulation Driveru dojít k chybě. Důsledkem špatně nainstalovaného Emulation Driveru nemusí být jen

nefunkční emulace síťových rozhraní ve VS2010, ale i neschopnost spouštět testy, nebo problémy s performance countery.

Tento limit lze navýšit v registrech systému Windows:

Klíč *MaxNumFilters* se nachází ve složce

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Network
```

a jeho hodnota představuje zmíněný limit.

6.9.9 MS Excel a chyba při generování load test reportu

MS Excel dokáže generovat reporty z výsledků load testů. Funkci je možné spustit buď přímo z okna pro prohlížení výsledků ve VS2010, nebo z Excelu ze záložky Load Test. V obou případech může nastat následující chyba:

```
Unspecified error (Exception form HRESULT: 0x80004005 (E_FAIL))
```

Dochází k ní, pokud je během konfigurace nového reportu typu *trend* vybrán více než jeden load test run najednou. Pro zabránění této chyby je třeba vytvořit report jen s jedním během testu a přes možnost *Edit Runs* přidat další.

Další problém s generováním reportů v MS Excel může nastat při spouštění této funkce z VS2010 a přepínání mezi více load test repository databázemi. Chyba se projeví zamrznutím aplikace a v případě přístupu do požadované databáze přímo prostřednictvím menu v Excelu k ní nedochází.

7 Vyhodnocení testů

7.1 Manuální testy a coded UI testy

Navržené a implementované testy uživatelského rozhraní fungovaly správně a nebyly jimi odhaleny žádné defekty. Prezentační vrstva Školy OnLine je velmi dobře zpracovaná a její funkce byla prověřena jak s využitím testů v MTM a VS2010, tak i explorativním testováním v prohlížečích Safari 5.1.2 a Maxthon 3.4.1. Výsuvné i kontextové menu pracuje dobře a bez zpoždění.

7.2 Web testy

U web testů se doba odezvy stránek Školy OnLine pohybovala v rámci limitů uvedených v kapitole 5. Aplikace reagovala uspokojivě a jediné mírné zpoždění bylo registrováno při načítání hlavní stránky s kalendářem; a to již při středně zatíženém serveru. Výrazné, avšak pochopitelné, zdržení také představovalo načítání rozsáhlejších tabulek z databáze, jako například při zobrazování stránky pro prohazování hodin v rozvrhu, nebo při zadávání suplování. Následující tabulka obsahuje časy některých načítaných stránek. Rozdíly mezi web testem spuštěným na lokálním X220_4286-CTO a na Server-X byly zanedbatelné.

Stránka	Doba načítání (s)
Načtení úvodního formuláře pro přihlášení	0,3 – 0,4
Přihlášení do aplikace	1,4 – 1,5
Změna období a načtení hlavní stránky s rozvrhem	4,1 – 7,5
Načítání kalendáře (hlavní strany s rozvrhem)	2,0 – 6,9
Načítání formuláře pro vyhledávání dat ve školní matrice	2,4 – 2,7
Vyhledání žáka ve školní Matrice	3,0 – 3,6
Načítání rozvrhu třídy	2,5 – 3,3
Zápis do třídní knihy	3,2 – 4,1
Výpis ze třídní knihy	0,7 – 1,0
Odhlášení	0,2 – 0,3

Zde uvedené hodnoty se mohou jevit jako poměrně vysoké. Doba odezvy stránek, které nenačítají data z databáze, byla vždy v rozsahu 0,2 až 1,0 s. U provedených load testů[9] se průměr doby odezvy pohyboval mezi 2,3 a 3,1 s. V případě stress testů byly průměrné hodnoty mnohem vyšší, ovšem u nich byl rozhodující. Následující tabulka nejpomalejší stránky při zátěžovém testu s 55 uživateli[10]:

Stránka	Test	Prům. čas
/KDO014_VypAbsPredmTridy.aspx {POST}	WebTestVypisDochazkyVeTridachLT	29,1
/KSU017_VymenyHodin.aspx {POST}	WebTestVymenaHodinLT	22
/Default.aspx	WebTestCiselnikyTridyLT	7,98
/KCI006_SeznamStudentu.aspx {POST}	WebTestVypisSkolniMatriky	6,91

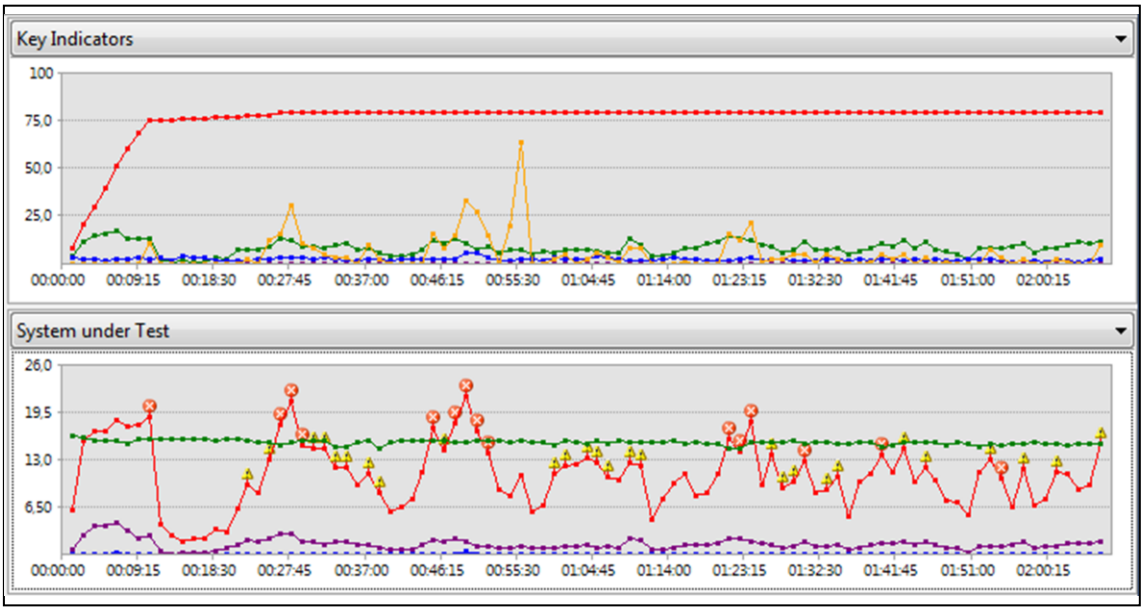
[9] *LoadTestSimulaceSOL.loadtest* a *LoadTestSimulaceSOL_II.loadtest* s 65 uživateli.

[10] Výsledky všech web testů a load testů jsou k dispozici na přiloženém DVD v adresářích *WebTesty_vysledky* a *LoadTesty_vysledky.*

7.3 Load testy

7.3.1 Optimální zátěž

Původní snahou bylo dosažení a udržení stanovené hodnoty 75 % vytížení procesoru zvláštním testem, řízeným goal-based vzorem (viz kapitola 6.7.3). Počet virtuálních uživatelů se však pohyboval mezi 20 a 40 a procesor dosahoval požadovaného vytížení jen výjimečně. Jeho průměrná hodnota se pohybovala těsně kolem 50 %. Příčinou byl měnící se počet zároveň prováděných testů v čase. Vedlo k tomu nastavení testů *LoadTestSimulaceSOL.loadtest* a *LoadTestSimulaceSOL_II.loadtest*, kde byly, v rámci snahy napodobit skutečný provoz, velké pauzy mezi testy každého uživatele. Vznikaly tak špičky a útlumy, jak je patrné například z následujícího obrázku.



Průběh testu *LoadTestSimulaceSOL_II.loadtest* s 80 uživateli

7.3.2 Kritická úroveň zatížení

Zjištění mezního zatížení, kdy už aplikace není schopna odbavovat požadavky klientů ve stanovaném čase, bylo provedeno stress testem *StressTestSimulaceSOL_intezivni.loadtest*, který měl nastaven step load pattern a postupně přidával virtuální uživatele. Jeho průběh je dobře patrný na následujícím obrázku.

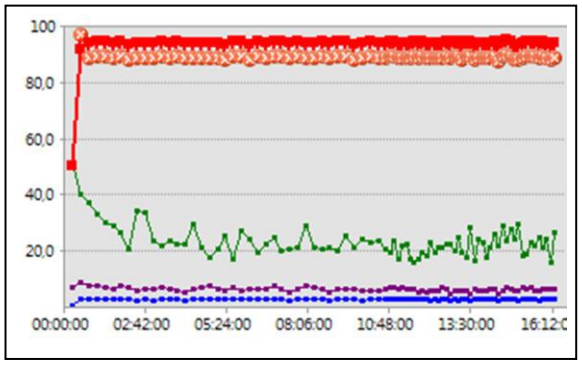
Průběh testu *StressTestSimulaceSOL_intezivni.loadtest*

Již ve 37. minutě byl při 60 uživatelích průměrný čas odpovědi na stránku 7,74 s a začínaly se objevovat první chyby. Jako hranice únosnosti by se pro tento test dala považovat hodnota zhruba 50 uživatelů, kdy byla průměrná doba odpovědi na stránku 4,51 s, a nedocházelo k žádným chybám. Nutno podotknout, že *StressTestSimulaceSOL_intezivni.loadtest* je do jisté míry nereálný model zátěže, kdy každý uživatel nepřetržitě provádí web testy vždy jen s dvou nebo tří minutovou přestávkou. V případě reálného modelu zatížení v testech *LoadTestSimulaceSOL_II.loadtest* a *LoadTestSimulaceSOL.loadtest* se maximální počet uživatelů pohyboval mezi 80 a 85 uživateli.

7.3.3 Vliv kontinuální zátěže

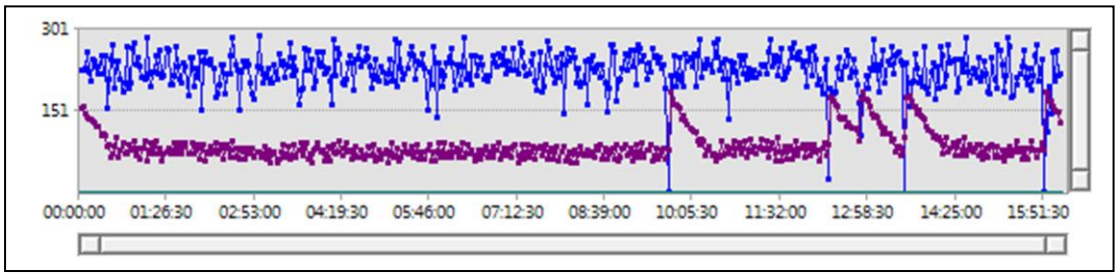
Během dlouhodobých testů, kdy byla aplikace vystavena zátěži alespoň po dobu 24 hodin, nebyly zjištěny problémy se stabilitou aplikace a k chybám docházelo jen výjimečně. Některé z těchto chyb byly způsobeny špatným vyhodnocením VS2010. Web test sice vrátil chybu, ale při prohlídce logu byly všechny dílčí požadavky i odpovědi v pořádku.

Dobře dopadly dlouhodobé stress testy provedené s intenzivní zátěží (viz obrázek vpravo). Mezi jednotlivými testy každého virtuálního uživatele byly jen malé prodlevy, a přestože docházelo k odmítnutí některých požadavků a tedy vyhodnocení těchto web testů jako neúspěšných. Aplikace nápor ustála a i po snížení náporu opět obsluhovala požadavky bezchybně.



Průběh intenzivního testu (16hodin).

Aplikaci se podařilo shodit jen jedním zátěžovým testem s 60 uživateli, který byl vytvořen experimentálně a byl složen jen z omezeného množství web testů, které se spouštěly opakovaně a intenzivně po dobu 16 hodin. Load test obsahoval jen zadávání a výpis docházky, výpis třídní knihy, vyhledávání v matrice a zápis do třídnice.



16 hodinový test s 60 uživateli (modře je čas procesoru, fialově paměť k dispozici)

Chyby, které se při pádech objevily, jsou v následující tabulce:

Typ	Podtyp	Chybová zpráva
Exception	WebTestException	Context parameter '$HIDDEN1.__EVENTTARGET' not found in test context
Exception	WebTestException	Context parameter '$HIDDEN1.__VIEWSTATE' not found in test context
Exception	SocketException	An existing connection was forcibly closed by the remote host
Exception	WebException	The request was aborted: The request was canceled.

Zprávy o nenalezeném context parametru znamenají, že se z předchozí odpovědi nepodařilo extrahovat skryté pole, obsahující zmíněný parametr. Příčinou bylo nejčastěji přesměrování, které sice vrátilo odpověď 200OK, ale zaslaná stránka byla chybová hláška bez hledaného parametru. Krom zmíněných chyb se během pádu objevila ještě následující chyba aplikace:

```
System.ArgumentException: Neplatný argument zpětného odeslání nebo zpětného
volání. Ověření události je povoleno pomocí kódu <pages
enableEventValidation="true"/> v konfiguraci nebo kódu <%@ Page
EnableEventValidation="true" %> na stránce. Pro účely zabezpečení tato funkce
ověřuje, že argumenty událostí zpětného odeslání nebo volání pocházejí z
ovládacího prvku serveru, který je původně vykreslil. Jsou-li data platná a
očekávaná, použijte k registraci dat zpětného odeslání a volání pro ověření
metodu ClientScriptManager.RegisterForEventValidation.

   v System.Web.UI.ClientScriptManager.ValidateEvent(String uniqueId, String
argument)
```

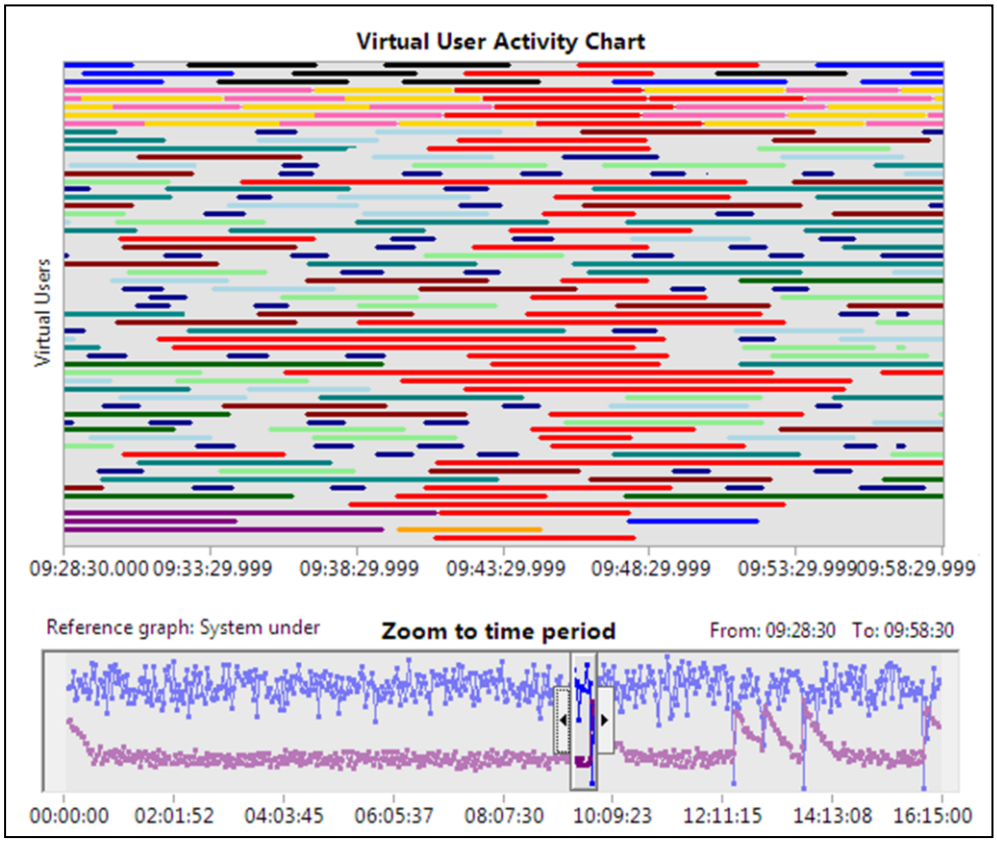
```
   v System.Web.UI.WebControls.DropDownList.LoadPostData(String postDataKey,
NameValueCollection postCollection)

   v System.Web.UI.Page.ProcessPostData(NameValueCollection postData, Boolean
fBeforeLoad)

   v System.Web.UI.Page.ProcessRequestMain(Boolean
includeStagesBeforeAsyncPoint, Boolean includeStagesAfterAsyncPoint)
```

7.3.4 Doba zotavení

Díky výše zmíněné chybě bylo možné zjistit dobu, kterou aplikace potřebuje pro obnovení své činnosti. Stav, kdy dochází k chybám, trvá zhruba 20 minut.



Okamžik pádu aplikace během 16 hodinového testu

Z obrázku je také dobře patrno, že při pádu docházelo k selhání především delších testů. Z logu bylo dále zjištěno, že se jednalo většinou o web testy obsahujících smyčky s více opakováním. Na základě tohoto poznatku byly smyčky u ostatních web testů zredukovány.

8 Alternativní nástroje

Alternativní nástroje byly instalovány a zkoušeny na čisté instalaci operačního systému Windows 7 Ultimate SP1 na počítači Acer Extensa 5220. Systému byl aktuální a neobsahoval žádný další software, který by mohl činnost instalovaných programů ovlivnit.

8.1 Fiddler

Fiddler je v některých případech doporučován [45] jako alternativa k zaznamenávání HTTP komunikace. Například, když *Web Test Recorder* ve VS2010 nedokáže při nahrávání web testů zaznamenat některé AJAX, ActiveX prvky nebo pop-up okna.

Program funguje na principu web debugging proxy a jako freeware je k dispozici na stránkách projektu [46]. Během testování Školy OnLine ve VS2010 nebylo jeho použití nutné, neboť zaznamenávalo HTTP komunikaci během testů správně. Fiddler byl úspěšně využit při nahrávání testů v Telerik Test Studiu (viz kapitola 8.5).

8.2 iMacros

iMacros je produktem společnosti iOpus [47]. Není určen jen k testování, ale obecně pro automatizaci práce na webových stránkách. V oblasti testování webových aplikací by měl program umožňovat tvorbu a provádění web testů a výkonnostního a regresního testování. Měl by také podporovat technologie Java, Flash, Flex nebo Silverlight a všechny AJAX elementy. Úplný výčet funkcí a přehled variant produktu je k dispozici na stránkách projektu [48].

iMacros zaznamenává činnost uživatele ve webovém prohlížeči a popisuje ji skriptem. Tento skript je možné libovolně upravovat v editoru *iMacrosEditor.exe*, který je součástí instalace. V nastavení je možné vybrat, do jaké míry program zaznamenává činnost uživatele na stránce. Vzhledem k vysoké interaktivitě stránek Školy OnLine (výsuvné menu, kontextová nabídka v rozvrhu), bylo během defaultního i během pokročilého nastavení nahráváno velké množství kroků. Nicméně se, přes vyzkoušení všech kombinací nastavení, nepodařilo kromě testů přihlášení a odhlášení uživatele úspěšně přehrát žádný další test. Program si nedokázal poradit s výsuvným moderním menu Školy OnLine. Poměrně často také docházelo během práce k pádům nebo k zasekávání celé aplikace.

iMacros je k dispozici ke stažení jako 30 denní trial verze, nebo ke koupi za $495 – $995 (USD), v závislosti na variantě.

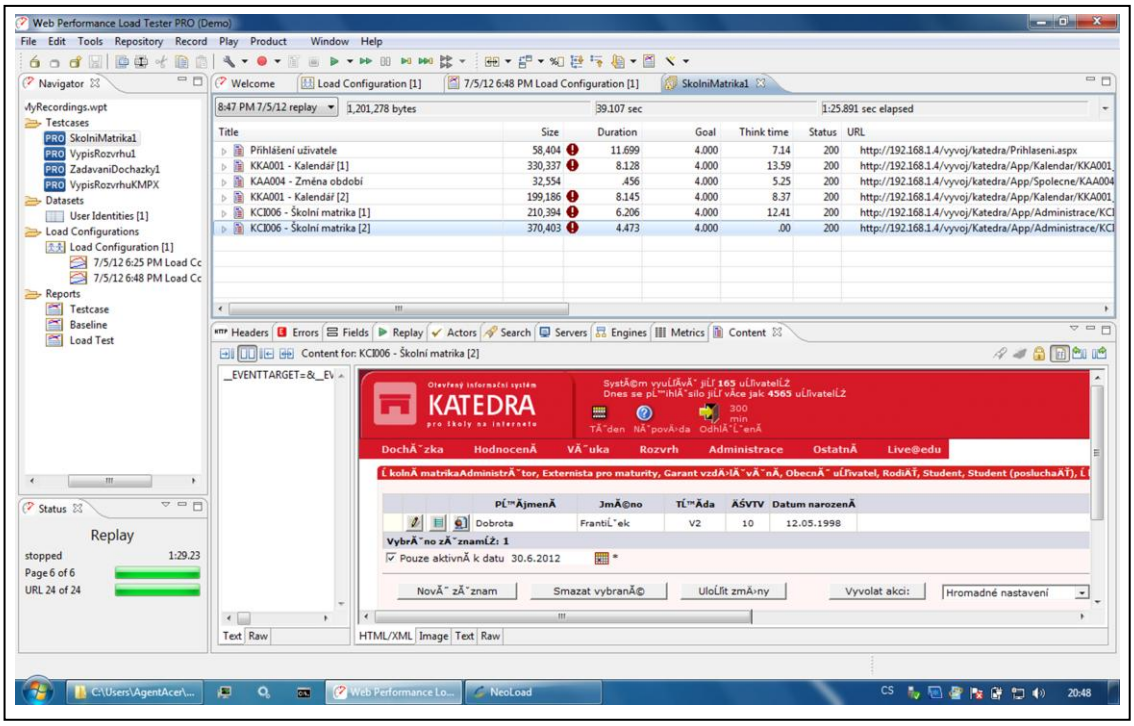
8.3 Web Performance Load Tester

Program *Web Performance Load Tester* (dále jen WPLT) je od společnosti *Web Performance, Inc.* [49] a je v současné době k dispozici ve verzi 5.1.10803. Jedná se o nástroj pro tvorbu web testů a load testů. Rozložení prvků uživatelského rozhraní je podobné VS2010.

Program poskytuje poměrně široké množství nastavení pro web testy a s nainstalovaným server agentem i podrobné sledování výkonu.

Specifikace modelu zátěže load testů je ovšem v porovnání s VS2010 slabší. Neumožňuje tolik kombinací a tak detailní nastavení jako produkt společnosti Microsoft a nelze s ním tedy simulovat zátěž na tak pokročilé úrovni. Analýza výsledků a reporty jsou naopak propracované a poskytují dobrý přehled o celém průběhu. Nahrávání web testů je prováděno v IE. Dynamické parametry (dynamic fields) jsou automaticky detekovány a datasets umožňují správu dat určených k vyplňování formulářů. Funkce je navíc doplněna o generátor údajů, což usnadňuje a zrychluje například automatizaci zadávání nových dat do testovaného systému. WPLT dále podporuje testování šířky pásma připojení a jeho analýzu. Na rozdíl od VS2010 neumožňuje spuštění z více lokálních serverů, ale s příslušnou verzí produktu a zřízeným účtem na Amazon EC2 je možné využívat spouštění load testů z cloudu a vytvořit tak zátěž odpovídající i mnoha tisícům uživatelů.

V rámci experimentu bylo ve WPLT vytvořeno několik web testů odpovídajících jejich předlohám hotovým ve VS2010 a z nich následně složen load test. Funkcionalita byla kvůli demoverzi programu omezena počtem virtuálních uživatelů jen na 25. Testy byly funkční a doby odezvy web testů se shodovaly s výsledky získanými ve VS2010. Během přehrávání se objevil i problém s parametrem __LASTFOCUS (viz kapitola 6.9.1). Problémy s modulem kalendáře však nenastaly.



Snímek obrazovky programu Web Performance Load Tester

Cena programu se pohybuje mezi $512 a $20 000 (USD) v závislosti na verzi, počtu virtuálních uživatelů a podpoře.

8.4 NeoLoad

Software *NeoLoad* společnosti *Neotys* [50] je určen především k zátěžovému testování. Provádění samotných web testů je v programu prováděno jen jako validace virtuálního uživatele. Licence NeoLoad 4.0.2 poskytnutá v rámci trial verze bohužel neopravňuje k spuštění testu s více virtuálními uživateli ani využití scénářů. Uživatel díky ní získá přehled o funkcích programu a o jeho grafickém prostředí. Trial verze mu umožňuje nahrání činnosti uživatele v prohlížeči, detekci obecných dynamických parametrů i parametrů typických pro .NET Framework. Dále přiřazení této činnosti virtuálnímu uživateli a její validaci (přehrání web testu). Takto vytvořené uživatele lze sice přidat do load testu a ten konfigurovat, ale jeho spuštění již není možné.

V rámci experimentů bylo v NeoLoad vytvořeno několik virtuálních uživatelů a k nim nahrány jednoduché úkoly prováděné ve Škole OnLine. Jejich přehrávání v rámci validace probíhalo až na několik drobných chyb s časovým limitem stránky správně.

Mezi funkce patří nejen podpora technologií .NET, JSP, AJAX a Flash, Flex a AIR, ale i Oracle E-Business, SAB Web a dalších ERP/ECM/CRM systémů. Monitorování je zajištěno specializovanými moduly, které jsou k dispozici pro celou řadu operačních systémů a serverů [51] včetně IIS a .NET. Pro prezentaci výsledků load testů je k dispozici i generování reportu ve formátu PDF.

Součástí produktu NeoLoad je také funkce generování zátěže z cloudu podobně, jako je tomu u WPLT (viz kapitola 8.3). Společnost Neotys ale pro tuto službu poskytuje vlastní platformu *Neotys Cloud Platform*.

Cena programu NeoLoad se pohybuje od $1 250 do $35 000 (USD) v závislosti na počtu virtuálních uživatelů a podpoře.

8.5 Telerik Test Studio

Test Studio společnosti Telerik [52] poskytuje kompletní řešení pro funkční, výkonnostní a zátěžové testování. Studio je vybaveno podobnou funkcionalitou jako zmíněná konkurenční řešení. Za zmínku navíc stojí možnost připojení k TFS serveru, anebo možnost exportovat testy v podobě projektu pro VS2008 a VS2010.

Přes prezentaci velkého množství funkcí si Test Studio verze 2012.1.528.0 při nahrávání testů Školy OnLine nevedlo příliš dobře. Nahrané testy obsahovaly nesprávně zaznamenané údaje (špatné jméno a heslo pro přihlášení apod.), které bylo třeba následně opravovat ručně. Jako dobrá alternativa pro nahrávání testů posloužil Fiddler (viz kapitola 8.1). Jím zachycený provoz byl uložen do .saz souboru a následně importován v Test Studiu při tvorbě load testů.

Základní verze Test Studia stojí $2 499 (USD), Test Studio Run-time pro servery $199 a v závislosti na velikosti generované zátěže je možné dále dokupovat balíčky pro virtuální uživatele.

8.6 Shrnutí alternativ

Kromě Fiddleru nabízí zmíněné produkty víceméně stejné funkce, které někde fungují lépe a někde hůře. Jako nejlepší alternativa ze zde uvedených, se jeví *Web Performance Load Tester*. Je to jednak díky pokročilým funkcím a jednak díky dobrému grafickému uživatelskému rozhraní. Program sice nedosahuje tak vysoké úrovně zpracování a tolika funkcí jako VS2010, ale testy pro Školu OnLine v něm bylo možné vytvořit snadno a fungovaly. Do jaké míry je adekvátní jeho cena, lze jen těžko posoudit bez informací o konkrétním projektu. Ceny VS2010 Test Professional začínají na $2 169 (USD), dokoupení balíku s licencí pro tisíc virtuálních uživatelů lze pořídit za dalších $4 500 [53].

Mnoho společností nabízí testování jen jako službu [54], kdy je daná aplikace testována z vně podnikové infrastruktury, jako je tomu u WPLT s Amazon EC2, nebo *Neotys Cloud Platform*. Jde především o výkonnostní a zátěžové testy, které mohou poskytnout informace, které laboratorním testováním získat nelze. Na rozdíl od laboratorního testování je prověřen systém jako celek, i včetně firewallu a sítě ISP. Ceny jsou různé a opět závisí na konkrétní dodávce služeb. Přínos takového testování z cloudu však rozhodně stojí minimálně za zvážení.

9 Závěr

Vlastní implementace a provádění testů neprobíhaly dle očekávání. Testy musely být mnohokrát přepracovány a poznatky této práce přepisovány. Z pohledu aplikace Školy OnLine představují získané informace spíše první kroky ve výkonnostním a zátěžovém testování. Ty další by však měly být díky této práci již snáze získatelné. Zajímavá byla i zkušenost s alternativními nástroji, která se často diametrálně lišila od prezentací dostupných na internetu a jejich skutečným potenciálem.

Cílem práce nebylo pouze dokumentovat návrh a implementaci, nýbrž také prezentovat možnosti, které testování v prostředí Visual Studia nabízí a současně tak vytvořit výchozí bod pro další vzdělávání v daném tématu. Velká pozornost byla proto věnována citovaným zdrojům. Jejich seznam není jen povinnou součástí dokumentu, ale i rozcestníkem pro čtenáře, který má zájem proniknout do problematiky hlouběji, než rozsah této práce dovoluje.

Seznam použitých zkratek

AJAX	Asynchronous JavaScript and XML
ASP	Active Server Pages
CRM	Customer relationship management
CSV	Comma-Separated Values
ECM	Enterprise Content Management
ERP	Enterprise Resource Planning
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IE	Internet Explorer
IEEE	Institute of Electrical and Electronics Engineers
IIS	Internet Information Services
IP	Internet Protocol
ISP	Internet Service Provider
LAN	Local Area Network
MS	Microsoft
MSDN	Microsoft Developer Network
MTM	Microsoft Test Manager
PDF	Portable Document Format
SQL	Structured Query Language
TFS	Team Foundation Server
UAC	User Account Control
UI	User Interface
URL	Uniform Resource Locator
VHD	Virtual Hard Disk
VS2008	Visual Studio 2008
VS2010	Visual Studio 2010
VSP	Visual Studio Profiler
WPLT	Web Performance Load Tester
WWW	World Wide Web
XML	Extensible Markup Language

Použitá literatura a zdroje

[1] R. Patton, Testování softwaru, Praha: COMPUTER PRESS, 2002, p. 314.

[2] L. Williams, „White-Box Testing," 2006. [Online]. Available: http://agile.csc.ncsu.edu/SEMaterials/WhiteBox.pdf. [Přístup získán 5 2012].

[3] „Gray Box Testing," December 2010. [Online]. Available: http://softwaretestingfundamentals.com/gray-box-testing/. [Přístup získán 5 2012].

[4] A. BUCHALCEVOVÁ a J. KUČERA, „Hodnocení metodik vývoje informačních systémů z pohledu testování," Katedra informačních technologií VŠE Praha, 2008. [Online]. Available: http://nb.vse.cz/~buchalc/clanky/testovani.pdf. [Přístup získán 5 2012].

[5] P. Ing. Luboš Král a Ing. Tomáš Hazdra, „Testování a diagnostika softwaru," Katedra kybernetiky ČVUT Praha, 2000. [Online]. Available: http://www.odbornecasopisy.cz/index.php?id_document=27838. [Přístup získán 2012].

[6] „Types of software Testing," April 2007. [Online]. Available: http://www.softwaretestinghelp.com/types-of-software-testing/. [Přístup získán 5 2012].

[7] M. Čermák, „Typy testů," 4 1 2009. [Online]. Available: http://www.cleverandsmart.cz/typy-testu/. [Přístup získán 18 3 2012].

[8] P. Glavich a C. Farrell, .NET Performance Testing and Optimization - The Complete Guide, Simple Talk Publishing, 2010.

[9] Institute of Electrical and Electronics Engineers, „829-2008 - IEEE Standard for Software and System Test Documentation," 18 07 2008. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4578383 nebo http://cow.ceng.metu.edu.tr/Courses/download_courseFile.php?id=4046. [Přístup získán 30 05 2012].

[10] J. P. Cem Kaner, „What Is a Good Test Case?," Florida Institute of Technology, Department of Computer Sciences, May 2003. [Online]. Available: http://www.kaner.com/pdfs/GoodTest.pdf. [Přístup získán 5 2012].

[11] „CSCI 577b: Software Engineering II - SOFTWARE TEST REPORT (STR)," The University of Southern California, Spring 2000. [Online]. Available: http://sunset.usc.edu/classes/cs577b_2000/EP/05/EP-05.pdf. [Přístup získán 5 2012].

[12] Microsoft, „Best Practices for Coded UI Tests," Microsoft, 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/dd380782.aspx. [Přístup získán 2 2012].

[13] Microsoft, „Anatomy of a Coded UI Test,“ Microsoft, 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/ff398062.aspx. [Přístup získán 1 2012].

[14] E. Glas, „Web Test Authoring and Debugging Techniques for Visual Studio 2010,“ Microsoft, 23 Mar 2010. [Online]. Available: http://blogs.msdn.com/b/edglas/archive/2010/03/24/web-test-authoring-and-debugging-techniques-for-visual-studio-2010.aspx. [Přístup získán 1 2012].

[15] Microsoft, „Visual Studio 2010 Feature Packs Frequently Asked Questions,“ Microsoft, 2012. [Online]. Available: http://msdn.microsoft.com/en-us/vstudio/ff520697.aspx. [Přístup získán 3 2012].

[16] Microsoft, „Anatomy of a Unit Test,“ Microsoft, 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/ms182517.aspx. [Přístup získán 6 2012].

[17] Microsoft, „Microsoft.VisualStudio.TestTools.UnitTesting Namespace,“ Microsoft, 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/microsoft.visualstudio.testtools.unittesting.aspx. [Přístup získán 6 2012].

[18] Microsoft, „Generic Tests Overview,“ Microsoft, 2010. [Online]. Available: http://msdn.microsoft.com/en-us/library/ms182623.aspx. [Přístup získán 2 2012].

[19] Microsoft, „Create Test Settings for Automated Tests as Part of a Test Plan,“ Microsoft, 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/ee257067.aspx. [Přístup získán 3 2012].

[20] Microsoft, „Microsoft Expression Encoder 4 with Service Pack 1 (SP1),“ Microsoft, 28 1 2011. [Online]. Available: http://www.microsoft.com/en-us/download/details.aspx?id=24601. [Přístup získán 12 2011].

[21] Microsoft, „Configuring Test Controllers and Test Agents for Load Testing,“ Microsoft, 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/ms243155.aspx. [Přístup získán 1 2012].

[22] Microsoft, „Test Scribe for Visual Studio Ultimate 2010 and Test Professional 2010,“ Microsoft, April 2011. [Online]. Available: http://visualstudiogallery.msdn.microsoft.com/e79e4a0f-f670-47c2-9b8a-3b6f664bf4ae. [Přístup získán 12 2011].

[23] Microsoft, „Testing the Application with Feature Pack 2,“ Microsoft, 2011. [Online]. Available: http://msdn.microsoft.com/cs-cz/library/gg413375.aspx. [Přístup získán 11 2011].

[24] Microsoft, „Take Visual Studio 2010 For a Test Drive*,“ Microsoft, 2011. [Online]. Available: http://www.microsoft.com/visualstudio/en-us/try. [Přístup získán 11 2011].

[25] Microsoft, „TechNet Virtual Labs," Microsoft, 2011. [Online]. Available: http://technet.microsoft.com/en-us/virtuallabs/bb467605.aspx. [Přístup získán 11 2011].

[26] Microsoft, „Microsoft Visual Studio 2010 (Virtual Labs)," Microsoft, 2011. [Online]. Available: http://msdn.microsoft.com/cs-cz/ff640662. [Přístup získán 11 2011].

[27] Microsoft, „Run IT on a Virtual Hard Disk – Test Drive Program," Microsoft, 2011. [Online]. Available: http://technet.microsoft.com/en-us/bb738372. [Přístup získán 11 2011].

[28] ŠKOLA ONLINE a.s., „Katedra - uživatelská příručka," ŠKOLA ONLINE a.s., 2010. [Online]. Available: https://aplikace.skolaonline.cz/dokumentace/KS/katedra/web/index.html. [Přístup získán 11 2011].

[29] J. Nielsen, Usability Engineering, San Francisco: Morgan Kaufmann, 1993.

[30] Microsoft, „Installing and Configuring Visual Studio Agents and Test and Build Controllers," Microsoft, August 2010. [Online]. Available: http://msdn.microsoft.com/en-us/library/dd648127.aspx. [Přístup získán 12 2011].

[31] Microsoft, „Ports Required for Installation of Team Foundation Components," Microsoft, 2011. [Online]. Available: http://msdn.microsoft.com/en-us/library/dd578664.aspx. [Přístup získán 12 2011].

[32] Microsoft, „Configuring a Test Controller and Test Agent Across a Firewall," Microsoft, 2011. [Online]. Available: http://msdn.microsoft.com/en-us/library/ff652627.aspx. [Přístup získán 1 2012].

[33] Microsoft, „Managing Test Controllers and Test Agents," Microsoft, 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/dd695837.aspx. [Přístup získán 2 2012].

[34] Microsoft, „How to: Configure Unit Tests to Target .NET Framework 3.5," Microsoft, 2011. [Online]. Available: http://msdn.microsoft.com/en-us/library/gg601487.aspx. [Přístup získán 12 2011].

[35] Microsoft, „Testing a Large Application with Multiple UI Maps," Microsoft, 2010. [Online]. Available: http://msdn.microsoft.com/en-us/library/ff398056.aspx. [Přístup získán 1 2012].

[36] Microsoft, „Walkthrough: Adding a Loop to a Web Performance Test," Microsoft, 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/ff820570(v=vs.110). [Přístup získán 4 2012].

[37] S. Kumar N. a S. Subashni , Software Testing using Visual Studio 2010, Birmingham, UK: Packt Publishing, 2010, p. 384.

[38] Microsoft, „Web and Load Test FAQs," Microsoft, 16 June 2006. [Online]. Available: http://social.msdn.microsoft.com/Forums/en-US/vstswebtest/thread/bd3b1caf-ca7b-408e-b415-3b8e3465bb03/. [Přístup získán 1 2012].

[39] B. Ozar, „SQL Server Perfmon (Performance Monitor) Best Practices," Brent Ozar PLF Inc., 30 December 2006. [Online]. Available: http://www.brentozar.com/archive/2006/12/dba-101-using-perfmon-for-sql-performance-tuning/. [Přístup získán 4 2012].

[40] B. McGehee, „Tips for Using SQL Server Performance Monitor Counters," SQL Server Performance, 2012. [Online]. Available: http://www.sql-server-performance.com/2005/sql-server-performance-monitor-coutners/. [Přístup získán 4 2012].

[41] Microsoft, „Specifying the Counter Sets for Computers in a Load Test," Microsoft, 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/tfs/ms404695(v=vs.100).aspx. [Přístup získán 1 2012].

[42] Microsoft, „How to: Configure ASP.NET Profiler for Load Tests Using Test Settings," Microsoft, 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/tfs/dd504817(v=vs.100).aspx. [Přístup získán 5 2012].

[43] Microsoft, „Strategies for Troubleshooting Test Controllers and Test Agents in Load Tests," Microsoft, 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/dd692842.aspx. [Přístup získán 2 2012].

[44] Applian Technologies, „Error when installing. "Error 1001. An exception occurred in the OnAfterInstall event handler of aas -> Applian driver installation failed." And code 2147180503.," Applian Technologies Inc., 6 April 2011. [Online]. Available: http://applian.com/faq-pro/?action=article&cat_id=003&id=7&lang. [Přístup získán 1 2012].

[45] J. Christie, „Web Test Authoring and Debugging Techniques," December 2005. [Online]. Available: http://msdn.microsoft.com/en-us/library/ms364082(VS.80).aspx. [Přístup získán leden 2012].

[46] E. Lawrence, „Fiddler Web Debugging Proxy," 2012. [Online]. Available: http://www.fiddler2.com/fiddler2/. [Přístup získán 3 2012].

[47] iOpus, „iMacros® 8.0 (domovská stránka produktu)," iOpus GmbH, 2012. [Online]. Available: http://www.iopus.com/imacros/. [Přístup získán 6 2012].

[48] iMacros, „iMacros Feature Comparison Chart," iMacros GmbH., 2012. [Online]. Available: http://www.iopus.com/imacros/compare/. [Přístup získán 6 2012].

[49] Web Performance, Inc., „(Domovská stránka společnosti)," Web Performance, Inc., 2012. [Online]. Available: http://www.webperformance.com/. [Přístup získán 6 2012].

[50] Neotys USA, Inc., „NeoLoad Overview," Neotys USA, Inc., 2012. [Online]. Available: http://www.neotys.com/product/overview-neoload.html. [Přístup získán 5 2012].

[51] Neotys USA, Inc., „Application Monitoring," Neotys USA, Inc., 2012. [Online]. Available: http://www.neotys.com/product/monitoring-web-stress-test.html. [Přístup získán 5 2012].

[52] Telerik, „Tools for Software Testing," Telerik, 2012. [Online]. Available: http://www.telerik.com/automated-testing-tools/. [Přístup získán 6 2012].

[53] Microsoft, „Visual Studio 2010 Test Professional with MSDN," Microsoft, 2012. [Online]. Available: http://www.microsoftstore.com/store/msstore/pd/Visual-Studio-2010-Ultimate-with-MSDN/productID.216637300/parentCategoryID.50804600/categoryID.50804700/list.true. [Přístup získán 6 2012].

[54] CloudTweaks, „Top 10 Cloud Computing Load Test and Performance Monitoring Companies," CloudTweaks, 20 August 2010. [Online]. Available: http://www.cloudtweaks.com/2010/08/top-10-cloud-computing-load-test-and-performance-monitoring-companies/. [Přístup získán 6 2012].

[55] J. Levinson, Software Testing with Visual Studio 2010, Boston, MA: Addison-Wesley, 2011.

[56] Institute of Electrical and Electronics Engineers, „829-2008 - IEEE Standard for Software and System Test Documentation," 18 07 2008. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4578383. [Přístup získán 30 05 2012].

[57] Microsoft, „Test Rig Requirements for Team System," Microsoft, 2012. [Online]. Available: http://msdn.microsoft.com/en-us/library/ms253092(v=vs.90).aspx. [Přístup získán 1 2012].

[58] Microsoft, „Visual Studio Agents 2010 - ISO," Microsoft, 4 12 2010. [Online]. Available: http://www.microsoft.com/en-us/download/details.aspx?id=1334. [Přístup získán 12 2011].

Příloha 1: Ukázka z dokumentu Test Plan Details

Test Plan Details

Test plan 1: SOL_test_plan_1

Testování nejčastěji užívaných částí web GUI
Status: Active | Active dates: 1.2.2012 0:00:00 - 1.4.2012 0:00:00 | Area: SOLTeamProject | Iteration: SOLTeamProject | Build: <Not assigned> | Build definition: <Not assigned> | Build quality: <Not assigned>

Available Configurations (1)

Config 2: Windows 7 and IE 9

Test Settings

Manual Settings 2: Základní diagnostika

Suite Hierarchy

SOL_test_plan_1 (0)
- Administrace (0)
  - Školní matrika (1)
- Dostupnost nejčastěji používaných stránek (9)
- Ověření nejčastěji používaných funkcí (10)

Suite Details

Suite 1: SOL_test_plan_1

State: In progress

Available Configurations (1)

Config 2: Windows 7 and IE 9

Test Cases (20)

Test Case 8: Vyhledani zaka ve skolni matrice (podle jmena)
Owner: Vaclav Javorsky | Design | Type: Manual | SOLTeamProject | Iteration: SOLTeamProject | Automated test: Not set | Assigned to: Vaclav Javorsky

Test Steps (12)

	#	Title	Expected Value
	1	Otevreni IE a prihlaseni do SOL (ada)	
	1.1	Spusteni IE	
	1.2	Otevreni http://localhost/vyvoj/katedra/Prihlaseni.aspx	
	1.3	Vyplneni jmena ada a hesla adad	
	1.4	Tlacitko Prihlasit	Uzivatel Adam Adamec je prihlasen do systemu
	2	Administrace - Osobni data - Skolni matrika	
	3	Jmeno - Zacina na	
	4	Vyplnit Fran	
	5	Tlacitko Zobrazit	
	6	Overeni (hledat Dobrota)	V seznamu se krom jinych objevi i Frantisek Dobrota
	7	Tlacitko odhlasit	
	8	Zavreni okna IE	

Příloha 2: Příklad testovacího případu vytvořeného v MTM

Test Case #9: Vypis dochazky z 2.4.2012

Title:	Vypis dochazky z 2.4.2012
Assigned To:	Vaclav Javorsky
State:	Design
Priority:	2
Automation Status:	Not Automated
Area:	SOLTeamProject
Iteration:	SOLTeamProject

Revisions

4/3/2012 3:51:28 PM, Edited by Vaclav Javorsky

4/2/2012 10:39:11 PM, Edited by Vaclav Javorsky

Field	Value
Rev	1 --> 2
Steps	Dochazka - Vypis dochazky - Denni mezisoucty Vyber tridy V1Od 2.4.2012 Do 2.4.2012 Tlacitko Zobrazit Objevi se tabulka dochazky z 2.4.2012 - Götzl Petr je prvni Tlacitko Odhlasit Zavrit IE --> Dochazka - Vypis dochazky - Denni mezisoucty Vyber tridy V1 Od 2.4.2012 Do 2.4.2012 Tlacitko Zobrazit Objevi se tabulka dochazky z 2.4.2012 - Götzl Petr je prvni Tlacitko Odhlasit Zavreni okna IE

4/2/2012 2:12:19 PM, Created by Vaclav Javorsky

Field	Value
Title	Vypis dochazky z 2.4.2012
State	Design
Rev	1
State Change Date	4/2/2012 4:12:19 PM
Activated Date	4/2/2012 4:12:19 PM
Activated By	Vaclav Javorsky
Reason	New
Assigned To	Vaclav Javorsky
Work Item Type	Test Case
Priority	2
Created Date	4/2/2012 4:12:19 PM
Created By	Vaclav Javorsky
Steps	Dochazka - Vypis dochazky - Denni mezisoucty Vyber tridy V1 Od 2.4.2012Do 2.4.2012 Tlacitko Zobrazit – Objevi se tabulka dochazky z 2.4.2012 - Götzl Petr je prvni Tlacitko Odhlasit Zavrit IE

Local Data Source	
Automation status	Not Automated
Iteration Path	SOLTeamProject
Iteration ID	1
Team Project	SOLTeamProject
Node Name	SOLTeamProject
Area Path	SOLTeamProject
Area ID	1

Příloha 3: Testovací případ

Hodnocení > Úprava hodnocení

Testovací případ			
Číslo:	13	**Název:**	WebTestUpravaHodnoceniK2.webtest
Aplikace:	Škola OnLine	**Část:**	Katedra
Navrhl:	Václav Javorský	**Vytvořeno:**	1. 5. 2012
Vytvořil:	Václav Javorský	**Provedeno:**	5. 7. 2012
Popis:	Menu > Hodnocení > Zadávání hodnocení > Úprava hodnocení		

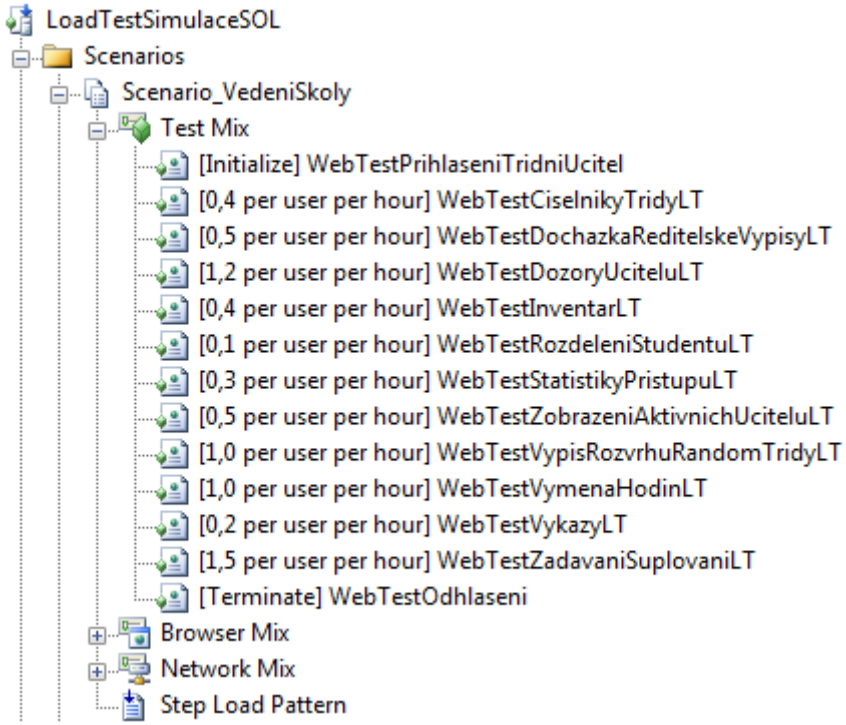
Podmínky pro test
Verze Školy OnLine 2.2 (build K0661.0.7), databáze VYV

Č.	Akce	Očekávaný výsledek	Ano/Ne
1.	Prohlížeč: {{TestWebServer}}/vyvoj/katedra/Prihlaseni.aspx	Načtení přihlašovacího formuláře Školy OnLine	
2.	Zadání jména ada		
3.	Zadání hesla adad		
4.	Tlačítko přihlášení	Přihlášení do systému	
5.	Klik na změnu období	Načtení stránky {{TestWebServer}}/vyvoj/katedra/App/Spolecne/KAA004_NastaveniProstr.aspx	
6.	Nastavení roku 2011/2012		
7.	Klik na tlačítko Nastavit	Načtení stránky {{TestWebServer}}/vyvoj/Katedra/App/Kalendar/KKA001_KalendarTyden.aspx	
8.	Menu > Hodnocení > Zadávání hodnocení > Úprava hodnocení	Načtení stránky {{TestWebServer}}/vyvoj/Katedra/App/Hodnoceni/KHO009_HodnVypisUcitel.aspx	
9.	Editovat 4.4.2012 MAT V1	Načtení stránky {{TestWebServer}}/vyvoj/Katedra/App/Hodnoceni/KHO001_HodnZapis.aspx	
10.	Přidat komentář webtest		
11.	Přidat Petříková Hana		
12.	Klik na tlačítko Uložit a zadat výsledky	Zobrazení tabulky pro editaci výsledků.	
13.	Zadat 5 bodů		
14.	Klik na Uložit výsledky	Zobrazení zprávy Úspěch: Výsledky uloženy	
15.	Klik na tlačítko Zpět	Načtení stránky {{TestWebServer}}/vyvoj/Katedra/App/Hodnoceni/KHO009_HodnVypisUcitel.aspx	
16.	Klik na tlačítko kalendář	Načtení stránky {{TestWebServer}}/vyvoj/Katedra/App/Default.aspx	
17.	Klik na odhlášení	Načtení stránky {{TestWebServer}}/vyvoj/Katedra/App/Logout.aspx	

Příloha 4: Složení load testů

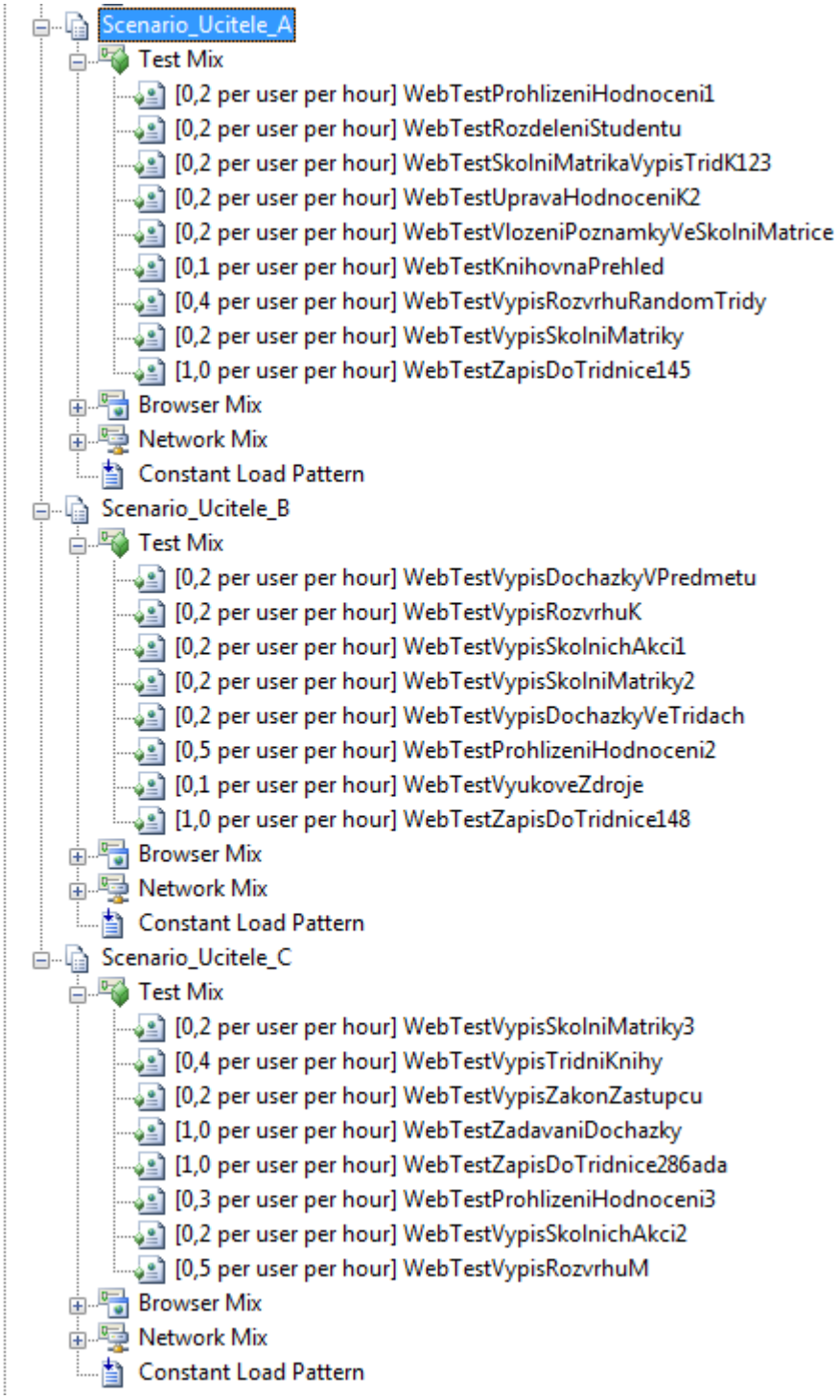
LoadTestSimulaceSOL.loadtest

Vedení školy



LoadTestSimulaceSOL.loadtest

Učitelé



LoadTestSimulaceSOL_intezivni.loadtest