Zápočtová úloha z předmětu KIV/ZSWI

DOKUMENT SPECIFIKACE POŽADAVKŮ

6. května 2015

Název týmu: **CrossCafé team**

Členové týmu:

Luboš Hubáček (vedoucí týmu)	lubos.hubacek@diginex.cz
Ondřej Pittl	ondrej.pittl@gmail.com
Jiří Homolka	jiri.homolka22@gmail.com
Jan Kohlíček	kohlicejan@gmail.com
Petr Kozler	pkozler@students.zcu.cz

JAVA aplikace pro monitoring běhu komponent

DOKUMENT SPECIFIKACE POŽADAVKŮ

Verze <1.2>

HISTORIE DOKUMENTU

Datum	Verze	Popis	Autor
9. 3. 2015	1.0	Vytvoření dokumentu	Luboš Hubáček, Petr Kozler
10. 3. 2015	1.1	Korektura, stylistika textu, doplnění chybějících informací	Ondřej Pittl
10. 3. 2015	1.2	Korektura textu	Jiří Homolka

Obsah dokumentu

1 Úvod

1.1 Předmět specifikace

Naprogramujte aplikaci v jazyce Java, která bude monitorovat stav daných komponent. Ke zjišťování stavu komponent použijte výměnu dat, předávaných ve formátu JSON, pomocí jednoduchých HTTP příkazů. Získané informace poté ukládejte do předpřipravených souborů (pro každou sledovanou instanci jeden soubor) s konstantní velikostí (ukládání bude probíhat způsobem, který se označuje jako tzv. *rolling file*). Pro vytváření souborů použijte knihovnu Log4J ve verzi 2. Součástí programu bude i grafické uživatelské rozhraní *(dále GUI)* implementované pomocí knihovny JavaFX. GUI bude obsahovat jednak strom obsahující instance komponent, ale také textové pole, které poslouží pro zobrazení výstupů z jednotlivých instancí filtrovaných označením příslušných položek ve stromu.

příklad URL pro získávání dat (komponent): http://peerfile.eu:3000/api/mon/memory_info

1.2 Typografické konvence

Během programování bude užíváno běžných typografických konvencí. Jedná se zejména o strukturu a obsah zdrojového kódu, programátorskou dokumentaci a rozhraní programu v anglickém jazyce. Tímto bude částečně zaštítěna kompatibilita s konvencemi využívanými u zadavatele.

1.3 Cílové publikum

Specifikace je určená zadavateli projektu a týmu, který jí zpracovává. Kromě zadavatele bude specifikaci číst ještě několik jeho kolegů, kteří později tuto aplikaci začlení do jiného většího systému.

1.4 Rozsah projektu

Cílem práce je vytvořit monitorovací aplikaci, která bude přehledně logovat data o stavu komponent do souborů pro archivaci těchto dat. Současně bude informovat i uživatele prostřednictvím výpisu na terminál.

1.5 Odkazy

Unicorn. Dokumentace Monitoring REST API. *Google dokumenty.* [online]. 2014 [cit. 2015-03-09]. Dostupné z: *http://bit.ly/1DeFsnZ*

CrossCafe team. ZSWI monitoring. *GitHub.* [online]. 2015 [cit. 2015-03-09]. Dostupné z: *http://github.com/Sekiphp/ZSWI_monitoring*

2 Obecný popis

2.1 Kontext systému

Jedná se o novou část již fungujícího stávajícího produktu, jejíž úlohou bude čtení dat vytvářených jinými částmi systému a unmarshalling z formátu JSON do snadno čitelné podoby.

2.2 Funkce produktu

Tato aplikace bude používána ke zjišťování aktuálního stavu daných komponent a ukládání výstupu do rotujících souborů. Bude poskytovat jednoduché grafické uživatelské rozhraní pro pohodlné ovládání programu.

2.3 Třídy uživatelů

Aplikace je navzdory předpokládané jednoduchosti ovládání určená pro technicky zdatné uživatele, neboť se jedná o podpůrnou monitorovací službu. Všichni uživatelé aplikace budou využívat stejného rozhraní – žádné administrátorské účty s vyššími oprávněními nejsou vyžadovány.

2.4 Provozní prostředí

Jelikož se jedná o klientskou aplikaci, hardware, typ a verze OS, zeměpisné umístění uživatelů: nelze určit a není směrodatné.

- zeměpisné umístění serverů a databází: ČR
- spolupracuje pouze s vystavenou REST WS

Službu mohou využívat lidé nezávisle na zeměpisné poloze, užitému anglickému jazyku porozumí všichni oprávnění uživatelé aplikace.

2.5 Omezení návrhu a implementace

Aplikace musí být naprogramována v jazyce Java, k vytvoření GUI musí být použita grafická knihovna JavaFX, pro logování knihovna Log4J 2 a pro výměnu dat framework Spring WS. Při výměně dat je pracováno s formátem JSON. Pro verzování a synchronizaci vývoje mezi členy týmu je použita služba GitHub a pro buildování projektu a správu závislostí programu nástroj Maven. Na použitá IDE nejsou kladeny žádné zvláštní požadavky.

2.6 Uživatelská dokumentace

Uživatelská dokumentace bude realizována pomocí wiki stránky na stránkách projektu na GitHubu *(viz kapitola 1.5)* a pomocí textové dokumentace ve formátu PDF.

2.7 Předpoklady a závislosti

Tato aplikace bude napojena na stávající komponenty většího systému. Funkce programu je tedy závislá na funkci těchto komponent (zejména Peerfile backend (*dále PF*)). Pokud dojde k výpadku služby PF, tento program nebude moci fungovat – zaznamená se výpadek systému.

3 Funkce systému

3.1 Webservice klient

3.1.1 Popis a priorita

Priorita: vysoká

3.1.2 Události a odpovědi

Událost: uživatel označí instanci komponenty ve stromě
Odpověď: program odešle příslušný požadavek pro získání dat

3.1.3 Funkční požadavky

Pokud se nepodaří odeslat požadavek, program zobrazí chybovou zprávu, zaloguje událost do souboru a vypíše na terminál.

3.2 Parsování JSON response

3.2.1 Popis a priorita

Priorita: vysoká

3.2.2 Události a odpovědi

Událost: je přijata odpověď ve formátu JSON
Odpověď: program provede parsování přijatých dat

3.2.3 Funkční požadavky

V případě neúspěšné operace program zareaguje obdobně jako v předchozím případě (*viz 3.1.3*).

3.3 Ukládání do rotujících souborů

3.1.1 Popis a priorita

Priorita: střední

3.1.2 Události a odpovědi

Událost: přijatá data ve formátu JSON byla naparsována a uložena do paměti
Odpověď: program zaloguje data v textové podobě do rotujícího souboru

3.1.3 Funkční požadavky

V případě neúspěchu při zápisu do souboru program zobrazí příslušnou chybovou hlášku.

3.4 Prezentace dat v GUI

3.1.1 Popis a priorita

Priorita: nízká

3.1.2 Události a odpovědi

Událost: přijatá data ve formátu JSON byla naparsována a uložena do paměti
Odpověď: program zobrazí přijatá data v čitelné podobě do textového pole

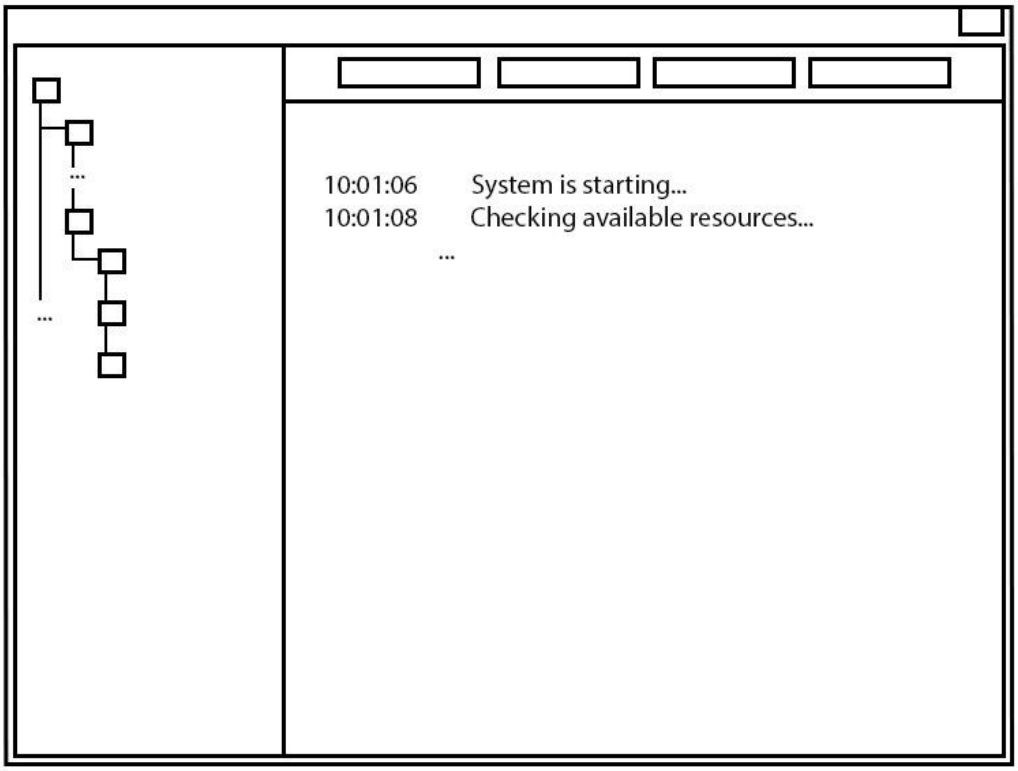
3.1.3 Funkční požadavky

U této části aplikace se neočekávají žádné chybové stavy.

4 Požadavky na vnější rozhraní

4.1 Uživatelská rozhraní

GUI aplikace bude sestávat z jediného hlavního okna a případně jednoduchých dialogových oken. Na vizuální podobu písma či ovládacích prvků nejsou kladeny žádné zvláštní požadavky, pravděpodobně budou použita výchozí nastavení komponent knihovny Java FX. Není vyžadována lokalizace softwaru ani možnosti usnadnění ovládání pro handicapované.



4.1 Obrázek — návrh grafického uživatelského rozhraní (GUI)

4.2 Hardwarová rozhraní

V této aplikaci není třeba HW rozhraní řešit.

4.3 Softwarová rozhraní

Pro přijímání dat bude využito rozhraní instancí PF (viz PF v sekci odkazy – dokumentace v daném dokumentu). O výměnu dat se v tomto případě postará protokol JSON za pomoci HTTP hlaviček. Toto je jediný způsob, jak bude program přijímat data.

API pro tento program, které by usnadnilo komunikaci s případnými dalšími programy, není vyžadováno.

4.4 Komunikační rozhraní

Jsou kladeny požadavky na funkčnost serveru (*viz Dodatek A — PF*), jehož komponenty jsou monitorovány. Server mimo provoz znemožní komunikaci s jeho komponentami a nebude možné jejich stav logovat.

Zprávy jsou přenášené ve formátu JSON (*viz Dodatek A — JSON*). Jedná se o způsob zápisu dat, jehož vstupem je libovolná datová struktura (číslo, řetězec, boolean, pole, objekty, …) a výstupem vždy řetězec. K přenosu využívaný JSON je čitelný člověkem. Nevýhodou bývá větší objem dat.

Aplikace není závislá na jiných faktorech.

5 Další mimofunkční požadavky

5.1 Výkonnostní požadavky

Nejedná se o paměťově či časově náročný program. Rychlost aplikace je závislá na rychlosti komunikace se serverem, tedy přenosu dat. Nejsou kladeny požadavky na časovou ani paměťovou náročnost.

5.2 Bezpečnostní požadavky

Nejedná se o business critical aplikaci, nicméně tato aplikace monitoruje business critical procesy, proto je vyžadována její 100% spolehlivost a v přípravě havárie by měla být provedena brzká oprava, ideálně proveditelná do jedné hodiny od zaznamenaného výpadku.

5.3 Kvalitativní požadavky

- požadavky na snadnost používání – nejsou
- požadavky na spolehlivost – je vyžadována na 100 %
- požadavky na robustnost – oprava po havárii musí proběhnout do 1 hodiny
- požadavky na udržitelnost – zdrojový kód a programátorská dokumentace (Javadoc) budou psány v anglickém jazyce; budou dodržovány konvence běžné v programovacím jazyce Java (například *camel case*)

5.4 Ostatní požadavky

Na aplikaci nejsou kladeny žádné další zvláštní požadavky.

6 Dodatek A: slovníček pojmů

PeerFile (PF) – server, k němuž se pomocí aplikace přistupuje a jehož komponenty monitoruje.

REST API – architektura rozhraní, navržená pro distribuované prostředí; použito pro jednotný a snadný přístup ke zdrojům.

JSON – způsob přenosu dat nezávislý na počítačové platformě. Vstupem je libovolná datová struktura, výstupem je vždy řetězec. Složitost hierarchie vstupní proměnné není teoreticky nijak omezena – JSON může mít různě složitou strukturu.

Rolling File – soubor s konstantní velikostí dat, v němž dochází k přepisování starších dat v případě zápisu dat překračujících velikostní limit.

7 Dodatek B: seznam úkolů

Seznam je průběžně měněn. Obsahuje všechny úkoly, jež je třeba splnit. Aktuální stav:

- načíst data z PF instancí
- zprovoznit Maven
- logování pomocí Log4J 2