Technická dokumentácia Logger modulu

Verzia 16.10.2015

Tabuľka 1. Autori

Autor	Rola
Tomáš Donko	

Tabuľka 2. História zmien

Verzia	Dátum	Autor	Popis
1.0	12.11.2015	Tomáš Donko	Vytvorenie dokumentu
1.1	09.12	Tomáš Donko	Doplnenie dokumentácie
1.2.	30.3.2016	Peter Vrana	Refaktoring REST služby pre zápis logov
1.3	17.04	Tomáš Donko	Podpora stránkovania, iné zobrazenie

Obsah

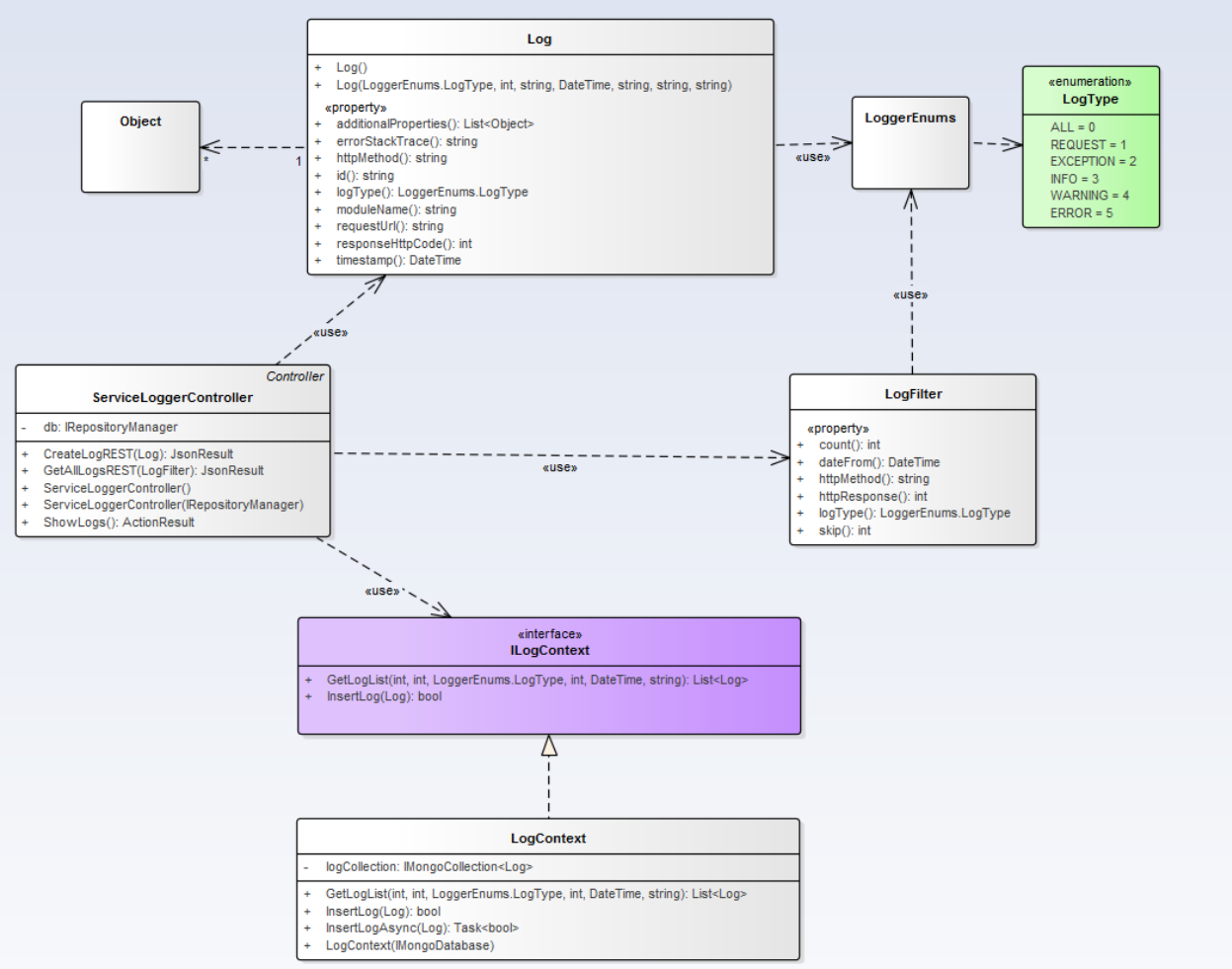
1 Analýza 1
2 Návrh 2
3 Implementácia 5
4 Testovanie 8

1 Analýza

Modul slúži na zapisovanie všetkých http requestov a výnimiek, alebo čohokoľvek, čo sa dá zalogovať. Tieto dáta sa odchytávajú a zapisujú do mongo databázy a je nad nimi možné vyhľadávanie a filtrovanie. Keďže máme architektúru postavenú na .NET MVC, všetky requesty je vhodné odchytávať v súbore Global.asax, v ktorej bude preťažujúca metóda Application_EndRequest(), v ktorej sa budú odchytávať všetky requesty. V tomto súbore bude aj metóda Application_Error(), ktorá bude odchytávať všetky neošetrené chyby. Taktiež v ostatných projektoch je potrebné odosielať logy na jedno miesto pomocou NLogger.

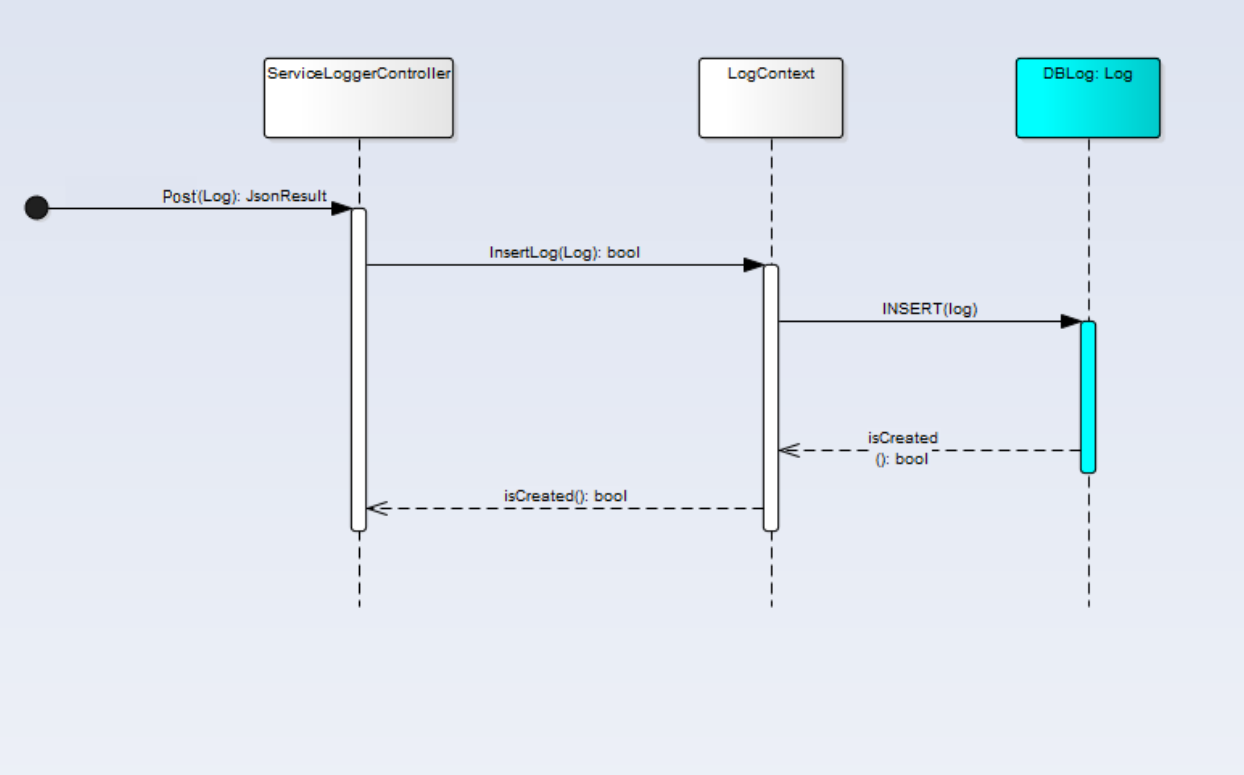
2 Návrh

Toto je reprezentácia tried na back-ende pomocou class diagramu:



Obrázok 1: Class diagram Logger modulu v EA

Diagram nezobrazuje úplne všetky závislosti, kvôli lepšej čitateľnosti. Controller trieda obsahuje všetky REST služby a metódy vhodné na volanie z vonka. Využíva sa trieda Log, ktorá predstavuje všetky uložené hodnoty. Log využíva LoggerEnums na ohraničenie typu logu. LogFilter slúži pri vyhľadávaní logov a filtrovaní. LogContext už priamo pracuje s databázou.

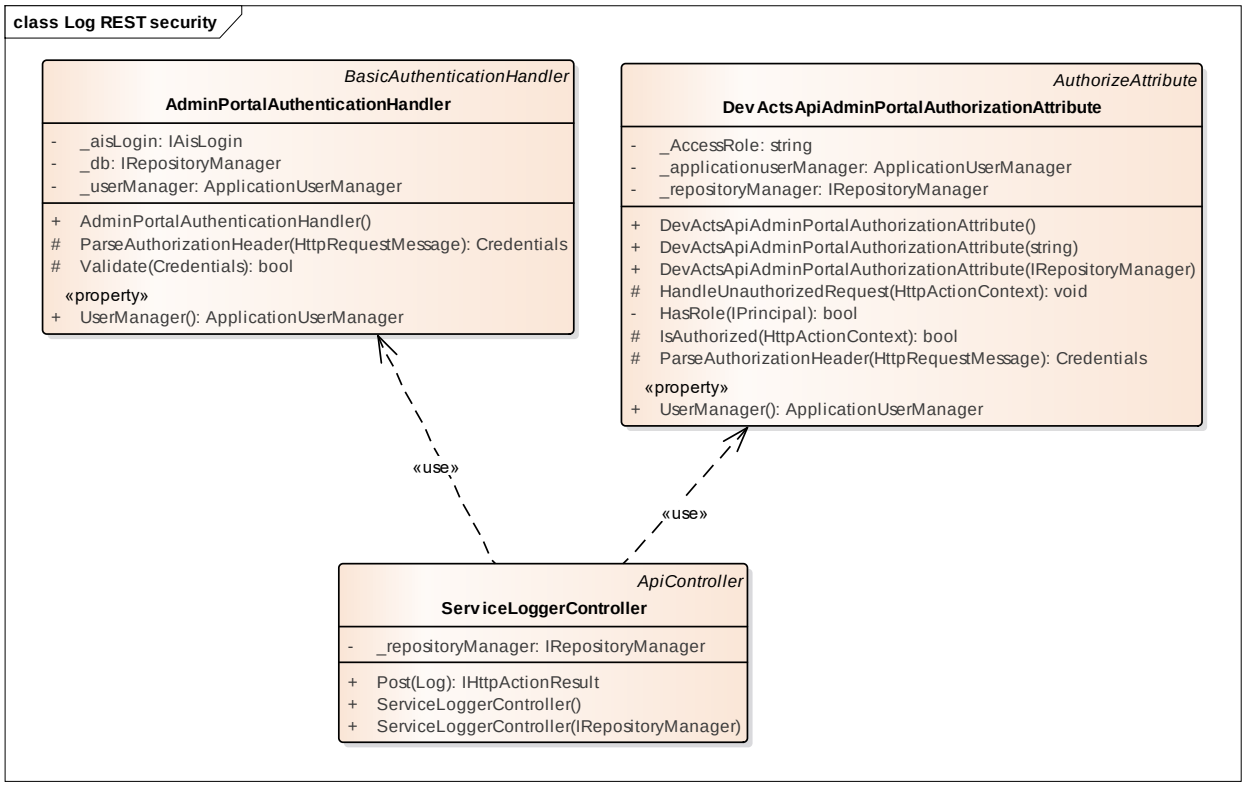


Obrázok 2: Sekvenčný diagram na vytvorenie logu

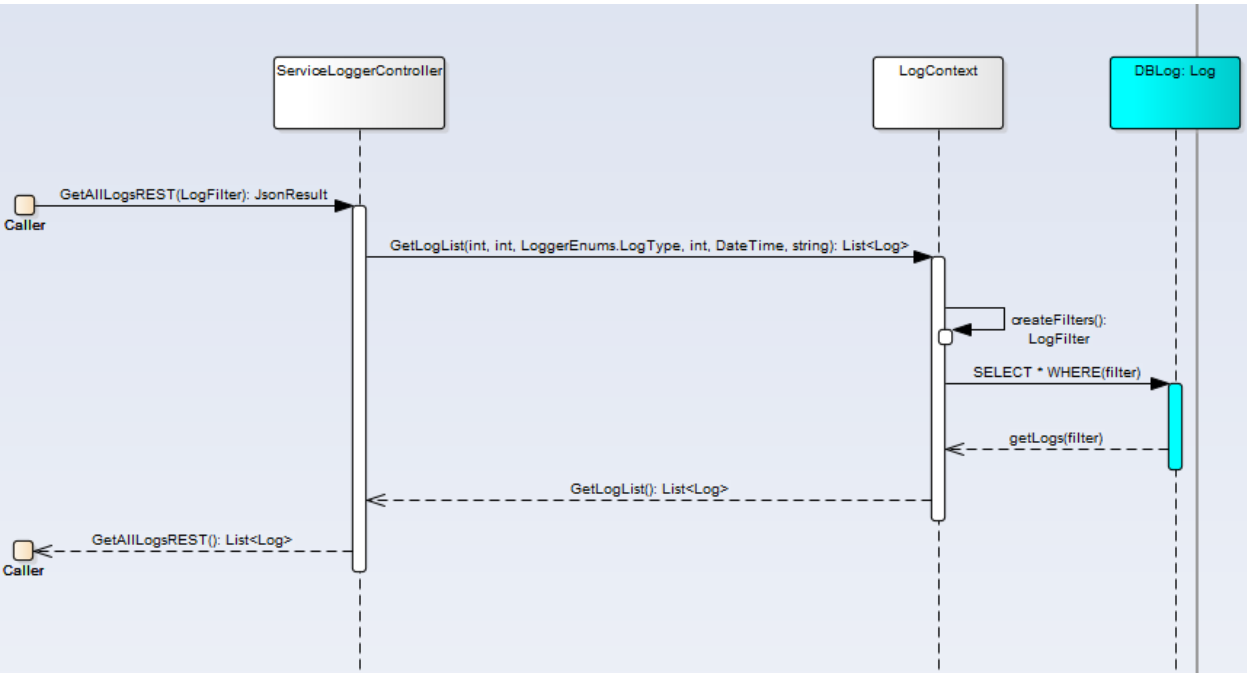
Po zavolaní REST služby Post(Log), ktorá sa nachádza na ceste *api/logger/post* sa automaticky zvaliduje objekt, ktorý prišiel do služby a následne sa pošle do LogContext a uloží sa do databázy. Metóda vracia boolean, podľa toho, či sa podarilo uložiť log, alebo nie. DBLog predstavuje už priamo databázovú tabuľku a ukladanie do nej. V prípade, že sa log podarí uložiť do DBS služba vracia HTTP status kód 201 – *Created*, v prípade, že sa nepodarí log uložiť do DBS služba vracia HTTP status kód 400 – *Bad request.* Aby sme zabránili napadnutiu služby pre zaznamenávanie logov, bolo potrebné vytvoriť zabezpečenie tejto služby. Za týmto účelom sme využili štandardnú zabezpečovaciu pipeline, ktorú poskytuje technológia Web API 2. Podrobnejší opis tejto pipeline sa nachádza v dokumentácii modulu CORD:

https://onedrive.live.com/redir?resid=957CC12FD13AF109!5765&authkey=!AOW8OJ8esRzQ9nU&ithint=file%2cdocx

Vzhľadom na to, že pri návrhu tejto autentifikácie a autorizácie sme využili analogický postup, nie je potrebné ho opisovať znova. Na obrázku 3 sa nachádza diagram tried pre autorizáciu a autentifikáciu požiadavky na vloženie logu do databázy.



Obrázok 3: Diagram tried pre autorizáciu požiadavky pre vloženie logu



Obrázok 4 Sekvenčný diagram na zobrazenie logov

Diagram zobrazuje vyhľadanie logov, ktorý sa aktivuje zavolaním REST metódy GetAllLogsREST(), do ktorej vchádza filter ako argument. Tento filter sa automaticky zvaliduje a pošle sa do LogContext triedy, kde sa spravia filtre čitateľné pre mongo databázu a následne sa zavolá select do databázy. Databáza vráti list logov.

3 Implementácia

Trieda Log v súčasnom stave obsahuje tieto premenné:

- **Log type** predstavuje typ logu, ktorý je ohraničený ako enum na obrázku 1 (LogType). Enum all je spravený kvôli vyhľadávaniu a dokáže nájsť všetky typy logov.
- **ResponseHttpCode** - vrátený http kód pri odchytenom logu.
- **RequestUrl** - daná URL pri odchytenom logu.
- **HttpMethod** - http metóda pri logu, napríklad POST, GET, PUT a podobne.
- **Timestamp** - premenná predstavuje aktuálny čas, kedy bol odchytený log.
- **ErrorStackTrace** - informačná správa, ktorá je odchytená pri výnimkach.
- **ModuleName** - je názov modulu, odkiaľ bol odchytený log.

- **AdditionalProperties** sú ďalšie premenné, ktoré by mohli byť odchytené pre budúce účely.

Konštruktor na vytvorenie logu má prednastavené hodnoty, kvôli filtrovaniu (ak bola zadaná default hodnota, nájde všetky logy):

```
public Log(LoggerEnums.LogType logType = 0,
int httpCode = 0,
string url = "",
DateTime timestamp = new DateTime(),
string httpMethod = "",
string errorStackTrace = "",
string module = "")
```

1.1. Zobrazovanie logov:

Všetky logy sa zobrazujú v súbore ShowLogs.cshtml v dive s id tabs-1, ktorý predstavuje prvú podstránku logov. Momentálne sa používa na zobrazovanie logov bootstrap zoznam a je ohraničený na 15 najnovších záznamov.

Na filtrovanie sú na stránke `<input/>` elementy, podľa typu filtra. Na stránke je aj implementovaný dateTime picker pomocou bootstrap.css a bootstrap.js, ktorý bol importovaný pomocou NuGet. Pomocou datetime picker sa dá vyhľadávať podľa dátumu a času. Momentálne je možné zvoliť dátum od aj dátum do kedy sa majú filtrovať logy. Taktiež je možné nastaviť filtre typ logu, http metóda, http kód odpovede. Všetky tieto filtre sa naplnia do payloadu filtrovacej metódy a sú ohraničené triedou LogFilter.

Po kliknutí na tlačidlo vyhľadaj sa zavolá REST metóda:

```
[HttpPost]
public JsonResult GetFilteredLogs(LogFilter payload)
```

, do ktorého sa vložia filtre, podľa vyplnených hodnôt na stránke. Všetky logovacie filtre sú opísané na obrázku 1 v triede LogFilter.

Skip je počet preskočených záznamov a count je počet záznamov, ktoré vráti táto služba. Tieto dve premenné slúžia na implementované stránkovanie.

Táto služba vráti záznamy vo formáte Json, ktoré sa následne spracujú a prekreslia bootstrap zoznam, ktorý sa vyskladáva pomocou javascriptu v súbore logger.js v metóde

createCollapsibleRecordElement(). V tomto súbore sa nachádzajú metódy na stránkovanie a populovanie dropdown listov na html stránke.

Ak boli zadané zlé dáta do tejto služby služba vráti http kód 400 (BAD REQUEST). Ak sa podarilo vrátiť validné dáta, prípadne prázdnu kolekciu služba vráti 200 (OK).

1.2. Vytvorenie logu:

Na vytvorenie logu z akéhokoľvek prostredia projektov je možné pomocou REST služby:

```
public IHttpActionResult Post(Log log)
```

, ktorá berie ako parameter objekt typu Log a uloží ho do databázy. Ak boli uložené nesprávne dáta služba vráti 400 (BAD REQUEST). Ak prebehlo všetko správne vráti daný objekt ako Json a http kód 201 (CREATED). Fungovanie je rozpísané na obrázku 2 v sekvenčnom diagrame. Pre potreby zabezpečenia služby bol implementovaný *DevActsApiAuthorizationAttribute*, ktorý slúži na zabezpečenie API kontrolórov prostredníctvom servisného účtu. Tento autorizačný atribút pracuje v spolupráci s autentifikačným handlerom a zabezpečuje, aby neautentifikovaný používateľ nemohol vkladať logy do aplikácie volaním danej REST služby. Celá zabezpečovacia pipeline API controllerov je opísaná v dokumentácii modulu CORD:

https://onedrive.live.com/redir?resid=957CC12FD13AF109!5765&authkey=!AOW8OJ8esRzQ9nU&ithint=file%2cdocx

V module DevActs bola použitá analogická metóda a preto ju nie je potrebné znova opisovať. Nutné je podotknúť, že služba na vkladanie logu sa nenachádza v MVC controlleri ServiceLoggerController, ale v API ServiceLoggerControlleri, ktorý je súčasťou Web api technológie pri implementácii bolo použité aktuálne najnovšie dostupné Web API 2, ktoré umožňuje nastavovať cesty priamo prostredníctvom atribútov nad metódami.

1.3. Štatistiky

Na frontende sa nachádza dropdown list, ktorý zobrazuje všetky moduly, z ktorých sú zozbierané logy. Napopuluje sa pomocou služby GetModules(), ktorá sa nachádza v controlleri logov. Táto metóda pristupuje do databázy vráti DISTINCT názvy všetkých modulov. Táto metóda sa zavolá len raz pre každý dropdown list pri načítaní danej stránky.

Pri zobrazovaní a filtrovaní štatistík je možné zadať aj počet dní dozadu, odkiaľ sa budú maximálne brať logy. Teda ak si používateľ zadá 7 tak bude brať štatistiky za posledný týždeň. Toto je implementované aj pre diagram aj pre zoznamové základné zobrazenie štatistík.

Na samotnú štatistiku sa volá služba GetModuleStatistics do ktorej vchádzajú dva parametre module a daysAgo, ktoré boli spomínané. Do tejto metódy sa dostanú pomocou URL, kde sa zapíšu ako parameter = hodnota oddelené znakom AND. Služba následne pristupuje k databáze a pomocou agregácie a zgrupovaniu vráti zgrupené počty podľa typu logu. Teda napríklad WARNING 354, ERROR 112 a podobne. Tieto dáta sa spracujú do rozumného formátu a vrátia na frontend, kde sa spracujú.

Po zavolaní tejto služby sa pomocou google api vytvorí dátový objekt dataTable, ktorý sa naplní pomocou získaných dát a zavolá sa samotné vykreslenie grafu (Histogram) na tabe číslo 3. Taktiež je možné zobraziť tieto štatistiky aj pomocou tabuľky na tabe číslo 2. Spracovanie štatistík sa vykonáva v súbore statistics.js.

4 Testovanie

Testovanie sa robilo v ServiceLoggerControllerTest.cs kde je metóda ShowLogsTest() a LogInsertTest() (táto sa nachádza v triede ServiceLoggerControllerTest.cs v priečinku API), kde sa testujú rest-ové metódy a nasimuluje sa RepositoryManager a TestLogContext.