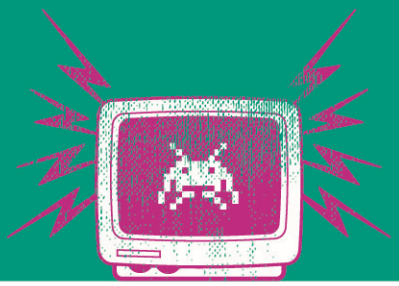
Informatik



BIF/SWE 1 - Übungsbeispiel

Arthur Zaczek

Sep 2015

1 Übungsbeispiel – embedded sensor cloud

1.1 Aufgabenstellung

Auf einem *[[tragen Sie bitte hier Ihr Embedded System Ihrer Wahl ein]]* soll ein Messdaten Erfassungssystem implementiert werden. Seine Aufgabe ist es, mittels Plugins Messdaten zu erfassen und zu speichern. Die Messdaten können dann mittels einer REST Schnittstellt von der Sensor Cloud ausgelesen werden. Die Konfiguration des Systems erfolgt über ein Web Interface.

Aus dieser kurzen Angabe ergeben sich folgende konkrete Aufgabenstellungen:

- Implementierung eines WebServer
  - Annahme: Auf einem Embedded System sind die Resourcen begrenzt, daher kann kein Windows bzw. LAMP Server ausgeführt werden
  - Annahme: Jedoch ist das System in der Lage, ein Linuxsystem mit Java bzw. dem Mono-Framework auszuführen
  - Für die Übung: Der WebServer ist selbst zu implementieren, einzig die Methoden URLEncode/URLDecode dürfen aus der BCL verwendet werden.
- Der WebServer kann über den Port 8080 erreicht werden
- Der WebServer muss multiuserfähig sein
- Der WebServer muss über ein Plugin System verfügen. Diese Plugins führen den konkreten Request aus.
  - Welche Plugins vom Webserver genutzt werden, muss konfigurierbar sein. Weitere, „Dritthersteller Plugins“, müssen ohne neue Kompilierung des Web-Servers hinzugefügt werden können.
- Die Messdaten müssen in einer Datenbank Ihrer Wahl gespeichert werden.
  - Um die Übung zu vereinfachen müssen die Plugins NICHT die Datenbank verwalten können. Sie dürfen sich auf ein korrektes Schema verlassen.

1.2 Plugins

Temperatur Messung

- Ein eigener Thread liest laufend einen Sensor aus
- Für die Übung: Es werden Zufallszahlen oder ein Sinus, etc. berechnet
- Diese Messwerte werden in der Datenbank ihrer Wahl gespeichert
  - Implementieren Sie den DAL selbst, es sind keine OR-Mapper erlaubt.
- Auf der WebOberfläche können die Messdaten dargestellt werden
- Für die Übung: Erzeugen Sie mindestens 10.000 Messwerte verteilt über die letzten 10 Jahre.
- Bei 10.000 Messwerten muss die Oberfläche zwangsläufig eine Suche bzw. Blättern unterstützen
- Eine REST Abfrage http://localhost:8080/GetTemperature/2012/09/20 soll alle Temeraturdaten des angegebenen Tages als XML zurück geben. Das XML Format ist frei wählbar.

Statische Dateien

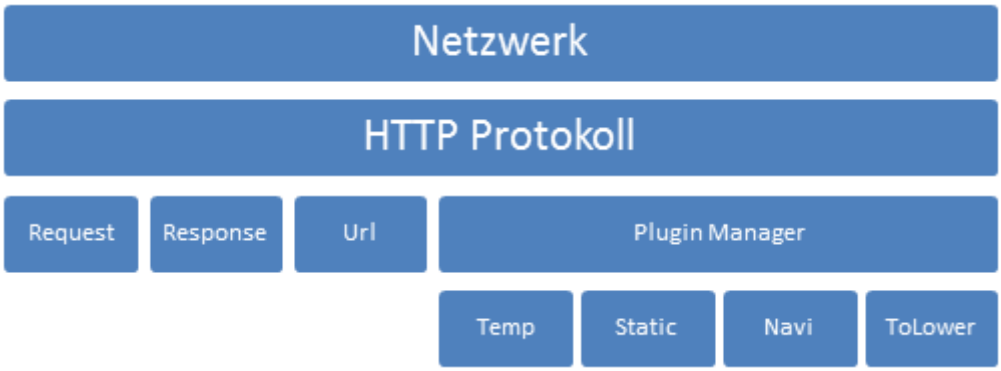


Figure 1: Architektur

- Dieses Plugin soll einfach nur Dateien aus dem Filesystem an den Browser zurück senden.
- Diese Dateien sollen dazu genutzt werden, um im HTML Bilder, stylesheets, scripte, etc. einbinden zu können
  - Es soll explizit kein “Downloadmanager” oder “Filemanager” sein.

Navi

- Als extra Feature, weil das Embedded System Speicherkarten aufnehmen kann
- In einem Textfeld wird ein Straßenname eingegeben
- Das Plugin gibt anschließend eine Liste aller Orte aus, in denen die Straße existiert.
  - Arbeiter Strandbad Straße
  - -> In Wien.
- Eine Karte mit allen Straßen sowie Orten kann von OpenStreetmap bezogen werden.
  - Die Datei müsste ein ca. 4 GB großes XML Dokument sein
  - http://download.geofabrik.de/osm/europe/
  - Es genügt, wenn einer/eine die Datei organisiert und allen anderen weitergibt.
- Eine eigene Seite im Navi Plugin kann den Befehl: „Staßenkarte neu aufbereiten“ auslösen
  - Dieser Befehlt liest die OpenStreetmap Karte (neu) ein und erstellt eine interne Straßen <-> Ort Zuordnung im Hauptspeicher (z.B. in einem Dictionary oder HashTable). Wer möchte, kann das auch in der Datenbank speichern, ist aber nicht Pflicht.
  - Dies ist mittels SAX Parsers zu realisieren
  - Es genügen die POI Tags! Keine geografischen Zuordnungen durchführen!
  - XPath: `//node|way/tag[@k,@v]`
  - Während dieser Aufbereitung darf das Plugin keine anderen Abfragen annehmen dürfen. Es muss stattdessen eine Warnmeldung ausgeben (Multiuserfähigkeit)

ToLower

- In einer textarea kann beliebig langer Text eingegeben werden
- Ein Submit-Button sendet den Text mittels POST Request an den Server
- Am Server wird der Text in Kleinbuchstaben konvertiert (.toLower())
- Das Ergebnis wird in einem PRE Tag unterhalb der textarea dargestellt

1.3 Architekturübersicht

1.4 20 Unit Tests

Implementieren Sie 20 eigene Unit Tests. Testen Sie z.B. Ihre Plugins genauer.

1.5 JavaDoc/Doxygen/XML Documentation Comments

Versehen Sie alle *public* Klassen/Methoden/Eigenschaften mit Kommentaren, aus denen mittels Tools eine API Dokumentation erstellt werden kann. Führen Sie das Tool Ihrer Wahl aus und zeigen Sie uns das Ergebnis.

Wenn Sie unsicher sind bzgl. des Tools dann fragen Sie uns.

Mögliche Tools:

- Doxygen
- Sandcastle

Falls Sie ein anderes Tool nutzen möchten, können Sie dies sehr gerne tun. Evtl. sprechen Sie sich mit uns ab.

1.6 Nicht Dokumentierte Anforderungen

Anforderungen, welche nicht explizit dokumentiert oder durch Unit-Test festgelegt sind, dürfen Sie selbst bestimmen, wie sie diese umsetzen.

Wenn Sie uns fragen, werden Sie Antworten erhalten.

1.7 Dokumentation

Ausgedruckt 1 ca. A4 Seite lang.

Inhalt

1. Benutzerhandbuch – Wie wird die Applikation verwendet
2. Lösungsbeschreibung – Wie wurde die Aufgabe gelöst
3. Worauf bin ich stolz
4. Was würde ich das nächste mal anders machen

1.8 Technologien

Bereich	Java	.net
Sprachen	Java	C#
Datenbank	JDBC	ADO.NET

2 Bewertung

2.1 Automatisierte Tests

Die Übung wird automatisiert getestet und bewertet. Näheres finden Sie im Moodle-Kurs.

Die Unit-Tests geben Ihnen eine Struktur vor, *wie* sie die Anforderungen zu implementieren haben.

2.2 Codereview

In der letzten Übung findet ein Code-Review statt welches gesondert bewertet wird. Benötigt wird:

- Ein lauffähiges Programm (.exe, .jar)
- SourceCode

Bewertung	Punkte	Max. Punkte
Zeiterfassung wurde kontrolliert und eingetragen! (Ja)		Ja
Sockets		
Multiuser fähig		1
PluginManager		
Plugins können dynamisch geladen werden		2
Plugin Temperaturmessung		
Laufende erzeugung von Testdaten		0,5
10.000 Messwerte bereits erzeugt		0,5
Speichern in einer Datenbank		1
REST Schnittstelle		1
Blättern		1
Datumauswahl		1
Plugin Statische Dateien		
CSS/Javascript/Html wird korrekt zurück gegeben		1
Beliebige Dateien aus einem Verzeichnis		1
Plugin Navi		
Suche nach Orten bei Eingabe einer Straße		1
Karte neu laden Befehl funktioniert wie beschrieben		2
Plugin ToLower		
toLower		1
Text wird mittels POST im HTTP Body übertragen		1
Nichtfunktionale Anforderungen		
20 eigene Unittests		3
Javadoc/Doxygen/XML Documentation Comments & Dokumentation generiert		1
Dokumentation der Architektur lt. Angabe (1 A4 Seite)		1
Abzüge		
Sonstiges 1		
Sonstiges 2		
Sonstiges 3		
Summen	**Zeiterfassung f**	**20**

Figure 2: Notenrechner Codereview

2.3 Notenrechner