VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Webová služba pro sběr a vizualizaci předpovědí počasí

Web service for collecting and visualizing weather forecasts

2021 Jan Jelička

Zadání bakalářské práce

Jiří Dvorský

Ukázka sazby diplomové nebo bakalářské práce

Diploma Thesis Typesetting Demo

+++

Podpis vedoucího katedry

+++

Podpis děkana fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 1. dubna 2016 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 1. dubna 2016 .....

Rád bych na tomto místě poděkoval vedoucímu bakalářské práce panu Ing. Janu Janouškovi, za pravidelné konzultace a poskytnutí mnoha rad a nápadů pro řežení samotné práce.

Abstrakt

Cílem bakalářské práce bylo vytvořit aplikaci, která bude schopna shromažďovat data o počasí z různých datových zdrojů v různých formátech (text XML, text JSON, bitmap). Agregovaná data jsou následně poskytována pomocí webové služby v jednom formátu (bitmap). Webová služba poskytuje data pro určité území v danném čase. Posledním bodem je vizualizační aplikace, která poskytuje uživateli možnost vykreslení počasí pro určité území v čase a také zobrazuje předpověď pro zadanou trasu.

Klíčová slova: XML; JSON; bitmap; počasí

Abstract

The aim of the bachelor thesis was to create an application that will be able to collect weather data from various data sources in various formats (XML text, JSON text, bitmap). The aggregated data is then provided using a web service in one format (bitmap). The web service provides data for a specific territory at a given time. The last point is a visualization application that provides the user with the ability to plot the weather for a certain area over time and also displays the forecast for the specified route.

Keywords: XML; JSON; bitmap; forecast

Obsah

Seznam použitých zkratek a symbolů

XML	–	Extensible Markup Language
JSON	–	JavaScript Object Notation
BMP	–	Bitmap
HTML	–	Hyper Text Markup Language

1 Úvod

Informace o počasí jsou v dnešní době distribuována mnoha službami v různých podobách. Nejčastěji narážíme na textové formáty (XML/JSON) kde sproztředkovávatelé dodávají kompletní výpis informací pro stát, město nebo konkrétní bod na základě zeměpisných souřadnic. Mimo textový formát narážíme i na snímky z radaru, které poskytují předpověď pro rozsáhlou plochu v konkrétním čase. Předpovědi jsou vytvářeny pro různé časové intervaly na rozdílnou dobu dopředu, můžete tedy například narazit na předpověď obsahující data na 24 hodin dopředu s hodinovými rozestupy nebo na týden s šesti hodinovými rozestupy. Vzhledem k tomu že ke změnám počasí dochází relativně pomalu tak není potřeba znát data pro každou minutu, stačí nám předpověď jednou za pár hodin.

Cílem této práce je tedy sjednotit různé datové zdroje do jednotného formátu a vytvořit předpověď počasí dle průměru těchto dat. Mimo to že každá služba může mít data ve svém formátu, je potřeba i sjednotit časy a především pozice pro které se data zjišťují.

2 Datové zdroje

V práci potřebujeme použít minimálně 3 různé datové zdroje. Jako první se tedy zvolil norský datový zdroj poskytovaný serverem yr.no, který dodává data v podobě XML. Druhým zvoleným zdrojem jsou JSON předpovědi od společnosti OpenWeather. Poslední zdroj poskytuje data ve formě bitmap reprezentujících snímky z radaru a pro tento účel byl vybrán český projekt Medard.

2.1 XML

2.1.1 yr.no

Tímto datovým zdrojem je norská meteorologická služba zvaná Yr. Poskytují data o počasí pokrývající celý svět, lze si stáhnout data pro určité město nebo bod založený na zeměpisných souřadnicích. Předpovědi jsou vždy od aktuálního času na týden dopředu a rozpetí mezi jednotlivými předpověďmi je 6 hodin. Veškeré informace jsou v podobě XML dokumentu.

2.1.2 ukázka

2.2 JSON

2.2.1 OpenWeather

Dalším datovým zdrojem v podobě textu je OpenWeather. Tato služba poskytuje data ve formátu JSON na týden dopředu s časovým rozmezím 3 hodin. Data se dají získat pro určité město či vesnici případně pro konkrétní bod na základě zeměpisnách souřadnic. Pro získání dat o počasí je potřeba vlastnit API key, který obdrží každý zaregistrovaný uživatel u této služby. Pokud používáte neplacenou verzi této služby tak jste omezeni na 60 dotazů na server za minutu. Tento limit stačí pokud zjišťujete pouze počasí pro jednotlivé body, při určování počasí na ploše (potřeba zjištění počasí na tisíci různých místech) je tento limit omezující a nutí nás čekat na stažení veškerých dat. Placené verze nám dovolují 600, 3 000, 30 000 a 200 000 dotazů na server dle koupeného balíčku.

2.2.2 ukázka

2.3 Bitmap

2.3.1 Medard

Medard je webová služba poskytující informace o počasí ve formě bitmap. Bitmapy pokrývají celou evropu a umožňují nám zjistit počasí na 5 dnů dopředu s pouze jedno hodinovým rozestupem. Díky nízkému časovému rozestupu je možné získlávat velice přesná data. Bitmapy s daty jsou uloženy vedle sebe do jednoho velkého obrázku který je následně potřeba rozdělit do jednotlivých částí.

2.3.2 Radar.bourky

Radar.bourky je datový zdroj který poskytuje data prostřednictvím snímků z radaru, data jsou tedy ve formátu bitmap. Tento datový zdroj je prakticky nepoužitelný, protože poskytuje data pouze v aktuálním čase a pokrávý jen českou republiku. Z toho důvodu se tento zdroj dá využít pouze pro zjištění aktuálního počasí v ČR.

2.3.3 ukázka

3 Agregace dat

Při agregaci dat bylo zapotřebí vyřešit pár otázek. První z nich byla potřeba jednotného výstupního formátu pro různorodá vstupní data, tímto formátem byly zvoleny bitmapy obsahující data o jednom typu počasí, data jsou reprezentována pomocí barev a pokrávají určitou plochu pro určitý čas. Další otázkou bylo jak převádět barvy na číselné hodnoty a hodnoty zpět na barvy, tento problém vyřešilo využití škál. A nakonec jsme potřebovali pokrýt celou bitmapu pokud známe hodnotu počasí pro určíté body, což nám vyřešila triangulace v kombinaci s interpolací.

3.1 Škála

Vzhledem k tomu že veškerá data o počasí jsou uložena do bitmap, ve který jsou data reprezentována barvou pixelů vzniká potřeba převádění barvy na číselnou hodnotu a naopak. Práce proto obsahuje metody, které pro určitou barvu vrátí desetinné číslo a pro určité číslo zase barvu.

Nejprve tyto metody obsahovaly desítky podmínek které staticky kontrolovaly zda se jedná o konkrétní barvu, později zda barva patří do určitého rozmezí pro R, G, B složky. Tento přístup ale není vhodný, protože při potřebě změnit barvy pro určitá desetinná čísla vznikala nutnost přepsat hodnoty barev pro všechny metody ve veškerých podmínkách.

Finálním řešením se stalo využití škál, škály jsou obrázky široké přesné tolik pixelů kolik hodnot uchovávají, kde každý pixel obsahuje unikátní barvu a reprezentuje jednu konkrétní číselnou hodnotu kterou daná barva reprezentuje. Výška škály může být i pouze 1 pixel. Výhoudou škál je, že změna barev reprezentujících hodnoty případně změna rozsahu uchovávanách hodnot se vyměnit pouze pomocí použití nové škály která bude opět orientovaná na šířku. Další z výhod je že metody které vrací hodnotu z barvy nemusí obsahovat desítky podmínek a stačí pouze vrátit hodnotu uloženou pro pixel. V programu jsou barvy ze škály uložený do slovníku, kde klíč reprezentuje barva a hodnotu desetinné číslo. Pro určení čísla z barvy stačí vypočítat index na kterém barva leží a tuto barvu vrátit.

3.2 Triangulace

Při vytváření bitmap dochází k určování hodnot počasí (teplota, srážky atp.) pro konkrétní zeměpisné souřadnice, které jsou následně převedeny na pixely bitmapy. Zjišťovat hodnotu pro

každý jednotlivý pixel by bylo velice časově náročné, a protože můžeme předpokládat že v okolí pixelu budou obdobné hodnoty jako na pixelu samotné stačí nám určit hodnoty pouze pro určité množství pixelů rozložených správně po bitmapě. Následně je potřeba tyto pixely nějakým způsobem propojit, zde nám problém řeší využité triangulace.

V práci se využívá S-hull Algoritmus pro triangulaci. Konkrétně Phil Atkinova implementace pro C#. Algoritmus nám množinu bodů, v našem případě pixelů, rozdělí na trojuhelníky. Respektive nám pro každý vrchol řekne s kterými vrcholi je spojen hranou, čímž nám vznikne síť trojuhelníků pokrývající většinu bitmapy.

S-hull algoritmus slouží pro vytvoření Delaunayovi triangulace z množiny 2D bodů s časovou složitostí $O(nlog(n))$. Algoritmus využívá radiálního šíření, které se postupně vytváří z radiálně seřezené množiny 2D bodů, a je zakončen převracením trojuhelníků čímž se získá Delaunayova trinaguzlace. Tento algoritmus ve srovnásí s Q-hull algoritmem dosahuje přibližně polovičního času při vytváření triangulace pro náhodně generované množiny 2D bodů. S-hull je pro množinu unikátních 2D bodů x_i implementován následovně:

1. Vybere počáteční bod x_0 z množiny bodů x_i.
2. Seřadí body dle vzdálenosti od tohoto bodu $|x_i - x_0|^2$.
3. Nalezne bod x_j, který je k bodu x_0 nejblíž.
4. Nalezne bod x_k, který vytvoří nejmenší kružnici opsanou s body x_0 a x_j současně i zaznamená střed kružnice opsané C.
5. Seřadí body x_0, x_j, x_k pro získání pravorukého systému, tohle je počateční prvek pro convex hull.
6. Přetřídí zbývající body na základě vzdálenosi bodů od středu kružnice opsané $|x_i - C|^2$ pro získání bodů s_i.
7. Postupně se přidávají body s_i do rostoucího 2D convex hull který je závislý na trojuhelníku vytvořeném z bodů x_0, x_j, x_k. Následné jsou přidány zkosené hrany pro 2D-hull, které jsou bodu viditelné z nově vytvořených trojuhelníků.
8. Vzájemně se nepřekrývající triangulace pro množinu bodů je nyní vytvořena. Tato metoda je velice rychlá mezi způsoby vytváření 2D triangulace.
9. Sousední páry trojuhelníků této triangulace musí být „převráceny“ aby došlo k vytvoření Delaunayovi triangulace z počáteční nepřekrývající se triangulace.

3.3 Interpolace

Bitmapa je pokryta sítí trojuhelníků kde známe hodnotu každého vrcholu. Nyní vzniká potřeba každý trojuhelní vyplnit barvamy, které reprezentují hodnotu pixelů uvnitř trojuhelníku. Tuto

práci řeší využití interpolace, nebole výpočet hodnoty uvnitř objektu na základě vzdálenosti od vrcholů.

Pro výpočet interpolace postačí jednoduchý vzorec, který na základě hodnot vrcholů trojuhelníku a vzdálenosti od nich pro zjišťovaná bod určí jakou hodnotu sám zjišťovaný bod má.

V prvním kroku výpočtu je zapotřebí spočítat vzdálenost $Distance$ zjišťovaného bodu P od všech tří vrcholů trojuhelníku V_1, V_2, V_3.

$$Distance_{v1} = \sqrt{(X_{v1} - P_x)^2 + (Y_{v1} - P_y)^2}$$

$$Distance_{v2} = \sqrt{(X_{v2} - P_x)^2 + (Y_{v2} - P_y)^2}$$

$$Distance_{v3} = \sqrt{(X_{v3} - P_x)^2 + (Y_{v3} - P_y)^2}$$

Následně se určí váha každého vrcholu W, neboli inverní hodnota jeho vzdálenosti od zjišťovaného bodu.

$$W_{v1} = \frac{1}{Distance_{v1}}$$

$$W_{v2} = \frac{1}{Distance_{v2}}$$

$$W_{v3} = \frac{1}{Distance_{v3}}$$

Ve finálníčásti výpočtu se určí hodnota našeho zjišťovaného bodu $Value_p$, která je úměrná podílu součtu součíná váh a hodnot jednotlivých vrcholů trojuhelníku se seoučtem jendotlivých váh.

$$Value_p = \frac{W_{v1}Value_{v1} + W_{v2}Value_{v2} + W_{v3}Value_{v3}}{W_{v1} + W_{v2} + W_{v3}}$$

4 Distribuce dat

5 Vizualizace dat