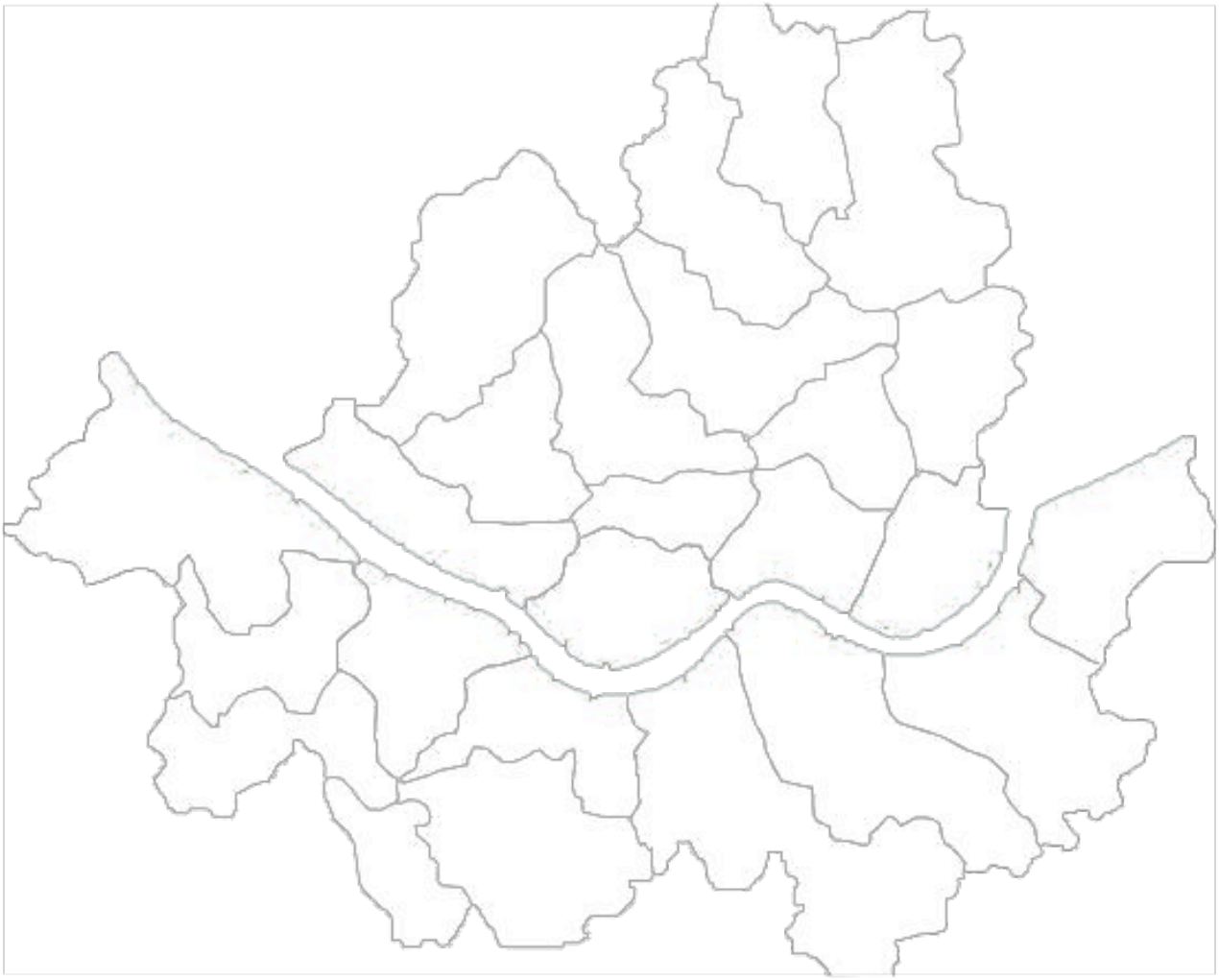


Alfeios 프로젝트

부제 - 서울시 하천 수위 관제 시스템

작성자: 김선호, Web 파트 연구원

2017년 7월 10일

프로젝트 요약

목적

서울 열린데이터 광장에서 제공되는 실시간 수도정보 수위 api를 통해 서울 전체의 하천 수위 상황을 쉽게 파악하고 관리할 수 있게 하는 것입니다.

목표

기존 서울 열린데이터 광장에서 제공되는 api는 하천에 따른 현재 수위와 보의 높이 등을 같이 제공하고 있으나, 그에 따른 위치정보나 하천의 이어짐이 나타나지 않아 전체적인 상황을 쉽게 파악할 수 없습니다. 제공되는 하천 정보를 바탕으로 “Wizeye” 에서 제공하는 인터넷 지도와 커스텀 맵 기능을 활용하여, 하천의 위치 별 상황, 지류의 연결 등을 쉽게 확인할 수 있도록 시각화 하는 것이 위 프로젝트의 목표입니다.

솔루션

Alfeios 프로젝트는 정부에서 제공하는 시도별 하천일람을 기초로 하천의 계층구조를 파악하고, 이에 따라 “Wizeye” 에서 제공하는 커스텀맵 기능을 이용하여, 실제 지형이 반영된 인터넷 지도에 line 요소를 이용하여 하천 지류를 쉽게 확인하도록 표시하도록 합니다. 또한, 하천 상황에 따라 Line 요소의 변화를 통해 하천 수위의 위험도를 표기할 수 있게 합니다. 또한 경고를 알리는 lamp의 위치를 실제 관측소가 있는 곳에 두어, 실제 관측소를 제어하는 데에 용이하도록 합니다.

계획 및 진행상황

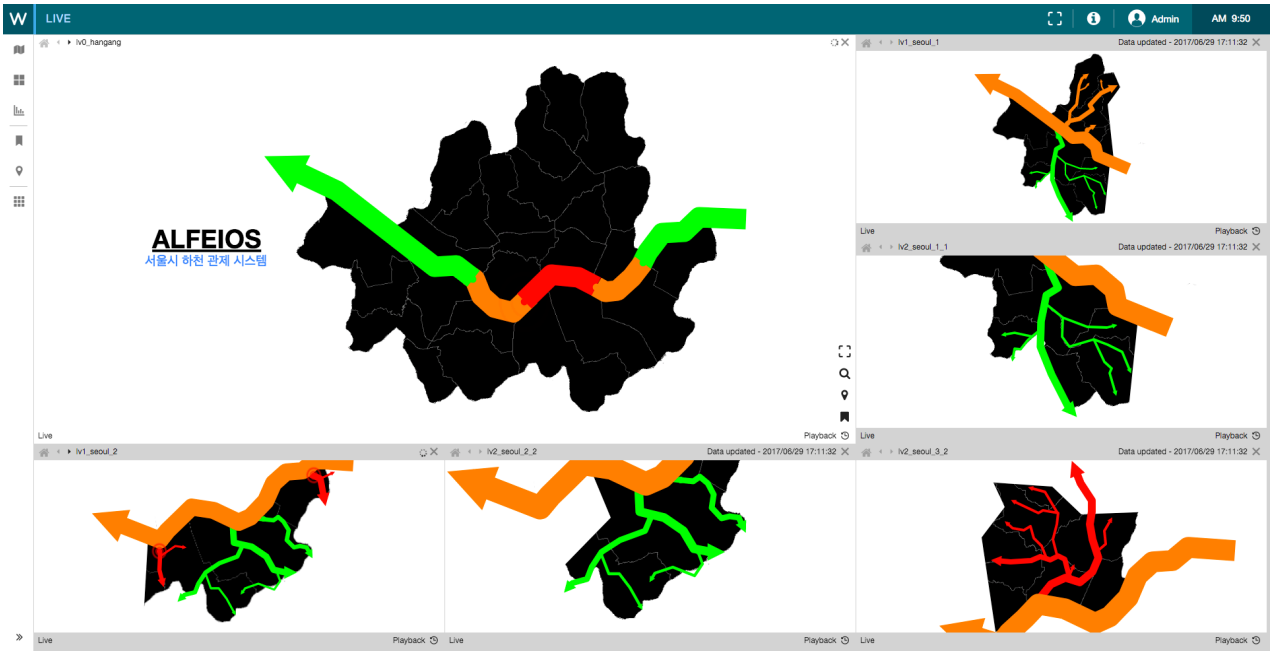
(달성한 부분은 취소선으로 표기)

- ~~서울 열린데이터 광장 실시간 수도정보 수위 API 활용신청~~
- ~~서울시 하천 계층 정보 확보~~
- ~~Node.js 로컬서버 구축을 통해, 1분마다 수위를 가져올 수 있도록 설정~~
- ~~하천 관측소의 위치 정보 확보~~
- ~~Wizeye 서버로 수위 데이터 전송 (http://shkim.dev.wizeye.io:9070/alfeios)~~
- ~~Metric 값으로 전송된 Raw Data 외의 필요한 값은 ADP의 function node를 통해 추가~~
- ~~Wizeye 맵 위에 지도와 함께 하천과 관측소 위치를 고려한 line 및 lamp 요소 배치~~
- ~~하천 관계에 따른 커스텀맵 구축과 계층구조 형성~~
- Data Flow 구성 (추후 나올 Aggregate Node를 감안하여 설계, pt 때 두가지 모델 모두 제시) - 70% 완성
- 시간 여유가 있을 시 하류가 상류에 미칠 위험도 분석을 위한 과거 데이터 수집 (필수 아님)
- 데이터 바인딩
- ppt 작성

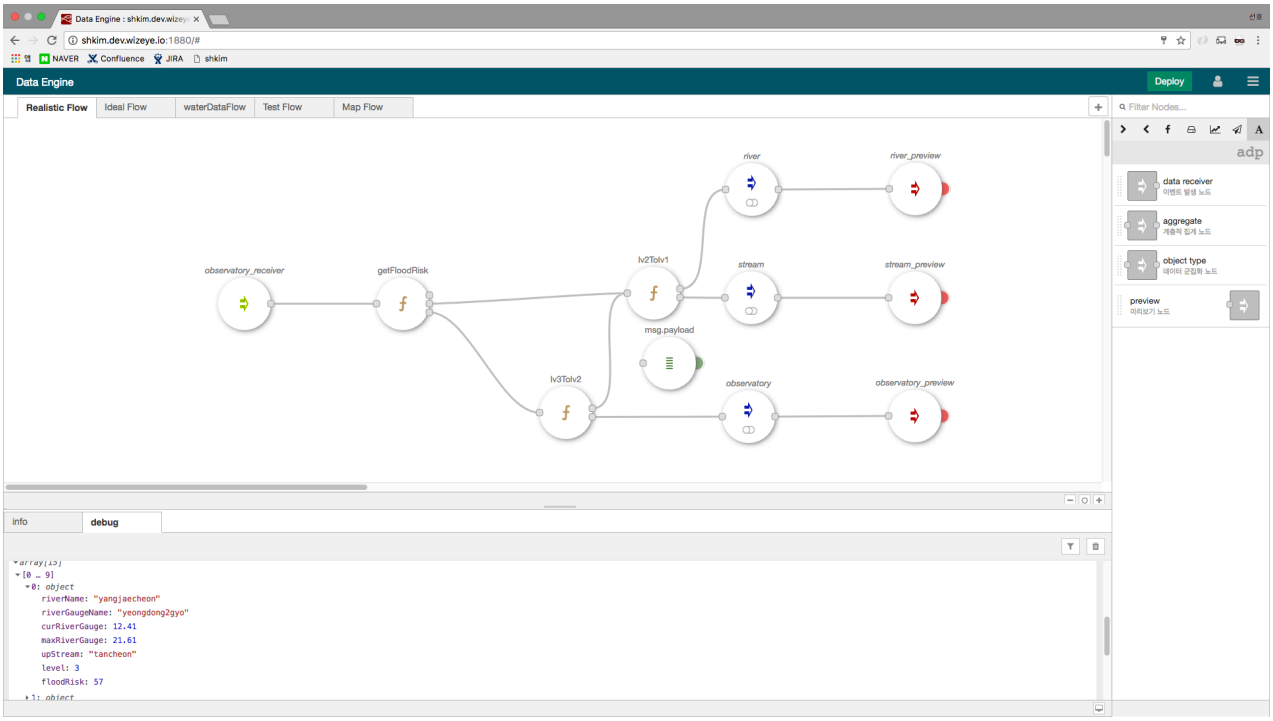
서울시 하천 계층 정보

위 자료를 바탕으로 제 1지류를 기준으로 계층을 분류하면, 서울 시내 하천 지류들은 고덕천, 성내천, 탄천, 중랑천, 반포천, 봉원천, 홍제천, 안양천, 향동천, 창릉천, 굴포천, 아라천 12개의 지류로 나뉩니다.

Custom Map Preview



Data flow Preview



Data Receiver로 보내는 Metric

riverName : 하천 이름, riverGaugeName : 관측소 이름, curRiverGauge: 현재 수위, maxRiverName: 최대 수위, upStream : 상류 이름 (최상류의 경우 null 값 배정, 너무 작은 지류의 영향은 미비하다고 판단하여 최대 lv3까지 형성)

```
var getRiverInfo = function(url) {

    request({ "url": url, "Content-type": "application/json" }, function(error, response, body) {
        if (error) throw new Error(error);

        data = JSON.parse(body);

        var riverInfo = data.ListRiverStageService.row,
            riverCount = data.ListRiverStageService.list_total_count,
            riverName, upStream,
            observatoryInfoObject = null,
            streamInfoObject = null,
            riverInfoObject = null,
            observatoryArray = [],
            streamArray = [],
            riverArray = [];

        for (i = 0; i < riverCount; i++) {
            riverGaugeName = riverInfo[i].RIVERGAUGE_NAME.replace(/(\s*)/g, "");
            riverName = riverInfo[i].RIVER_NAME.replace(/(\s*)/g, "");
            curRiverGauge = riverInfo[i].CURRENT_LEVEL;
            maxRiverGauge = riverInfo[i].LEVEE_LEVEL;

            // 관측소 정보
            observatoryInfoObject = {
                riverName: convertRiverNameKorToEng(riverName),
                riverGaugeName: convertGaugeNameKorToEng(riverGaugeName),
                curRiverGauge: Number(curRiverGauge),
                maxRiverGauge: Number(maxRiverGauge),
                upStream: getUpstream(riverName)
            };

            observatoryInfoObject.level = getLevel(riverName);
            observatoryArray.push(observatoryInfoObject);
        }
        requestData(observatoryArray);
    });
};

//1분 마다 데이터 가져옴
getRiverInfo(URL);
setInterval(function() { getRiverInfo(URL); }, 60000);

```