GET WEATHER SERVICE

(Las2peer Template)

**MỤC LỤC**

**A. Về Las2peer project**…………………………………………………

1. Tổng quan về project Las2peer ……………………………………
   * 1. Tìm hiểu chung …………………………………………………
     2. Kỹ thuật được sử dụng …………………………………………
2. Template Las2peer ………………………………………………..

**B. Về getWeatherService**: ……………………………………………..

1. Ý tưởng …………………………………………………………...

2. Triển khai code …………………………………………………....

* + 1. Xây dựng backend……………………………………………...
    2. Xây dựng giao diện …………………………………………….

**C. Tài liệu tham khảo: …………………………………………………**

**A. Về Las2peer project:**

1. **Tổng quan về project Las2peer**
2. *Tìm hiểu chung:*

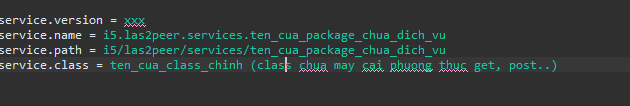
* Cung cấp nền tảng để mỗi cá nhân hoặc doanh nghiệp sử dụng và cung cấp dịch vụ của mình.
* Sử dụng mạng ngang hang P2P
* Dễ dàng triển khai các dịch vụ mới và tích hợp chúng với những dịch vụ khác.
* Mỗi mạng sẽ là một node trên hệ hệ thống dịch vụ. Người khác thông qua las2peer có thể dễ dàng sử dụng các dịch vụ khác có sẵn. Lưu trữ và giao tiếp dữ liệu là bảo mật bằng cách sử dụng mã hóa bất đối xứng.

1. *Kỹ thuật được sử dụng:*

* Nền tảng: Java 8
* Kĩ thuật mạng ngang hang P2P
* Sử dụng RestFul của Jax-rs
* Build và chạy trên Apache Ant

1. **Template Las2peer**

* Thư mục src/main
* Chứa code java: chứa các class (dịch vụ mà mình muốn triển khai)
* Lưu ý, package tạo ra cần phải đặt là con của package cha là **i5.las2peer.services**
* Có thể chứa nhiều code java. Nhưng tối thiểu phải chứa một class mà bên trong đó sẽ xây dựng (kế thừa) từ class RESTService của Jax-rs cung cấp.
* Phương thức GET: lấy dữ liệu
* POST: tạo một dữ liệu mới và đẩy lên (upload)
* PUT: sửa dữ liệu (update)
* DELETE : xóa dữ liệu thỏa mãn các tiêu chí mà người dùng truyền vào
* Thư mục src/test:
* Khởi động server
* Đóng server
* Chạy thử các phương thức được cho ở class chính
* Một điều lưu ý là: kết quả được xuất dưới file TEST-i5.las2peer.services…..trong đường dẫn: export/test\_reports/file.
* JRE System Library: chứa thư viện jdk1.8
* Referened Libraries: chứa các thư viện cần thiết khác trong quá trình xây dựng code (gson,v.v).
* Bin: chứa file khởi động và kết thúc của: mạng hiện tại (network), các tác nhân (agents), các dịch vụ sử dụng dịch vụ hiện tại (services).
* Etc
* Thư mục ant\_configuration:
* Chứa 2 file là service.properties và user.properties để chỉ ra đường dẫn, name và class được sử dụng trong cái đống code java đã nói ở mục src/main.
* Câu hỏi đặt ra là nếu có nhiều dịch vụ mà mình cài đặt ở trong đống code java thì làm sao để có thể kết nối hết? Đơn giản làm mình chỉ cần thêm đoạn mã như sau:



* Thư mục frontEnd: chứa giao diện hiển thị lên web
* Thư mục ivy: chứa các file cài đặt, các file trung gian giúp tìm và cài đặt các thư viện cần thiết
* Thư mục start up: chứa đường link (api của dịch vụ mà bạn đang xây dựng) và các tài khoản, mật khẩu mở khóa dịch vụ
* Export
* Thư mục doc: chứa các giao diện để test api (những phương thức mà mình cung cấp). Thư mục này được tạo tự động nhờ anotition : *@SwaggerDefinition*
* Thư mục jars: chứa file dịch vụ mà mình đã xuất ra ( dùng để upload lên node mạng trên hệ thống las2peer). File được lưu dưới dạng file.jar
* Thư mục test\_reports: chứa kết quả chạy thử ở thư mục src/test
* Lib: các thư viện khác.
* Log: ghi lại toàn bộ hành động khi sử dụng service ( bao gồm cả việc bật tắt nút mạng)
* Service: file nén lại của toàn bộ dịch vụ mình viết ( giống hệt file được lưu ở export/jars)
* Tmp: lưu dự phòng code (ko cần quan tâm lắm :v)
* File build.xml: cài đặt các đường dẫn để:
* Lấy thông tin về tên, đường dẫn đến các dịch vụ được lưu trong file properties
* Bắt đầu nút, tác nhân (dịch vụ, người dùng).
* Lấy và lưu khóa (tài khoản, mật khẩu) để bắt đầu
* Sử dụng các dịch vụ
* Tải và load các thư viện trung gian
* Xóa và khởi tạo các thư mục như tmp, export, lib, startup
* V.v.v

**B. Về GetWeather Service:**

***1)Ý tưởng:***

+)Sử dụng api thời tiết của “openweathermap.io” để lấy dữ liệu thời tiết của các thành phố lớn trên thế giới trong các ngày qua.

+)Sử dụng phương thức GET trong bộ restful của jax-rs.

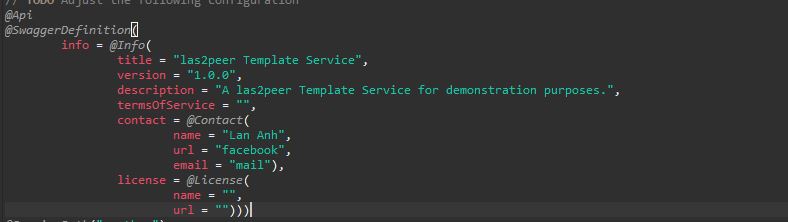
***2) Triển khai code***: do sử dụng Template cảu Las2peer nên những phần chung như build file hay lưu trữ file đã được nói ở mục A của guide này. Trong phần này, chúng ta chỉ quan tâm đến cách xây dựng code trong hàm src/main và thư mục forntEnd trong etc.

a) Xây dựng các class trong thư mục src/main và được đóng gói trong thư mục i5.las2peer.services.getWeatherService

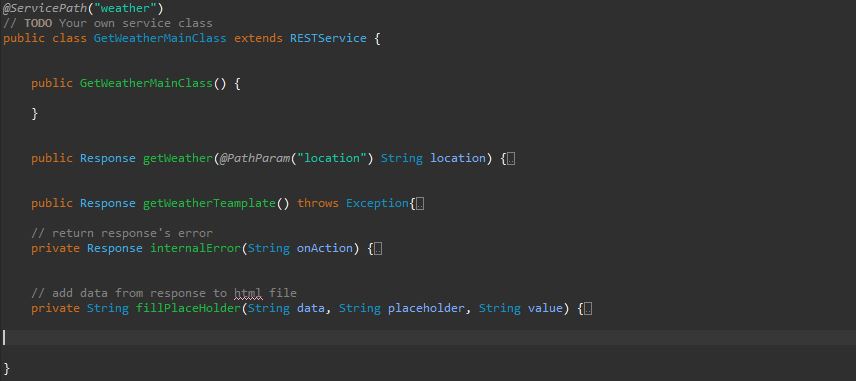
* Xây dựng class Main:
* Gồm các thuộc tính cơ bản cần có về thông tin thời tiết như: nhiệt độ (Temp), độ ẩm (humidity), áp suất (pressure), nhiệt độ lớn nhất, nhỏ nhất trong ngày, v..v..
* Xây dựng các hàm setter và getter để cập nhật là lấy được các dữ liệu có trong hàm main.
* Xây dựng class Weather: bao gồm thông tin của một kiểu thời tiết: icon, mô tả (cái này có thể có hoặc không cũng được
* Xây dựng class JsonResult:
* Gồm những thuộc tính cơ bản (chứa trong class Main) và bổ sung thêm các thuộc tính khác như tên thành phố, mã ,thậm chí là những kiểu thời tiết trong một ngày :v
* Xây dựng hàm chính để thao tác với dữ liệu: GetWeatherMainClass.

(extend phần dịch vụ của RESTService mà Jax-rs cung cấp. Để có thể hiểu rõ hơn về các annotation được sử dụng của jax-rs, có thể truy cập đường link: <https://www.tutorialspoint.com/restful/restful_jax_rs.html> )

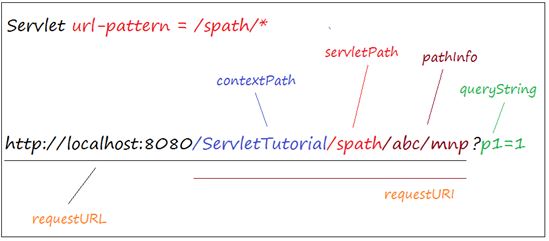
1. Thông tin về Api của getWeatherService (có thể có hoặc không):



1. Cấu trúc code và các hàm có trong code:



* Hiểu hơn về URL và URI:



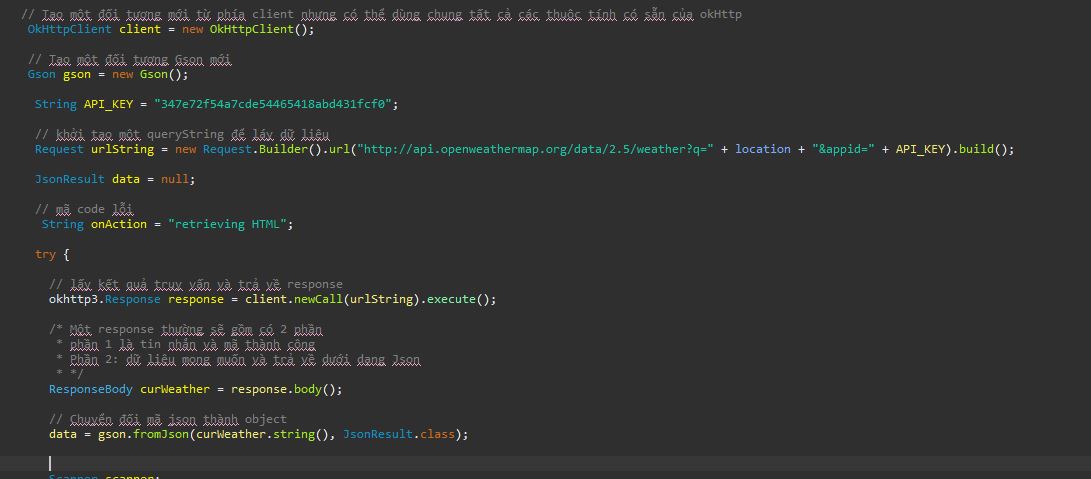
* PathService : weather => các dịch vụ cung cấp bên trong sẽ được bắt đầu bằng localhots../weather/
* Hiểu đúng theo hình thì PathService là contextPath
* Các dịch vụ liên quan đến các chủ đề khác nhau cần có PathService khác nhau, tránh trường hợp viết chung một Class. Ví dụ nếu như lấy thông tin về covid thì cần viết ra một file khác là CovidService và cung cấp PathService cho nó là “covid”.
* Nếu một request được gửi lên, nếu ta không viết rõ phương thức của nó là gì thì mặc định đó là phương thức get. Về cụ thể, ta sẽ tìm hiểu hai phương thức Get được cung cấp trong code dưới đây.
* **getTemplate của service;**



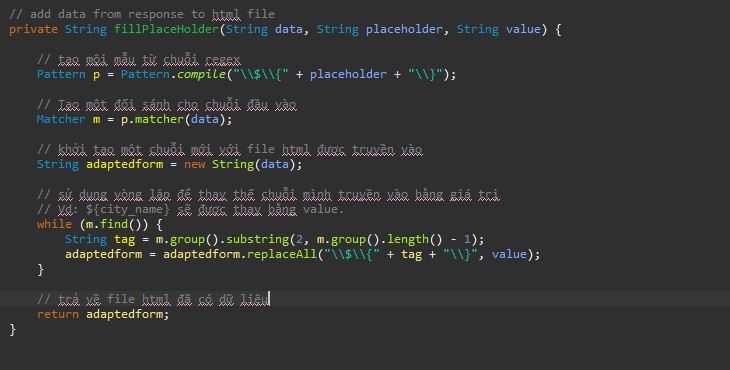
* Phương thức sẽ trả về một response dưới dạng file html để có thể nhìn thấy trên màn hình.
* Sử dụng scanner để quét toàn bộ file html mình cung cấp và so khớp toàn bộ dữ liệu đọc được bằng phương thức useDelimiter(“\\A”). Tham số \A được truyền vào ở đây là một dạng của REGEX. Sau khi quét xong toàn bộ cần đóng scanner.
* Sử dụng hàm fillPalacHolder viết thêm để bổ sung các thông tin dưới dạng các tham số truyền vào bên trong file HTML. Ở đây là NAME\_CITY và có giá trị là ~.
* Trả về một response dưới dạng giao diện html hoặc báo lỗi bằng hàm internalError
* **Phương thức getWeather**



* Link : …/weather/getTemp/location
* Phương thức trả về một giao diện hoàn chỉnh bao gồm cả dữ liệu lấy được từ api.
* Giải thích đoạn code đầu:

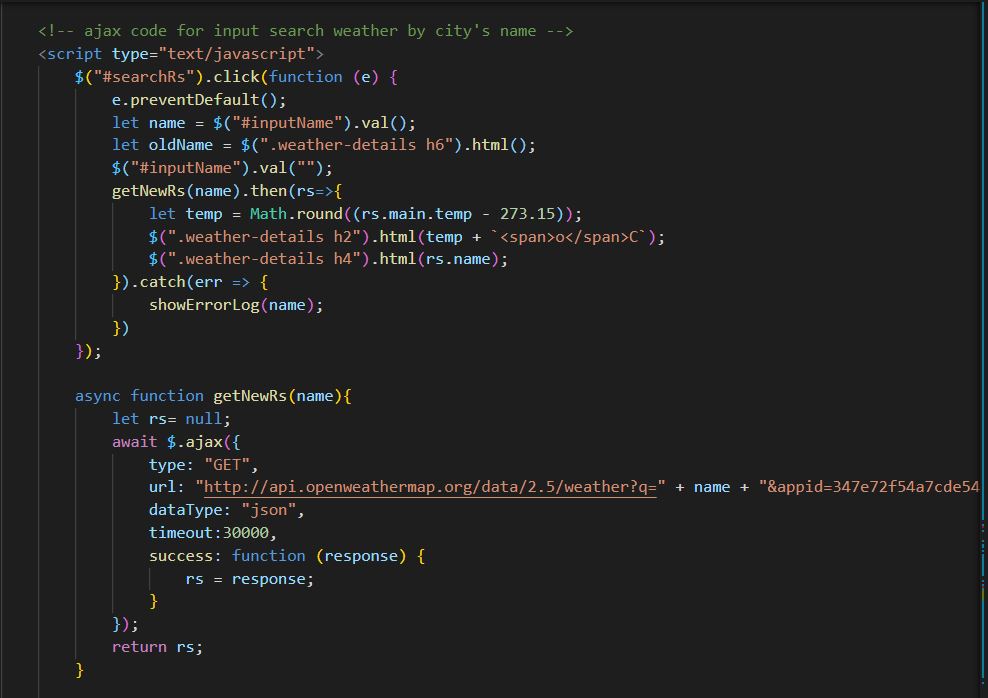


* Hiểu rõ hơn về okHttp : <https://square.github.io/okhttp/>
* Hiểu rõ hơn về Gson: <https://gpcoder.com/3251-huong-dan-su-dung-thu-vien-gson/>
* **Hàm bổ sung : fillPlaceHolder:**

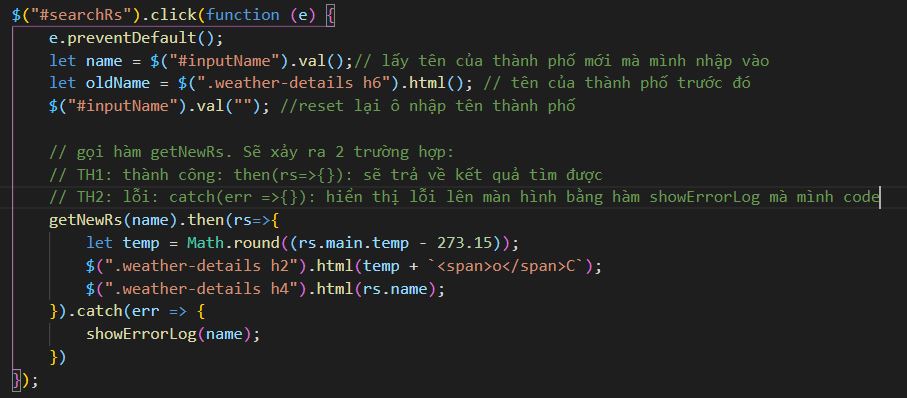
****

b) Tạo giao diện cho website:

* Tạo thư mục frontEnd trong mục etc
* Tạo một file index.html trong đó
* ở đây, tất cả các phần liên quan đến css và js đã được nhúng hết vào file html với mục đích: giảm thời gian link file khi gọi request và tránh lỗi ko link được các file với nhau (một lỗi khá phổ biến khi chạy trên localhost).
* Giải thích thêm về 2 hàm js viết thêm để có thể lấy được dữ liệu cập nhật:



* Hàm getNewRs: đây là một hàm bất đồng bộ trong js, cụ thể là ajax. Hàm có tác dụng lấy dữ liệu mới từ API thời tiết mà ta sử dụng ở phía backend với tên thành phố mà ta nhập mới vào
* Khi click vào nút tìm kiếm ( có id là #searchRs):



**C. Tài liệu tham khảo**

1. Trang chủ của Las2peer : <https://las2peer.org/>

2. Link github của dự án Las2peer : <https://github.com/rwth-acis/las2peer-Template-Project/wiki>

3. Các services của Las2peer cung cấp.

4. Các bài viết liên quan đến lập trình được nhắc đến trong bài.