Revision history

|  |  |  |
| --- | --- | --- |
| Revision # | Date of Revision | Changes Made |
| 1.0 | NA | First Release |
|  |  |  |

Reference documents

|  |  |  |
| --- | --- | --- |
| S.No | Document Title | Date |
| 1 |  | 05 Jun 2017 |

Glossary

| **Terms** | **Description** |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Table of contents

[1 Predix 6](#_Toc489465976)

[2 Back End 6](#_Toc489465977)

[3 Front End 6](#_Toc489465978)

[4 Deployment 6](#_Toc489465979)

[5 Dev Environment Setup 6](#_Toc489465980)

[6 Project management 6](#_Toc489465981)

THIS PAGE IS INTENTIONALLY LEFT BLANK

# Predix

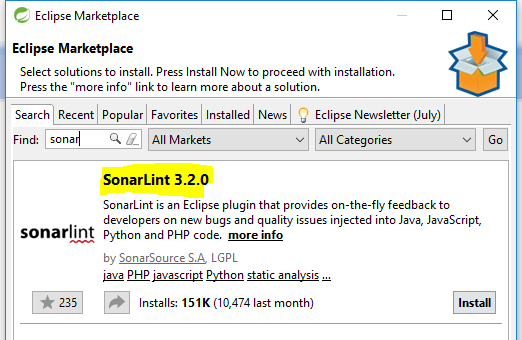
TBD

# Back End

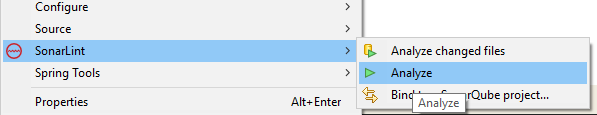
## Learning

1. Plugged in SonarLint for code quality check in STS

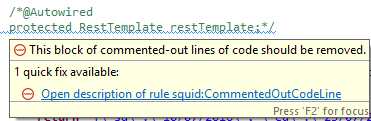
* Go to Eclipse Marketplace, search for sonar and install latest SonarLint.



* Once installation completes, restart Eclipse/STS and you will be able to see SonarLint on right-click of the project.



* After completely analyzed your project by SonarLint, you will be able to see Sonar defects and fix them while in development phase. Ref. below snapshot of issue captured.



## Issue Resolution

1. Identifying the HTTP port number which should connect Consumer and Provider services through Register Service.

**Resolution:**

* No need to explicitly declare any port number in Consumer service. Although port number is needed in Provider and Register services.
* Define default constructor for all the Application Beans in Consumer microservices. No need to define default constructor for application beans in Provider services.
* Declare @LoadBalanced annotation for restTemplate() method in Consumer service bootstrap class.
* Declare dependence for Ribbon in Consumer’s pom.xml

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-ribbon</artifactId>

</dependency>

# Front End

## Learning

1. Before closing the browser we have to trigger some event to warn user about the unsaved Changes(if any in application) , same thing we need it for invalidating the session also by calling the REST API.
2. We have faced issue in implementing and importing the Highcharts library with  our application, which we have resolved it by importing it with alternate way.
3. Angular – 2 command to make component

* ng g component nameComponent --module=app.module.ts(specific to a module)

1. Lifecycle hook of Angular2
   1. constructor
   2. ngOnChanges − When the value of a data bound property changes, then this method is called.
   3. ngOnInit − This is called whenever the initialization of the directive/component after Angular first displays the data-bound properties happens.
   4. ngDoCheck − This is for the detection and to act on changes that Angular can't or won't detect on its own.
   5. ngAfterContentInit − This is called in response after Angular projects external content into the component's view.
   6. ngAfterContentChecked − This is called in response after Angular checks the content projected into the component.
   7. ngAfterViewInit − This is called in response after Angular initializes the component's views and child views.
   8. ngAfterViewChecked − This is called in response after Angular checks the component's views and child views.
   9. ngOnDestroy − This is the cleanup phase just before Angular destroys the directive/component.

## Issue Resolution

1. We have faced issue while fetching the latitude and longitude from table , we have written the code of map in ngOninit(one of lifecycle hooks which comes into the control while loading of component) so at the time of page rendering map get loaded  and we unable to bind the data in the map which already got loaded

**Resolution:**

* Soltn=> Now we are using the different component for map which is now a child component(map-child component) and by the help of ngOnchanges(it triggered on changes only) we will be able to identify the changes happens in child component

# Deployment

## Learning

## Issue Resolution

# Dev Environment Setup

## Learning

## Issue Resolution

# Project management

## Learning

## Issue Resolution

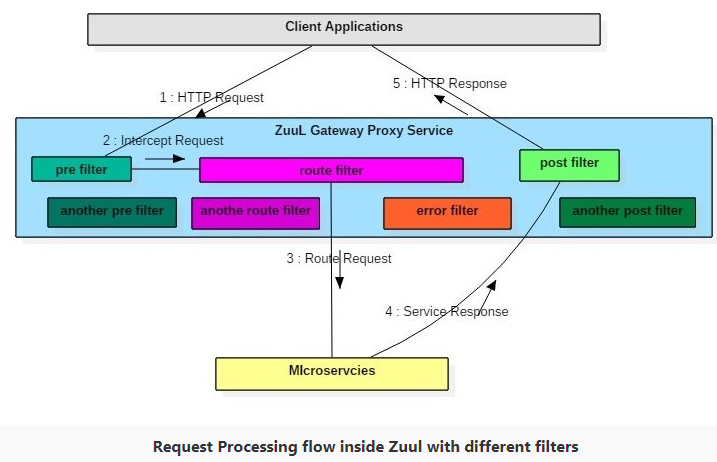
# Predix Cloud

## UAA

# Microservices

## Netflix ZUUL

* Zuul is the front door for all requests from devices and websites to the backend of the Netflix streaming application.
* As an edge service application, Zuul is built to enable dynamic routing, monitoring, resiliency, and security.
* Zuul uses a range of different types of filters that enables us to quickly and nimbly apply functionality to our edge service. These filters help us perform certain functions described below.
* Authentication and Security - identifying authentication requirements for each resource and rejecting requests that do not satisfy them.
* Insights and Monitoring - tracking meaningful data and statistics at the edge in order to give us an accurate view of production.
* Dynamic Routing - dynamically routing requests to different backend clusters as needed.
* Stress Testing - gradually increasing the traffic to a cluster in order to gauge performance.
* Load Shedding - allocating capacity for each type of request and dropping requests that go over the limit.
* Static Response handling - building some responses directly at the edge instead of forwarding them to an internal cluster
* Multiregional Resiliency - routing requests across AWS regions in order to diversify our ELB usage and move our edge closer to our members
* Zuul gives us a lot of insight, flexibility, and resiliency, in part by making use of other Netflix OSS components:
* [Hystrix](http://github.com/Netflix/Hystrix) is used to wrap calls to our origins, which allows us to shed and prioritize traffic when issues occur
* [Ribbon](http://github.com/Netflix/ribbon) is our client for all outbound requests from Zuul, which provides detailed information into network performance and errors, as well as handles software load balancing for even load distribution
* [Turbine](http://github.com/Netflix/Turbine) aggregates fine­grained metrics in real­time so that we can quickly observe and react to problems
* [Archaius](http://github.com/Netflix/archaius) handles configuration and gives the ability to dynamically change properties
* You can integrate Zuul with other Netflix stack components like Hystrix for fault tolerance and Eureka for service discovery or use it to manage routing rules, filters and load balancing across your system
* Zuul has mainly four types of filters that enable us to intercept the traffic in different timeline of the request processing for any particular transaction. We can add any number of filters for a particular url pattern.
* pre filters – are invoked before the request is routed.
* post filters – are invoked after the request has been routed.
* route filters – are used to route the request.
* error filters – are invoked when an error occurs while handling the request.



* Ref. Links:

<https://spring.io/guides/gs/routing-and-filtering/>

<https://blog.heroku.com/using_netflix_zuul_to_proxy_your_microservices>

<https://github.com/Netflix/zuul/wiki>

### Issue faced-1:

* Create a new spring boot application and add the dependencies for spring-cloud and Netflix-zuul.
* Now, if you notice the pom.xml file you can see spring boot version is 1.5.6.RELEASE.
* Now enable spring cloud embedded zuul-proxy using @EnableZuulProxy annotation. This will turn the Gateway application into a reverse proxy that forwards relevant calls to other services.
* Now try running the boot application and you will end up with below error.

Caused by: java.lang.ClassNotFoundException: org.springframework.boot.context.embedded.ServletRegistrationBean

at java.net.URLClassLoader.findClass(URLClassLoader.java:381)

at java.lang.ClassLoader.loadClass(ClassLoader.java:424)

at org.springframework.boot.loader.LaunchedURLClassLoader.loadClass(LaunchedURLClassLoader.java:94)

at java.lang.ClassLoader.loadClass(ClassLoader.java:357)

* Cause of issue:

org.springframework.boot.context.embedded.ServletRegistrationBean was deprecated in spring-boot version 1.4 and its replacement, org.springframework.boot.web.servlet.ServletRegistrationBean, was introduced. The deprecated class was removed in spring-boot version 1.5. @EnableZuulProxy has dependency of this class which, needs to be updated to use the replacement.

* Resolution:

Downgrade the spring boot version to 1.4.x in pom-xml and rebuild the app.

## Ribbon