package hello;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@JsonIgnoreProperties(ignoreUnknown = true)

public class Quote {

private String type;

private Value value;

public Quote() {

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public Value getValue() {

return value;

}

public void setValue(Value value) {

this.value = value;

}

@Override

public String toString() {

return "Quote{" +

"type='" + type + '\'' +

", value=" + value +

'}';

}

}

package hello;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@JsonIgnoreProperties(ignoreUnknown = true)

public class Value {

private Long id;

private String quote;

public Value() {

}

public Long getId() {

return this.id;

}

public String getQuote() {

return this.quote;

}

public void setId(Long id) {

this.id = id;

}

public void setQuote(String quote) {

this.quote = quote;

}

@Override

public String toString() {

return "Value{" +

"id=" + id +

", quote='" + quote + '\'' +

'}';

}

}

package hello;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.web.client.RestTemplate;

public class Application {

private static final Logger log = LoggerFactory.getLogger(Application.class);

public static void main(String args[]) {

RestTemplate restTemplate = new RestTemplate();

Quote quote = restTemplate.getForObject("https://gturnquist-quoters.cfapps.io/api/random", Quote.class);

log.info(quote.toString());

}

}

package hello;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.web.client.RestTemplateBuilder;

import org.springframework.context.annotation.Bean;

import org.springframework.web.client.RestTemplate;

@SpringBootApplication

public class Application {

private static final Logger log = LoggerFactory.getLogger(Application.class);

public static void main(String args[]) {

SpringApplication.run(Application.class);

}

@Bean

public RestTemplate restTemplate(RestTemplateBuilder builder) {

return builder.build();

}

@Bean

public CommandLineRunner run(RestTemplate restTemplate) throws Exception {

return args -> {

Quote quote = restTemplate.getForObject(

"https://gturnquist-quoters.cfapps.io/api/random", Quote.class);

log.info(quote.toString());

};

}

}

import clr  
clr.AddReference(r"C:\Program Files (x86)\PIPC\AF\PublicAssemblies\4.0\OSIsoft.AFSDK")  
import pandas as pd  
  
import matplotlib.pyplot as plt  
import time  
from OSIsoft import AF  
from OSIsoft.AF import \*  
import datetime as datetime  
import statsmodels.api as sm  
  
#DATABASE BAĞLANTISI  
piDB = AF.PI.PIServers().DefaultPIServer  
piPoint = AF.PI.PIPoint.FindPIPoint(piDB,"BA:CONC.1")  
i=0  
future\_data = []  
starttime = time.time()  
  
#while loop başlangıcı...  
while 1:  
 #EĞİTİM VERİLERİ ZAMAN DİLİMİ  
 startTime = AF.Time.AFTime("\*-3d")  
 endTime = AF.Time.AFTime("\*")  
 timeRange = AF.Time.AFTimeRange(startTime, endTime)  
 #VERİLERİN ARALIĞI  
 span = AF.Time.AFTimeSpan.Parse("15m")  
 boundaryType = AF.Data.AFBoundaryType.Inside  
 #TAHMİN VERİLERİ  
 startPredictTime = AF.Time.AFTime("\*")  
 sp = datetime.datetime.strptime(startPredictTime.LocalTime.ToString(), '%m/%d/%Y %I:%M:%S %p')  
 endPredictTime = AF.Time.AFTime("\*+2d")  
 ep = datetime.datetime.strptime(endPredictTime.LocalTime.ToString(),'%m/%d/%Y %I:%M:%S %p')  
  
 #EĞİTİM VERİLERİNİN ÇEKİLMESİ  
 recordedValues = piPoint.InterpolatedValues(timeRange,span, "",False)  
  
 recordedValuesDict = dict()  
 #EĞİTİM VERİLERİNİN DİCTİONARY'e ATILMASI  
 for event in recordedValues:  
 dt = datetime.datetime.strptime(event.Timestamp.LocalTime.ToString(),'%m/%d/%Y %I:%M:%S %p')  
 recordedValuesDict[dt] = event.Value  
  
 #EĞİTİM VERİLERİNDEN OLUŞAN DATAFRAME'in OLUŞTURULMASI  
 df = pd.DataFrame(recordedValuesDict.items(),columns=["TimeStamp","Value"])  
 df['TimeStamp'] = pd.to\_datetime(df['TimeStamp'])  
 indexed\_df = df.set\_index(['TimeStamp']).sort\_index()  
 #EĞİTİM VERİLERİNİN GRAFİK GÖSTERİMİ  
 plt.plot(indexed\_df, color = 'blue')  
  
 #REGRESYON  
 ar\_model = sm.tsa.AR(indexed\_df)  
 pandas\_ar\_res = ar\_model.fit(maxlag=200,method='cmle', disp=True)  
 #TAHMİN  
 pred = pandas\_ar\_res.predict(start=str(sp) , end=str(ep))  
 #TAHMİN VERİLERİNİN GRAFİK GÖSTERİMİ  
 plt.ion()  
 plt.show()  
 plt.plot(pred, color='orange')  
 plt.draw()  
 time.sleep(0.1)  
 plt.pause(0.0001)  
  
 plt.gcf().clear()

C:\Users\komon>python

Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> import requests

>>> from requests.auth import HTTPBasicAuth

>>> requests.get('https://devdata.osisoft.com/piwebapi', auth=HTTPBasicAuth('webapiuser', '!try3.14webapi!'))

<Response [200]>

>>> a = requests.get('https://devdata.osisoft.com/piwebapi', auth=HTTPBasicAuth('webapiuser', '!try3.14webapi!'))

>>> print(a)

<Response [200]>

>>> print(a.text)

{"Links":{"Self":"https://devdata.osisoft.com/piwebapi/","AssetServers":"https://devdata.osisoft.com/piwebapi/assetservers","DataServers":"https://devdata.osisoft.com/piwebapi/dataservers","Search":"https://devdata.osisoft.com/piwebapi/search","System":"https://devdata.osisoft.com/piwebapi/system"}}

>>> print(a.text[1])

"

>>> print(a.text)

{"Links":{"Self":"https://devdata.osisoft.com/piwebapi/","AssetServers":"https://devdata.osisoft.com/piwebapi/assetservers","DataServers":"https://devdata.osisoft.com/piwebapi/dataservers","Search":"https://devdata.osisoft.com/piwebapi/search","System":"https://devdata.osisoft.com/piwebapi/system"}}

>>> restTemplate.getInterceptors().add(new BasicAuthorizationInterceptor("user", "password"));

 Rest servisleri oluşturmak için aşağıdaki anatasyonlar kullanılır.  
**@Controller**  
**@RequestMapping**  
**@PathVariable**  
**@RequestBody**  
**@ResponseBody**  
**@ResponseStatus**

public void shouldReturn200WhenSendingRequestToController() throws Exception {

@SuppressWarnings("rawtypes")

ResponseEntity<Map> entity = this.testRestTemplate.getForEntity(

"http://localhost:" + this.port + "/hello-world", Map.class);

then(entity.getStatusCode()).isEqualTo(HttpStatus.OK);

}

|  |
| --- |
| public class SpringRestTemplateBasicAuthExample { |
|  |  |
|  | public static void main(String... args){ |
|  |  |
|  | RestTemplate rt = new RestTemplate(); |
|  | rt.getMessageConverters().add(new MappingJackson2HttpMessageConverter()); |
|  | rt.getMessageConverters().add(new StringHttpMessageConverter()); |
|  | String uri = new String("https://some.api.provider.com/rest/authenticate"); |
|  |  |
|  | String plainCreds = "user@awesome.com:sfdfsdf$%&^$%4"; |
|  | byte[] plainCredsBytes = plainCreds.getBytes(); |
|  | byte[] base64CredsBytes = Base64.encodeBase64(plainCredsBytes); |
|  | String base64Creds = new String(base64CredsBytes); |
|  |  |
|  | HttpHeaders headers = new HttpHeaders(); |
|  | headers.add("Authorization", "Basic " + base64Creds); |
|  |  |
|  | HttpEntity<String> request = new HttpEntity<String>(headers); |
|  | ResponseEntity<String> response = rt.exchange(uri, HttpMethod.POST, request, String.class); |
|  | log.info("Account Content:" + response.getBody()); |
|  | } |
|  | } |

restTemplate.getInterceptors().add(new BasicAuthorizationInterceptor("user", "password"));

String plainCreds = "willie:p@ssword";

byte[] plainCredsBytes = plainCreds.getBytes();

byte[] base64CredsBytes = Base64.encodeBase64(plainCredsBytes);

String base64Creds = new String(base64CredsBytes);

HttpHeaders headers = new HttpHeaders();

headers.add("Authorization", "Basic " + base64Creds);

HttpEntity<String> request = new HttpEntity<String>(headers);

ResponseEntity<Account> response = restTemplate.exchange(url, HttpMethod.GET, request, Account.class);

Account account = response.getBody();

@Bean

RestOperations rest(RestTemplateBuilder restTemplateBuilder) {

return restTemplateBuilder.basicAuthentication("user", "password").build();

}

restTemplate.getInterceptors().add(new BasicAuthorizationInterceptor("user", "password"));

String username = "willie";

String password = ":p@ssword";

HttpHeaders headers = new HttpHeaders();

headers.setBasicAuth(username, password);

URL url = new URL(“location address”);

URLConnection uc = url.openConnection();

String userpass = username + ":" + password;

String basicAuth = "Basic " + new String(Base64.getEncoder().encode(userpass.getBytes()));

uc.setRequestProperty("Authorization",basicAuth);

InputStream in = uc.getInputStream();

public static final String REST\_SERVICE\_URI = "http://localhost:8080/SecureRESTApiWithBasicAuthentication";

/\*

\* Add HTTP Authorization header, using Basic-Authentication to send user-credentials.

\*/

private static HttpHeaders getHeaders() {

String plainCredentials = "bill:abc123";

String base64Credentials = new String(Base64.encodeBase64(plainCredentials.getBytes()));

HttpHeaders headers = new HttpHeaders();

headers.add("Authorization", "Basic " + base64Credentials);

headers.setAccept(Arrays.asList(MediaType.APPLICATION\_JSON));

return headers;

}