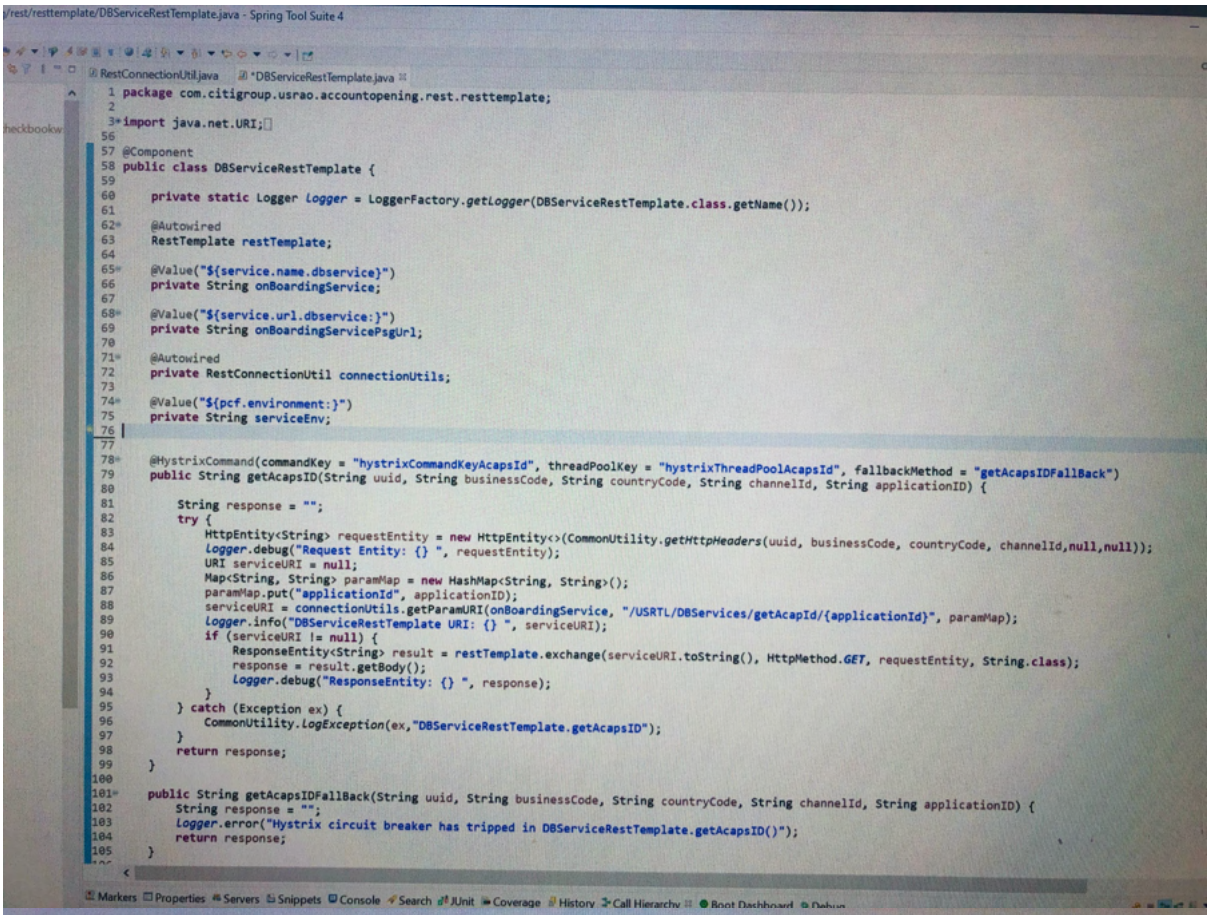
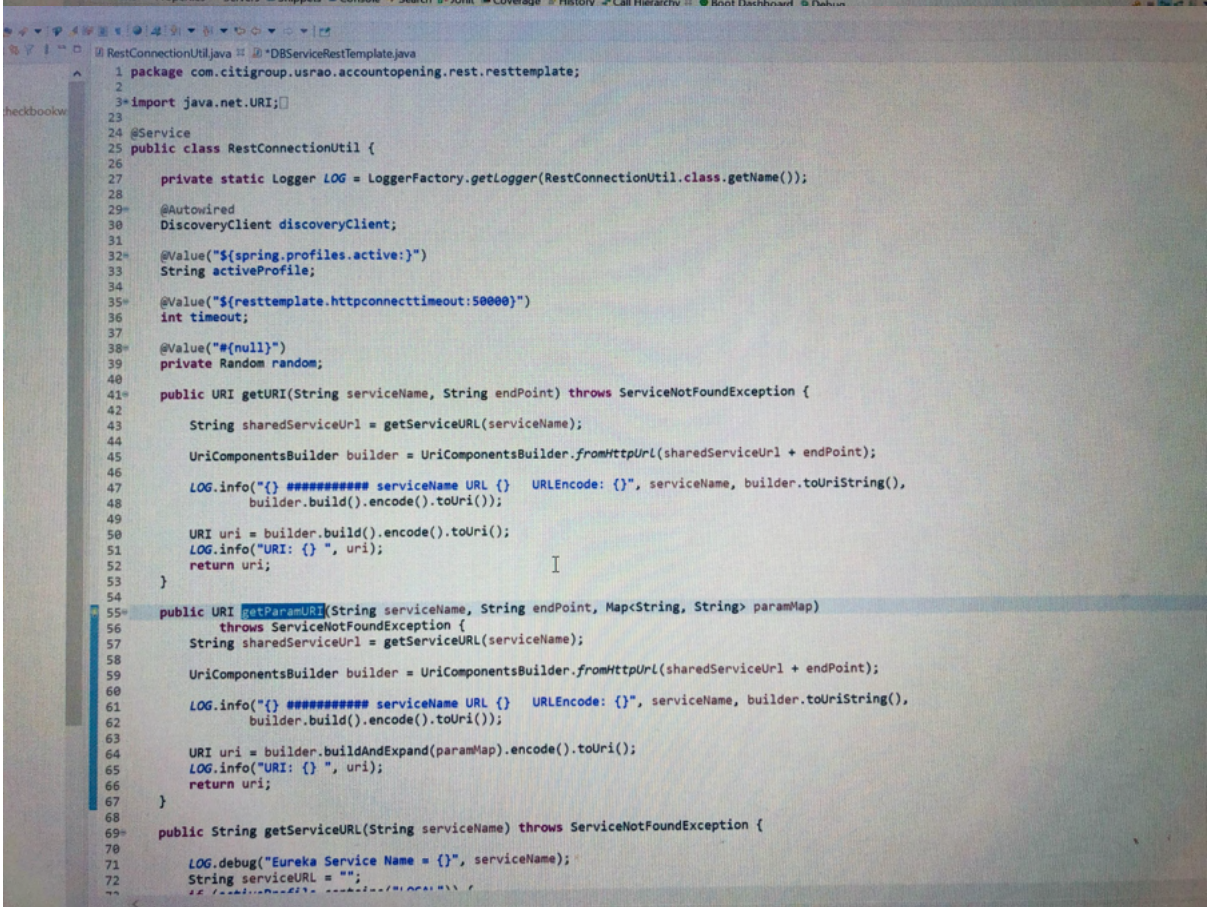
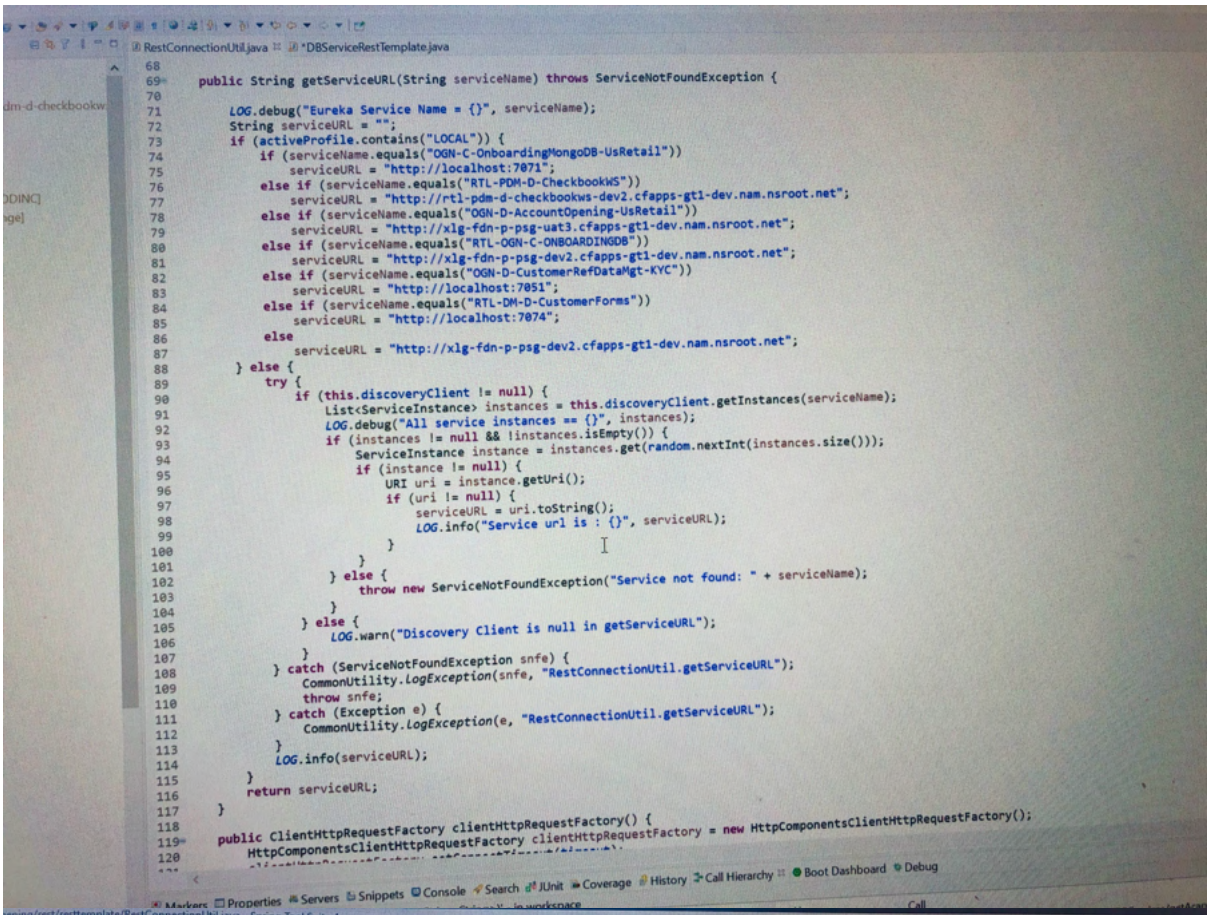

/rest/resttemplate/DBServiceRestTemplate.java - Spring Tool Suite 4
RestConnectionUtil.java
*DBServiceRestTemplate.java
package com.citigroup.usrao.accountopening.rest.resttemplate;
import java.net.URI;
@Component
public class DBServiceRestTemplate {
private static Logger logger = LoggerFactory.getLogger(DBServiceRestTemplate.class.getName());
@Autowired
RestTemplate restTemplate;
@Value("${service.name.dbservice}")
private String onBoardingService;
@Value("${service.url.dbservice:}")
private String onBoardingServicePsgUrl;
@Autowired
private RestConnectionUtil connectionUtils;
@Value("${pcf.environment:}")
private String serviceEnv;
@HystrixCommand(commandKey = "hystrixCommandKeyAcapsId", threadPoolKey = "hystrixThreadPoolAcapsId", fallbackMethod = "getAcapsIDFallBack")
public String getAcapsID(String uuid, String businessCode, String countryCode, String channelId, String applicationID) {
String response = "";
try {
HttpEntity<String> requestEntity = new HttpEntity<>(CommonUtility.getHttpHeaders(uuid, businessCode, countryCode, channelId,null,null));
logger.debug("Request Entity: {} ", requestEntity);
URI serviceURI = null;
Map<String, String> paramMap = new HashMap<String, String>();
paramMap.put("applicationId", applicationID);
serviceURI = connectionUtils.getParamURI(onBoardingService, "/USRTL/DBServices/getAcapId/{applicationId}", paramMap);
logger.info("DBServiceRestTemplate URI: {} ", serviceURI);
if (serviceURI != null) {
ResponseEntity<String> result = restTemplate.exchange(serviceURI.toString(), HttpMethod.GET, requestEntity, String.class);
response = result.getBody();
logger.debug("ResponseEntity: {} ", response);
}
} catch (Exception ex) {
CommonUtility.logException(ex,"DBServiceRestTemplate.getAcapsID");
}
return response;
}
public String getAcapsIDFallBack(String uuid, String businessCode, String countryCode, String channelId, String applicationID) {
String response = "";
logger.error("Hystrix circuit breaker has tripped in DBServiceRestTemplate.getAcapsID()");
return response;
}
Markers Properties Servers Snippets Console Search JUnit Coverage History Call Hierarchy Boot Dashboard Debug


RestConnectionUtil.java
*DBServiceRestTemplate.java
package com.citigroup.usrao.accountopening.rest.resttemplate;
import java.net.URI;
@Service
public class RestConnectionUtil {
private static Logger LOG = LoggerFactory.getLogger(RestConnectionUtil.class.getName());
@Autowired
DiscoveryClient discoveryClient;
@Value("${spring.profiles.active:}")
String activeProfile;
@Value("${resttemplate.httpconnecttimeout:50000}")
int timeout;
@Value("#{null}")
private Random random;
public URI getURI(String serviceName, String endPoint) throws ServiceNotFoundException {
String sharedServiceUrl = getServiceURL(serviceName);
UriComponentsBuilder builder = UriComponentsBuilder.fromHttpUrl(sharedServiceUrl + endPoint);
LOG.info("{} ########## serviceName URL {} URLEncode: {}", serviceName, builder.toUriString(),
builder.build().encode().toUri());
URI uri = builder.build().encode().toUri();
LOG.info("URI: {} ", uri);
return uri;
}
public URI getParamURI(String serviceName, String endPoint, Map<String, String> paramMap)
throws ServiceNotFoundException {
String sharedServiceUrl = getServiceURL(serviceName);
UriComponentsBuilder builder = UriComponentsBuilder.fromHttpUrl(sharedServiceUrl + endPoint);
LOG.info("{} ########## serviceName URL {} URLEncode: {}", serviceName, builder.toUriString(),
builder.build().encode().toUri());
URI uri = builder.buildAndExpand(paramMap).encode().toUri();
LOG.info("URI: {} ", uri);
return uri;
}
public String getServiceURL(String serviceName) throws ServiceNotFoundException {
LOG.debug("Eureka Service Name = {}", serviceName);
String serviceURL = "";



```
RestConnectionUtil.java  *DBServiceRestTemplate.java

68
69      public String getServiceURL(String serviceName) throws ServiceNotFoundException {
70
71          LOG.debug("Eureka Service Name = {}", serviceName);
72          String serviceURL = "";
73          if (activeProfile.contains("LOCAL")) {
74              if (serviceName.equals("OGN-C-OnboardingMongoDB-UsRetail"))
75                  serviceURL = "http://localhost:7071";
76              else if (serviceName.equals("RTL-PDM-D-CheckbookWS"))
77                  serviceURL = "http://rtl-pdm-d-checkbookws-dev2.cfapps-gt1-dev.nam.nsroot.net";
78              else if (serviceName.equals("OGN-D-AccountOpening-UsRetail"))
79                  serviceURL = "http://xlg-fdn-p-psg-uat3.cfapps-gt1-dev.nam.nsroot.net";
80              else if (serviceName.equals("RTL-OGN-C-ONBOARDINGDB"))
81                  serviceURL = "http://xlg-fdn-p-psg-dev2.cfapps-gt1-dev.nam.nsroot.net";
82              else if (serviceName.equals("OGN-D-CustomerRefDataMgt-KYC"))
83                  serviceURL = "http://localhost:7051";
84              else if (serviceName.equals("RTL-DM-D-CustomerForms"))
85                  serviceURL = "http://localhost:7074";
86              else
87                  serviceURL = "http://xlg-fdn-p-psg-dev2.cfapps-gt1-dev.nam.nsroot.net";
88          } else {
89              try {
90                  if (this.discoveryClient != null) {
91                      List<ServiceInstance> instances = this.discoveryClient.getInstances(serviceName);
92                      LOG.debug("All service instances == {}", instances);
93                      if (instances != null && !instances.isEmpty()) {
94                          ServiceInstance instance = instances.get(random.nextInt(instances.size()));
95                          if (instance != null) {
96                              URI uri = instance.getUri();
97                              if (uri != null) {
98                                  serviceURL = uri.toString();
99                                  LOG.info("Service url is : {}", serviceURL);
100                             }
101                         }
102                     } else {
103                         throw new ServiceNotFoundException("Service not found: " + serviceName);
104                     }
105                 } else {
106                     LOG.warn("Discovery Client is null in getServiceURL");
107                 }
108             } catch (ServiceNotFoundException snfe) {
109                 CommonUtility.LogException(snfe, "RestConnectionUtil.getServiceURL");
110                 throw snfe;
111             } catch (Exception e) {
112                 CommonUtility.LogException(e, "RestConnectionUtil.getServiceURL");
113             }
114             LOG.info(serviceURL);
115         }
116         return serviceURL;
117     }
118
119     public ClientHttpRequestFactory clientHttpRequestFactory() {
120         HttpComponentsClientHttpRequestFactory clientHttpRequestFactory = new HttpComponentsClientHttpRequestFactory();
```

Markers Properties Servers Snippets Console Search JUnit Coverage History Call Hierarchy Boot Dashboard Debug

ening/rest/resttemplate/RestConnectionUtil.java - Spring Tool Suite 4

```
RestConnectionUtil.java  *DBServiceRestTemplate.java

78              else if (serviceName.equals("OGN-D-AccountOpening-UsRetail"))
79                  serviceURL = "http://xlg-fdn-p-psg-uat3.cfapps-gt1-dev.nam.nsroot.net";
80              else if (serviceName.equals("RTL-OGN-C-ONBOARDINGDB"))
81                  serviceURL = "http://xlg-fdn-p-psg-dev2.cfapps-gt1-dev.nam.nsroot.net";
82              else if (serviceName.equals("OGN-D-CustomerRefDataMgt-KYC"))
83                  serviceURL = "http://localhost:7051";
84              else if (serviceName.equals("RTL-DM-D-CustomerForms"))
85                  serviceURL = "http://localhost:7074";
86              else
87                  serviceURL = "http://xlg-fdn-p-psg-dev2.cfapps-gt1-dev.nam.nsroot.net";
88          } else {
89              try {
90                  if (this.discoveryClient != null) {
91                      List<ServiceInstance> instances = this.discoveryClient.getInstances(serviceName);
92                      LOG.debug("All service instances == {}", instances);
93                      if (instances != null && !instances.isEmpty()) {
94                          ServiceInstance instance = instances.get(random.nextInt(instances.size()));
95                          if (instance != null) {
96                              URI uri = instance.getUri();
97                              if (uri != null) {
98                                  serviceURL = uri.toString();
99                                  LOG.info("Service url is : {}", serviceURL);
100                             }
101                         }
102                     } else {
103                         throw new ServiceNotFoundException("Service not found: " + serviceName);
104                     }
105                 } else {
106                     LOG.warn("Discovery Client is null in getServiceURL");
107                 }
108             } catch (ServiceNotFoundException snfe) {
109                 CommonUtility.LogException(snfe, "RestConnectionUtil.getServiceURL");
110                 throw snfe;
111             } catch (Exception e) {
112                 CommonUtility.LogException(e, "RestConnectionUtil.getServiceURL");
113             }
114             LOG.info(serviceURL);
115         }
116         return serviceURL;
117     }
118
119     public ClientHttpRequestFactory clientHttpRequestFactory() {
120         HttpComponentsClientHttpRequestFactory clientHttpRequestFactory = new HttpComponentsClientHttpRequestFactory();
121         clientHttpRequestFactory.setConnectTimeout(timeout);
122         return clientHttpRequestFactory;
123     }
124
125     @PostConstruct
126     public void initialiseRestConnectionUtil() {
127         random = new Random();
128     }
129 }
130
```

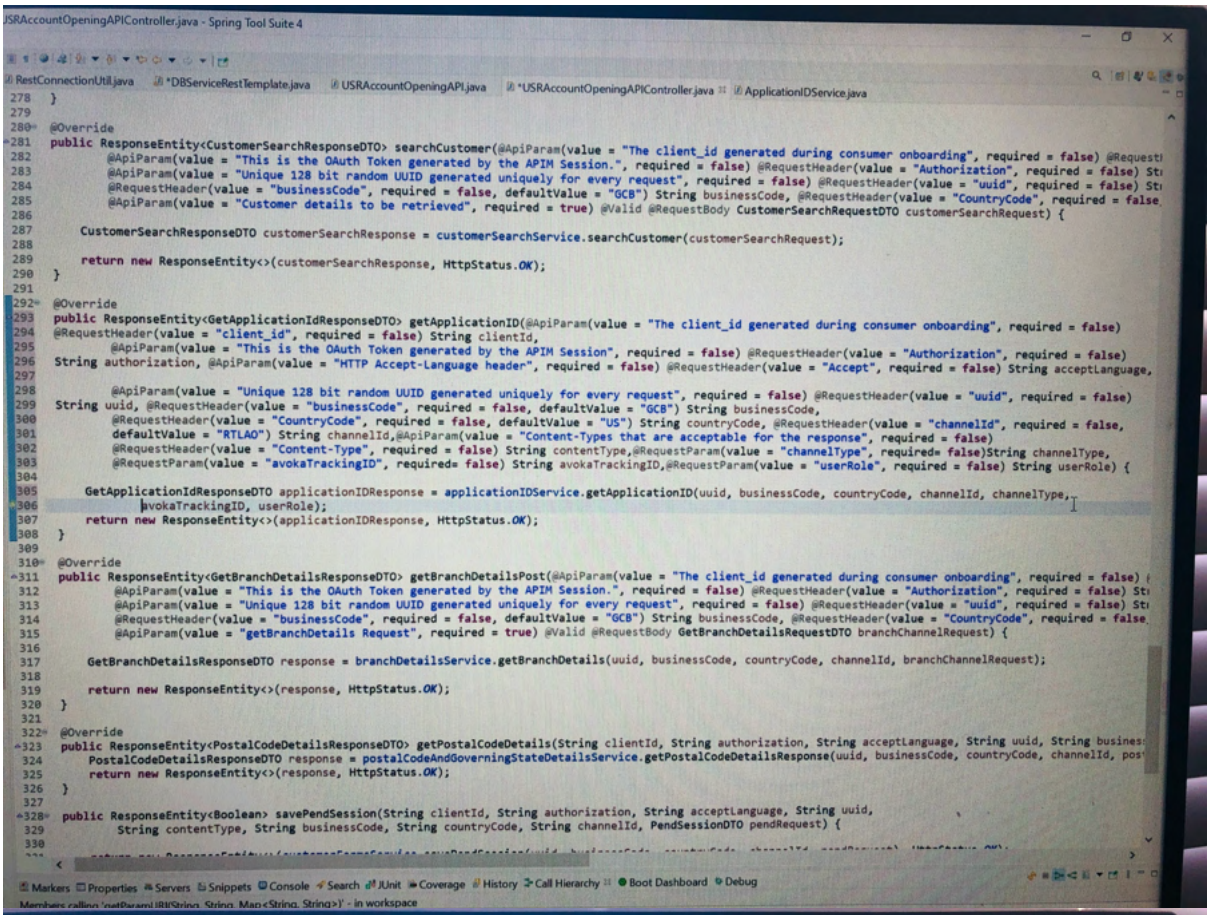
Markers Properties Servers Snippets Console Search JUnit Coverage History Call Hierarchy Boot Dashboard Debug

Members calling 'getParamURI(String, String, Map<String, String>)' - in workspace

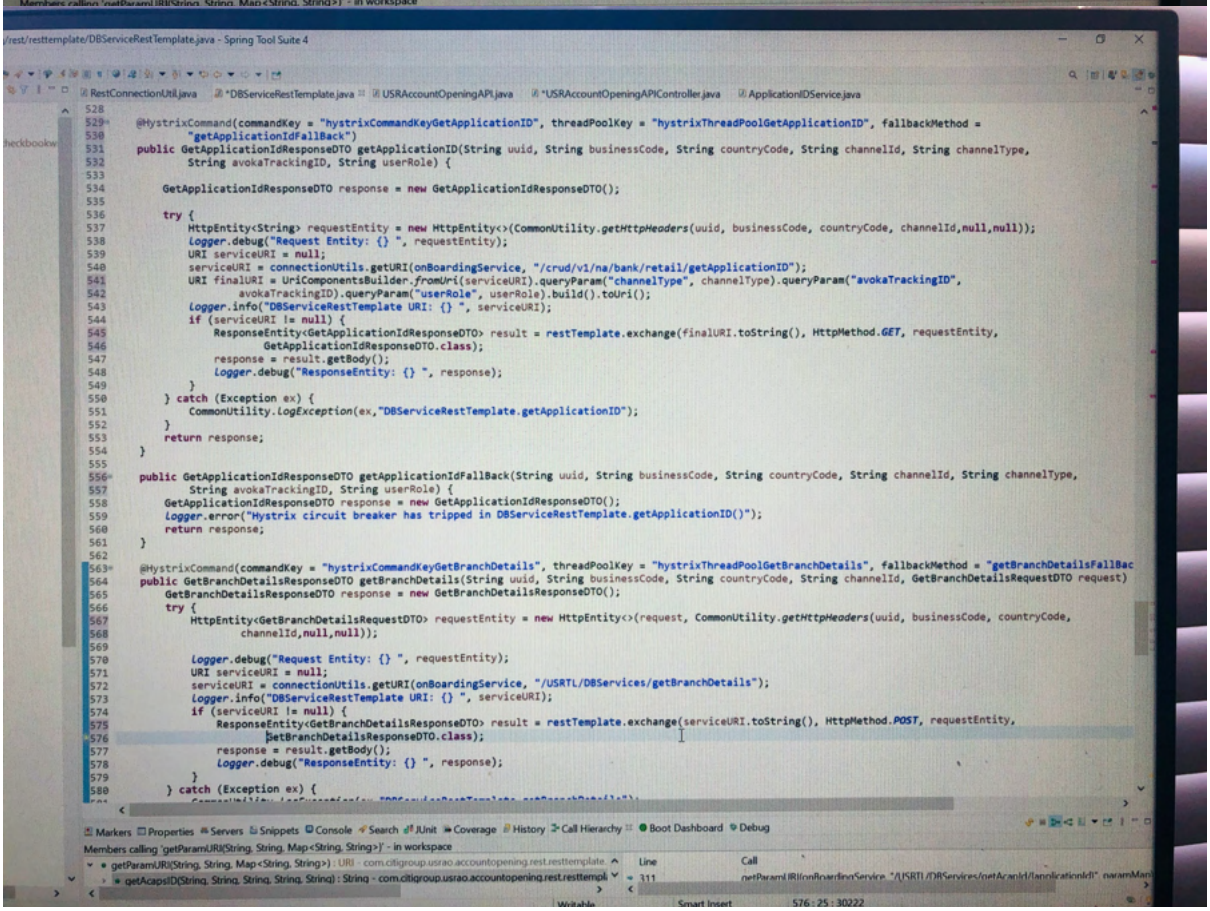
getParamURI(String, String, Map<String, String>) : URI - com.citigroup.usrao.accountopening.rest.resttemplate.

Line 311

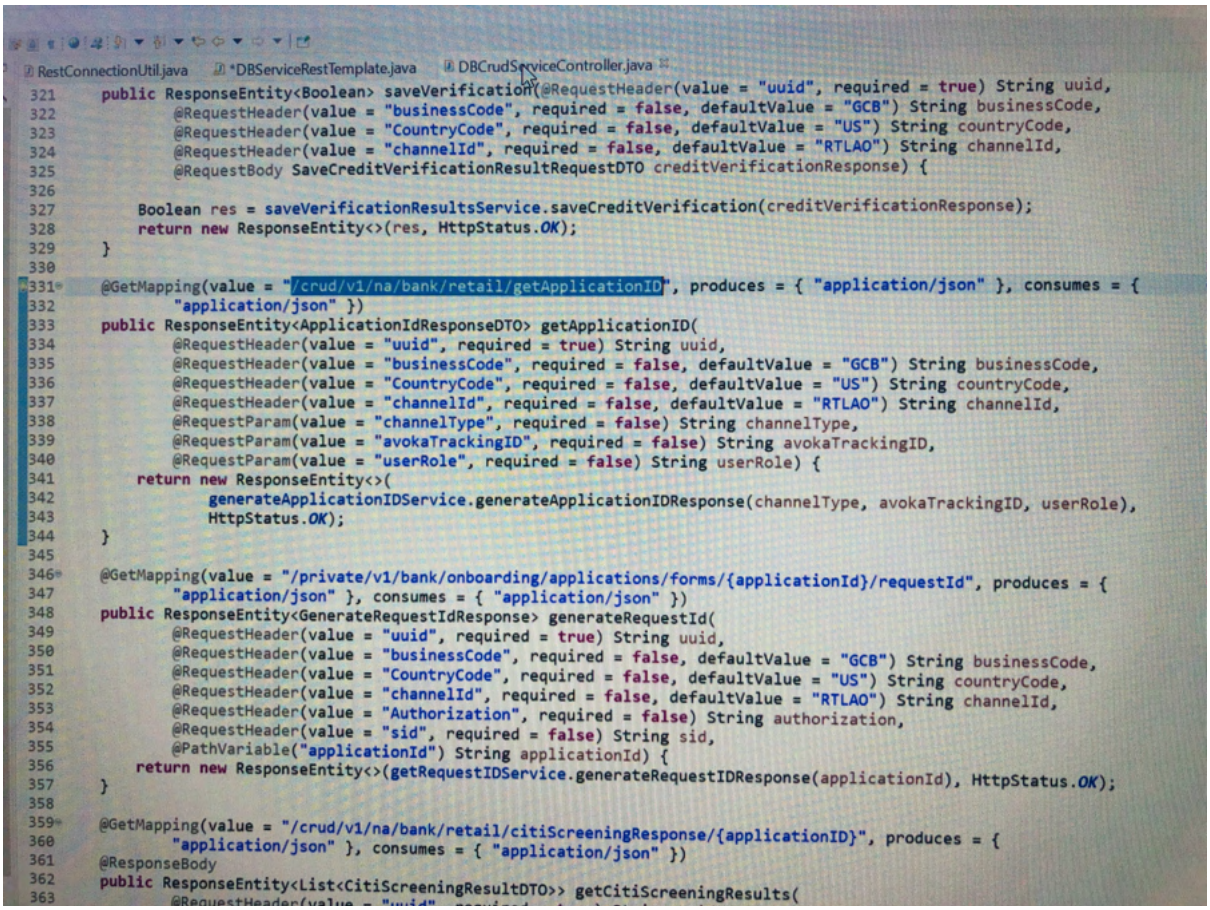
Call



```
}

@Override
public ResponseEntity<CustomerSearchResponseDTO> searchCustomer(@ApiParam(value = "The client_id generated during consumer onboarding", required = false) @Requesti
        @ApiParam(value = "This is the OAuth Token generated by the APIM Session.", required = false) @RequestHeader(value = "Authorization", required = false) Str
        @ApiParam(value = "Unique 128 bit random UUID generated uniquely for every request", required = false) @RequestHeader(value = "uuid", required = false) Str
        @RequestHeader(value = "businessCode", required = false, defaultValue = "GCB") String businessCode, @RequestHeader(value = "CountryCode", required = false,
        @ApiParam(value = "Customer details to be retrieved", required = true) @Valid @RequestBody CustomerSearchRequestDTO customerSearchRequest) {

    CustomerSearchResponseDTO customerSearchResponse = customerSearchService.searchCustomer(customerSearchRequest);

    return new ResponseEntity<>(customerSearchResponse, HttpStatus.OK);
}

@Override
public ResponseEntity<GetApplicationIdResponseDTO> getApplicationID(@ApiParam(value = "The client_id generated during consumer onboarding", required = false)
@RequestHeader(value = "client_id", required = false) String clientId,
        @ApiParam(value = "This is the OAuth Token generated by the APIM Session", required = false) @RequestHeader(value = "Authorization", required = false)
String authorization, @ApiParam(value = "HTTP Accept-Language header", required = false) @RequestHeader(value = "Accept", required = false) String acceptLanguage,

        @ApiParam(value = "Unique 128 bit random UUID generated uniquely for every request", required = false) @RequestHeader(value = "uuid", required = false)
String uuid, @RequestHeader(value = "businessCode", required = false, defaultValue = "GCB") String businessCode,
        @RequestHeader(value = "CountryCode", required = false, defaultValue = "US") String countryCode, @RequestHeader(value = "channelId", required = false,
        defaultValue = "RTLAO") String channelId,@ApiParam(value = "Content-Types that are acceptable for the response", required = false)
        @RequestHeader(value = "Content-Type", required = false) String contentType,@RequestParam(value = "channelType", required= false)String channelType,
        @RequestParam(value = "avokaTrackingID", required= false) String avokaTrackingID,@RequestParam(value = "userRole", required = false) String userRole) {

    GetApplicationIdResponseDTO applicationIDResponse = applicationIDService.getApplicationID(uuid, businessCode, countryCode, channelId, channelType,
            avokaTrackingID, userRole);
    return new ResponseEntity<>(applicationIDResponse, HttpStatus.OK);
}

@Override
public ResponseEntity<GetBranchDetailsResponseDTO> getBranchDetailsPost(@ApiParam(value = "The client_id generated during consumer onboarding", required = false) (
        @ApiParam(value = "This is the OAuth Token generated by the APIM Session.", required = false) @RequestHeader(value = "Authorization", required = false) Str
        @ApiParam(value = "Unique 128 bit random UUID generated uniquely for every request", required = false) @RequestHeader(value = "uuid", required = false) Str
        @RequestHeader(value = "businessCode", required = false, defaultValue = "GCB") String businessCode, @RequestHeader(value = "CountryCode", required = false,
        @ApiParam(value = "getBranchDetails Request", required = true) @Valid @RequestBody GetBranchDetailsRequestDTO branchChannelRequest) {

    GetBranchDetailsResponseDTO response = branchDetailsService.getBranchDetails(uuid, businessCode, countryCode, channelId, branchChannelRequest);

    return new ResponseEntity<>(response, HttpStatus.OK);
}

@Override
public ResponseEntity<PostalCodeDetailsResponseDTO> getPostalCodeDetails(String clientId, String authorization, String acceptLanguage, String uuid, String busines
    PostalCodeDetailsResponseDTO response = postalCodeAndGoverningStateDetailsService.getPostalCodeDetailsResponse(uuid, businessCode, countryCode, channelId, pos
    return new ResponseEntity<>(response, HttpStatus.OK);
}

public ResponseEntity<Boolean> savePendSession(String clientId, String authorization, String acceptLanguage, String uuid,
        String contentType, String businessCode, String countryCode, String channelId, PendSessionDTO pendRequest) {
```



```
@HystrixCommand(commandKey = "hystrixCommandKeyGetApplicationID", threadPoolKey = "hystrixThreadPoolGetApplicationID", fallbackMethod =
        "getApplicationIdFallBack")
public GetApplicationIdResponseDTO getApplicationID(String uuid, String businessCode, String countryCode, String channelId, String channelType,
        String avokaTrackingID, String userRole) {

    GetApplicationIdResponseDTO response = new GetApplicationIdResponseDTO();

    try {
        HttpEntity<String> requestEntity = new HttpEntity<>(CommonUtility.getHttpHeaders(uuid, businessCode, countryCode, channelId,null,null));
        Logger.debug("Request Entity: {} ", requestEntity);
        URI serviceURI = null;
        serviceURI = connectionUtils.getURI(onBoardingService, "/crud/v1/na/bank/retail/getApplicationID");
        URI finalURI = UriComponentsBuilder.fromUri(serviceURI).queryParam("channelType", channelType).queryParam("avokaTrackingID",
                avokaTrackingID).queryParam("userRole", userRole).build().toUri();
        Logger.info("DBServiceRestTemplate URI: {} ", serviceURI);
        if (serviceURI != null) {
            ResponseEntity<GetApplicationIdResponseDTO> result = restTemplate.exchange(finalURI.toString(), HttpMethod.GET, requestEntity,
                    GetApplicationIdResponseDTO.class);
            response = result.getBody();
            Logger.debug("ResponseEntity: {} ", response);
        }
    } catch (Exception ex) {
        CommonUtility.LogException(ex,"DBServiceRestTemplate.getApplicationID");
    }
    return response;
}

public GetApplicationIdResponseDTO getApplicationIdFallBack(String uuid, String businessCode, String countryCode, String channelId, String channelType,
        String avokaTrackingID, String userRole) {
    GetApplicationIdResponseDTO response = new GetApplicationIdResponseDTO();
    Logger.error("Hystrix circuit breaker has tripped in DBServiceRestTemplate.getApplicationID()");
    return response;
}

@HystrixCommand(commandKey = "hystrixCommandKeyGetBranchDetails", threadPoolKey = "hystrixThreadPoolGetBranchDetails", fallbackMethod = "getBranchDetailsFallBac
public GetBranchDetailsResponseDTO getBranchDetails(String uuid, String businessCode, String countryCode, String channelId, GetBranchDetailsRequestDTO request)
    GetBranchDetailsResponseDTO response = new GetBranchDetailsResponseDTO();
    try {
        HttpEntity<GetBranchDetailsRequestDTO> requestEntity = new HttpEntity<>(request, CommonUtility.getHttpHeaders(uuid, businessCode, countryCode,
                channelId,null,null));

        Logger.debug("Request Entity: {} ", requestEntity);
        URI serviceURI = null;
        serviceURI = connectionUtils.getURI(onBoardingService, "/USRTL/DBServices/getBranchDetails");
        Logger.info("DBServiceRestTemplate URI: {} ", serviceURI);
        if (serviceURI != null) {
            ResponseEntity<GetBranchDetailsResponseDTO> result = restTemplate.exchange(serviceURI.toString(), HttpMethod.POST, requestEntity,
                    GetBranchDetailsResponseDTO.class);
            response = result.getBody();
            Logger.debug("ResponseEntity: {} ", response);
        }
    } catch (Exception ex) {
```



RestConnectionUtil.java | *DBServiceRestTemplate.java | DBCrudServiceController.java

```
321     public ResponseEntity<Boolean> saveVerification(@RequestHeader(value = "uuid", required = true) String uuid,
322             @RequestHeader(value = "businessCode", required = false, defaultValue = "GCB") String businessCode,
323             @RequestHeader(value = "CountryCode", required = false, defaultValue = "US") String countryCode,
324             @RequestHeader(value = "channelId", required = false, defaultValue = "RTLAO") String channelId,
325             @RequestBody SaveCreditVerificationResultRequestDTO creditVerificationResponse) {
326
327         Boolean res = saveVerificationResultsService.saveCreditVerification(creditVerificationResponse);
328         return new ResponseEntity<>(res, HttpStatus.OK);
329     }
330
331     @GetMapping(value = "/crud/v1/na/bank/retail/getApplicationID", produces = { "application/json" }, consumes = {
332             "application/json" })
333     public ResponseEntity<ApplicationIdResponseDTO> getApplicationID(
334             @RequestHeader(value = "uuid", required = true) String uuid,
335             @RequestHeader(value = "businessCode", required = false, defaultValue = "GCB") String businessCode,
336             @RequestHeader(value = "CountryCode", required = false, defaultValue = "US") String countryCode,
337             @RequestHeader(value = "channelId", required = false, defaultValue = "RTLAO") String channelId,
338             @RequestParam(value = "channelType", required = false) String channelType,
339             @RequestParam(value = "avokaTrackingID", required = false) String avokaTrackingID,
340             @RequestParam(value = "userRole", required = false) String userRole) {
341         return new ResponseEntity<>(
342                 generateApplicationIDService.generateApplicationIDResponse(channelType, avokaTrackingID, userRole),
343                 HttpStatus.OK);
344     }
345
346     @GetMapping(value = "/private/v1/bank/onboarding/applications/forms/{applicationId}/requestId", produces = {
347             "application/json" }, consumes = { "application/json" })
348     public ResponseEntity<GenerateRequestIdResponse> generateRequestId(
349             @RequestHeader(value = "uuid", required = true) String uuid,
350             @RequestHeader(value = "businessCode", required = false, defaultValue = "GCB") String businessCode,
351             @RequestHeader(value = "CountryCode", required = false, defaultValue = "US") String countryCode,
352             @RequestHeader(value = "channelId", required = false, defaultValue = "RTLAO") String channelId,
353             @RequestHeader(value = "Authorization", required = false) String authorization,
354             @RequestHeader(value = "sid", required = false) String sid,
355             @PathVariable("applicationId") String applicationId) {
356         return new ResponseEntity<>(getRequestIDService.generateRequestIDResponse(applicationId), HttpStatus.OK);
357     }
358
359     @GetMapping(value = "/crud/v1/na/bank/retail/citiScreeningResponse/{applicationID}", produces = {
360             "application/json" }, consumes = { "application/json" })
361     @ResponseBody
362     public ResponseEntity<List<CitiScreeningResultDTO>> getCitiScreeningResults(
363             @RequestHeader(value = "uuid", required = true) String uuid,
```