BTP Cloud SDK를 활용한ERP 확장 개발

Java Application

Exercises / Solutions

Speakers:   
한정우 (SAP SE)

Table of ContentS

[1. 실습 개요 3](#_Toc121855693)

[2. Prerequisite 3](#_Toc121855694)

[2.1 CAP Java + Cloud SDK Hands-on 3](#_Toc121855695)

[2.2 Cloud SDK & CAP CDS를 활용한 프로그래밍 9](#_Toc121855696)

[2.3 RFC를 위한 Cloud SDK 활용 13](#_Toc121855697)

[2.4 RFC를 위한 Cloud SDK 활용(프로젝트 생성부터) 15](#_Toc121855698)

# 실습 개요

# Prerequisite

https://sap.github.io/cloud-sdk/docs/java/overview-cloud-sdk-for-java

## CAP Java + Cloud SDK Hands-on

| Explanation | Screenshot |
| --- | --- |
| 1. 프로젝트(Application) 생성 | CAP 템플릿을 이용해 CAP Java프로젝트 생성 |
| 1. Pom.xml 파일 수정   srv > pom.xml | 아래 파일 참조  [https://github.com/micol92/btpenable\_capjava/blob/main/srv/pom.xml](https://github.com/micol92/backend_app_1029/blob/main/srv/pom.xml)  <dependency>  <groupId>com.sap.cloud.sdk.cloudplatform</groupId>  <artifactId>scp-cf</artifactId>  </dependency>  <dependency>  <groupId>com.sap.cloud.sdk.s4hana</groupId>  <artifactId>s4hana-all</artifactId>  </dependency>  <dependency>  <groupId>com.sap.cds</groupId>  <artifactId>cds-integration-cloud-sdk</artifactId>  </dependency>  <dependency>  <groupId>com.sap.cloud.sdk.s4hana</groupId>  <artifactId>s4hana-api-odata-onpremise</artifactId>  <version>4.0.0</version>  </dependency>  <dependency>  <groupId>com.sap.cloud.sdk.datamodel</groupId>  <artifactId>odata-core</artifactId>  </dependency>  <dependency>  <groupId>org.projectlombok</groupId>  <artifactId>lombok</artifactId>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>javax.inject</groupId>  <artifactId>javax.inject</artifactId>  <scope>provided</scope>  </dependency> |
| 1. 프로젝트 루트 밑에 pom.xml파일 수정   (parent pom.xml) | 아래 파일 수정  <https://github.com/micol92/backend_app_1029/blob/main/pom.xml>  <dependency>  <groupId>com.sap.cloud.sdk</groupId>  <artifactId>sdk-bom</artifactId>  <version>3.72.0</version>  <type>pom</type>  <scope>import</scope>  </dependency> |
| 1. CF Login |  |
| 1. 빌드 | $>mvn clean install |
| 1. 배포를 위한 mta.yaml파일 수정 | Hdi-container 서비스 생성  아래 파일 참조  [https://github.com/micol92/btpenable\_capjava/blob/main/mta.yaml](https://github.com/micol92/backend_app_1029/blob/main/mta.yaml)  # ------------------------------------------------------------  - name: demo02-db  # ------------------------------------------------------------  type: com.sap.xs.hdi-container  parameters:  service: hana # or 'hanatrial' on trial landscapes  service-plan: hdi-shared  config:  database\_id : 2ec0a886-207d-4c67-bde0-bbec20c27977  properties:  hdi-service-name: ${service-name} |
| 1. Domain model 생성 | namespace my.bookshop;  entity Books {  key ID : Integer;  title : String;  stock : Integer;  }  entity PurchaseOrders {  key poid : String;  potype : String;  pogroup : String;  posupplier : String;  } |
| 1. Service model 생성 | using my.bookshop as my from '../db/data-model';  service CatalogService @(requires: 'any') {  entity Books as projection on my.Books ;  entity POrders as projection on my.PurchaseOrders ;    type BookTypes : {  ID : Integer;  title : String;  stock : Integer;  }  action SaveBook ();  action SaveMultipleBooks (Books : array of BookTypes) returns array of BookTypes;  action SaveSingleBook (ID : Integer, title : String, stock : Integer) returns BookTypes;  } |
| 1. Mta.yaml파일 구성 |  |
| 1. 빌드 | Mta.yaml파일 우클릭 > “Build MTA Project” 클릭 |
| 1. CF Login |  |
| 1. 배포 | Mta\_archives > *file\_name.mtar* 우클릭 > “Deploy MTA Archive” 클릭 |
| 1. 배포하지 않고 로컬테스트 수행하기 | <https://bnheise.medium.com/sap-tutorial-serving-data-from-an-on-premise-system-in-a-cap-java-application-part-6-5b4a79927774>   1. 프로젝트 홈에 default-env.json 파일생성 2. BTP Cockpit을 통해 본인이 배포한 인스턴스 확인.     VCAP\_SERVICES부분 복사   1. 복사한 내용을 default-env.json 파일에 붙여넣기 2. Pom.xml 파일 수정 🡺 위에서 수행. 3. cf enable-ssh <my-app-name>   처음인 경우 new terminal 수행 🡪 default-env.json 환경정보가 수행될 수 있게끔.   1. cf restart <my-app-name> 2. SSH tunnel셋업(default-env.json파일에 있는 host, port참조)   Cf ssh <app-name> -L <onpremise\_proxy\_port>:<onpremise\_proxy\_host>:<onpremise\_proxy\_port>  <<예제>>  cf ssh demo02-srv -L 20003:connectivityproxy.internal.cf.ap12.hana.ondemand.com:20003  위 내용을 cflogin.sh파일에 저장   1. 아래와 같이 수행.      1. Default-env.json파일 수정.   “onpremise\_proxy\_host” 주소를 “localhost”로 수정   1. Mvn clean install 수행 2. Mvn spring-boot:run 수행 3. 이제 부터는 로컬 수행이 가능해 집니다. |
| 1. Event Handler Class파일생성 |  |
| 1. CDS QL(Query Language) 적용  * Action 정의 | action SaveBook (); |
| 1. EventHandler 로직 구현 | @On(event = SaveBookContext.CDS\_NAME)  public void onSaveBook(SaveBookContext context) {  Map<String, Object> book = new HashMap<>();  book.put("ID", 101);  book.put("title", "Capire 2");  book.put("stock", 100);  CqnInsert insert = Insert.into("CatalogService.Books").entry(book);  db.run(insert);  context.setCompleted();  } |
| 1. 어플리케이션 로컬 구동 | $>mvn spring-boot:run  CF 로그인 필요. |
| 1. 로컬 테스트(Requests.http 활용) | ### Post Books via Action  POST http://localhost:8080/odata/v4/CatalogService/SaveBook  Content-Type: application/json  {  } |
| 1. 1건 입력 : CDS QL(Query Language) 적용   Action 정의 | type BookTypes : {  ID : Integer;  title : String;  stock : Integer;  }  action SaveSingleBook (ID : Integer, title : String, stock : Integer) returns BookTypes; |
| 1. EventHandler로직 구현 | @On(event = SaveSingleBookContext.CDS\_NAME)  public void onSaveBookTypesEntityProc(SaveSingleBookContext context) {  BookTypes booktypes = BookTypes.create();  Books books = Books.create();  books.setId(context.getId());  books.setTitle(context.getTitle());  books.setStock(context.getStock());  booktypes.setId(context.getId());  booktypes.setTitle(context.getTitle());  booktypes.setStock(context.getStock());    Result res = db.run(Insert.into(BOOKS).entry(books));  cds.gen.catalogservice.Books inserted = res.single(cds.gen.catalogservice.Books.class);  context.setResult(booktypes);  } |
| 1. 어플리케이션 로컬 구동 | $>mvn spring-boot:run  CF 로그인 필요. |
| 1. Requests.http파일 | ### Post Books via Action  POST http://localhost:8080/odata/v4/CatalogService/SaveSingleBook  Content-Type: application/json  {  "ID" : 1004,  "title" : "test0001",  "stock" : 12  } |
| 1. 여러건 입력 : CDS QL(Query Language) 적용   Action 정의 | action SaveMultipleBooks (Books : array of BookTypes) returns array of BookTypes; |
| 1. EventHandler 구현 | @On(event = SaveMultipleBooksContext.CDS\_NAME)  public void onSaveBookTypesEntityProc(SaveMultipleBooksContext context) {  Collection<BookTypes> bookTypes = context.getBooks();  for (BookTypes bookType : bookTypes) {  //bookType.getId();  //bookType.getTitle();  //bookType.getStock();  Result res = db.run(Insert.into(BOOKS).entry(bookType));  cds.gen.catalogservice.Books inserted = res.single(cds.gen.catalogservice.Books.class);    }  context.setResult(bookTypes);  } |
| 1. 어플리케이션 로컬 구동 | $>mvn spring-boot:run  CF 로그인 필요. |
| 1. Requests.http 구현 | ### Post Books via Action  POST http://localhost:8080/odata/v4/CatalogService/SaveMultipleBooks  Content-Type: application/json  {  "Books" :[ {  "ID" : 1003,  "title" : "test0001",  "stock" : 12  },  {  "ID" : 1005,  "title" : "test0002",  "stock" : 16  }]  } |
| 1. 어플리케이션 로컬 구동 | $>mvn spring-boot:run  CF 로그인 필요. |
|  |  |

## Cloud SDK & CAP CDS를 활용한 프로그래밍

….

| Explanation | Screenshot |
| --- | --- |
| 1. 구매Data Model | entity PurchaseOrders {  key poid : String;  potype : String;  pogroup : String;  posupplier : String;  qty : Integer;  } |
| 1. 구매 Service Model | entity POrders as projection on my.PurchaseOrders ; |
| 1. Build | Mvn clean install |
| 1. Event Handler로직 추가   : ERP Processing후, 결과를 CDS Service에 담아서 전송.  : API.sap.com 에서 API 확인. 프로그래밍을 위한 Cloud SDK확인.  CAP event : EVENT\_READ  CAP entity : POrders  예제: 구매오더 조회. | “POrders”라는 CDS조회시, ERP구매오더를 읽어서 POrders 서비스에 담아서 결과로 회신.  @On(event = CqnService.EVENT\_READ, entity = POrders\_.CDS\_NAME)  public void onRead(CdsReadEventContext context) throws ODataException{  //CdsModel model = context.getModel();  Map<String, Object> v\_row = new HashMap<>();  Map<Object, Map<String, Object>> v\_result = new HashMap<>();  HttpDestination httpDestination = DestinationAccessor.getDestination("S4H\_30").asHttp();  purchaseOrderService = new DefaultPurchaseOrderService();  PurchaseOrderFluentHelper helper = purchaseOrderService.getAllPurchaseOrder().top(5);  List<PurchaseOrder> purchaseOrders = helper.executeRequest(httpDestination);  //logger.info("purchaseOrderItemTexts size:" + purchaseOrders.size());    //StringBuffer stringBuffer = new StringBuffer();  for (PurchaseOrder item: purchaseOrders) {  //logger.info(item.toString());  POrders capPOrders = com.sap.cds.Struct.create(POrders.class);  capPOrders.setPoid(item.getPurchaseOrder());  capPOrders.setPotype(item.getPurchaseOrderType());  capPOrders.setPogroup(item.getPurchasingGroup());  capPOrders.setPosupplier(item.getSupplier());    capPOrderss.add(capPOrders);  v\_row.put("poid",item.getPurchaseOrder());  v\_row.put("potype",item.getPurchaseOrderType());  v\_row.put("pogroup",item.getPurchasingGroup());  v\_row.put("posupplier",item.getSupplier());  v\_result.put(item.getPurchaseOrder(), v\_row);  }  //context.setResult(v\_result.values());  context.setResult(capPOrderss);  } |
| 1. Event Handler로직 추가   : ERP Processing후, 결과를 CDS Service에 저장.  구매오더 생성  CAP event : EVENT\_CREATE  CAP entity : POrders  예제: 구매오더 생성. | @On(event = CdsService.EVENT\_CREATE, entity = POrders\_.CDS\_NAME)  public void onCreate(CdsCreateEventContext context) throws ODataException {  HttpDestination httpDestination = DestinationAccessor.getDestination("S4H\_30").asHttp();  PurchaseOrderService service = new DefaultPurchaseOrderService();  //LocalDateTime ldtime = new LocalDateTime().now();  LocalDateTime currentDateTime = LocalDateTime.now();  //BigDecimal qty = new BigDecimal(1);  // qty = 1;  Map<String, Object> m = context.getCqn().entries().get(0);  BigDecimal qty2 = new BigDecimal(m.get("qty").toString());  PurchaseOrderItem poitem = PurchaseOrderItem.builder().  purchaseOrderItem("10".toString()).  material("TG11".toString()).  plant("1710".toString()).  incotermsLocation1("VENDER".toString()).  orderQuantity(qty2).build();    PurchaseOrder po = PurchaseOrder.builder().purchasingGroup("001".toString()).  companyCode("1710".toString()).  //purchaseOrderType("NB".toString()).  purchaseOrderType(m.get("potype").toString()).  supplier("17300001".toString()).  purchasingOrganization("1710".toString()).  purchaseOrderItem(poitem).  build();  //BusinessPartner bp = BusinessPartner.builder().firstName(m.get("firstName").toString()).lastName(m.get("surname").toString()).businessPartner(m.get("ID").toString()).build();  try {  service.createPurchaseOrder(po).executeRequest(httpDestination);  }  catch( ODataServiceErrorException e) {  ODataServiceError odataError = e.getOdataError();  logger.debug("The OData service responded with an error: {}", odataError);  } catch( ODataDeserializationException e ) {  // handle failures in deserialization  System.out.println("The OData Deserialization responded with an error: {}"+ e.getMessage());  } catch( ODataResponseException e ) {  // handle all other errors originating from handling the HTTP response  int httpCode = e.getHttpCode();  logger.debug("The OData Response1 responded with an error: {}", e.getHttpHeaders());  logger.debug("The OData Response2 responded with an error: {}", e.getHttpBody());  }  capPOrderss.clear();  POrders capPOrders = com.sap.cds.Struct.create(POrders.class);  capPOrders.setPoid("NEW PO");  capPOrders.setPotype("NB");  capPOrders.setPogroup("001");  capPOrders.setPosupplier("17300001");  capPOrderss.add(capPOrders);  context.setResult(capPOrderss);  } |
| 1. 어플리케이션 로컬 구동 | $>mvn spring-boot:run  CF 로그인 필요. |
| 1. Requests.http 구현 | ### Get Purchase Orders  ### Get POrders  GET http://localhost:8080/odata/v4/CatalogService/POrders  ### Post POrders via Action  POST http://localhost:8080/odata/v4/CatalogService/POrders  Content-Type: application/json  {  "potype" : "NB",  "posupplier" : "17300001",  "qty": 2  } |
| 1. BusinessPartner 조회 & 생성 | <https://github.com/micol92/btpenable_capjava> |
|  |  |

## RFC를 위한 Cloud SDK 활용

자료는 blog와 cloud sdk library & help guide를 참조로 정리되었습니다.

<https://blogs.sap.com/2020/07/30/sap-cloud-foundry-to-on-premise-rfc-connection-using-jco/>

<https://sap.github.io/cloud-sdk/docs/java/features/bapi-and-rfc/bapi-and-rfc-overview>

| Explanation | Screenshot |
| --- | --- |
| 1. 프로젝트 생성. | mvn archetype:generate "-DarchetypeGroupId=com.sap.cloud.sdk.archetypes" "-DarchetypeArtifactId=scp-cf-tomee" "-DarchetypeVersion=RELEASE" "-DgroupId=com.sap.cloud.sdk.tutorial" "-DartifactId=myrfc01" "-Dpackage=com.sap.cloud.sdk.tutorial" |
| 1. RFC관련 서블릿 추가 | package com.test.jco;  import java.io.IOException;  import java.io.PrintWriter;  import javax.servlet.ServletException;  import javax.servlet.annotation.WebServlet;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;    import com.sap.conn.jco.JCoDestination;  import com.sap.conn.jco.JCoDestinationManager;  import com.sap.conn.jco.JCoFunction;  import com.sap.conn.jco.JCoParameterList;  import com.sap.conn.jco.JCoRepository;    @WebServlet("/ConnectRFC/\*")  public class ConnectRFC extends HttpServlet {  private static final long serialVersionUID = 1L;  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  PrintWriter responseWriter = response.getWriter();  try {  // access the RFC Destination "Test"  JCoDestination destination = JCoDestinationManager.getDestination("Test");  // make an invocation of STFC\_CONNECTION in the backend  JCoRepository repo = destination.getRepository();  JCoFunction stfcConnection = repo.getFunction("STFC\_CONNECTION");  JCoParameterList imports = stfcConnection.getImportParameterList();  imports.setValue("REQUTEXT", "JCO successful");  stfcConnection.execute(destination);  JCoParameterList exports = stfcConnection.getExportParameterList();  String echotext = exports.getString("ECHOTEXT");  String resptext = exports.getString("RESPTEXT");  responseWriter.println(echotext+":"+resptext);  } catch (Exception e) {  e.printStackTrace();  }  }  } |

## RFC를 위한 Cloud SDK 활용(프로젝트 생성부터)

자료는 blog와 cloud sdk library & help guide를 참조로 정리되었습니다.

<https://blogs.sap.com/2020/07/30/sap-cloud-foundry-to-on-premise-rfc-connection-using-jco/>

<https://sap.github.io/cloud-sdk/docs/java/features/bapi-and-rfc/bapi-and-rfc-overview>

| Explanation | Screenshot |
| --- | --- |
| 1. 프로젝트 생성. | mvn archetype:generate "-DarchetypeGroupId=com.sap.cloud.sdk.archetypes" "-DarchetypeArtifactId=scp-cf-tomee" "-DarchetypeVersion=RELEASE" "-DgroupId=com.sap.cloud.sdk.tutorial" "-DartifactId=myrfc01" "-Dpackage=com.sap.cloud.sdk.tutorial" |
| 1. RFC관련 서블릿 추가 | package com.test.jco;  import java.io.IOException;  import java.io.PrintWriter;  import javax.servlet.ServletException;  import javax.servlet.annotation.WebServlet;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;    import com.sap.conn.jco.JCoDestination;  import com.sap.conn.jco.JCoDestinationManager;  import com.sap.conn.jco.JCoFunction;  import com.sap.conn.jco.JCoParameterList;  import com.sap.conn.jco.JCoRepository;    @WebServlet("/ConnectRFC/\*")  public class ConnectRFC extends HttpServlet {  private static final long serialVersionUID = 1L;  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  PrintWriter responseWriter = response.getWriter();  try {  // access the RFC Destination "Test"  JCoDestination destination = JCoDestinationManager.getDestination("Test");  // make an invocation of STFC\_CONNECTION in the backend  JCoRepository repo = destination.getRepository();  JCoFunction stfcConnection = repo.getFunction("STFC\_CONNECTION");  JCoParameterList imports = stfcConnection.getImportParameterList();  imports.setValue("REQUTEXT", "JCO successful");  stfcConnection.execute(destination);  JCoParameterList exports = stfcConnection.getExportParameterList();  String echotext = exports.getString("ECHOTEXT");  String resptext = exports.getString("RESPTEXT");  responseWriter.println(echotext+":"+resptext);  } catch (Exception e) {  e.printStackTrace();  }  }  } |
| 1. POM.XML 수정   JCO library 다운로드 | <dependencies>  <dependency>  <groupId>com.sap.cloud</groupId>  <artifactId>neo-java-web-api</artifactId>  <version>4.40.5</version>  <scope>provided</scope>  </dependency>  </dependencies> |
| 1. Mvn clean install | $>mvn clean install |
| 1. Manifest.yml 파일   Destintation, connectivity, uaa서비스 생성 및 바인딩 | MYERPS20-xsuaa 생성시, 혹은 생성후에 파라미터로 추가  {  "xsappname" : "connect-rfc",  "tenant-mode": "dedicated",  "scopes": [  {  "name": "$XSAPPNAME.all",  "description": "all"  }  ]  }  ---  applications:  - name: myrfc01  memory: 1500M  timeout: 300  random-route: true  path: application/target/myrfc01-application.war  buildpacks:  - sap\_java\_buildpack  env:  USE\_JCO: true  TARGET\_RUNTIME: tomee7  SET\_LOGGING\_LEVEL: '{ROOT: INFO, com.sap.cloud.sdk: INFO}'  JBP\_CONFIG\_SAPJVM\_MEMORY\_SIZES: 'metaspace:128m..'  SAP\_JWT\_TRUST\_ACL: '[{"clientid":"\*","identityzone":"\*"}]'  xsuaa\_connectivity\_instance\_name: "MYERPS20-xsuaa"  services:  - MYERPS20-dest  - MYERPS20-conn  - MYERPS20-xsuaa |
| 1. 배포 | $> cf push |

| Explanation | Screenshot |
| --- | --- |
| 1. AppRouter프로젝트 생성. | $>mkdir approuter-rfc  $>unzip approuter.zip |
| 1. NPM 구성 | $>npm install |
| 1. Xs-app.json파일 수정 | {  "routes": [ {  "source": "/",  "destination": "dest"  }  ]  } |
| 1. Manifest.yml파일 수정   URL부분을 앞선 RFC servlet이 호출되도록 수정 | applications:  - name: approuter-jco  path: ./  buildpacks:  - nodejs\_buildpack  memory: 120M  env:  NODE\_TLS\_REJECT\_UNAUTHORIZED: 0  destinations: >  [  {"name":"dest", "url" :"https://myrfc01-quick-leopard-ct.cfapps.ap12.hana.ondemand.com/ConnectRFC", "forwardAuthToken": true }  ]  services:  - MYERPS20-xsuaa |
|  |  |