**SAP BTP CAP Introduce**

**In this session You will learn:**

* Setup BTP BAS Environment
* Build a Odata or Restful API
* Build and Deploy CAP Java APP to BTP Cloud Foundry

**TABLE OF CONTENTS**

[STEP 0: PREREQUISITES 3](#_Toc116659550)

[STEP 1: CREATE A CAP PROJECT FOR JAVA IN BAS 4](#_Toc116659551)

[STEP 2: DEPLOY TO BTP CLOUDFOUNDRY 11](#_Toc116659552)

[STEP 3: BUILD DOMAIN & SERVICE 24](#_Toc116659553)

[STEP 4: BUILD CUSTOM HANDLER 29](#_Toc116659554)

[STEP 5: BUILD RESTFUL 32](#_Toc116659555)

[STEP 6: ACCESS CONTROLS 34](#_Toc116659556)

[STEP 7: EXERCISE 35](#_Toc116659557)

# STEP 0: PREREQUISITES

## To go through the exercise, you need a BTP trial account. Of course, it also works for enterprise account. Please make sure:

## You have created a subaccount with Cloud Foundry enabled.

## You have subscribed Business Application Studio(BAS)

## You have HANA Cloud and HDI container entitled

## [Help Thomas Get Started with SAP HANA Cloud | Tutorials for SAP Developers](https://developers.sap.com/tutorials/hana-trial-advanced-analytics.html)

图形用户界面, 应用程序

描述已自动生成

Get a Free Account on SAP BTP Trial: [Get a Free Account on SAP BTP Trial | Tutorials for SAP Developers](https://developers.sap.com/tutorials/hcp-create-trial-account.html)

# STEP 1: CREATE A CAP PROJECT FOR JAVA IN BAS

|  |  |
| --- | --- |
| **Explanation** | **Screenshot** |
| Go to the SAP BTP Cockpit on Trial.  Choose SAP Business Application Studio under Quick Tool Access. | enter subaccount |
| Create a Dev Space with Full Stack Cloud Application template.  Once the Dev Space created, enter the development UI.  **Full Stack Cloud Application**: Build business services, business applications, and extend S/4HANA using the SAP Cloud Application Programming Model, Fiori and Java or Node.js.  [Dev Space Types | SAP Help Portal](https://help.sap.com/docs/SAP%20Business%20Application%20Studio/9d1db9835307451daa8c930fbd9ab264/4142f786f3d345699c3d5fbebda5ded6.html) |  |
| Create a project with wizard.   * Choose CAP Project * Check on HANA, MTA and CF related options * Check on Samples for starting   Click Finish to start the project generation |  |

|  |  |
| --- | --- |
| **Explanation** | **Screenshot** |
|  |  |
| The generated project is already runnable.  **db**:  The Objects associated with db. Defining the table structure.  **srv**:  Defining the service and build customer logic.  **manifest.yml**:  Defining appName, memory and env.  **mta.yaml**:  Defining module and resource.  **pom.xml**:  Maven dependency |  |
| Open the root folder. Set hdi-container and permissions here.  The 1.26.0 version will force the Service to validate, so you need to add the XSUAA  Settings |  |

|  |  |
| --- | --- |
| **Explanation** | **Screenshot** |
| This file generates the corresponding credentials . |  |

# STEP 2: DEPLOY TO BTP CLOUDFOUNDRY

|  |  |  |  |
| --- | --- | --- | --- |
| **Explanation** | | **Screenshot** | |
| Build the deployable mtar file with command mbt.  When it’s done, you can find the mtar file in project folder mta\_archives/. . | |  | |
| Right click on the mtar file and select Deploy MTA Archive to deploy it into BTP CF environment. | |  | |
| You need to enter your username and password to log in (**Valid for 24 hours**). | |  | |
| You can get Endpoint from BTP Subaccount overview. | |  | |
| When you are signed in to Cloud Foundry. You should select CF organization and space. Then click Apply | |  | |
| When the log prints Process finished. The deployment is successful. | |  | |
| In your CF space, you can find the running application. | | 图形用户界面, 应用程序  描述已自动生成 | |
| Click the application url and you will see the similar entry page but this time it’s running on CF with HANA Cloud. | |  | |
|  | |  | |
| Your access will be blocked(401) when you click the **Books**. Because you do not have permission. | |  | |
| Click the application **Environment Variables** and you will see the xusaa information in **System-Provided** json file. | |  | |
| Open the **Postman**. you should select method **GET** and select Auth: **Oauth 2.0**.  Grant Type: **Client Credentials**  Enter the **Access Token URL** and **Client ID** and **Client Secret** message from xusaa.  Then **Get New Access Token** and use it to **Send** request | |  | |
| You will find that it is already accessible and it sends the data back to you. | |  | |
| Go to SAP HANA Data Explorer and review the HDI container.  Check the tables created automatically. | |  | |
|  | |  | |

# STEP 3: BUILD DOMAIN & SERVICE

|  |  |
| --- | --- |
| **Explanation** | **Screenshot** |
| Open db  data-model.cds file Add 2 new entities: |  |
| Open srv  cat-service.cds file Expose 2 entities as odata services: |  |

|  |  |
| --- | --- |
| **Explanation** | **Screenshot** |
| Again, rebuild and redeploy project.  You can find the new created services. |  |
| CAP provides out-of-box support for CRUD operations on entity defined. We could use POST request to insert some data.  Open the **Postman** and input below json on **Body**:  {      "ID": 1,      "name": "TechEd",      "descr": "TechEd related topics",      "children": [          {              "ID": 10,              "name": "CAP Java",              "descr": "Run on Java"          },          {              "ID": 11,              "name": "CAP Node.js",              "descr": "Run on Node.js"          }      ]  }  This request trigger a api request and insert several records in Category table. |  |
| In the **Postman**,, send **GET** request, you can also find the data |  |

# STEP 4: BUILD CUSTOM HANDLER

|  |  |
| --- | --- |
| **Explanation** | **Screenshot** |
| Folder srv/src is the place we put all the java classes.   * src/gen contains the code generated by CAP * src/main contains the code developed by ourselves * src/resources is the standard resource folder, like configurations.   In java package customer.firstcapjava.handlers, we create a new java file AdminServiceHandler to add some additional logic    CAP is using event driven mode. So here we added some logic before an entity created(before it was inserted into DB) |  |
| Rebuild and redeploy project to CF and **Send** request to create a product object.  {      "title": "Pen",      "descr": "Pen used in office",      "stock": 8,      "price": 2.41,      "currency": {          "code": "USD"      },      "category": {          "ID": 10      }  } |  |
| In the **Postman** result we can see this new entity if the logic run. |  |

# STEP 5: BUILD RESTFUL

|  |  |
| --- | --- |
| **Explanation** | **Screenshot** |
| In java package customer, we create a new java file MyControoler to add some additional logic |  |
| In the postman we can see this restful API if the logic run. |  |

# STEP 6: ACCESS CONTROLS

|  |  |
| --- | --- |
| **Explanation** | **Screenshot** |
| In java package customer, we create a new java file MyWebSecurityConfig to resolve permissions issues |  |
| It is now accessible to all. |  |

# STEP 7: EXERCISE

|  |  |
| --- | --- |
| **Explanation** | **Screenshot** |
| Finish Step 0 to Step 6.  And use restful to implement the following functions.  Source Code: |  |