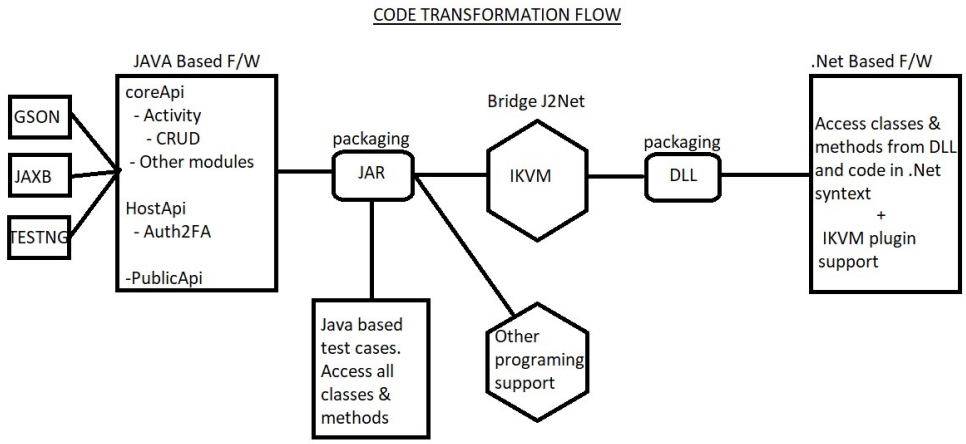
Automation Framework Integration

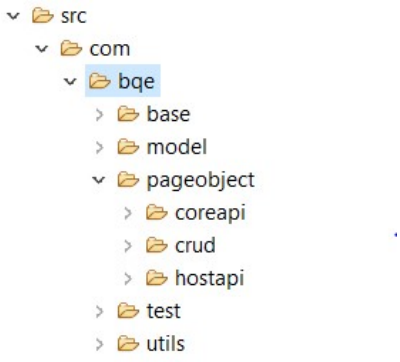
Java to .Net

IKVM

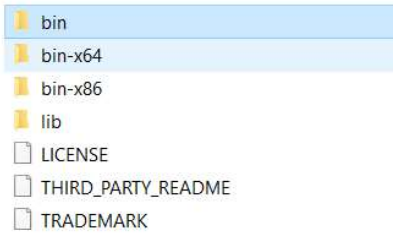
- **Objective**
  - We want to automation framework which support all aspects of QA Activities.
  - We are trying to create “Automation Framework” as “Language agnostic” which means any QA automation engineer comes with their expertise and start contributing to automation script writing with their skill set.
    - **Example**: we have framework written in Java. Which should be pluggable with any programming language. Today we have .Net support tomorrow we can extend it with any other programming language.
  - Objective behind this practise is fast automation with multiple skill of resources.
- **Language Agnostic Automation Framework**
  - **Tools**
    - Core Java as programming language,
    - HTTP Client library for HTTP calls,
    - GSON, JAXB for serialize & deserialize of JSON & XML,
    - TestNG for internal assertion.
  - **Editor**
    - Eclips or IntelliJ
  - **Packaging**
    - JAR file
  - **Currently Support Languages**
    - Java
    - .Net
  - **Code Transformation Flow**

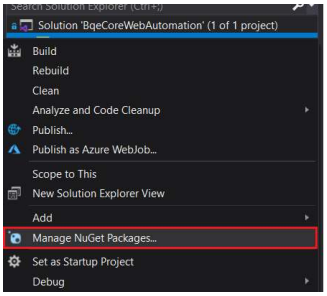


- **Folder structured of Automation Framework**

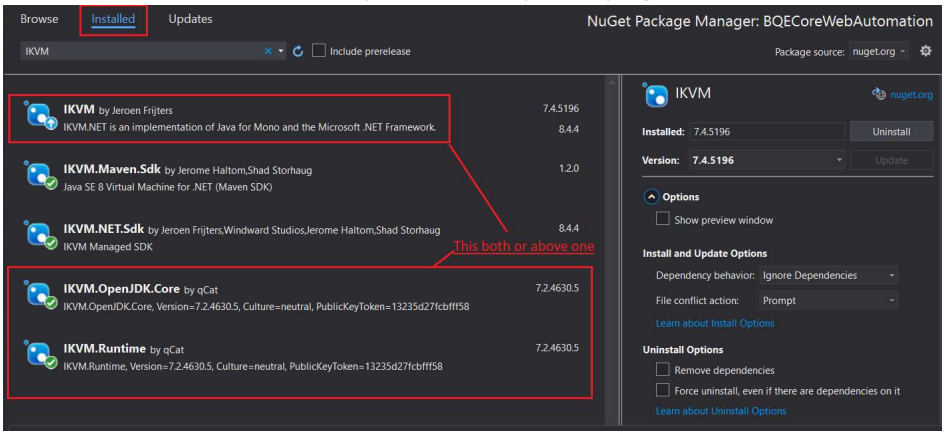


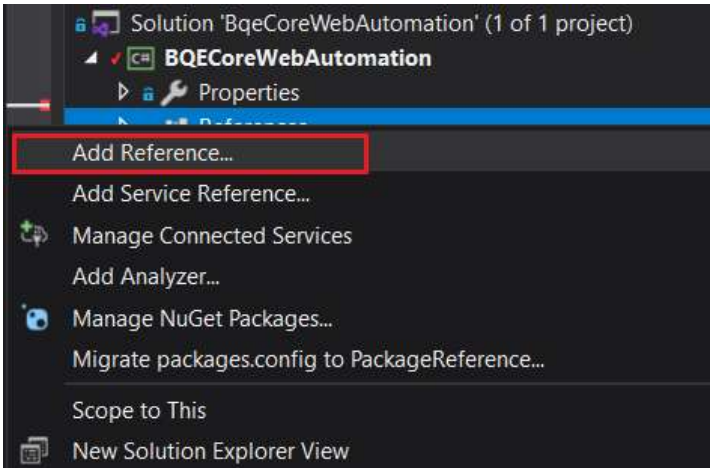
- **Communication between Java to .Net**
  - **DLL**
    - Java support *. Jar, *.ZIP, and *.WAR file as packaging and integration system while .Net support DLL as packaging and integration 3rd party library.
  - **JAR to DLL**
    - If we want to communicate between .Net and java then DLL format of packing is required. Hence, we are converting our Jar into DLL.
  - **IKVM**
    - We are using 3rd party tool for converting our JAR file into DLL
- **IKVM**
  - **What is IKVM?**
    - It is possible to convert a Java Jar file to .Net Dynamic-link library (DLL) file and use it in Visual Studio Projects. IKVM.Net an implementation of Java for Mono and the Microsoft .Net Framework makes this possible.
  - **Software Requirement**
    - IKVM library
    - (JDK) Java 7 or above
    - MS .Net Framework 3.5 or above
  - **Download IKVM & Generate DLL – Onetime activitys**
    - https://sourceforge.net/projects/ikvm/files/ikvm/
    - Download as zip file > extract it.

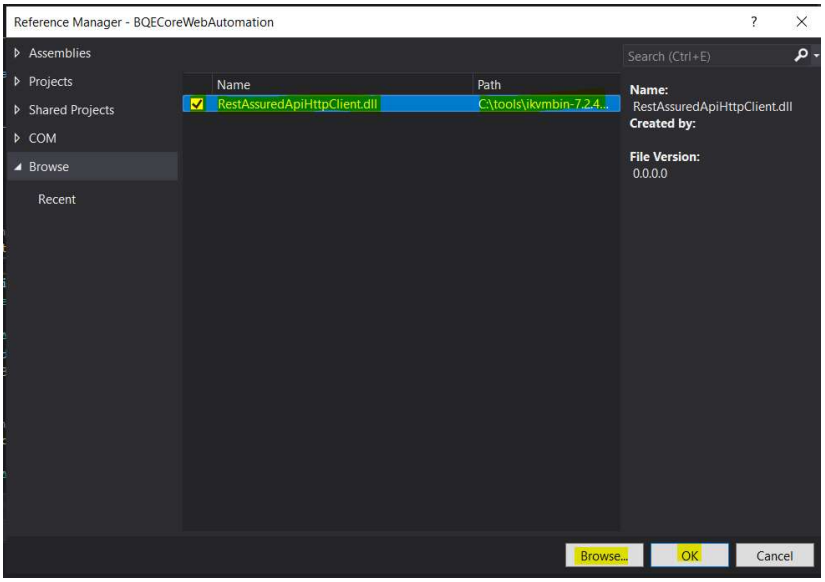


    - Go inside “bin” folder
    - Open CMD terminal as administrator
    - Type below **command** in CMD
      - ikvmc -target:library -out:{NameOfDLL_Destination}.dll -recurse:"{PathOfJar_source}*.jar"
  - **IKVM plugin for Visual Studio – Onetime activity**
    - IKVM is bridge between the java and .Net so that in editor we required to add plugin for compile and runtime support.
    - **Steps to integrate IKVM plugin**
      - Open VS with our automation project (BQECoreWebAutomation)
      - Open solution explorer
      - Right click on project and select “Manage NuGet Packages…”
      - 

- Search IKVM under Browse tab
- Install (IKVM.Runtime & IKVM.OpenJDK.Core) | IKVM plugin



- Once we add both IKVM plugins then time to add our generated DLL file.
- Adding DLL file as reference.
  - 

  - Browse the DLL file
  - 

- **Script writing approach**
  - Once we done with plugin and DLL integration, now we good write script (reuse all centralized code into our .Net and java code)
  - <u>We have 2 approaches for writing the script.</u>
  - **<u>Approach – 1</u>**
    - All Pages, methods, business logic, utilities, POJO, everything (method implementations & definitions) on new framework side and on test script side we will just pass required arguments, accept the response and validate the response. So, anyone can write the script (manual team can also quickly start writing the script by this approach)
    - **Advantages of Approach 1**
      - All business logic and complexity will be decoupled and hide under pages.
      - No technical knowledge required for script writer, because they just have to create object, invoke methods, pass arguments and validate the response.
      - Manual team can also start contributing to the automation.
      - It will maintain script format across different programming languages.
        - **Java Script snipped.**

```
System.out.println("---------------- [POST] GET AUTH TOKEN ----------------");
Validate2FAUserDto dto = ObjectMapper.getObject(response, ApiConst.ACCEPT_TYPE_JSON, Validate2FAUserDto.class);
System.out.println(dto.getBusinessToken());

System.out.println("");
System.out.println("---------------- [GET] GET ALL ACTIVITY ----------------");
String res = CoreApi.getInstanceOf().activityApi().getAllActivities(dto.getBusinessToken());
System.out.println("Get All Activities : "+res);

System.out.println("");
System.out.println("---------------- [GET] GET ACTIVITY BY ID=f1M:ouv ----------------");
String activityDto = CoreApi.getInstanceOf().activityApi().getActivityByID("f1M:ouv", dto.getBusinessToken(), String.class);
System.out.println("Activity ByID : "+activityDto);

System.out.println("");
System.out.println("---------------- [POST] CREATE ACTIVITY ----------------");
String createActivity = CoreApi.getInstanceOf().activityApi().createActivity(activityDto, dto.getBusinessToken());
System.out.println("Newly Created Activity : "+createActivity);

System.out.println("");
System.out.println("---------------- [DELETE] DELETE ACTIVITY ----------------");
CoreApi.getInstanceOf().activityApi().deleteActivityByID("FakeID");
```

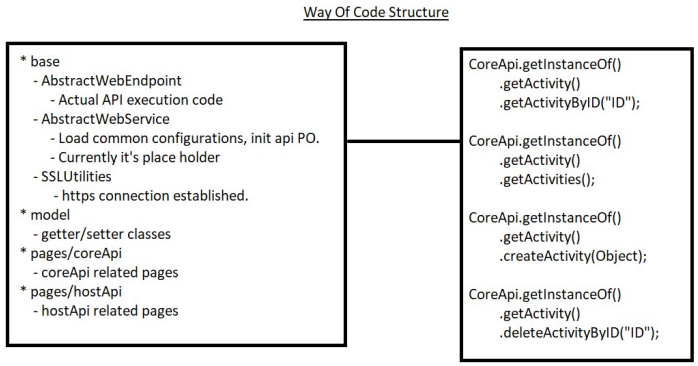
        - **.Net Script Snipped.**

```
1 reference | 0 changes | 0 authors, 0 changes
public static String getAuthToken()
{
    Validate2FAUserDto validateDto = new Validate2FAUserDto();
    validateDto = (Validate2FAUserDto) HostApi.getInstanceOf().validate2FAUser().getAuthToken(new AuthCredential(), validateDto);
    Console.WriteLine(validateDto.getBusinessToken());
    return validateDto.getBusinessToken();
}

1 reference | 0 changes | 0 authors, 0 changes
public static void getAllActivities(String token)
{
    String response = CoreApi.getInstanceOf().activityApi().getAllActivities(token);
}

1 reference | 0 changes | 0 authors, 0 changes
public static String getActivityByID(String token, String activityID)
{
    return CoreApi.getInstanceOf().activityApi().getActivityByID("f1M:ouv", token);
}

1 reference | 0 changes | 0 authors, 0 changes
public static String createActivity(String token, String activityBody)
{
    CoreApi.getInstanceOf().activityApi().createActivity(activityBody, token);
    return "FakeID";
}

0 references | 0 changes | 0 authors, 0 changes
public static void deleteActivityByID(String token, String id)
{
    CoreApi.getInstanceOf().activityApi().deleteActivityByID("FakeID");
}
```

    - **Disadvantages of approach 1**
      - It might generate duplication of code.

- If in case, there is any change in business logic then we required to change framework code, generate DLL again and link updated version.

- **Example**
  - HostApi have auth api & CoreApi has Activity and more Apis.
  - **Example of Get Token**
  - String token = HostApi.getInstanceOf().auth().getToken();
    - AuthToken authToken = HostApi.getInstanceOf().auth()
    - .getToken();
  - **Example of Activity**
    - CoreApi.getInstanceOf().activity().getAllActivities();
    - List<Activity> allActivities = CoreApi.getInstanceOf().activity().getAllActivities();
    - AssertThat(allActivities.size()>10);
- **Approach 1 Diagram**



- **Approach – 2**
  - Language agnostic framework responsibility is only executing HTTP call.
  - We will have all classes, business logic, models, everything on .Net side.
  - We will create business logic, pass the arguments to language agnostic framework and that framework will accept all valid args, execute API calls and return the response.
  - **Advantages of Approach 2**
    - No duplicate of code, we can reuse existing models, code and easy to maintain because everything under same language.
    - If any new feature introduces or deprecate in this case, we don't required to update DLL because DLL contains only HTTP calls.
  - **Disadvantages of Approach 2**
    - This way is losing its language agnostic power.
    - Sometimes it will be difficult to accept arguments because of cross platform.
      - **Example**: .Net has Dictionary concept and Java don't support it. So, If .Net user wants to pass Key value pair then it will difficult to pass with such objects.

- **Reference Links**
  - https://sourceforge.net/p/ikvm/wiki/Tutorial/#setup-your-environment
  - https://www.craftedforeveryone.com/convert-jar-file-to-dll-for-use-in-visual-studio-projects/