CHAPTER 8

Best Practices and FAQs

In every programming and technical platform, there is a recommended approach by the product owner. Similarly with SAP MII, there are dos and don'ts that need to be followed. SAP MII is a platform, so there are few recommendations from SAP. The best practices explained here should be followed for best performance and effectiveness of the solution.

Best Practices

It is recommended you follow the best approaches explained in the following sections.

Project Handling

From the project creation perspective in SAP MII, it is always recommended best to use NWDI (NetWeaver Development Infrastructure) as for the proper version controlling mechanism, use the central repository feature for the entire development component, and to have zero code overwrite risk. If NWDI is used, then a shared project is created and check in and check out of the code check-in and check-out is mandatory in it. If you create a local project and store it on the local server, then anyone with access to the server can access the code. There is a good chance that the code will get overwritten, as multiple developers can edit and save the code. The latest instance of a code change will be saved.

Project Folder Structure

When creating a project, it is recommended you use a specific, easily understandable structure. For example, in the Catalog tab under Project, there should be separate folders defining different functionalities of the project, such as a report name. Under each functionality, there should be separate folders for different object types like Query Template and BLS. The exact same structure needs to be followed for the Object and Web tabs, but the Query Template and BLS is replaced with MDO, KPI objects etc., and with the web files HTML (.irpt), JavaScript, etc.

S. Mukherjee and S. Mukherjee (Das), *SAP MII*, DOI 10.1007/978-1-4842-2814-2_8

Naming Conventions

Certain naming conventions should also be followed as part of your best practices. Here are a few examples:

- Scheduler: `<Project Name>_<Functionality>_<Plant (if any)>`
- Data Server: `DS_<Project Name>_<Server Name>`
- Credential Store: `<Project Name>_<Connection Name>`
- BLS: `BLS_<Functionality>`
- Query: `SQL_<Functionality>, [MDOQry/KPIQry/AlertQry]_ <Functionality>,PCO_<Functionality>, XCT_<BLS Name>` etc.
- Display Template: `LineChrt_<Functionality>, iGrd_<Functionality>` etc.
- Object: `MDOObj_<Table Name>, KPIObj_<KPI Name>`

Note that if the customer has some pre-defined naming conventions as per company policy, those conventions should be given priority.

Development

During development, follow these recommendations to write clean code and experience better and more effective performance:

- Only the input and output properties used in the BLS should be declared as *transaction* variables. All other variables temporarily used in the BLS should be declared as *local* variables.
- Always use the `Exceptional Enabler` and `Catch` action blocks to capture any exceptions within the logic.
- Avoid using a repeater under another repeater. Avoid loops as much as possible in the logic. Instead, use XSLT to achieve good performance.
- Use XPath and XQueries to avoid looping.
- Remove all the tracers and loggers once the development is finalized (i.e., before migration), otherwise, the performance will be very poor and sometimes can lead to memory crash issues due to heavy logging.
- Avoid writing file and loader action blocks as much as possible because when they execute, they consume a considerable amount of memory.
- Avoid using a terminate action block.

- You must remove breakpoints and watchpoints once debugging is done; otherwise, the execution of BLS will stop at these breakpoints.
- If any BLS is executed in debug mode, close it with the Debug Close button. Do not close it directly by clicking the cross (X) button. Even if the BLS is closed, the cache memory will continue caching the debug log whenever the BLS executes and a Java heap space error could occur.
- Avoid storing large input in the local variable again, even if it is already coming from source, because once a value comes in a BLS, it exists until the end of the BLS execution.
- Avoid creating a local XML structure as a local variable and appending the value in it if it is possible to achieve the same effect through MII Output XML action blocks.
- Try to use asynchronous transaction calls as much as possible if they fit in the logic. Synchronous transaction calls hold the execution of primary BLS until the execution of the called BLS is complete and thus adversely affect performance.
- Use meaningful naming conventions with *Camelcasing* for all action blocks. If possible, provide action block descriptions.
- Avoid calling BLS from a scheduler where, within that BLS, a synchronous transaction call is present. It holds the execution of scheduler instance and keeps the scheduler in a running state for a long time. In such cases, try to use a wrapper BLS, which will call the primary BLS asynchronously and put the wrapper BLS in the scheduler. The scheduler instance will not be queued until the next execution since it works asynchronously.
- Do not hardcode usernames and passwords in an action block. Try to use credential aliases and refer to them in the action block. On a similar note, this is also applicable to the connection alias.
- Avoid hardcoding in the action block configuration, and try to use it dynamically as much as possible.
- Do not use `select *` in a SQL query if it is not required. Try to define the specific column while querying.
- Do not use hardcoded timeout in the UI. Try to handle timeouts using logic.
- Do not hardcode usernames and passwords in a URL used in the UI.
- Do not repeat any logic part in BLS and in UI. In UI try to call the function in multiple places.

- It is suggested to use sessioning during an ECC call using JCO, JRA, and similarly for query templates. It helps to commit and roll back the call.
- It is recommended to use the MVC model for UI, which helps keep code clean and easily understandable to other developers.
- Try to use separate property files for different user locales in the Web tab.
- Try to develop a utility file that will have the common functionality code for different modules so as to reuse it in other object development settings.
- Explain in the comments section the use of the code and the functions being used in the UI frontend coding. This will help other developers understand the code later.

FAQs (Frequently Asked Questions)

Here are some common queries that SAP MII professionals and practitioners have:

1. *What is SAP MII?*

   SAP MII is a platform developed in Java by Lighthammer and acquired by SAP. SAP MII can work as middleware with real-time integration and high-end intelligence capabilities. SAP MII is so advanced that lots of innovations are possible using IOT.

2. *How do I install SAP MII?*

   SAP MII can be installed on top of NetWeaver or the SAP HANA Web application server.

3. *What versions are available in SAP MII?*

   SAP MII was introduced by SAP as SAP xMII 11.5. Later on, there was SAP MII 12.0, SAP MII 12.1, SAP MII 12.2, SAP MII 14.0, and SAP MII 15.0. The latest version is SAP MII 15.1.

4. *Does SAP MII support SAP HANA?*

   Yes, from SAP MII 15.1, it is possible to expose the MII data in HANA using the OData service.

5. *What is the base component of SAP MII?*

   SAP MII is developed using Java Swing and SAP MII supports XML for all internal processing and handling.

6. *What kind of integration is possible using SAP MII?*

   SAP MII can connect with SAP ERP; any kind of databases; systems that support any kind of REST service like HTTP Post, Web service, JMS etc.; and any kind of shopfloor system that supports UDC connector.

7. *What kind of intelligence does SAP MII have?*

   SAP MII can show rich and high-end reporting using various charts and it is also possible to develop a user interactive interface using SAP MII.

8. *What kinds of applications are possible using SAP MII?*

   SAP MII based applications are always web-based applications because SAP MII runs on Web Application Server.

9. *Does SAP MII provide real-time data?*

   Yes, SAP MII is very much capable of providing near real-time data. However, as applications developed on SPA MII are web based, it is difficult to expect exact real-time data in a SAP MII-based application because the data flow is dependent on the network speed.

10. *How flexible is a SAP MII based application?*

    From an integration perspective, SAP MII can connect and fetch data bi-directionally from most of the systems. From a UI perspective, SAP MII now uses SAP UI5 for its user interface development. SAP UI5 is browser independent, which helps the application run in any kind of window or any mobile browser, regardless of the operating system.

11. *What OS servers does SAP MII support?*

    SAP MII can be executed on Windows, on Linux, and on SAP PCo, which is one of the freely distributable components with SAP MII. It's also an important integration tool with shopfloor, but it supports only Windows Server, as SAP PCo is developed using the Microsoft .NET technology.

12. *How is MDO different from a database?*

    MDO stands for Manufacturing Data Object. It is an internal database or cache for SAP MII applications. Instead of querying a database, data can be stored in MDO and queried locally to fetch data faster. There are two types of MDO—persistent and on-demand. These differ in data storing ways.

13. *How is HANA integration done with SAP MII?*

    Data from plant or manufacturing systems can be obtained in SAP HANA using SAP MII Smart Data Adapter (SDI). Using the adapter data from PCo, queries in SAP MII will be available on virtual tables in SAP HANA and the tables can be used for various analytics and calculation views in SAP HANA.

14. *What is shared memory?*

    It is a persistent memory that can be used to store limited values and can be used in transactions while developing logic as a persistent variable. Shared memory can be accessible to the users through the SAP MII menu page. To change a value in shared memory, no code change is required.

15. *What is a custom action block?*

    A custom action block is customized code developed in the Java platform to handle certain functionalities which are not available as standard functions in SAP MII. Using the standard SDK available for the custom action block, it is possible to create the .JAR file containing the Java code with the special handling functions and upload and use the file as an action block in SAP MII workbench.

16. *Is it possible to access Excel files?*

    Yes it is possible, in two ways. If the Excel filename is fixed, even if the content is refreshed periodically, it is possible to integrate the file through PCo, the file system connector and OLEDB connector in SAP MII. Another way is to develop a custom action block through Java to integrate it.

17. *Can SAP MII Transaction be called remotely?*

    Yes, it is possible to call SAP MII Transaction using runner services. A runner service is an illuminator service that can help to call MII transaction from outside using this URL format:

    ```
    http://<server>:<port>/XMII/Runner/Transaction=<fold
    er>/<TransactionName>&<InputParamName1>=<value1>&<In
    putParamNameN>=<valueN>&OutputParameter=<OutputParam
    Name>&Content-Type=<content-type>&isBinary=<true|fal
    se>&XacuteLoginName= <username>&XacuteLoginPassword=
    <password>
    ```

18. *Is it possible to expose SAP MII Transaction as a REST service?*

    Yes, it is possible in SAP MII to expose any SAP Transaction as a SOAP web service.

The MII transaction can be converted into WSDL using this URL:

```
http://<server>:<port>/XMII/WSDLGen/<projectName>/
<folderName>/<transactionname>
```

To run the SAP MII transaction as a Web Service, use the following URL:

```
http://<server>:<port>/XMII/SOAPRunner/<projectName>
/<folderName><transactionname>
```

19. *Does SAP MII support synchronous and asynchronous calls?*

    Yes, it supports both synchronous and asynchronous calls. Inside a SAP MII Transaction, other transactions can be called depending on the requirements. Transactions can be called in both asynchronous and synchronous ways. In a synchronous call, parent transactions call the child transactions and wait for the child transaction to complete execution. Based on the executed result, the parent transaction completes the logic and closes the session. In an asynchronous call, the parent transaction triggers the child transactions and the parent transaction completes its full logic and closes the session. It does not wait for the completion of the child transaction being triggered and thus the child transaction is always executed independently.

20. *What is the difference between the default and active memory of a shared property?*

    Shared properties in SAP MII have both a default and active memory. Default memories have default values whereas active memories have active values. Active values can be checked either from the Tools tab from workbench or from the Data Servers sections of the SAP MII Admin page. At the time of transaction execution, a value is always fetched from active memory. Initially when the default memory is created for any variable, the same value gets replicated immediately and automatically to active memory the first time.

21. *What is the difference between an iterator and a repeater?*

    SAP MII Transaction gives us a wide variety of action blocks for logic assignments. To loop in, there are two main action blocks—an iterator and a repeater. An iterator is used to loop over lists, as its input source is a list. The repeater is used to loop over XML nodes.

22. *What are scheduled transactions?*

    Scheduled transactions are the ones that enable execution of MII Transaction as per defined schedules for required logic. They can be enabled or disabled any time as needed.

23. *Is there a platform from which plant hierarchy can be directly maintained?*

    Yes, SAP MII provides the PIC (Plant Information Catalog) from SAP MII version 15+. Using this, it is possible to define the plant hierarchy along with the tags and details from SAP MII.

24. *How is PCo used in SAP MII?*

    SAP Plant Connectivity (PCo) is a software component that enables the exchange of data between the shopfloor and SAP MII. In PCo, it is possible to define the shopfloor tags to get real-time time-series data directly from the machines; it can maintain the historical data too. There are two ways to exchange the data from PCo—one is the push methodology to create agents with groups of tags and define certain conditions based on which a notification is triggered to the specific defined BLS of SAP MII. The second is the Pull mechanism by which it is possible to directly fetch data from the tags by using the PCo query via PCo connector from SAP MII.

25. *Does SAP MII support any other SAP manufacturing product?*

    Yes, it supports SAP manufacturing products—SAP ME and SAP OEE for their integration components MEINT and OEEINT. As both the products are developed on top of SAP MII, the custom logic hooking (activity hook) and custom dashboard (POD for ME and operator dashboard for OEE) development is also possible through SAP MII and with the standard solutions. Hence, to summarize, SAP MII is tightly coupled with both manufacturing products.

26. *What is bootstrapping?*

    To begin any application in UI, it must first be loaded and initialized. This process of loading and initializing is called bootstrapping.

27. *What are the supported data models for SAP UI5 development?*

    SAP UI5 supports XML model, JSON model, OData model, and JS model view.

28. *What is the difference between applet-based UI coding (.irpt) and UI5 coding?*

    In Applet-based UI coding, to load any kind of display template like chart, grid etc., you need Java JRE to load initially, which consumes a significant amount of time during execution and can lead to cache issues. Sometimes JRE versions also create issues for the applet version. UI5 is a Java-independent coding for all the UI components like HTML5, as it follows MVC model. This means the processing time for UI5 is much faster compared to .irpt files and there are no cache issues in UI5.

29. *What is the difference between an i5 display template and a normal display template?*

    SAP MII introduced new display templates that are UI5 based and they have been named as i5 templates whereas the existing ones are applet-based templates.

30. *What is an IRPT page?*

    An IRPT page, also referred to as an SAP MII report, is a web page that uses the SAP MII .irpt file extension rather than the standard .html file extension. IRPT pages facilitate the dynamic generation of web pages through server-side processing. The web server sends the .irpt page to a specialized servlet, called `ReportServlet`, for processing. There are some special features used by IRPT like session variables and dynamic content insertion in web pages. This makes it different from HTML.

31. *Is there any functionality in SAP MII for tracing and logging activities?*

    Logging in SAP MII can be done with NWA logs with some configurations done in NetWeaver and this tracing can be initiated. Event Logger action blocks trace custom messages in logs from SAP MII Transaction as well.

32. *What is an illuminator service?*

    Illuminator service is the backbone of SAP MII. Using this web service, any of the SAP MII standard components can be called, modified, and updated. Apart from that, the components of SAP MII, like query template, display template, and the components of the UI, can be called directly to access them. It is the only way to work with SAP standard data stored in an internal table.

33. *How do I use illuminator services?*

    Illuminator services contain two major parameters—one is the service and the other one is the mode. There are multiple services available, including admin, systemInfo, scheduler, BLS manager services, and more. Various modes are also available for each service. The way to write it is as follows:

```
http://<server>:<port>/XMII/
Illuminator?service=<service name>&mode=<mode
name>&content-type=text/xml
```

Summary

In this chapter, you learned about the best practices that should be followed in SAP MII. Along with that, the frequently asked questions are also provided. They can help you have a batter grasp of SAP MII.

Now you are ready to work in SAP MII. You have an understanding of manufacturing industries. You also know the history and types of manufacturing industries. You have gone through how SAP MII came into manufacturing industries and how it is progressed day by day along with the versioning. It will provide you with the understanding of changes and requirements for SAP MII upgrade projects. You learned why SAP MII is important and where to use it, so you can now relate the business requirements and can easily understand the probable scenarios for the business. As you got the flavor on the basics, components of SAP MII, and how to develop technical objects in the SAP MII IDE, you should be confident working in it.

Along with it, you got a detailed understanding of the various manufacturing domains and their requirements, as well as how SAP MII and other SAP manufacturing products can fulfill those requirements. Last but not least, you are also aware of various integration scenarios using SAP MII and the new features in SAP MII. So to conclude this book, you will be your perfect guide to understanding and exploring SAP MII. Further, to gain more knowledge on SAP MII, you can visit the SCN—SAP Community Network—at https://www.sap.com/community/topic/manufacturing-integration-and-intelligence.html, wherein you can read various blogs, documents, and discussions.