ORACLE

Dynamic Components and Concepts

Mustafa Cayci
Senior Principal Product Manager

When You Should use Dynamic Components

Dynamic Components should help application developers in the following use cases:

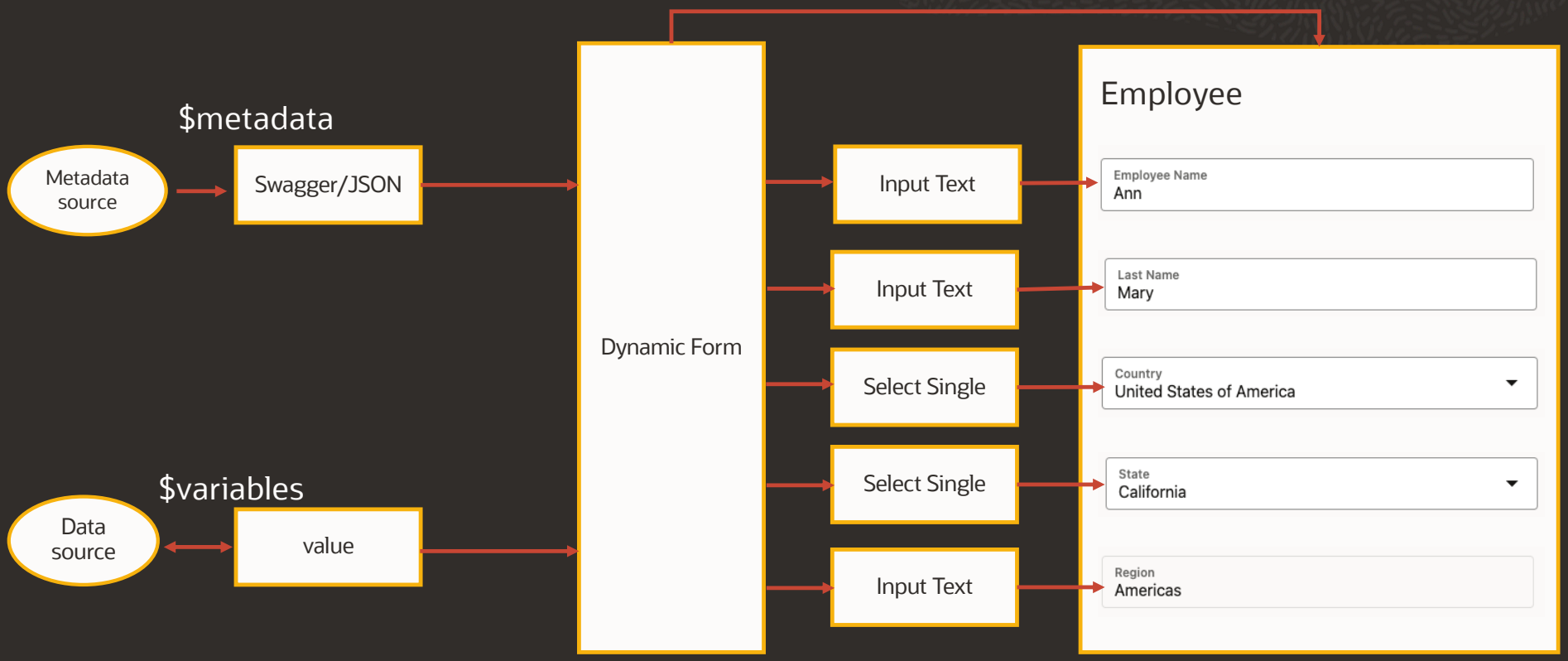
- Customizing applications in a declaratively way
- Extending and customizing Fusion Cloud Applications with low code or no code
- Achieve high degree of customization with low code or no code
- Quick Start capability in Visual Builder Studio Design Time for various use cases
  - Include table data
  - Create, Edit and Details Form
  - Editors, Rule Set, Templates and Fields

How do I access Dynamic Components

Part of Oracle Visual Builder (VB) Application Development Architecture

- First it was introduced with VB version 21.07 which was Dynamic Components version 11.07
- Dynamic Components include:
  - Dynamic Container – Dynamically decide what to include inside a page
  - Dynamic Table – Dynamically display data in a table format leveraging oj-table under the cover
  - Dynamic Form – Dynamically display data in a form leveraging oj-form-layout under the cover
- Supporting Components:
  - Field Binding
  - Section Binding
- Dynamic Components are available in Visual Builder Studio
- Under VB Components section, you can search for Dynamic as a keyword to see versions and update to latest version. Updates should mention Dynamic UI Pack as the title

High Level Dynamic Components Architecture



Client Metadata Features

Dynamic components are designed

- Evaluate and process metadata obtained from business services
- Business Service Driven
- Augment Business Services using Client Metadata Capabilities

Client Metadata Features (Continued)

Dynamic Components provide

- Extension support for Oracle Fusion Applications to augment metadata in Application Composer
- Design and runtime support in Visual Builder to support metadata changes
  - Add Fields
  - Add Metadata
  - Replace Metadata
- Expression Support

Dynamic Layout

Dynamic Layout defines the structure and set of rules where dynamic components are rendered

Dynamic layouts allows to control which fields to render, order of fields to render and more

```
"layouts": {
      "default": {
            "description": "default layout for employee form",
            "layoutType": "form",
            "layout": {
                  "direction": "row",
                  "maxColumns": 2,
                  "labelEdge": "start",
                  "labelWidth": "16.6%",
                  "displayProperties": [ "firstName", "lastName", "employeeId",
"hireDate", "email", "location" ]
```


Context Variables

Application developers needs various information to develop applications This information is gathered from various sources such as metadata from business services

Context is a way to expose this information to developers to implement dynamic layouts or templates

Some more examples around context variables

- $fieldMetadata – References a metadata associated with a field
  - `"displayProperties": [ "name", { "email": { "helpHints": { "definition": "[[$fieldMetadata.description]]" } } } ]`
- $readonly – Corresponds to ”readonly” property in Dynamic Form or Dynamic Table (11.0.0 and up) and 12.0.0 as a field level context variable

  `"readonly": "[[ $fields.ToReassignFlag.value() === false ]]”`

Visual Builder Built-In Contexts

- Some examples of context variables in Visual Builder
  - `$user, $response, $translations, $responsive, and $info`

Context Variables (Continued)

Some more examples around context variables

- $fieldMetadata – References a metadata associated with a field
  - `"displayProperties": [ "name", { "email": { "helpHints": { "definition": "[[$fieldMetadata.description]]" } } } ]`
- $readonly – Corresponds to ”readonly” property in Dynamic Form or Dynamic Table (11.0.0 and up) and 12.0.0 as a field level context variable

  `"readonly": "[[ $fields.ToReassignFlag.value() === false ]]”`

Visual Builder Built-In Contexts

- Some examples of context variables in Visual Builder
  - `$user, $response, $translations, $responsive, and $info`

Examples of Using Context In Layout and Metadata

Using context to define a rule

```
"layouts": {
      "isDefault": {
            "description": "Default Rule",
            "type": "employees_10",
            "expression": "{{$value.employeeId > 2000 ? 'alternate' : 'default'}
```

Using context to determine displayProperties

```
"emp-table": {
      "description": "employee table layout",
      "layoutType": "table",
     "layout": {
          "displayProperties": [
                "{{$responsive.smOnly ? '!employeeId' : 'employeeId'}}",
                "{{$responsive.smOnly ? '!email' : 'email'}}"
           ]
       }
```


Global Field Templates

Global field templates are field templates based on similar fields in dynamic components across different business objects

Global field templates are available in oj-dyn-form (VDOM based Dynamic Form)

field-templates-overlay.json and field-templates-x.json files capture properties and the mapping. They both have the same format

- field-templates-overlay.json is used for Base Application
- field-templates-x.json is reserved for Application Extensions

```
"addTemplates": {
       "globalEmailTmpl": {
             "description": "template for rendering fields with
layoutDiscriminant of 'email’”,
             "extensible": "none"
        },
        "addTemplateMap": {
      "singleValue": { "$byLayoutDiscriminant": { "email": "globalEmailTmpl"
},
```


Readonly and Required

Fields can be configured either “readonly” or required which means updateable

”readonly” and required attributes are configured in several ways:

- Swagger or JSON base metadata as they are configured in REST service

```
},
"oracle_apps_crmCommon_salesParties_accountService_view_AccountVO-readOnlyFields": {
    "properties": {
        "PartyId": {
            "type": "integer",
            "format": "int64",
            "readOnly": true,
            "title": "Party ID",
            "nullable": false,
            "x-hints": {
                "precision": 18
            },
            "description": "The unique identifier of the account."
        },
```


Readonly and Required

"readonly" and required properties are configured in several ways:

- readonly and required properties can be used in the Client Metadata Augmentation
- Controls the rendering of fields in VB applications
- Controls fields by defining field templates and giving full control how fields are rendered
- Used in Dynamic Tables Metadata determines if a field is readonly or required Readonly behaves differently for operations that have a response body and operations that do not have response body

Server Metadata Features

Metadata is fetched from the services that define business objects. For example, if a VB application is concern with Accounts object, the metadata is retrieved for that object from some back end REST service.

The REST service should be OpenAPI 3.0 compliant.

Once the OpenAPI document is obtained from the REST service, the dynamic component transforms the document to metadata that is recognizable by dynamic components.

This is called Metadata Transformation.

The API documentation outlines the properties present in the metadata:

- componentType, converter, dataProvider, defaultValue, readonly, description and many more are given in the API documentation

Server Metadata Features

Metadata by backend services is consumed by dynamic components.

Backend service is described as OpenAPI 3.0 specification.

The mapping between metadata property in dynamic component and OpenAPI 3.0 specification.

- For example: dataProvider in dynamic component metadata is defined as x-lov in OpenAPI 3.0 documentation for that service.

Custom Metadata Transform allows support for specific use cases by applications.

Server Metadata Features

Sample of OpenAPI 3.0 documentation with link definition

```
"links": {
        "stateByCountryLOV": {
                "operationId": "getStatesByCountry",
                "parameters": {
                        "countryId": "$this#/country",
                        "status": "[[$componentContext?.status]]" },
```

Dynamic component documentation

```
"state": {
        "type": "string",
        "labelHint": "State",
        "dataProvider": {
                "factory": "vb/types/factories/serviceDataProviderFactory",
                 "parameters": [{
                        "endpoint":
"https://sample.us.oracle.com/country/{countryId}/states?status={status}",
                        "uriParameters": { "countryId": "[[$value.country]]",
                        "status": "[[$componentContext?.status]]" } }] }
```


Learn More

- JET API Documentation - https://www.oracle.com/webfolder/technetwork/jet/index.html
- Visual Builder Cookbook - https://vbcookbook.oracle.com/
- blogs.oracle.com/vbcs
- Oracle University Learning Subscription – Build Visual Applications using Oracle Visual Builder Studio
- Cloud Customer Connect Forums:
  - Visual Builder
  - Visual Builder Studio
- Dynamic Components API Reference Accessible from Components Palate in Visual Builder Studio
- Join the Oracle Visual Builder group on LinkedIn

ORACLE

DEMO