Outsystems Overview

Outsystems is a platform that helps us to build,deploy,run and manage applications using visual code development and low code model.

Outsystems Components:

1. The Platform Server
2. Development – Service Studio, Integration Studio
3. Admin & Operations – Service Central, Lifetime
4. Forge and Community

The Platform server: A set of server that compiles, deployes, manages, runs and monitors the application

Service Studio – An IDE provided by outsystems for developers to develop visual code and logic

Integration Studio – An editor to develop and integrate with external application like .NET and databased.

Service Central - A server management and Admin Console. Can be accessed via web browser. ,manages running applications, and server configuration.

Lifetime – manages the lifetime of applications across an infrastructure from deployment, production and quality aspects.

Forge – provides extensions, pre written codes, libraries, narketplace from community to integrate our apps with.

Community – A public community to ask questions, interact in forums

Outsystems features and capabilities:

1. Integrate with anything
2. Visual Design and development
3. Standard and optimized code
4. Continuous Integration/Continuous Development
5. Flexible Deployements

Service Studio –

Application Layer:

1. Processes
2. Interface
3. Logic
4. Data

Modular Programming: Application Can be structured in modules. Different types of modules like app module, service module, blank module, etc.

Elements can be reused and shared between modules and other applications as well.

Module that shares an elemtent is called producer.

Module which consumes an elements is called consumer.

MODELING DATA:

Modeling data for an app:

1. An applications contains important information and business concepts.
2. We need to store this information(persist) and retrieve at times.
3. So, these concepts are modeled and refered to as ENTITIES in outsystems

Entities are created in development environment.

Outsystem Entities	----->	Database Table
Attributes		Columns Data types
Id(IdentifierAttributes)		Primary Key
Reference Attribute		Foreign Key
Index		Index
Record or Instance		Row or Tuple

Entities have Id attribute by default.

Each entity must have atleast one attribute

Basic Data types:

Alphanumeric: Text, Email, PhoneNumber

Numberic: Interger, Long Integer, Decimal, Currency

DateTime: Date, Time, DateTime

Logic: Boolean

Large Object: Binary Data

Referential: Entity Identifier

All dataypes have default value if not specified.

Outsystem also inferes datatype on its own depending on attribute name eg: *Email ---> PersonalEmail

Also default automatic CRUD actions which CANNOT be modified

To work with entities, we can access the Data layer / Entities

We can bootstrap / load entities as well as data from excel sheet

When we load, a bootstrap timer is created in process layer/ timers which is executed when we PUBLISH.

STATIC ENTITIES are special type of entities that create a PREDEFINED LIST OF VALUES that can be used in our application.

It acts like a enumeration. We call it a record(list of items).

They have attribute and records.

Defined during development and CANNOT BE CHANGED AT RUNTIME.

ONLY HAS ONE ACTION - GET (only be read)

They have attributes - ID, LABEL, ORDER, IS_ACTIVE

These attributes CAN BE CHANGED

Each record has : Identifier(Id) Eg: LOW, MEDIUM, HIGH

A Record can be changed/added anytime

Right click Data layer/Databases/Add static entities

TO BOOTSTRAP DATA FROM EXCEL:

1. Right click entity

2. Click advances
3. Select create action to bootstrap data from excel
4. Auto mapping is performed by outsystems. Click Proceed

What is a screen?

A screen is a build block or a part of an interface layer.

It is composed of various elements OR WIDGETS like buttons, links, images, etc.

End user can interact with this screen and can follow a particular screen flow according to design.

Eg: Starting with login, then homepage, then CRUD screen.

Screen can start from emplty canvas or templates.

Screen has WIDGETS like TITLE, TABLE, BUTTONS,ICONS.

Widgets are reactive and have diff formats on diff screen sizes.

Screen inputs and variables:

Input parameters: Some data can be passed to screens

Local Variables: in the scope of screen (inp para are also in scope)

Buttons: have triggers or links to actions

OTHER SCREENS CANNOT ACCESS LOCAL VARIABLE ON ONE SCREEN

These screen actions are CLIENT SIDE LOGIC AND ARE IN THE SCOPE OF THE SCREEN

UI elements immediately react to data changes.

Links and Buttons can either NAVIGATE OR TRIGGER SCREEN ACTIONS

A container widgets allow grouping of several widgets

To work:

Interface Layer/ Mainflow/ Add widgets from left side tool box.

Eg. Expression widget

Publish to see in runtime

Logic

What is Action?

An Action is an element that helps define a logic flow.

This action can run on client side or server side as well.

Action Flow is where logic is defined.

IT CAN HAVE ONLY ONE START NODE.

An action can end with mulitple nodes: End, Destination(Screen Actions only), Downloads(Screen Actions Only).

THERE ARE 3 TYPES OF ACTIONS:

SCREEN ACTIONS: Logic specific to single screen. Contains logic in widgets, aggregates, input parameters, local var

CLIENT ACTIONS: Logic used on a device. Used Throughout the module and not limited to screen.

SERVER ACTIONS: Logic used on server side.

Screen action have inp para, local var, but client and server ALSO HAVE OUTPUT PARA IN ADDITION.

There can be multiple screens with multiple screen action (with screen limited scope).

Screen ---?(cannot call actions) OTHER Screen

Screen --->(can call actions) client, server

client --->(can call actions) Server

NOTE: Internet connection is needed from screen/client to server.

CLIENT ACTIONS AND SERVER ACTIONS CAN BE SET AS FUNCTIONS

-restricted to ONLY ONE OUTPUT PARAMETER.

-available in expressions.

Client actions that are set as functions CANNOT CALL SERVER ACTIONS.

Exceptions: an exception is thrown when an operation fails at runtime.

It causes and INTERRUPTION IN THE EXECUTION of the flow.

The execution is then moved to the exception handler flow.

What is an Aggregate?

An aggregate is a VISUAL element which allows us to define a QUERY TO FETCH DATA from entities with SOURCES, FILTERS, AND SORTING.

It creates query in a visual way.

4 tabs:

Add sources: from where data is retrived. Can be one or more entities

Create filters: adds one or more conditions to query to filter the output records. Can also use built in func like If(), CurrDateTime(), etc.

Define sorting: Sort output in particular direction i.e asc or desc.

Can also define multiple sorts

Test: Used to test aggregate and PREVIEW records. No influence at runtime.

Aggregates Output and Properties:

AGGREGATES RETURN LIST OF RECORDS.

List has attributes:

1. Iterator: Current Cursor, CurrentRow,EOF,BOF

Current points to first row by default.

Filled with default values if no rows are returned by query.

2. Length: number of elements returned

3. Empty: True if no records were returned.

4. Count: total number of records that match criterion defined in aggregate. CAN BE DIFFERENT THAN LENGTH OF LIST.

List has properties:

MAX. RECORDS: limit the records to be fetched

SQL DIALECT: a read only sql statement generated from aggregate. Source in FROM and filter in WHERE clause sort in ORDER BY

STEPS:

OPTION 1: USE aggregate widget from left side tool bix

OPTION 2: Directly drag entity in the flow to create GetEntity() aggregate.

What is a variable?

A variable is a location in memory that can hold data.

They exist in particular scope.

Three types: Input variable, Local Varible, Output Variable.

Input parameter: allow to pass value from outside scope.

Can be be set as mandatory.

Output parameter: returns a value to the outside scope.

OUTPUT PARAMETER CONTINUES TO EXIST EVEN IF ITS PARENT ELEMENT'S SCOPE IS GONE whereas input are destroyed.

Local variable exists only within the scope of its element.

OUTSYSTEM LANGUAGE IS STRONGLY TYPED.

TYPES MUST BE DECALRED AND CANNOT BE CHANGED.

Outsystems supports three types of datatypes:

1. Basic types: Interger, Date, text,etc

2. Compound Types: Structures, Entities

3. Lists

Structures are custom compund data types

-Includes other Structueres, Entities and Lists

-It is only a defination of data type and DOES NOT HOLD ANY VALUE

LIST is a collection of elements of same data types: basic, compound or Union record

A block is a reusable piece of code.

They are user interface elements. Its can hold widgets, patterns and other blocks.

They are designed like screen. They also do not have output parameters.

Events can be used to communicate.

It has placeholder and events.

A placeholder reserves space for dynamic interface content.

Every instance of a block can have different content inside placeholders.

Blocks promote maintainability and reusability

Events:

Interaction between blocks can be done using parent

A block triggers an event and the parent handles it

DEVOPS IN OUTSYSTEMS

PARTS OF TEAM:

1. Business
2. Demand and Governance
3. Development
4. Architecture
5. User Experience
6. Operations

Members Involved:

1. Business Sponsor: top decision making authority, provides business vision,direction,guidance
2. Key Users / SME(Subject matter experts): End users and crucial to agile projects. Active role in project from requirements to testing
3. Project Owners: Primary person responsible for **functional solution**. Understand impact of business. Create vision and align solution towards it. Backlog Mnaagement. Change Mnaagement. Budget control, quality control.
   Need to have organisation awaarenes, REAL BUSINESS VISION
4. Tech lead(TL) : Solution Architect, development team lead. Deliver high quality solution. Together with product owner, protect development plan . SOLUTION ARCHITECT, PROJECT STEERING, TEAM BUILDING.
5. Developers: Implement technical sols and perform iteration tests
6. Project Managers: responsible for program and strategy.
7. Experts: Work closely with development team
8. Business Analyst – seek complete definition of business processes
9. Tester/QA Coordinators
10. IT managers: Coordinates cross team dependencies
11. DEVOPS: Responsible for
    a. The compliance of the devops model
    b. Define and optimize factory processes.

    Enabler for:

    TECH LEADS:

    a. : Enforce a valid application architecture
    b. Enforce Best practices
    c. Enforce realease testing

    TESTERS:

    a. Devise test strategy
    b. Iteration testing

    OPERATIONS:

    a. Deploy to pre-production and production
    b. Monitor and troubleshoot applications

MAINTAINANCE:

a. Execute urgent maintenance task
b. Pre production (hotfixes)

Infrastructure:

a. Provision for developing and running

Architects:

Operations:

a. Manage factory permissions in lifetime
b. Coordinate and execute deployments
c. Coordinate platform upgrades
d. Monitor and troubleshoot problems
e. Perform environment Configs

PROJECT TIMEBOX:

NO AGILE PRACTICE IS LIKE ONE SIZE FITS ALL.

The methodology can be customized or optimized.

OUTSYSTEMS AGILE METHODOLOGY:

1. Timebox
2. Collaboration
3. Iterations
4. User feedback is gold
5. Transparency
6. Visibility
7. Flexibility

Timebox:

INITIATION →	ITERATION DEVELOPMENT →	SOLUTION RELEASE →	POST PRODUCTION
1-2 WEEKS	2-3 WEEKS	1-4 WEEKS	1 WEEK

Initiation: vision document, app skeleton,

Iteration Development: SHAPE-BUILD-ACCEPT. Shape next iteration, deliver current iteration, validation of previous iteration

Solution Release: Go live at phase end

Post production: MONITORING, TUNING. FEEDBACK.

Outsystem Platform server:

The core of outsystems.

It generates, optimizes, compiles, and deploys applications

Has 3 services:

1. Code Generator
2. Deployment Services
3. Application Services

Generator: generates code for all layers like sql, backend, html n javascript.

Triggered by pressing 1 click publish button

SERVICES:

Code generator: optimizes the application

Adds logging and monitoring capabilities

Runs scripts to update database before deploying application

Deployment services: deploys thr generated application to the application server of every front end server

Scheduler Service: It allows having ASYNCHRONOUS TASKS SUCH AS:

a. Timers
b. Emails
c. Business Process Activities

Scheduler keeps checking for the next pending job and executes them

It used JOBS MESSAGE QUEUE for task management.

Scehduler fetches the jobs according to their schedule and triggers their execution in a front end

Log service: Outsystem apps automatically generate error and performance logs, which can be complemented by custom logs.

LOGGING RUNS ASYNCHRONOUSLY.

OUTSYSTEMS HAS 4 ENVIRONMENTS:

1. DEVELOPMENT
2. QUALITY ASSURANCE
3. PRODUCTION
4. LIFETIME