Capgemini

OUTSYSTEMS WEB

04/2024

Outsystems

OVERVIEW

Outsystems Web

OUTSYSTEMS WEB

Introduction

- Web Development in Outsystems
- Web Apps in Outsystems
- Best Practices - Data
- Best Practices - Queries
- Best Practices - UI
- Best Practices – Apps & Modules

Outsystems

WEB DEVELOPMENT IN OUTSYSTEMS

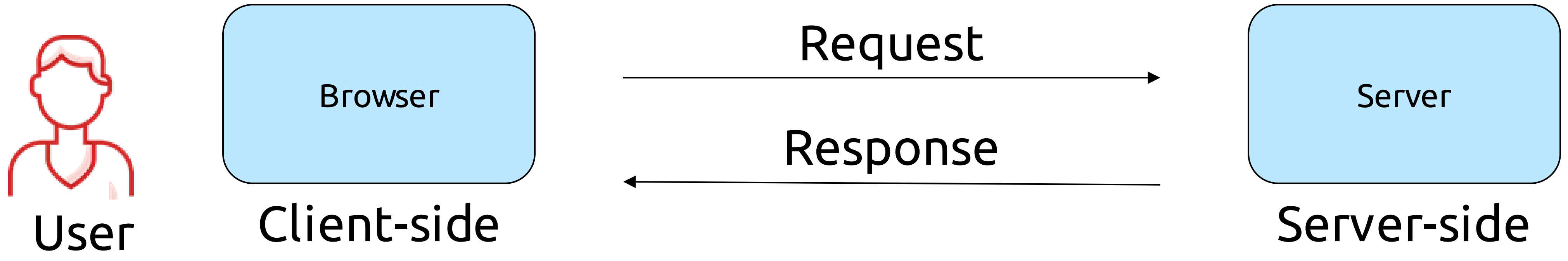
Outsystems Web

OUTSYSTEMS WEB

User Interaction

The user opens a **browser**:

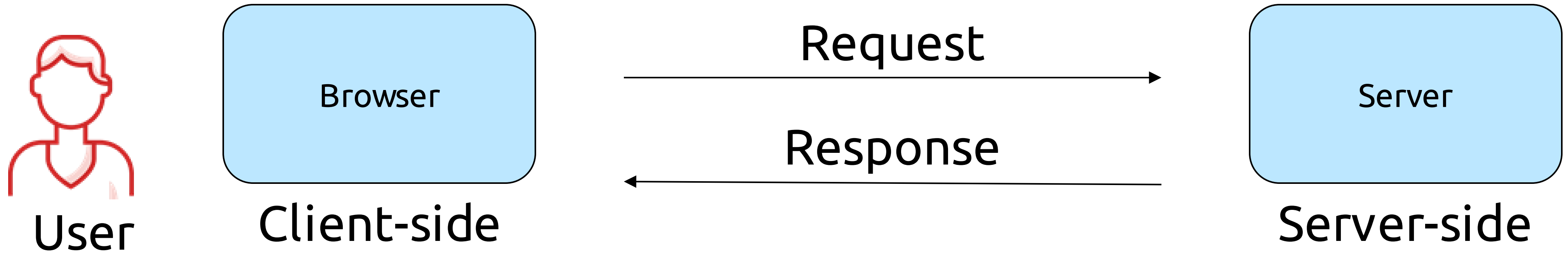
1. Types an URL in the address bar or clicks a link in a web page
2. The browser **(client-side)** sends a **request** to the server
3. The server **(server-side)** returns a **response** back to the browser
4. Browser receives the response and renders the web page to the user



OUTSYSTEMS WEB

Web requests & responses

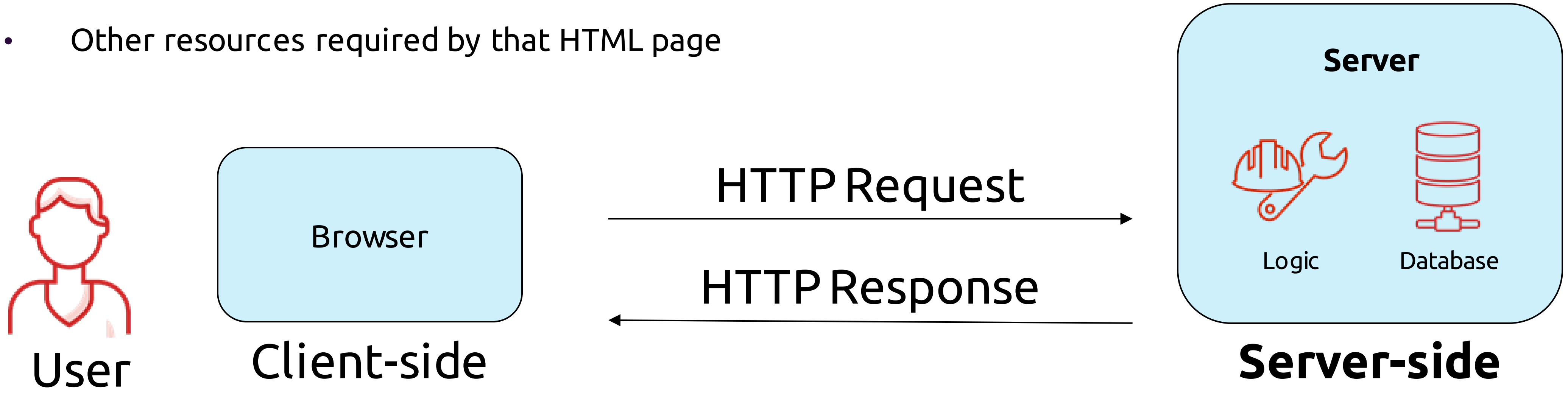
- Browsers and web servers communicate using the **HTTP protocol**
- An **HTTP request** is sent when users interact with a web page
  - E.g., click a link on a web page, submit a form, a search on a web page
- The server waits for requests and sends back an **HTTP response** to the browser
- Successful HTTP responses contain the requested resource (e.g. HTML page)



OUTSYSTEMS WEB

Server Side

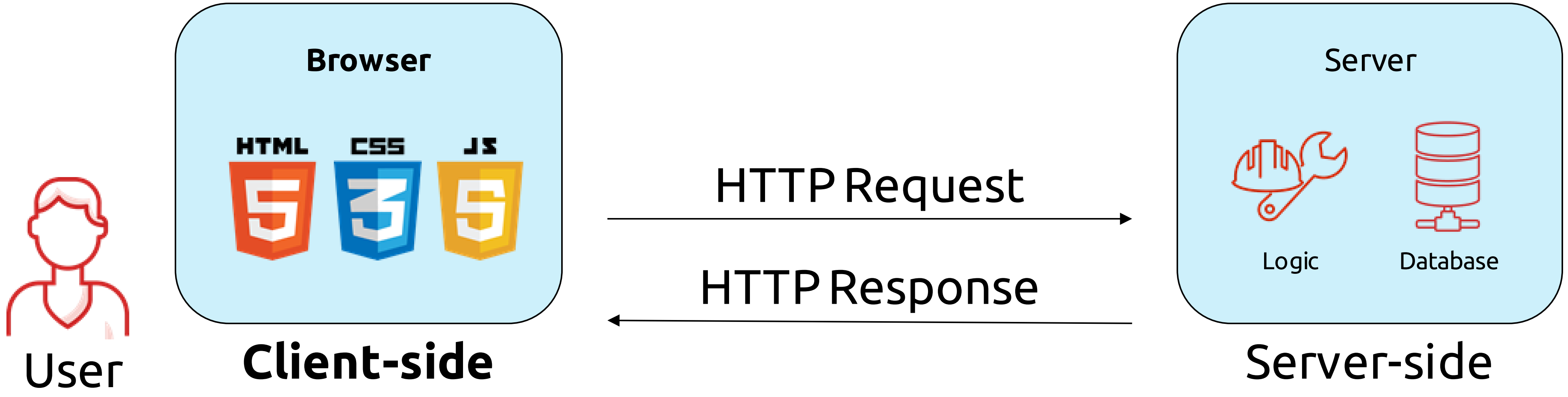
- Listens for incoming requests
- Processes requests
  - Retrieves needed data and stores relevant info
  - Controls access to data and customizes responses
- Sends response back to the browser
  - Dynamically built HTML page
  - Other resources required by that HTML page



OUTSYSTEMS WEB

Client Side

- Client-side code runs in a browser
- Browser renders the response to the end-user
  - Basic web page components (HTML)
  - Styling of those components (CSS)
  - Behavior and interactivity of those components (JS)



Outsystems

WEB APPS IN OUTSYSTEMS

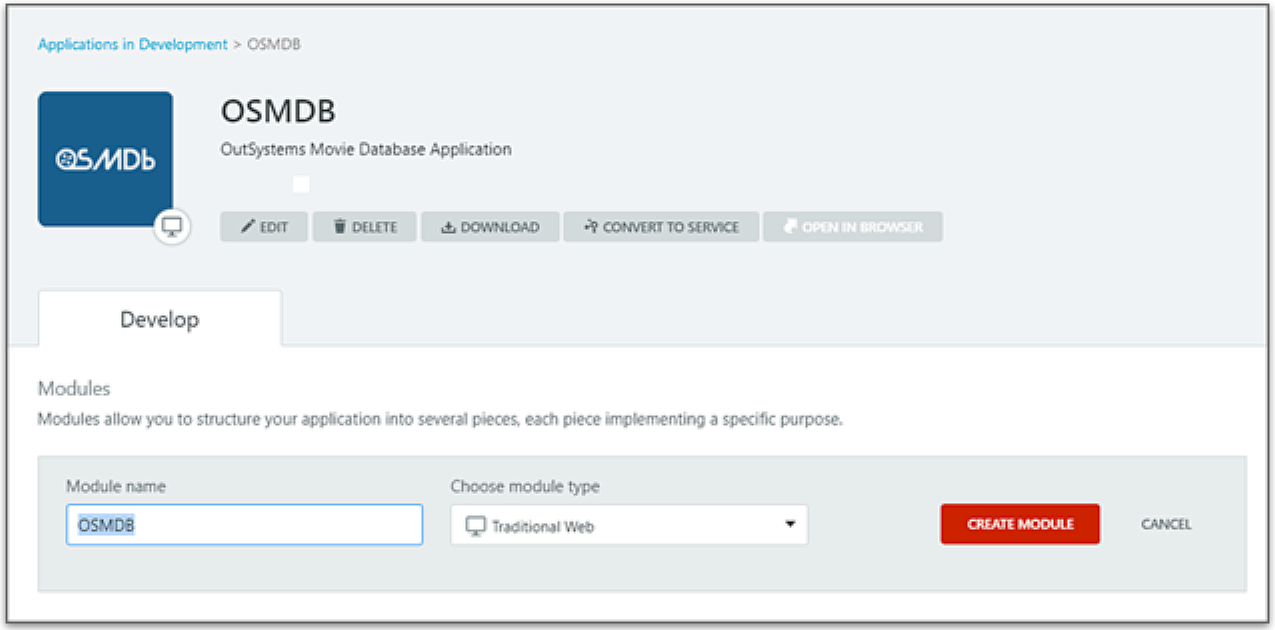
Outsystems Web

OUTSYSTEMS WEB

Web Applications – Module Types

- Applications should have at least one module

- Web application modules can be:
  - Traditional Web
  - Blank
  - Service
  - Extension


Applications in Development > OSMDB

OSMDB

OutSystems Movie Database Application

Modules

Modules allow you to structure your application into several pieces, each piece implementing a specific purpose.

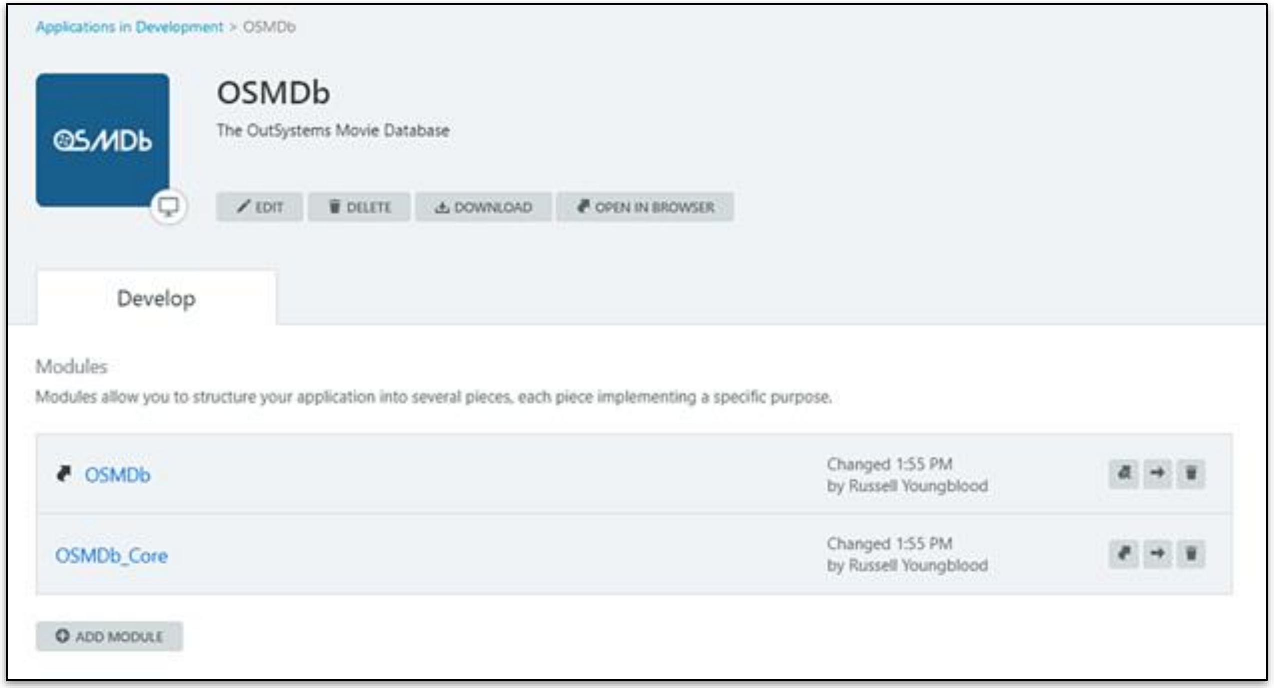
Module name: OSMDB

Choose module type: Traditional Web

OUTSYSTEMS WEB

Modules

- Applications can have one or more modules
- Modules are where developers:
  - Create the data model
  - Define business logic
  - Build web pages



Applications in Development > OSMDb

OSMDb

The OutSystems Movie Database

EDIT | DELETE | DOWNLOAD | OPEN IN BROWSER

Develop

Modules

Modules allow you to structure your application into several pieces, each piece implementing a specific purpose.

OSMDb	Changed 1:55 PM by Russell Youngblood
OSMDb_Core	Changed 1:55 PM by Russell Youngblood

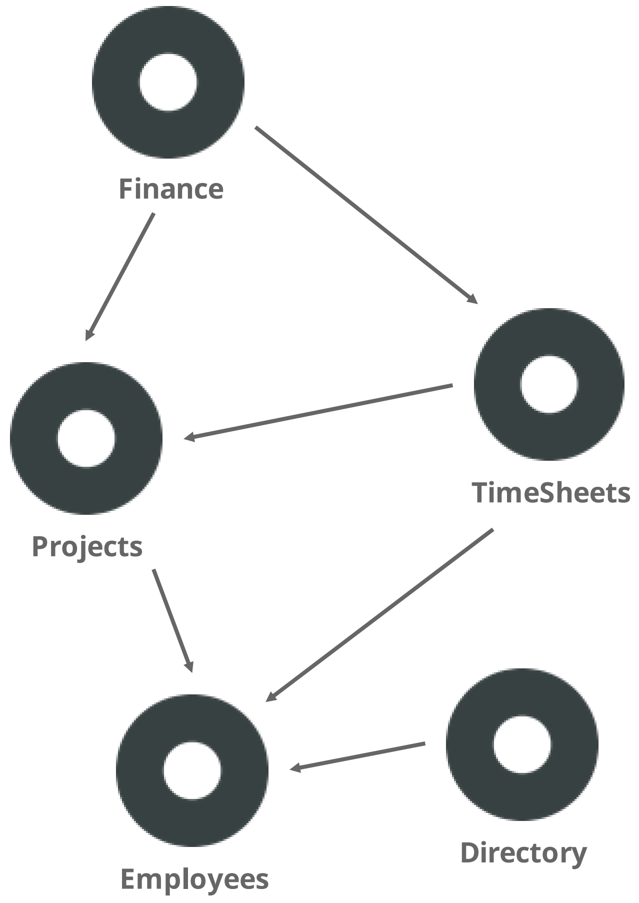
ADD MODULE

OUTSYSTEMS WEB

Modular Programming

Software design technique that allows each module to...

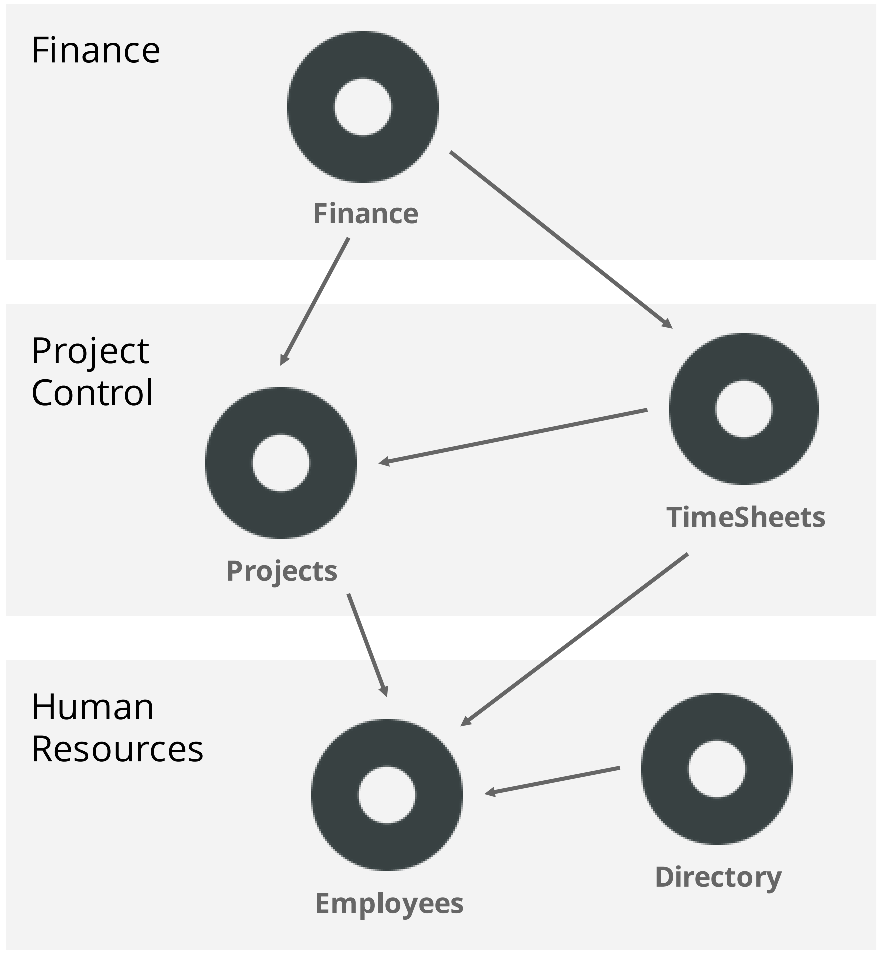
- **Encapsulate** everything necessary to execute **one aspect** of functionality
- Separate functionality by **independent** and (potentially) **replaceable** pieces of code



OUTSYSTEMS WEB

Modular Programming

- An Application groups a set of related modules
- Modules can share elements with other modules
    - Modules that share features are called **Producers**
    - Modules that use features from others are called **Consumers**
- Producers and Consumers can be in different applications



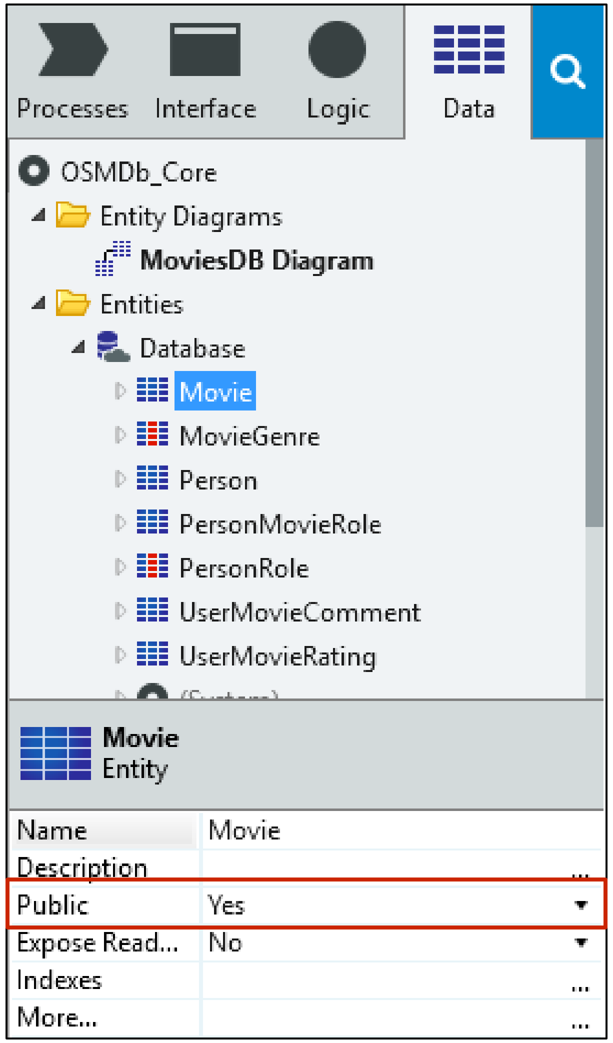
OUTSYSTEMS WEB

Producers

Make features **Public** to share them

The following elements can be made public:

- Data
  - Entities
  - Structures
- Logic
  - Server Actions
  - Roles
- Interface
  - Web Blocks, Web Screens
  - Images
  - Themes
- Processes
  - (Business) Processes

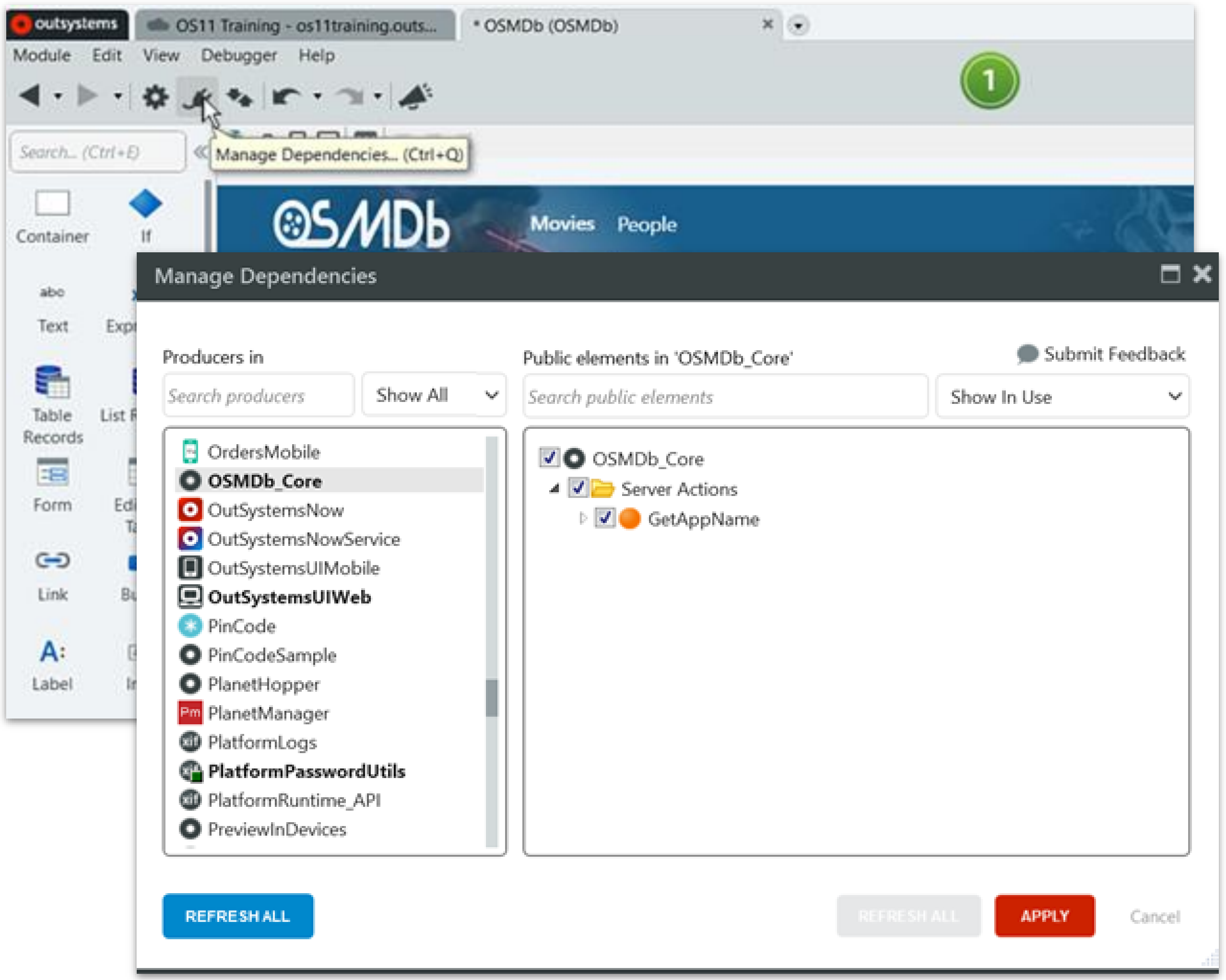


Name	Movie
Description	...
Public	Yes
Expose Read...	No
Indexes	...
More...	...

OUTSYSTEMS WEB

Consumers

- Reuse public elements from producers
- Manage Dependencies
  - Modules with public features are displayed
  - Select the desired elements to be able to use them (consume)
- Selected elements become available in the module



Outsystems

BEST PRACTICES - DATA

Outsystems Web

OUTSYSTEMS WEB

Best Practices - Data

Problem

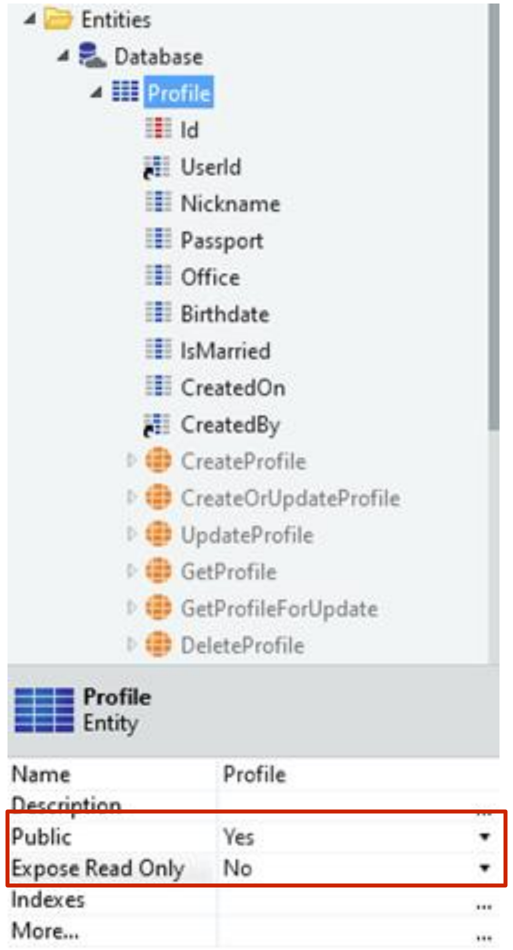
- Business rules for public entity modification scattered in all modules that use the entity are hard to enforce

Solution

- Make the entity not Public or mark it as Read Only
- Create Server Actions to "Wrap" CRUD operations

OUTSYSTEMS WEB

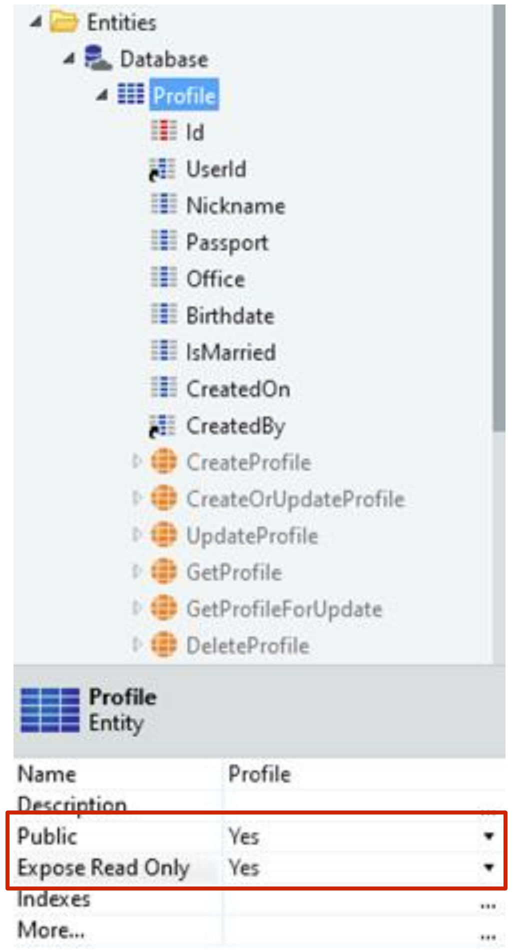
Best Practices – Data – Entity Wrappers



Entities
Database
Profile
Id
UserId
Nickname
Passport
Office
Birthdate
IsMarried
CreatedOn
CreatedBy
CreateProfile
CreateOrUpdateProfile
UpdateProfile
GetProfile
GetProfileForUpdate
DeleteProfile

Profile
Entity

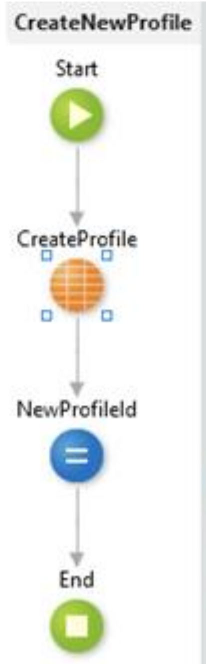
Name	Profile
Description	
Public	Yes
Expose Read Only	No
Indexes	
More...	

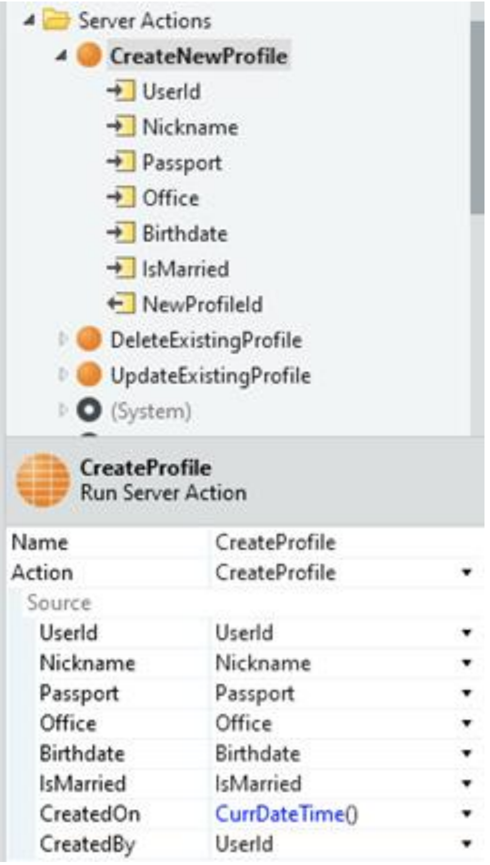


Entities
Database
Profile
Id
UserId
Nickname
Passport
Office
Birthdate
IsMarried
CreatedOn
CreatedBy
CreateProfile
CreateOrUpdateProfile
UpdateProfile
GetProfile
GetProfileForUpdate
DeleteProfile

Profile
Entity

Name	Profile
Description	
Public	Yes
Expose Read Only	Yes
Indexes	
More...	





Server Actions
CreateNewProfile
UserId
Nickname
Passport
Office
Birthdate
IsMarried
NewProfileId
DeleteExistingProfile
UpdateExistingProfile
(System)

CreateProfile
Run Server Action

Name	CreateProfile
Action	CreateProfile
Source	
UserId	UserId
Nickname	Nickname
Passport	Passport
Office	Office
Birthdate	Birthdate
IsMarried	IsMarried
CreatedOn	CurrDateTime()
CreatedBy	UserId

OUTSYSTEMS WEB

Best Practices - Data

Pros

● Common Actions for all modules / applications

- Its use is mandatory

● Centralized Validation and Security Checking

● Default values/Standard Fields (“CreatedOn”, “UpdatedBy”, etc)

● Pre- and post-processing (cascade deletes, clean ups, etc)

Cons

● Entity Scaffolding will not work

- Use Screen Templates + Replace Data

● Changing wrapper logic will impact all consumers

- Rethinking Data Model can help
- Create a set of specialized wrappers

OUTSYSTEMS WEB

Best Practices – Site Properties

Problem

- Performance degrades when updating Site Properties often
  - Module and all Consumers will have their caches invalidated on all front-ends

Solution

- Use Site Properties for application global configuration only (almost constants)
  - Same configuration for all users
  - No need to update them often
- Use Entities for other configurations

OUTSYSTEMS WEB

Best Practices – Complex Data Type Session Variables

Problem

- Performance degrades when using Session Variables with complex data type
  - Database overhead

Solution

- Avoid using Session Variables with complex data types
- Implement your own Session Variables management system using Entities

OUTSYSTEMS WEB

Best Practices – Large Attributes

Problem

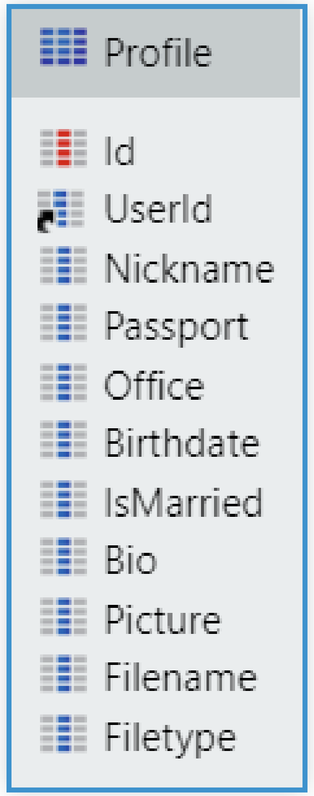
- Updating Binary Data and large Text attributes is slower
  - Updating entity records with large Text or Binary Data attributes is also slow

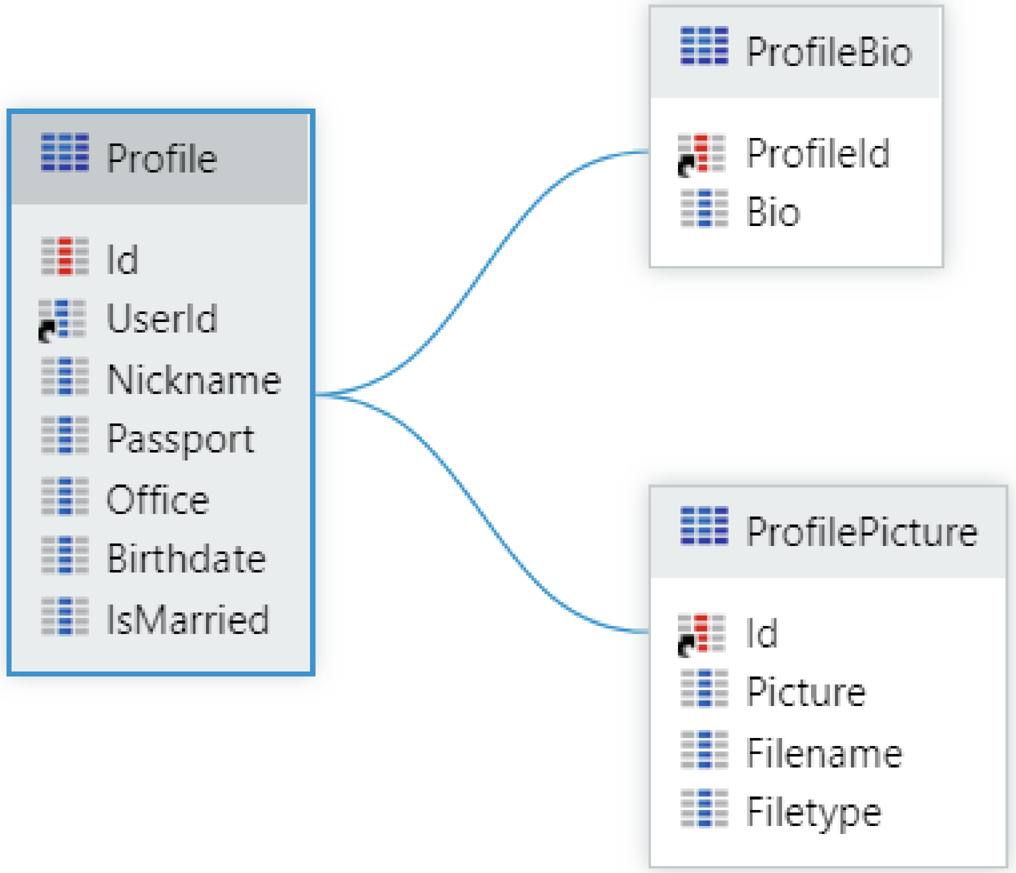
Solution

- Avoid Text attributes with length 2000+
- Isolate large Text & Binary Data attributes in separate Entities

OUTSYSTEMS WEB

Best Practices - Data





Outsystems

BEST PRACTICES - QUERIES

Outsystems Web

OUTSYSTEMS WEB

Best Practices – Record Counting

Problem

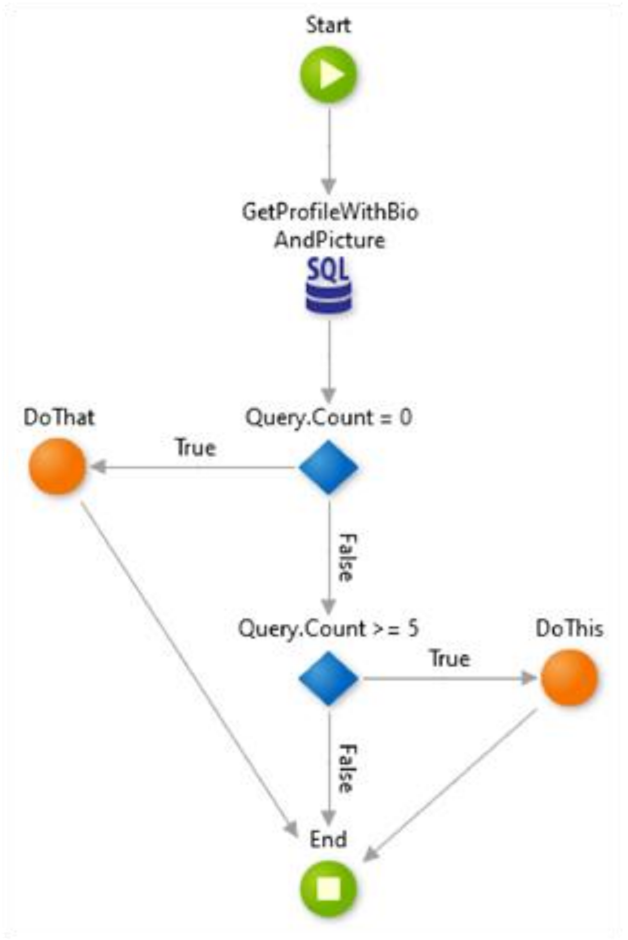
- Performance degrades when using the **.Count** property of Aggregate/SQL query
  - When **.Count** is used, a second query is executed to count the records

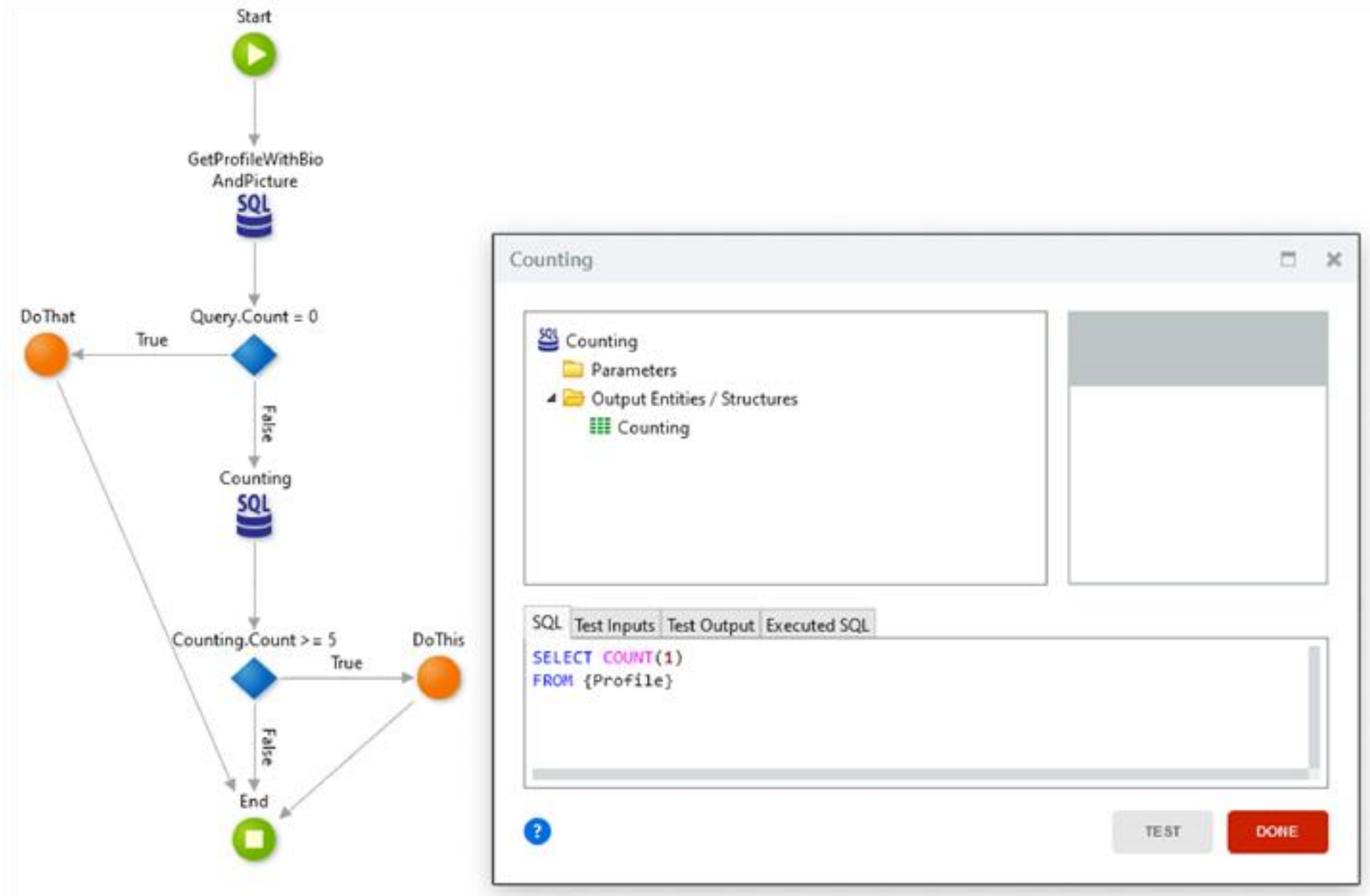
Solution

- Create a separated simpler version of the query to count the records
  - No attributes, no restrictive joins, no sort
- When testing to see if the output list is not empty, use the **.List.Empty** instead

OUTSYSTEMS WEB

Best Practices – Record Counting





```
SELECT COUNT(1)
FROM {Profile}
```

OUTSYSTEMS WEB

Best Practices – SQL Injection (Expand Inline)

Problem

- User information must be used as part of a SQL through inline parameters, creating a security flaw
  - Using inline parameters coming from UI

Solution

- Use the **EncodeSQL()** function to escape the string literals passed by the user
  - Enclose it in " ' " (e.g. " ' " + EncodeSql(Variable) + " ' ")
- Use the following functions (Sanitization extension) to prepare the IN Clause parameters
  - **BuildSafe_InClauseIntegerList** and **BuildSafe_InClauseTextList**

OUTSYSTEMS WEB

Best Practices – Indexes

Problem

- Slow queries with filters and join conditions

Solution

- Create Indexes for the most commonly used attributes
- Find best candidates for indexing
  - Identify attributes used in query/join conditions in slow queries
  - Favor sparser attribute values (Date & small Text vs. Boolean)
  - Can be groups of attributes commonly queried together
- Also, rewrite aggregates as highly optimized sql queries

OUTSYSTEMS WEB

Best Practices – Slow Queries When Fetching Large Records

Problem

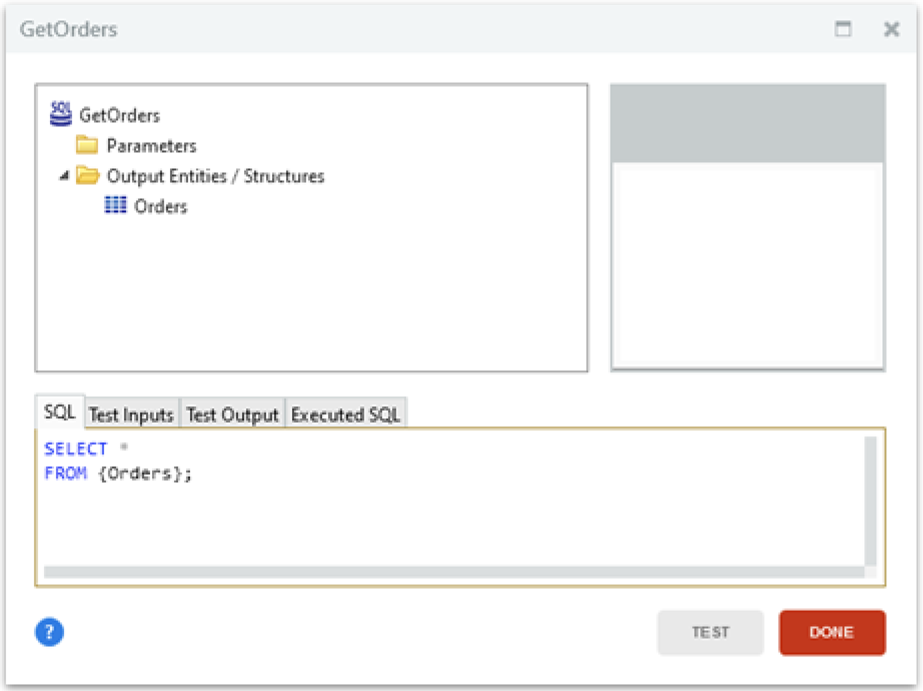
- Fetching entire entity records with large number of attributes when only a small subset is needed

Solution

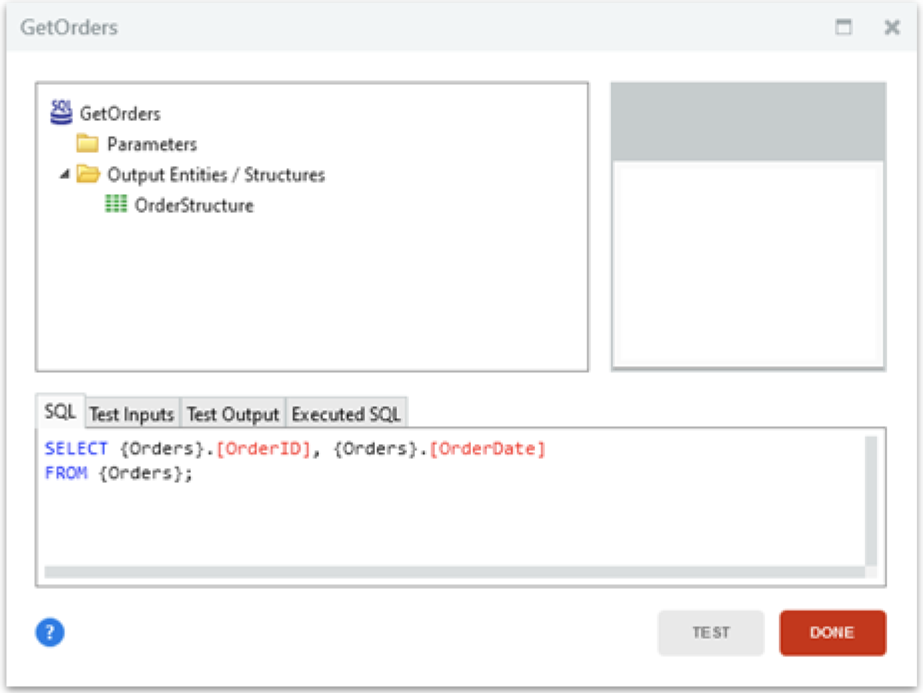
- Only fetch relevant attributes from the database
  - Platform optimizes, except when dealing with actions that receive or output records or lists of records with large numbers of attributes
  - Consider using structures with relevant attributes instead of entities

OUTSYSTEMS WEB

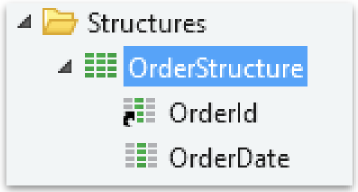
Best Practices – Slow Queries When Fetching Large Records



```
SELECT *
FROM {Orders};
```



```
SELECT {Orders}.[OrderID], {Orders}.[OrderDate]
FROM {Orders};
```



OUTSYSTEMS WEB

Best Practices – Slow Execution of Bulk Entity Operations

Problem

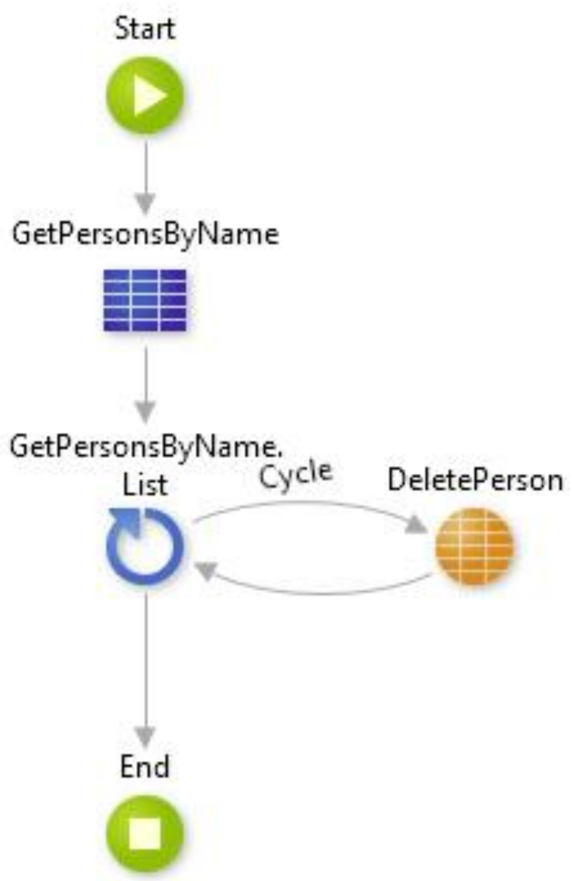
- For Each loops, with Aggregates and Entity Actions that update the Entities, are very slow and inefficient

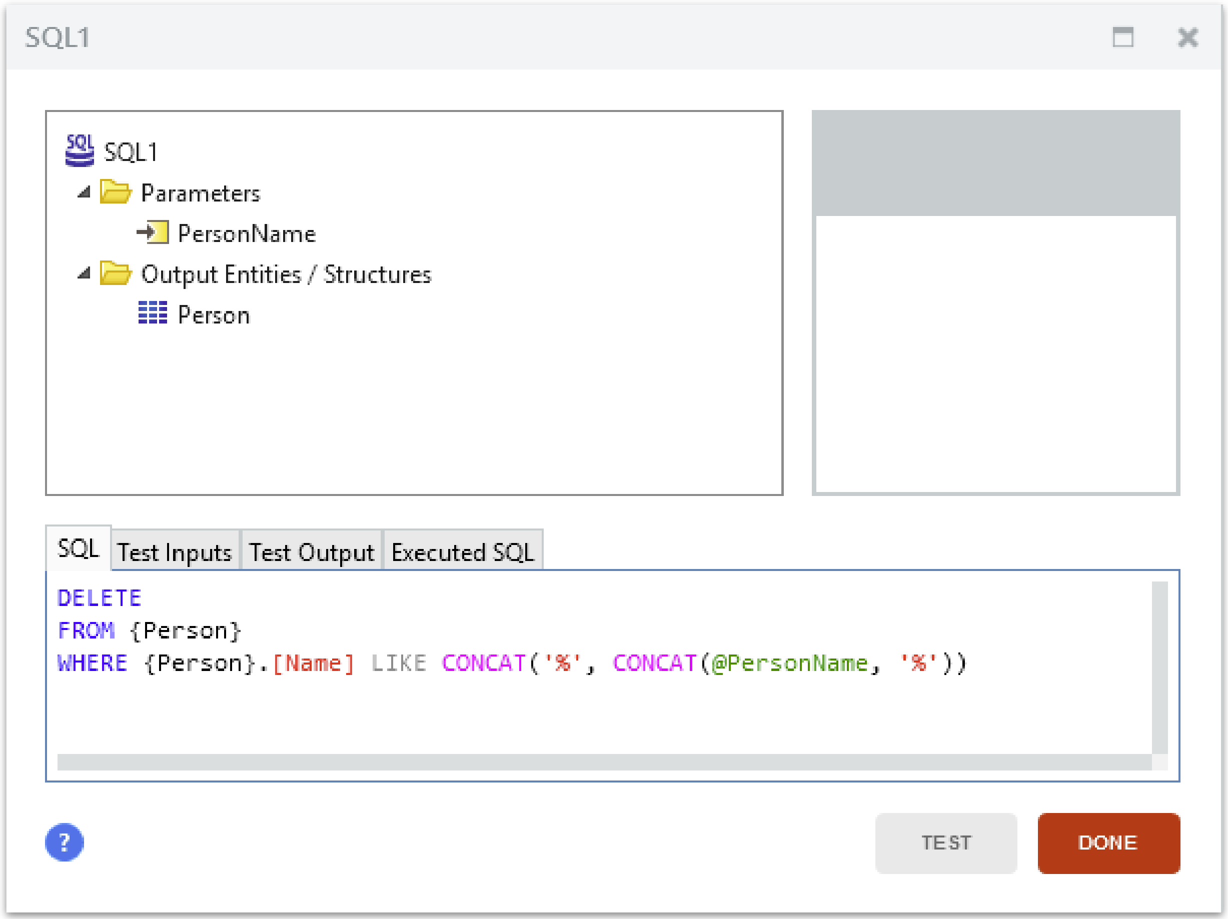
Solution

- If you need to update only some attributes
  - use a SQL query with an UPDATE statement for required attributes instead of Update<Entity>
- If you need to delete multiple records
  - use a SQL query with a single DELETE statement instead of a For Each and Delete<Entity>

OUTSYSTEMS WEB

Best Practices – Slow Execution of Bulk Entity Operations





```
DELETE
FROM {Person}
WHERE {Person}.[Name] LIKE CONCAT('%', CONCAT(@PersonName, '%'))
```

OUTSYSTEMS WEB

Best Practices – Preparation (Traditional)

Problem

- Long page loads have a big impact on the user experience

Solution

- Simplify Screen Preparations
  - Avoid queries inside cycles
    - better have a more complex one
  - Avoid complex security permissions per page
    - do it on session start
  - Avoid complex pre-processing of query data
    - do it on writes
  - Avoid write operations

OUTSYSTEMS WEB

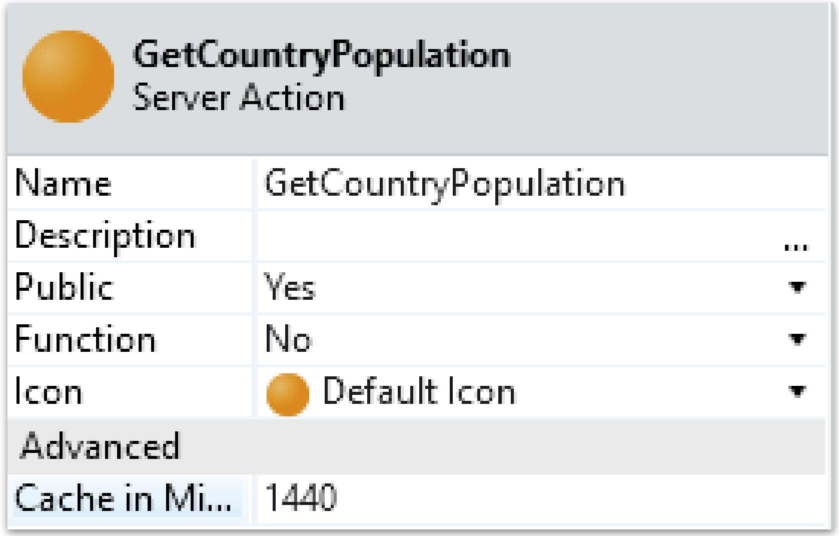
Best Practices – Caching

Problem

- High number of hits on the same server action (e.g. called inside a **ForEach** or **TableRecords**)

Solution

- Cache the server action, for the maximum amount of time that will make the value valid



GetCountryPopulation
Server Action

Name	GetCountryPopulation
Description	...
Public	Yes
Function	No
Icon	● Default Icon
Advanced	
Cache in Mi...	1440

OUTSYSTEMS WEB

Best Practices – Isolated Aggregates (Avoid)

Problem

- After isolating reusable Aggregates in an action, where the output of the action is the same as the output of the aggregate, they seem slower
  - Platform optimizes Aggregates based on what attributes are used on screen
  - If it cannot determine those, it will fetch the entire records from the DB

Solution

- Create structure with the needed attributes and return a list of elements of that structure as output

Outsystems

BEST PRACTICES - UI

Outsystems Web

OUTSYSTEMS WEB

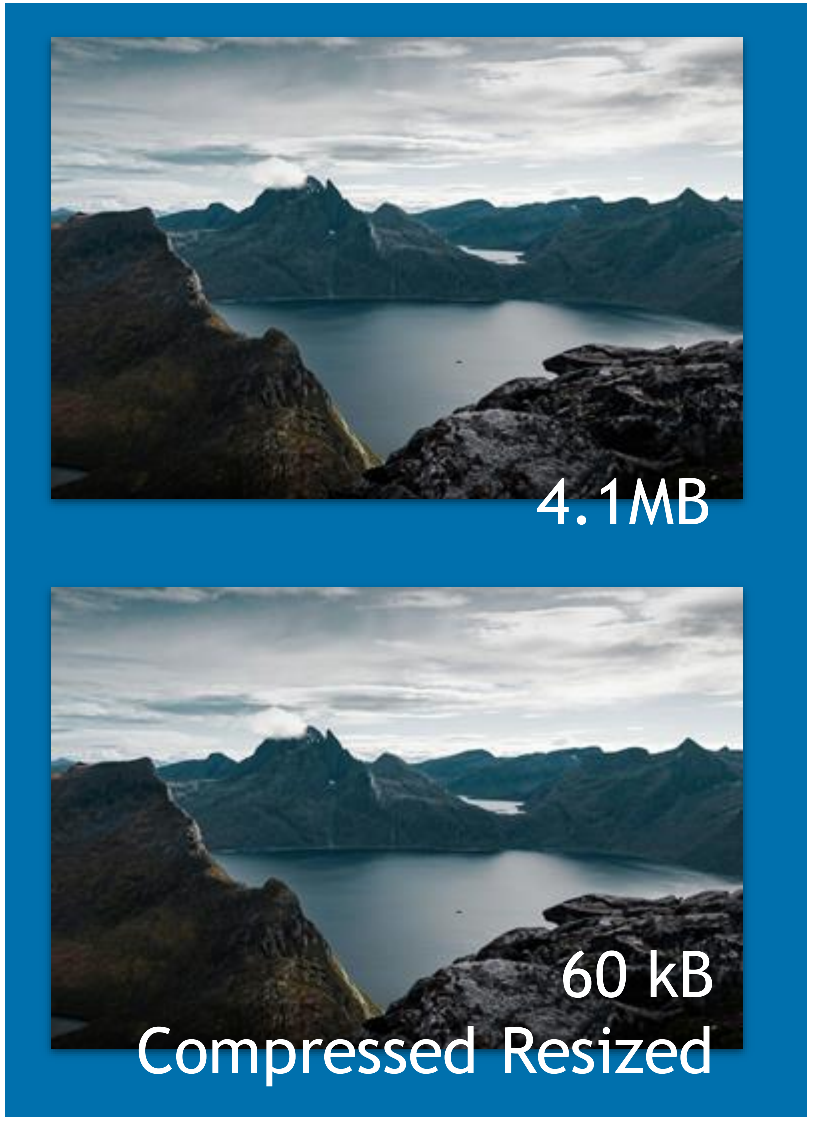
Best Practices – Control Image Sizes

Problem

- First access is too slow on screens with images

Solution

- Compress images



OUTSYSTEMS WEB

Best Practices – Avoid Inline Javascript and Styles

Problem

- Poor user experience during page load due to placing large chunks of JavaScript code or inline CSS in your screen through expressions
  - significantly increasing the screen size
  - no cache possible
  - increases maintenance complexity

Solution

- Avoid using large JavaScript in screen expressions or inline CSS
- Place them in files and include it through extensions or module resources, and place a link script to it
  - Or use a centralized theme for the CSS

OUTSYSTEMS WEB

Best Practices – Validate Permissions

Problem

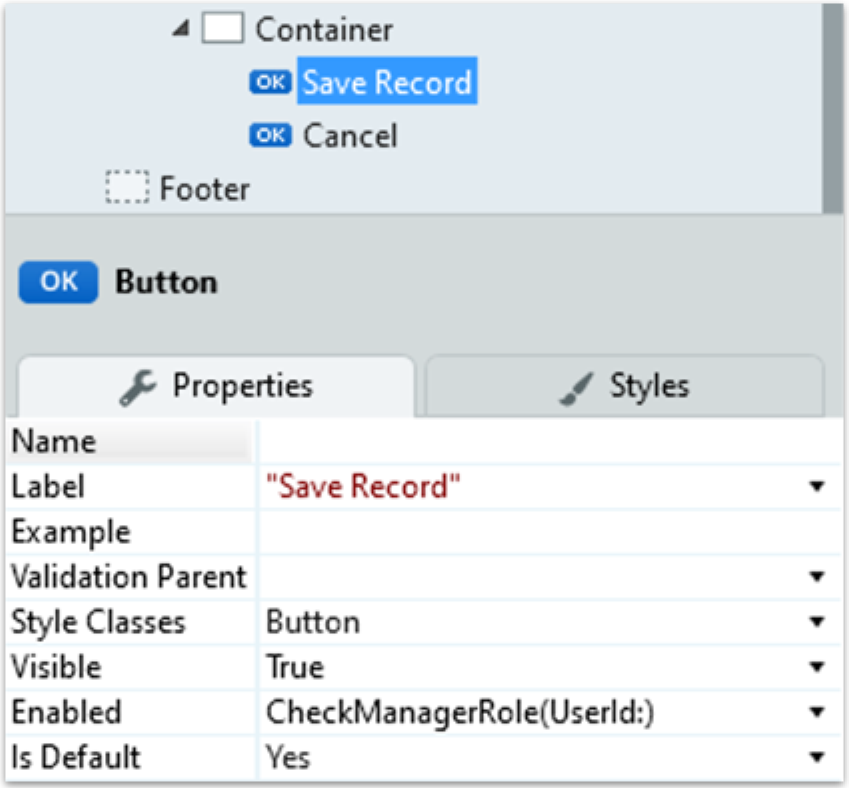
- Relying on UI to control permissions is not safe
- A skilled user will be able to overcome those limitations (e.g. buttons not displayed or disabled)

Solution

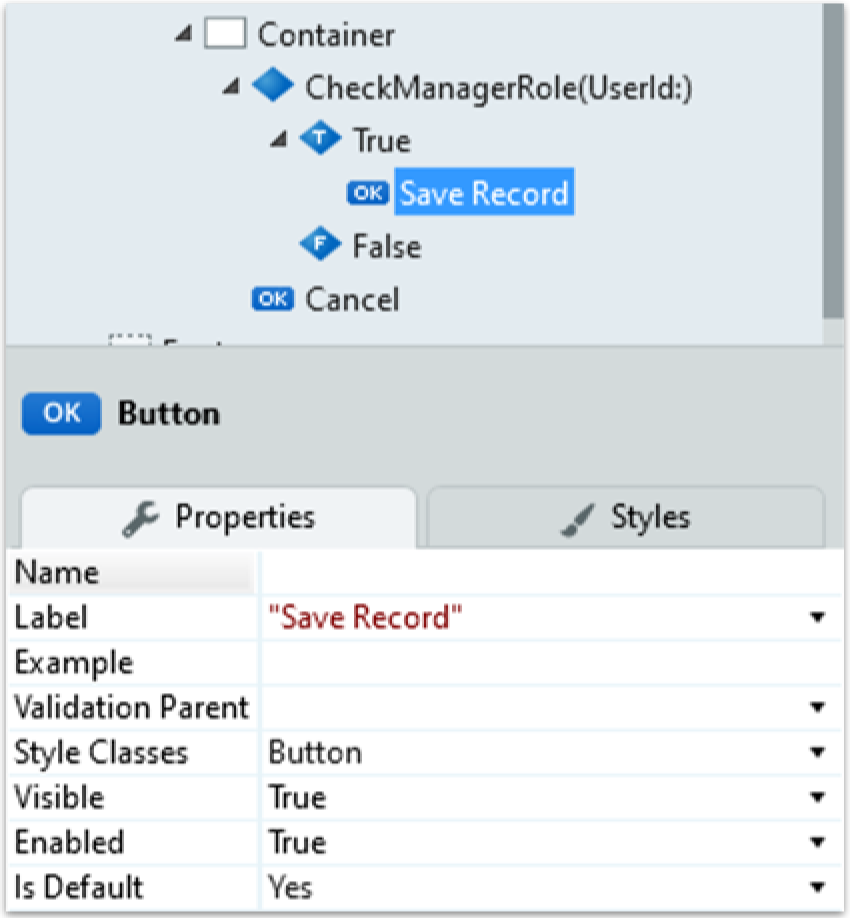
- Use the **Visible** or an ***If* widget** to build or not the widget instead of disabling it
- On every action that requires authorization (e.g. Save action), check the logged in user's roles to guarantee that they have the authorization to perform the action

OUTSYSTEMS WEB

Best Practices – Validate Permissions



Name	
Label	"Save Record"
Example	
Validation Parent	
Style Classes	Button
Visible	True
Enabled	CheckManagerRole(UserId:)
Is Default	Yes



Name	
Label	"Save Record"
Example	
Validation Parent	
Style Classes	Button
Visible	True
Enabled	True
Is Default	Yes

OUTSYSTEMS WEB

Best Practices – UI Code Injection (Escape Content in Expressions)

Problem

- Writing custom HTML can make it susceptible to HTML / Javascript injection
  - Also increases maintenance complexity

Solution

- Avoid writing custom HTML using unescaped expressions
- Enclose the screen user inputs and variables with the EncodeJavascript(), EncodeHTML() or SanitizeHtml() functions
- Prefer to write/use components that implement the required feature
  - Better maintainability

OUTSYSTEMS WEB

Best Practices – Refactor Large Screens

Problem

- Long screen loading time in screens that have a large amount of content
  - Long screen processing time
    - Using many records and record lists
  - Long network transmission time
  - Long HTML rendering page time

Solution

- Refactor large screens
  - Show information as needed
  - Use smaller blocks / different screens

OUTSYSTEMS WEB

Best Practices – Minimize Screen Parameters

Problem

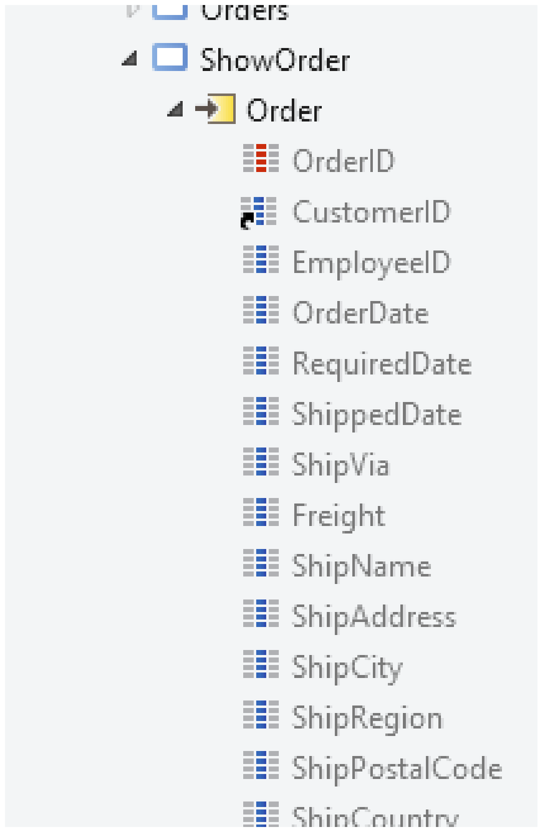
- General slowness in overall user experience due to a large amount of data being passed through screen input parameters

Solution

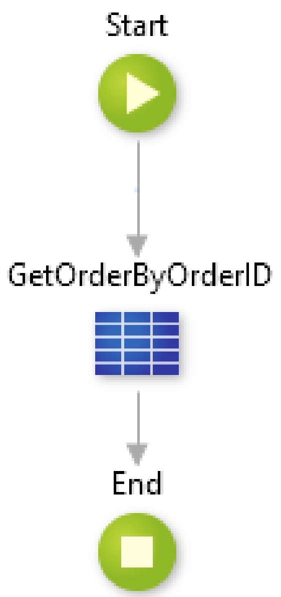
- Avoid passing lots of information using screen parameters
- Try to pass only entity identifiers and user inputs between screens to improve performance

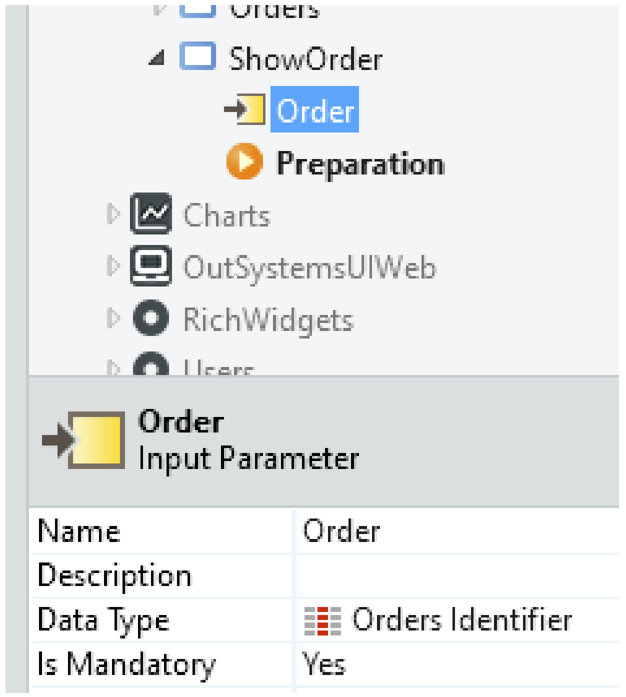
OUTSYSTEMS WEB

Best Practices – Minimize Screen Parameters



Orders
ShowOrder
Order
OrderID
CustomerID
EmployeeID
OrderDate
RequiredDate
ShippedDate
ShipVia
Freight
ShipName
ShipAddress
ShipCity
ShipRegion
ShipPostalCode
ShipCountry





Orders
ShowOrder
Order
Preparation
Charts
OutSystemsUIWeb
RichWidgets
Users

Order
Input Parameter

Name	Order
Description	
Data Type	Orders Identifier
Is Mandatory	Yes

Outsystems

BEST PRACTICES – APPS & MODULES

Outsystems Web

OUTSYSTEMS WEB

Best Practices – Module Naming

Problem

- When listing modules on an application, it is hard to identify on which layer of the Architecture Canvas they are located
    - It also makes it difficult to use tools like Discovery to evaluate architecture

Solution

- Adopt a module naming convention that reflects the position and type of module (e.g., *_CS for core services, *_Th for theme modules, etc)

OUTSYSTEMS WEB

Best Practices – Isolate Large Image and Resources

Problem

- Large images and resources impact download or publishing of modules
  - Images and resources are stored in the Module, increasing its size

Solution

- Move large images and resources to different module(s) to avoid increasing the main module's size
  - Publish is faster

OUTSYSTEMS WEB

Best Practices – Missing Descriptions in Public Elements

Problem

- Development and maintenance complexity increases when public elements do not have appropriate descriptions

Solution

- Set appropriate descriptions on public elements to make easier the task of identifying and understanding the purpose and utility of these elements
  - E.g., Blocks and Server actions
  - Don't forget input and output parameters
  - Also useful for OutDoc

Outsystems

LINKS & RESOURCES

Outsystems Web

OUTSYSTEMS WEB

Useful Links

Best Practices

https://success.outsystems.com/Documentation/Best_Practices/Injection_and_Cross_Site_Script_(XSS)

https://success.outsystems.com/Documentation/Best_Practices/Building_dynamic_SQL_statements_the_right_way

https://success.outsystems.com/Documentation/11/Reference/OutSystems_APIs/Sanitization_API#BuildSafe_InClauseIntegerList

https://www.outsystems.com/forums/discussion/36202/use-case-of-encodesql/

About Capgemini

Ricardo Machado

ricardo-manuel.machado@capgemini.com

Tech Lead / Scrum Master / Outsystems Trainer
Capgemini Portugal

Capgemini

GET THE
FUTURE
YOU WANT

capgemini.com