outsystems

Role-based Security

Topics

- Access Control
- Users and Roles
  - Checking Permissions
  - Client-side Role-based Authorization
- Endpoints

Access Control

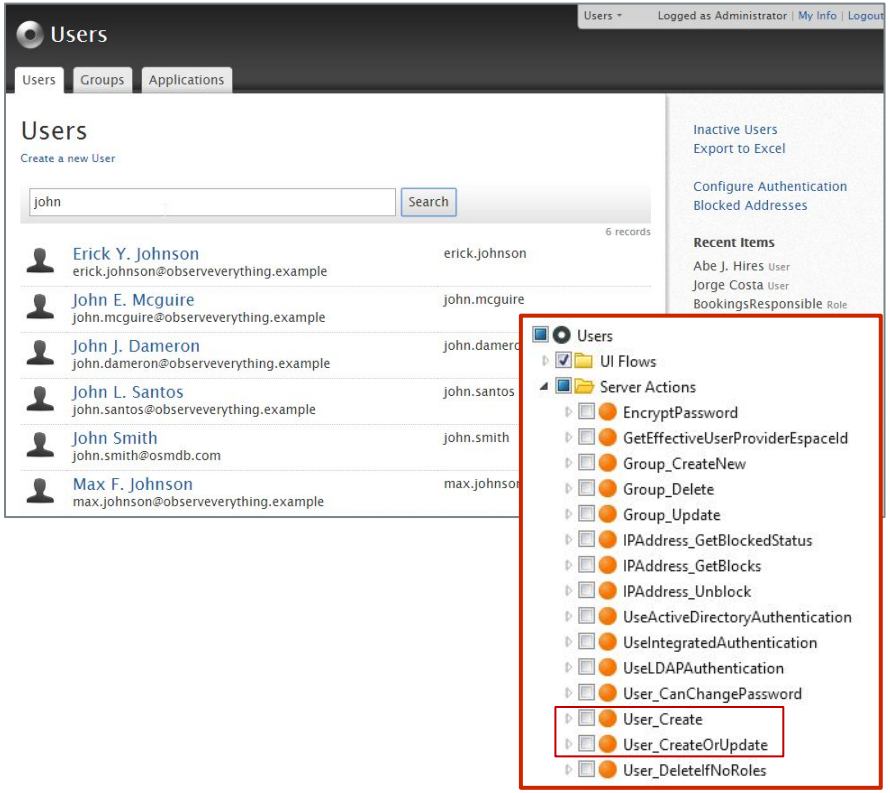
- Access control determines and enforces **who can do what**
  - **Who**: Business Users and Roles
  - **What**: Access Screens, trigger Actions, etc.

- It comprises **Authentication** …
  - Identify who is accessing the application
  - Validate user credentials (e.g. username and password)

- … and **Authorization**
  - Check if a user can execute a task
  - Validate if user was granted the required permissions

Users

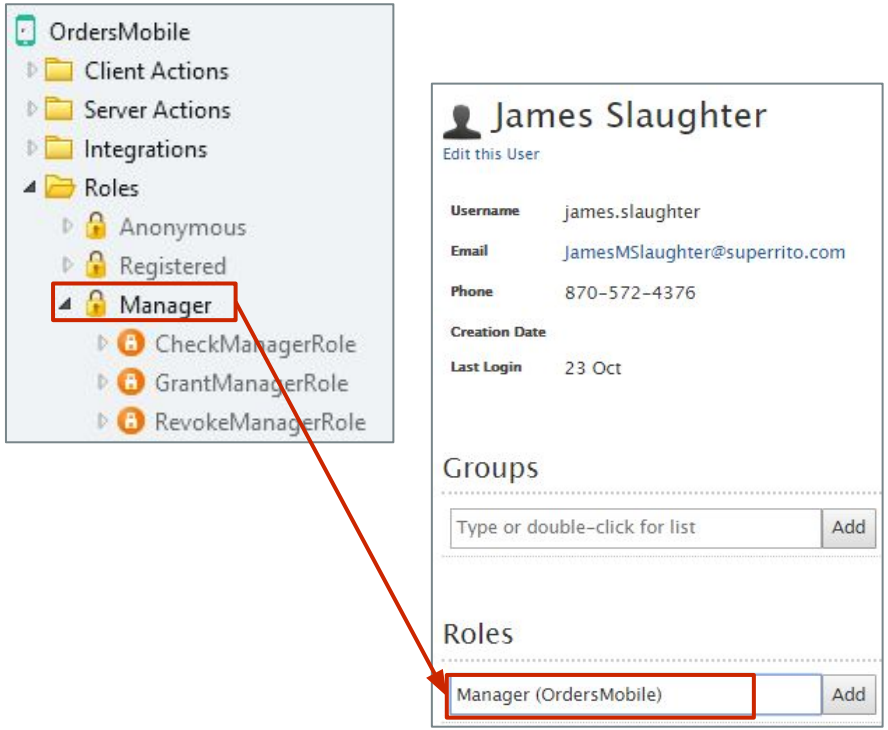
- End-users are created and managed in the built-in *Users* application
  - Default OutSystems end-user provider
- The core information of a user is
  - Username
  - Password
  - Name
- Can be created programmatically
  - Actions from Users module
- Authorization is granted via **Roles**



outsystems

Roles

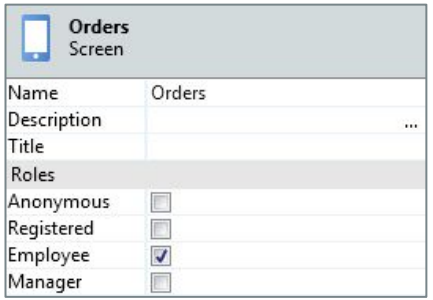
- Anonymous and Registered built-in Roles
- All end-users have the **Registered** Role
- Application-specific Roles can be created
  - Each Role has **Check**, **Grant** and **Revoke** Actions
  - A Role can be set as Public
- Authorization can be managed
  - Programatically
  - Manually in the Users application

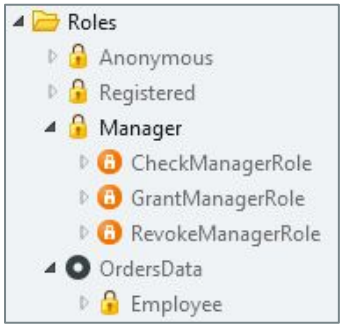


outsystems

Checking Permissions

- Roles can be checked in Screens and Actions
- Screen Roles property
  - Enforces a user to have one of the Roles to access the Screen
- *Check<RoleName>Role(UserId)* Action
  - Checks if a user has the *RoleName* Role
  - Server Action
  - Can be used in Boolean conditions
    - If statement on Server Action logic flows

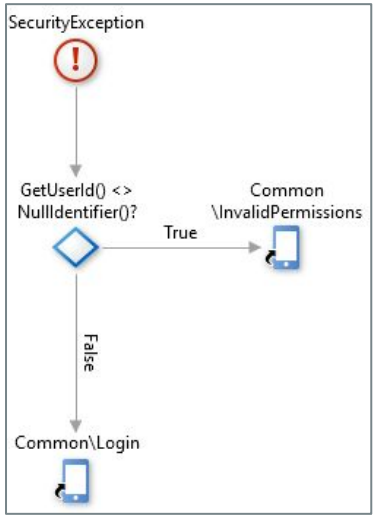




outsystems

Client-side Role-based Authorization

- User's Roles are stored in local storage at login and logout
  - Could be tampered with in local storage
  - Checks should be for UI purposes only
- Role validation is checked upon navigation
  - Access control for UI is enforced without having to go to the server
  - Available to be checked while offline
  - Raises a SecurityException when invalid
- Exceptions on client side can be handled in several ways:
  - Redirect to another Screen (or navigate back???)
  - Remain in the original Screen (do nothing)



outsystems

Endpoints

- Server endpoints are generated for:
  - Screen Aggregates (data)
  - Server Actions used by Client Actions (logic)
- Access control is enforced by the server
- Endpoints are exposed as REST APIs
  - Difficult to discover / enumerate how many endpoints there are
  - No single place where the information is aggregated and easy to visualize
- Flowless Actions are hard to secure with business logic
  - Screen Aggregates, Entity Actions, integrations can be used in client flows
  - No place for authorization logic on these elements

Summary

- Access Control
- Users and Roles
  - Checking Permissions
  - Client-side Role-based Authorization
- Endpoints


Role-based Security
Thank You!