Mendix 7

- Smart Apps - Make your apps intelligent, proactive, and contextual
- Mendix Connector Kit - Defining integrations is now a breeze
- Mendix Web Modeler - Experience the value of collaborative development
- Native Mobile Experience – Anywhere, Always On
- Mendix Application Test Suite - Embed testing in your application life-cycle
- Mendix Application Performance Monitor - Get a grip on performance early on
Mendix 7

- Resilient and scalable Apps – Stateless Mendix Runtime
- Smart Apps - Make your apps intelligent, proactive and contextual
- Mendix Connector Kit - Defining integrations is now a breeze
- Mendix Web Modeler - Experience the value of collaborative development
- Native Mobile Experience - Anywhere. Always On
- Mendix Application Test Suite - Embed testing in your application lifecycle
- Mendix Application Performance Monitor - Get a grip on performance from early on
Mendix 7

- Resilient and scalable Apps – Stateless Mendix Runtime
  - Why a stateless runtime?
  - How does it work?
  - Best practices
- Migration to Mendix 7
Mendix 7

- Resilient and scalable Apps – Stateless Mendix Runtime
  - Why a stateless runtime?
  - How does it work?
  - Best practices
- Migration to Mendix 7
- REST consume
Why a Stateless Runtime?
Mendix 6 – Shared State Database

- Runtime state shared in a state database: Redis

- Pros
  - Transparent to client

- Cons
  - Does not scale to large amounts of objects in session
  - Complex garbage collection
  - Complex infrastructure
Mendix 7 - Stateless runtime
Mendix 7 - Stateless Runtime

- State is stored in client:
  - Objects in JavaScript memory
  - Garbage collected as soon as possible
  - Protected against changes of read-only data
  - Server communication optimized:
    - Minimal roundtrips
    - Minimal dataset needed by server
  - Model analysis to ensure server requests are as small as possible
How Does This Work?
Important State Concepts

- What will be kept in state by client?
- What will be included in server communication?
- When to create association with session?
State in Mendix 6

Client

Create: MyEntity

MyEntity: id=1

Runtime

State:
• MyEntity
  • id: 1
State in Mendix 6

client

Change: 1; name=abc

Runtime

State:
- MyEntity
  - id: 1
  - name: abc
State in Mendix 6

client

action: doSomething
param: id 1
Refresh: 1

Runtime

State:
- MyEntity
  - id: 1
  - name: def

- MyEntity
  - id: 1
  - name: abc
State in Mendix 6

- Session state stored server side

- Downsides
  - Scaling out (state synchronization)
  - Garbage collection
  - Chatty protocol
  - Runtime Memory usage
State in Mendix 7

State:
- MyEntity
  - id: 1

Create: MyEntity
MyEntity: id=1
State in Mendix 7

State:
- MyEntity
  - id: 1
  - name: abc

Change: object 1:
- name=abc
State in Mendix 7

action: doSomething
param: id 1
objects:
  • MyEntity: id=1
    • name=abc

objects:
  • MyEntity: id=1
    • name=def
State in Mendix 7

- All user session state stored client side

Benefits
- Fewer limitations on server scaling
- Lower memory requirements in Runtime
- Fewer roundtrips
- More efficient garbage collection
- Better insight for developers

Potential side effects
- Larger requests and responses
  - Due to optimizations in Mendix some apps actually have smaller requests!
Mendix 7: Impact on Your Projects

- Request input
  - What client state needs to be included when calling the server

- Reachable network
  - What client state can be accessed from pages
Mendix 7: Impact on Your Projects

- Request input calculated based on the reachable network of:
  - Request parameters (like microflow inputs and associations used in a microflow)
    - Optimizations are disabled for java actions and service calls
  - Current User object
  - Current Session object

- Reachable network is calculated based on objects available on the client
  - Garbage collection (GC) limits the reachable network
  - Static analysis of the model is used to determine server side data usage
  - GC keeps all reachable …
    - … NPE objects from current user, session and subscribed objects
    - … changed objects from current user, session and subscribed objects
Best Practices
Best practice

- Minimize the number of in-use objects in your session
- Commit or roll back all changes to persistable objects before the end of the main microflow
- Link non-persistable objects that have long life spans to the current Session object
- Map only those parts of a web service integration that are necessary
- Delete any non-persistable objects as soon as they are no longer necessary
- Don’t use non-persistable objects in layouts
Best Practices for App Performance in Mendix 7

Overview

Limiting the Number of Objects

Changed Objects

Workflow Objects

Integrations
Minimize In-Use Objects in Your Session

- In-use (dirty) objects cannot be garbage collected
  - Will increase memory usage of your client
  - May increase request size when calling server
- Consider
  - NPEs pointing to many objects used in layout
    - prevents GC as they stay reachable from subscribed object in layout
  - NPEs pointing to current user object and current session
    - they need to be manually deleted when no longer necessary
Minimize Dirty State at End of Microflow

- Dirty state (new or changed persistent objects) needs to be held by the browser until it’s saved to the database.

- Commit or roll back the changes to persistable objects before the end of the main microflow.

```
[actionResult: null]
  changes: [{'EAN': '123', 'Description': 'A desk', 'Name': 'Table'},
             {'EAN': '124', 'Description': 'A desk chair', 'Name': 'Chair'}]
```
Link Long-Lived NPEs to Current Session

- By linking to $currentSession
  - Garbage collection knows it cannot be garbage collected
  - You can easily retrieve it
- Delete the NPE when no longer used
Integration: Calling Services

- NPEs resulting from a service call will all be sent to the browser.

Tips

- Map only those parts of a web service integration that are necessary.
- Delete NPEs as soon as they are no longer necessary.
- Use database (persistent entities) to cache service call results.
Don’t Use NPEs in Layouts

- Objects in layouts can be on the screen for a long time
- These will be sent back and forth between the client and Runtime very often
Browser State

- Browser refresh loses any unsaved changes
- Multiple browser tabs behave like separate browsers
  - Each browser tab has own client state
Custom Widgets: Use widget.subscribe

- Objects with subscribed widgets will not be garbage collected
  - widget.subscribe will automatically unsubscribe if no longer used
  - data.subscribe needs manual data.unsubscribe

- More info:
  - https://apidocs.mendix.com/7/client/mx.data.html
  - https://apidocs.mendix.com/7/client/mxui_widget__WidgetBase.html
Developer Support
Monitoring Request Size in Server Log

**State:**
- MyEntity
  - id: 1
  - name: def

**Action:** doSomething
**Param:** id 1
**Objects:**
- MyEntity: id=1
  - name=abc
- MyEntity: id=1
  - name=def
Browser Developer Tools
Browser Developer Tools

Hashed read-only values


Headers

Response

Cookies

Timing

Name: qer qe

Address: er qe

Total amount: 0.00

Product name: reg wreg

Quantity: 3
Browser Developer Tools

```json
{  
actionResult: null  
commits: ["12103423998558612", "11258999068426645", "12103423998558613"]  
0: "12103423998558612"  
1: "11258999068426645"  
2: "12103423998558613"  
instructions: [target: "system", type: "close", args: {},...]  
0: [target: "system", type: "close", args: {}],...  
1: [target: "system", type: "refresh_object_list",...],...  

objects: [objectType: "Main.OrderLine", guid: "12103423998558612",...]

0: [objectType: "Main.OrderLine", guid: "12103423998558612",...]


- amount: {value: 5}
  value: 5
- Main.OrderLines_Orders: {value: "11258999068426645"
  value: "11258999068426645"
- Main.OrderLines_Products: {value: "1182194021847554"
  value: "1182194021847554"
- Price: {value: "50", readonly: true}
  readonly: true
  value: "50"
  guid: "12103423998558612"
  hash: "8Qw6kn4L/1Y4Fag6DKXjx38/dqcT1/FjW+Qw5w="
objectType: "Main.OrderLine"

1: [objectType: "Main.Order", guid: "11258999068426645",...]

attributes: [Main.Orders_BillingAddress: {value: "11540474045136897"}, TotalPrice: {value: "140", readonly: true},...]

- Main.Orders_BillingAddress: {value: "11540474045136897"}
- Main.Orders_ShippingAddress: {value: "11540474045136897"}
- Name: {value: "Rutte"}
- OrderDate: {value: "1182194021847554"}
- OrderNumber: {value: 11, readonly: true}
- TotalPrice: {value: "140", readonly: true}
  guid: "11258999068426645"
  hash: "e3Q3cp9x2ZNo1X7Dd615dHy+T7yuU2rg1Tk9k9w="
objectType: "Main.Order"

2: [objectType: "Main.OrderLine", guid: "12103423998558613",...]


- Amount: {value: 6}
  guid: "12103423998558613"
  hash: "8Qw6kn4L/1Y4Fag6DKXjx38/dqcT1/FjW+Qw5w="
```
Browser Developer Tools

**Ctrl + Alt + G**

More client object state info:
* What objects are in client state?
* What widgets are using these objects?
Objects kept in client state depends on needs of page and modified state.
Migrating to Mendix 7
Migration

- Make sure you have a backup!
Upgrade to the latest 6, i.e. 6.10.5

Fix errors, warnings and deprecations

Replace legacy layouts

Upgrade App Store modules
  - Location of App Store modules in your project has moved to Project node

Update your use of java APIs
  - Mendix API is more strict -> API may have been renamed or removed
  - Classloader is more strict -> You cannot use all the jars shipped with Mendix runtime, just the jars in userlib
  - Classloader only loads newest jar of a particular library
Migration

- Upgrade to the latest 6, i.e. 6.10.5
- Fix errors, warnings and deprecations
- Replace legacy layouts
- Upgrade App Store modules
  - Location of App Store modules in your project has moved to Project node
- Update your use of java APIs
  - Mendix API is more strict -> API may have been renamed or removed
  - Classloader is more strict -> You cannot use all the jars shipped with Mendix runtime, just the jars in userlib
  - Classloader only loads newest jar of a particular library
Migration

- Upgrade to the latest 6, i.e. 6.10.5
- Fix errors, warnings and deprecations
- Replace legacy layouts
- Upgrade App Store modules
  - Location of App Store modules in your project has moved to Project node
- Update your use of java APIs
  - Mendix API is more strict -> API may have been renamed or removed
  - Classloader is more strict -> You cannot use all the jars shipped with Mendix runtime, just the jars in userlib
  - Classloader only loads newest jar of a particular library
**Migration**

- Upgrade to the latest 6, i.e. 6.10.5
- Fix errors, warnings and deprecations
- Replace legacy layouts
- Upgrade App Store modules
  - Location of App Store modules in your project has moved to Project node
- Update your use of java APIs
  - Mendix API is more strict -> API may have been renamed or removed
  - Classloader is more strict -> You cannot use all the jars shipped with Mendix runtime, just the jars in userlib
  - Classloader only loads newest jar of a particular library
Migration

- Upgrade to the latest 6, i.e. 6.10.5
- Fix errors, warnings and deprecations
- Replace legacy layouts
- Upgrade App Store modules
  - Location of App Store modules in your project has moved to Project node
- Update your use of java APIs
  - Mendix API is more strict -> API may have been renamed or removed
  - Classloader is more strict -> You cannot use all the jars shipped with Mendix runtime, just the jars in userlib
  - Classloader only loads newest jar of a particular library
Migration

- Upgrade to the latest 6, i.e. 6.10.5
- Fix errors, warnings and deprecations
- Replace legacy layouts
- Upgrade App Store modules
  - Location of App Store modules in your project has moved to Project node
- Update your use of java APIs
  - Mendix API is more strict -> API may have been renamed or removed
  - Classloader is more strict -> You cannot use all the jars shipped with Mendix runtime, just the jars in userlib
  - Classloader only loads newest jar of a particular library
Migration

- Upgrade to the latest 6, i.e. 6.10.5
- Fix errors, warnings and deprecations
- Replace legacy layouts
- Upgrade App Store modules
  - Location of App Store modules in your project has moved to Project node
- Update your use of java APIs
  - Mendix API is more strict -> API may have been renamed or removed
  - Classloader is more strict -> You cannot use all the jars shipped with Mendix runtime, just the jars in userlib
  - Classloader only loads newest jar of a particular library
Migration – Impact of Stateless Runtime

- Every session is a persistent session
  - After logout it may take up to 30 seconds before the logout is visible on all runtime instances
  - You can configure this using SessionValidationTimeout
- NPE attributes need to have read access
- Autocommitted objects not supported in system sessions
- Sign-in microflow has been removed
Rest
New Rest Features Mendix 7

- HTTP response metadata
- Optionally raw response payload
- Access to headers
- Access to cookies (via headers)
- Access to status
- urlEncode & urlDecode
New Rest Features Mendix 7

- HTTP response metadata
- Optionally raw response payload
- Access to headers
- Access to cookies (via headers)
- Access to status
- urlEncode & urlDecode
More Info
More info

- https://docs.mendix.com/releasenotes/desktop-modeler/7.0
- https://docs.mendix.com/refguide7/moving-from-6-to-7
- https://docs.mendix.com/refguide7/clustered-mendix-runtime
Thank You!

QUESTIONS?