Hands-On with Mendix 7

March 30 2017

Andrej Koelewijn, Bart Luijten, Jan de Vries

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

mx mendix

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

mx mendix

Mendix 7

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

mx mendix

Mendix 7

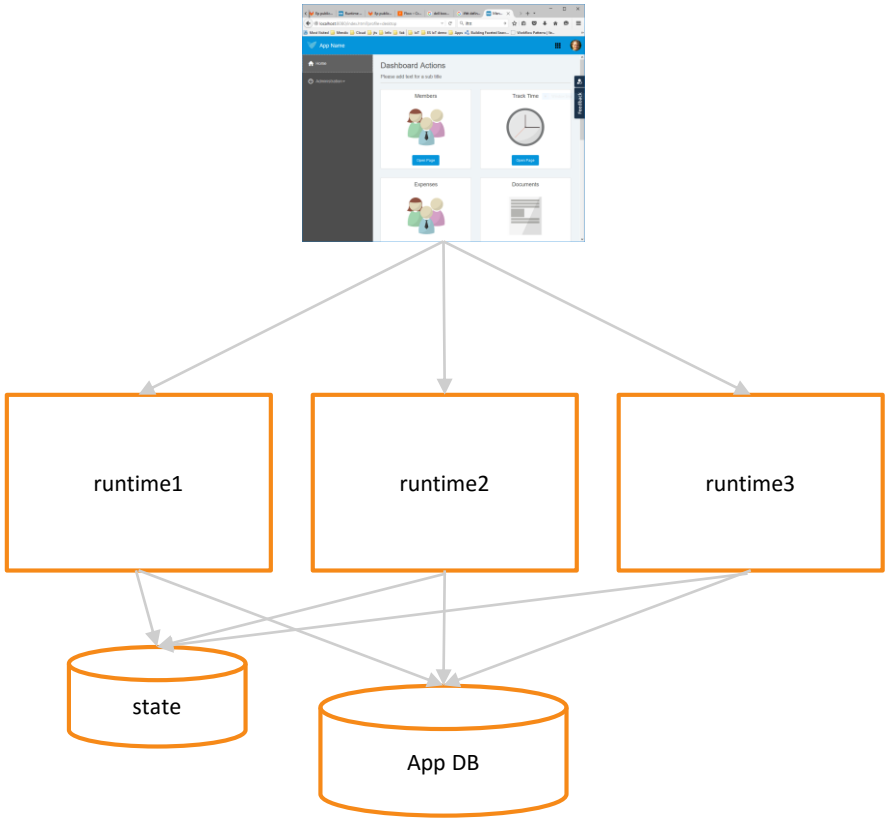
- Resilient and scalable Apps – Stateless Mendix Runtime
  - Why a stateless runtime?
  - How does it work?
  - Best practices
- Migration to Mendix 7
- REST consume

mx mendix

Why a Stateless Runtime?

mx mendix

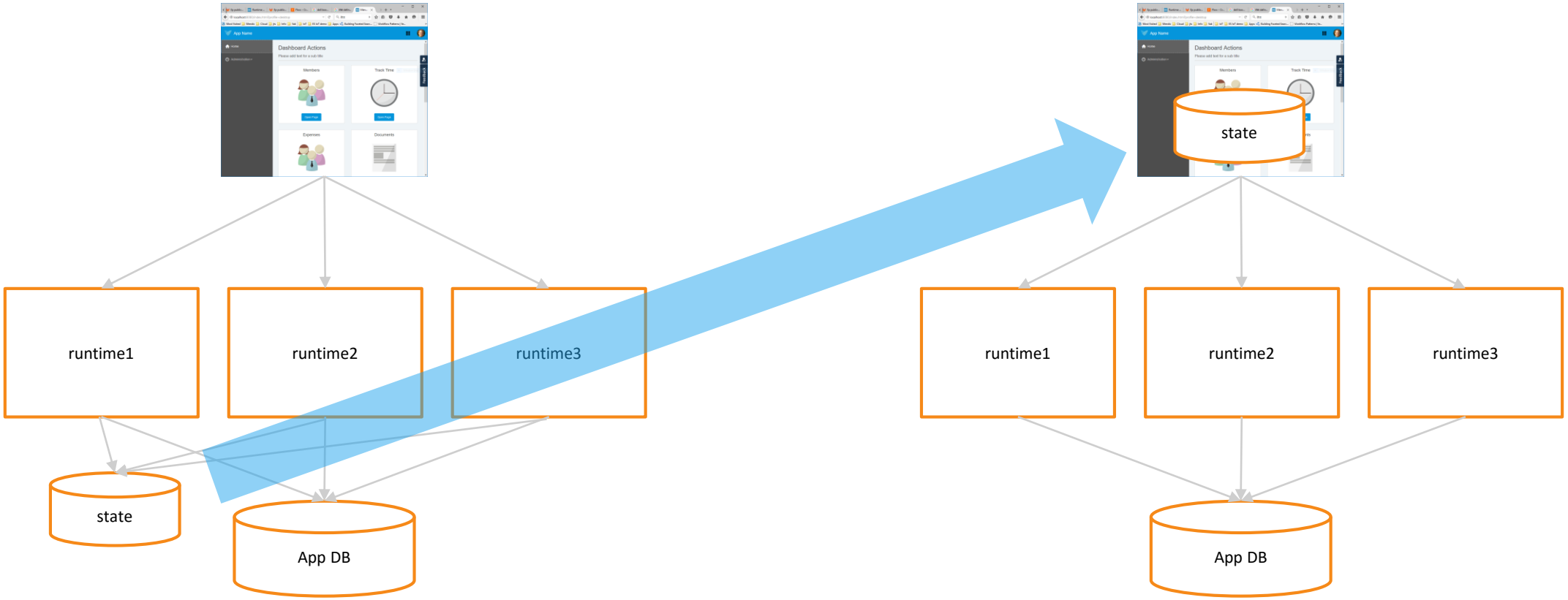
Mendix 6 – Shared State Database



- 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

mx mendix

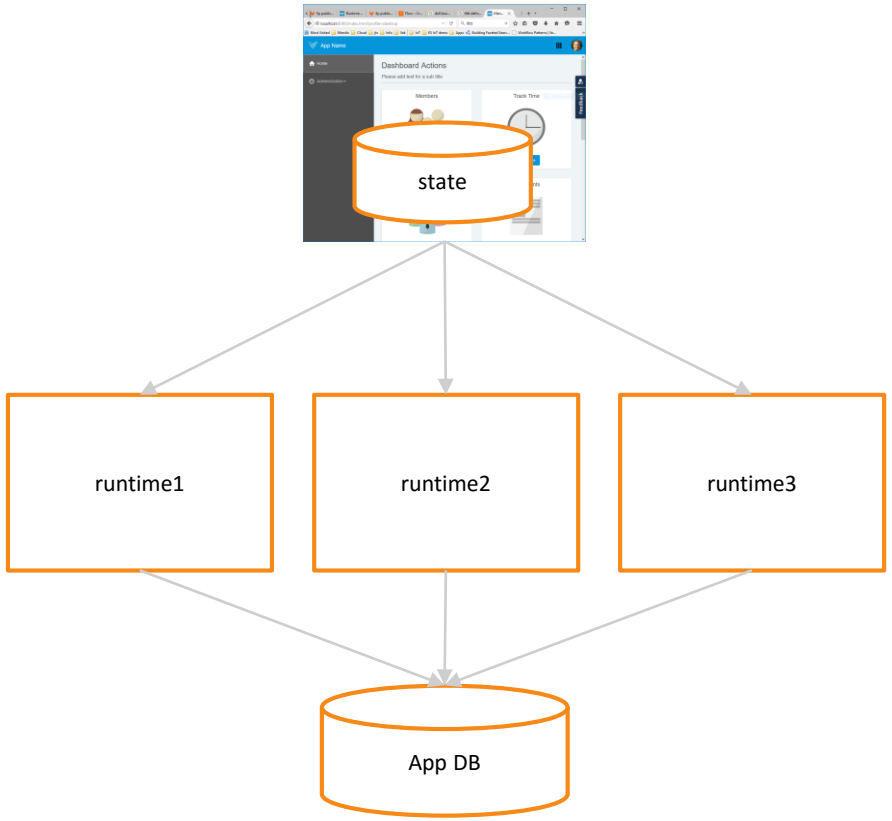
Mendix 7 - Stateless runtime



mx mendix

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



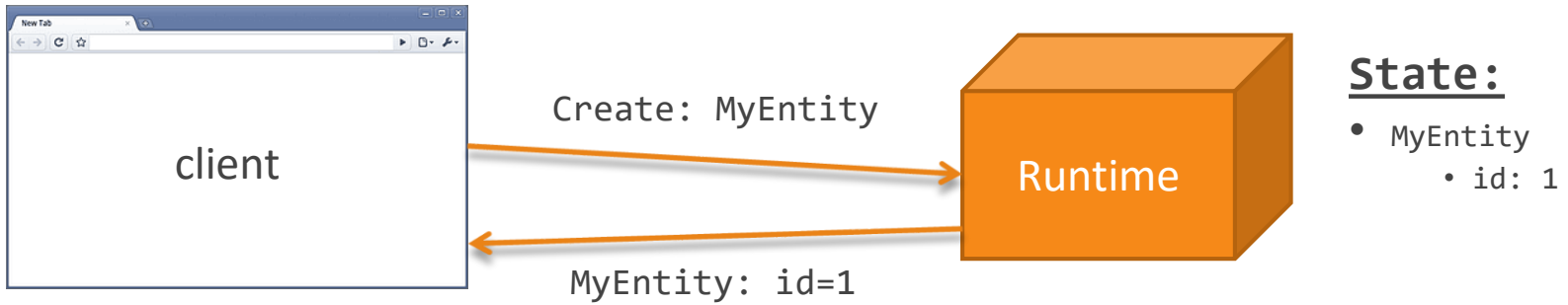
mx mendix

How Does This Work?

Important State Concepts

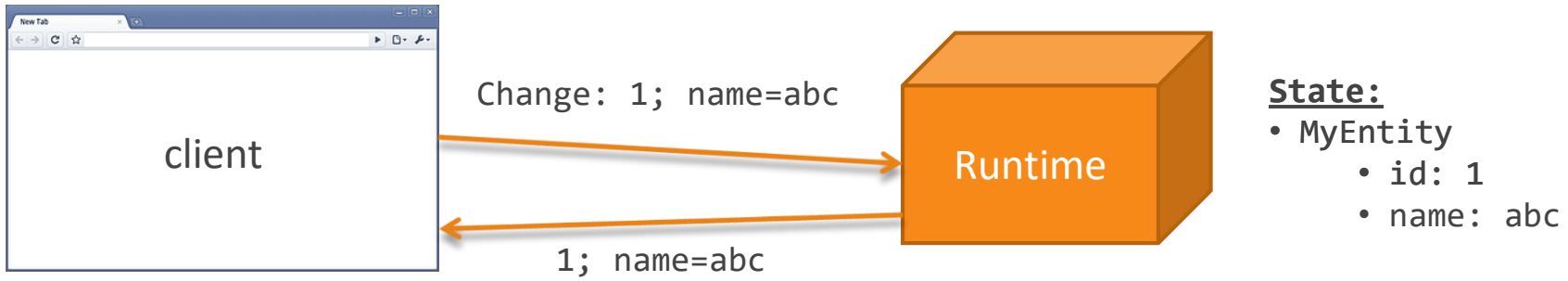
- What will be kept in state by client?
- What will be included in server communication?
- When to create association with session?

State in Mendix 6



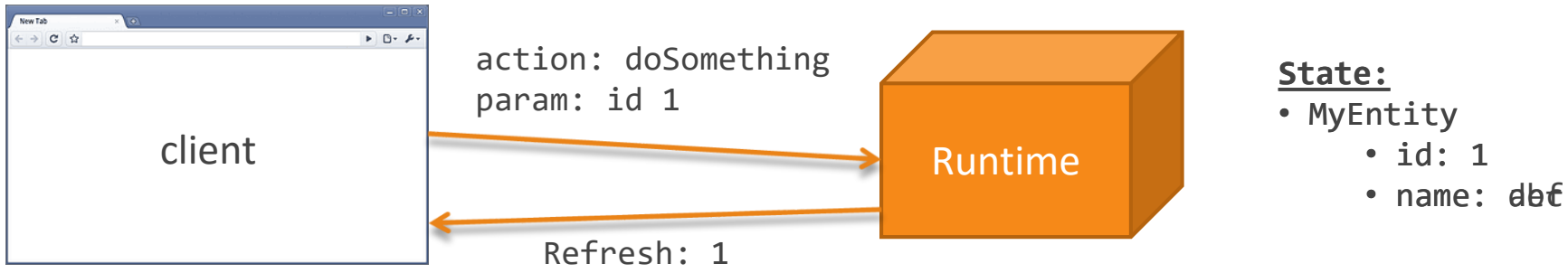
mx mendix

State in Mendix 6



mx mendix

State in Mendix 6

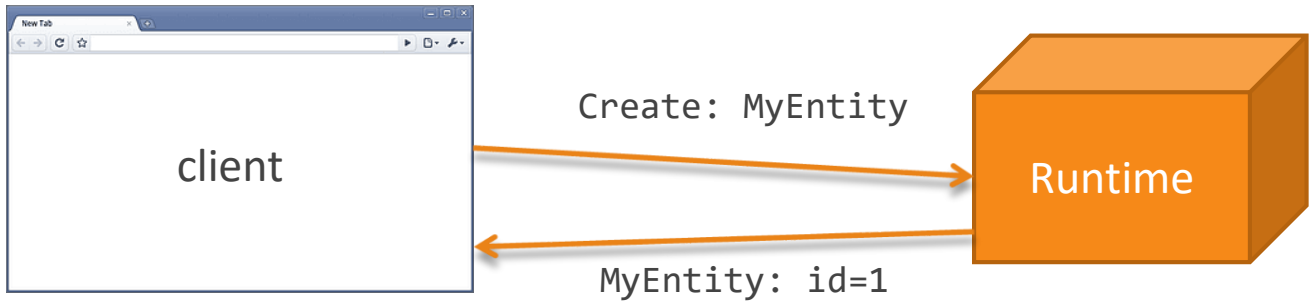


mx mendix

State in Mendix 6

- Session state stored server side
- Downsides
  - Scaling out (state synchronization)
  - Garbage collection
  - Chatty protocol
  - Runtime Memory usage

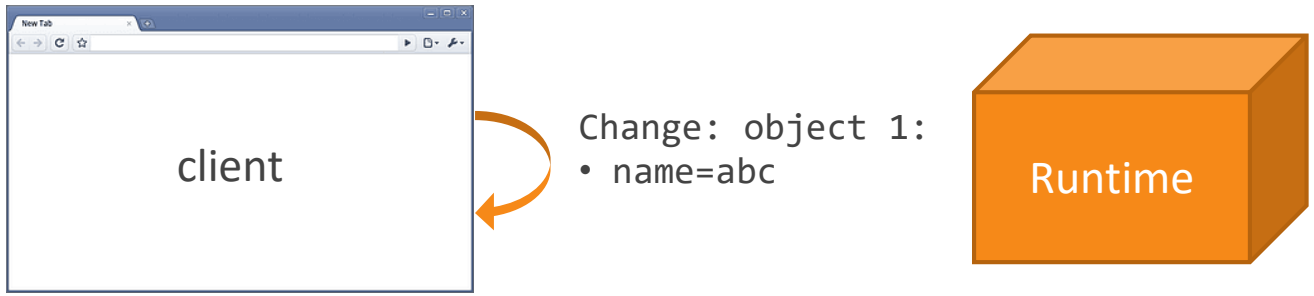
State in Mendix 7



State:

- MyEntity
  - id: 1

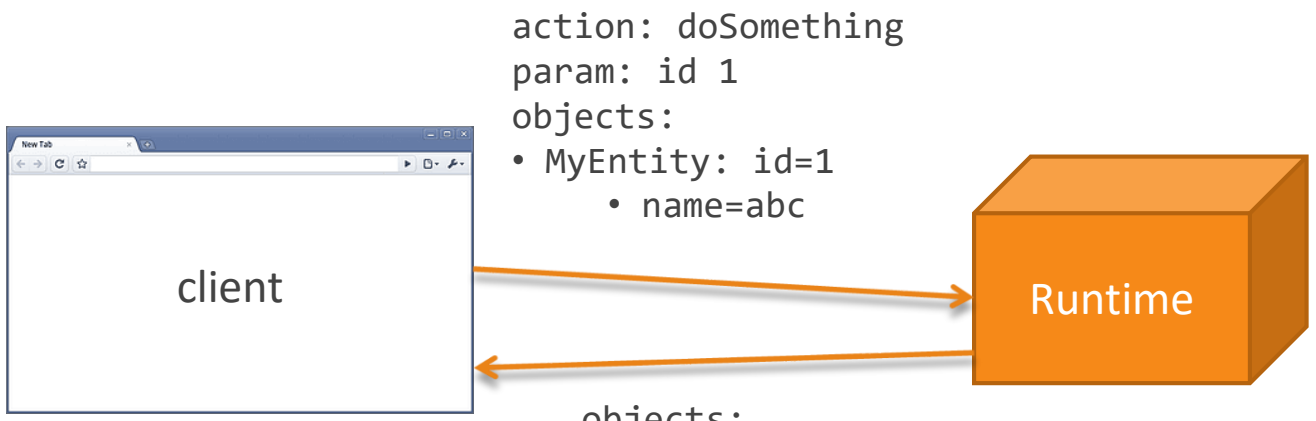
State in Mendix 7



State:

- MyEntity
  - id: 1
  - name: abc

State in Mendix 7



mx mendix

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

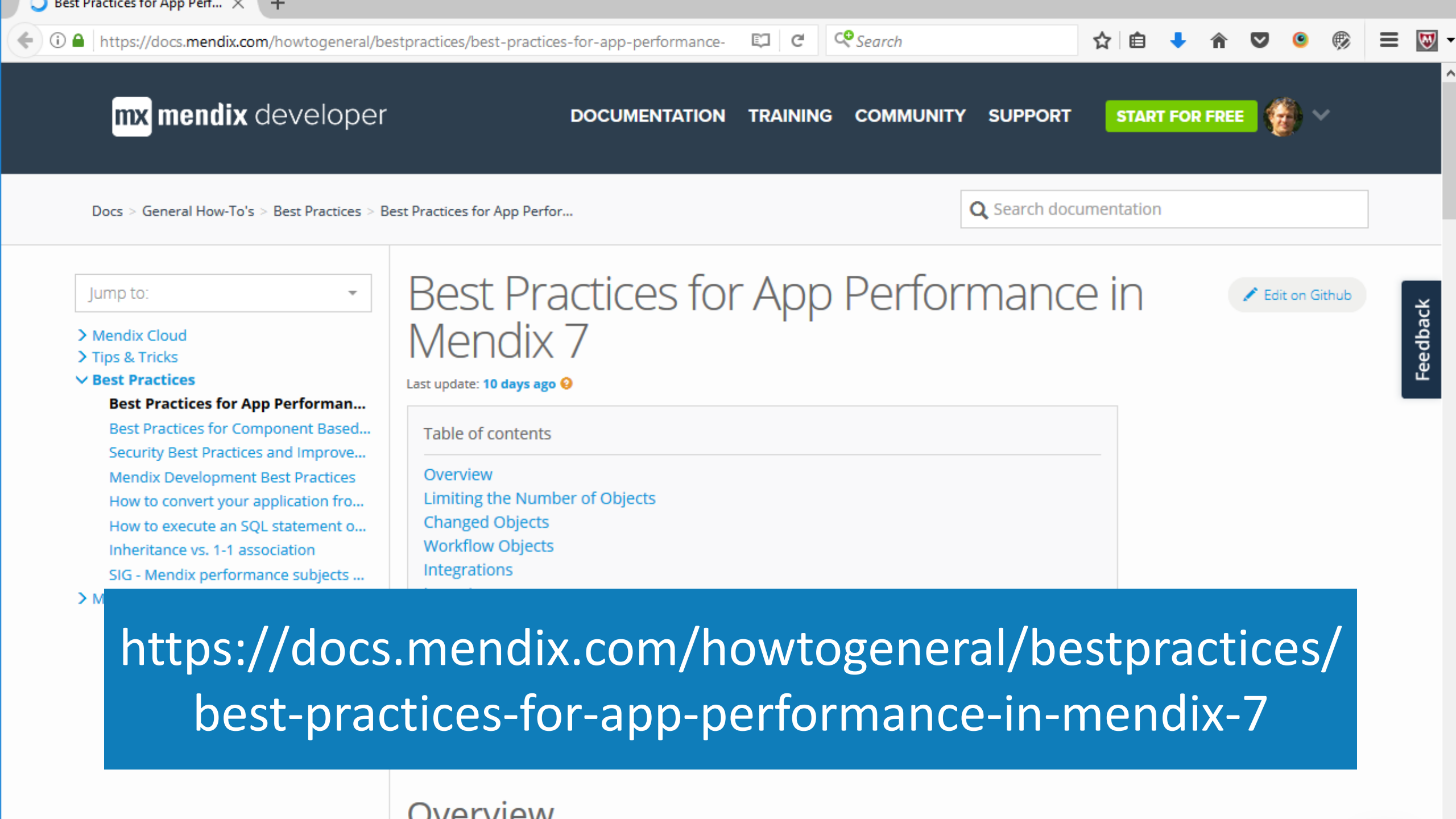
- Reachable network is calculated based on objects available on the client
  - Garbage collection (GC) limits the reachable network
  - Static analysis of the model is used to determine server side data usage
  - GC keeps all reachable …
    - … *NPE objects* from current user, session and subscribed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

Best Practices

mx mendix

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

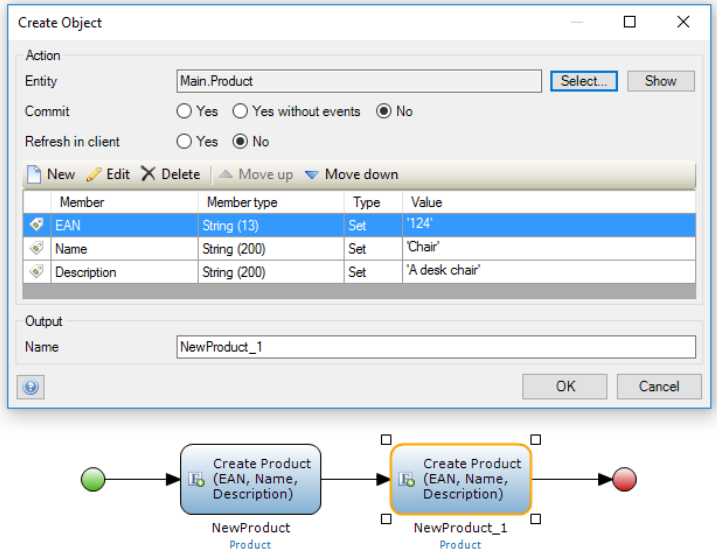

Best Practices for App Perf...
https://docs.mendix.com/howtogeneral/bestpractices/best-practices-for-app-performance-
Search
mx mendix developer
DOCUMENTATION
TRAINING
COMMUNITY
SUPPORT
START FOR FREE
Docs > General How-To's > Best Practices > Best Practices for App Perfor...
Search documentation
Jump to:
Mendix Cloud
Tips & Tricks
Best Practices
Best Practices for App Performan...
Best Practices for Component Based...
Security Best Practices and Improve...
Mendix Development Best Practices
How to convert your application fro...
How to execute an SQL statement o...
Inheritance vs. 1-1 association
SIG - Mendix performance subjects ...
Best Practices for App Performance in Mendix 7
Edit on Github
Feedback
Last update: 10 days ago
Table of contents
Overview
Limiting the Number of Objects
Changed Objects
Workflow Objects
Integrations
https://docs.mendix.com/howtogeneral/bestpractices/best-practices-for-app-performance-in-mendix-7
Overview

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



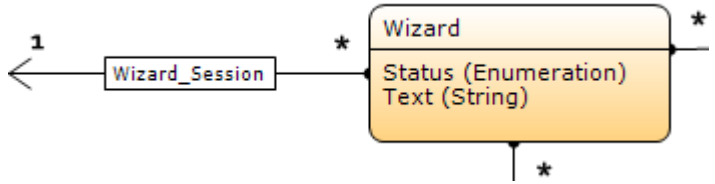
Headers Preview Response Cookies Timing

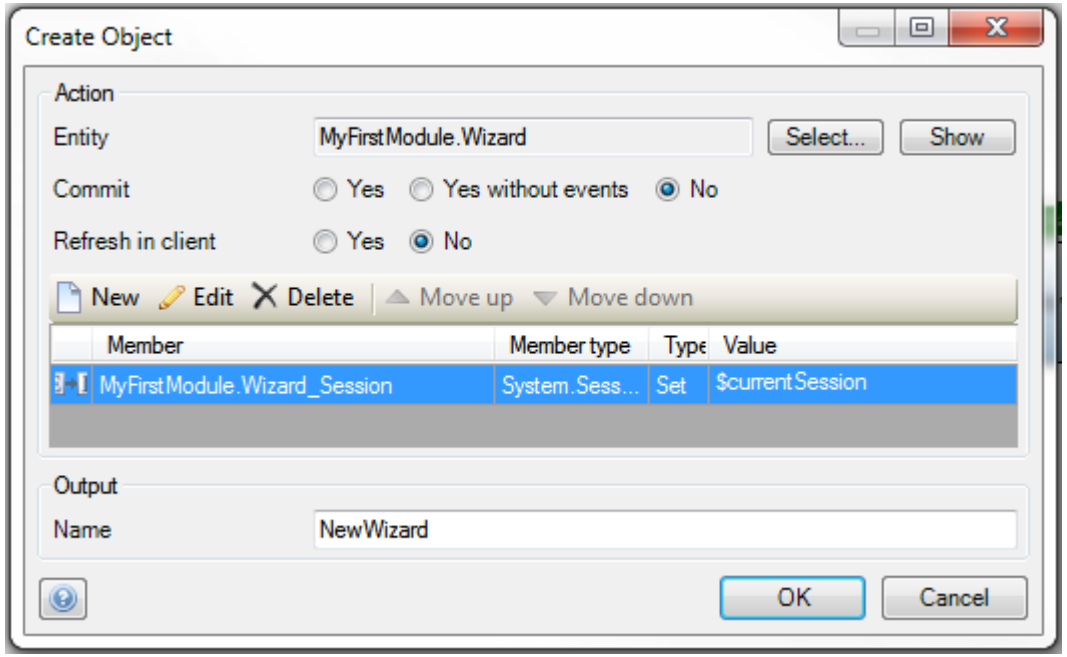
```
▼{,…}
  actionResult: null
▼changes: {8444249301319781: {EAN: {value: "123"}, Description: {value: "A desk"}, Name: {value: "Table"}},…}
  ▼8444249301319781: {EAN: {value: "123"}, Description: {value: "A desk"}, Name: {value: "Table"}}
    ▶Description: {value: "A desk"}
    ▶EAN: {value: "123"}
    ▶Name: {value: "Table"}
  ▼8444249301319782: {EAN: {value: "124"}, Description: {value: "A desk chair"}, Name: {value: "Chair"}}
    ▶Description: {value: "A desk chair"}
    ▶EAN: {value: "124"}
    ▶Name: {value: "Chair"}
▼newpersistable: ["8444249301319781", "8444249301319782"]
    0: "8444249301319781"
    1: "8444249301319782"
▶objects: [{objectType: "Main.Product", guid: "8444249301319781",…},…]
```


mx mendix

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

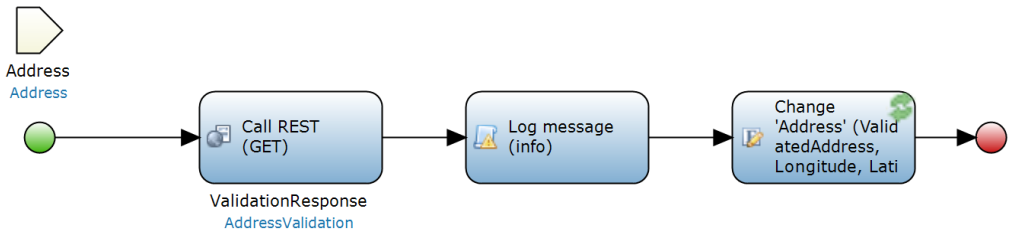




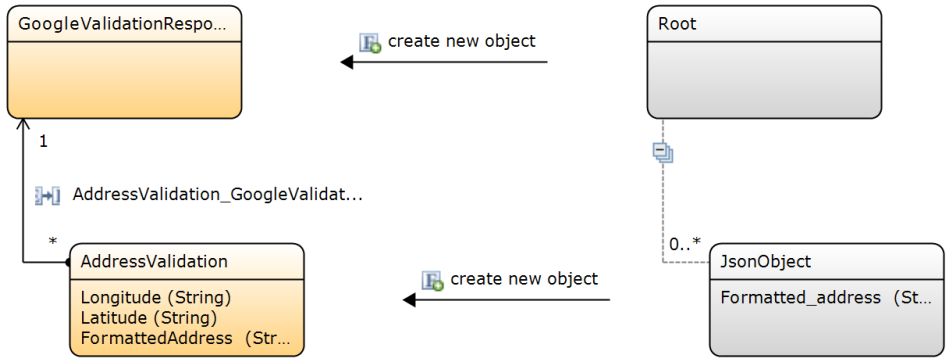
mx mendix

Integration: Calling Services

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



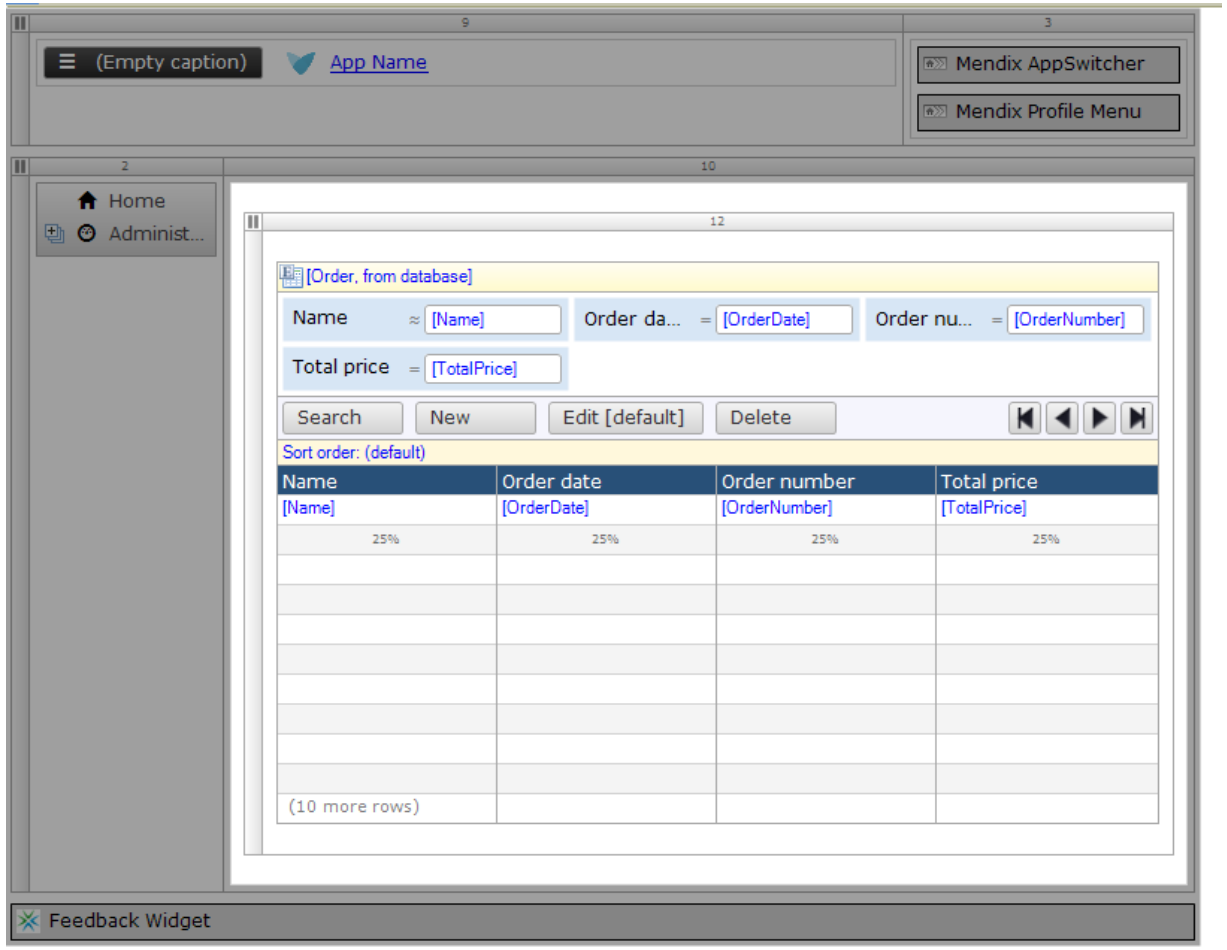
- Tips
  - Map only those parts of a web service integration that are necessary
  - Delete NPEs as soon as they are no longer necessary
  - Use database (persistent entities) to cache service call results



mx mendix

Don’t Use NPEs in Layouts

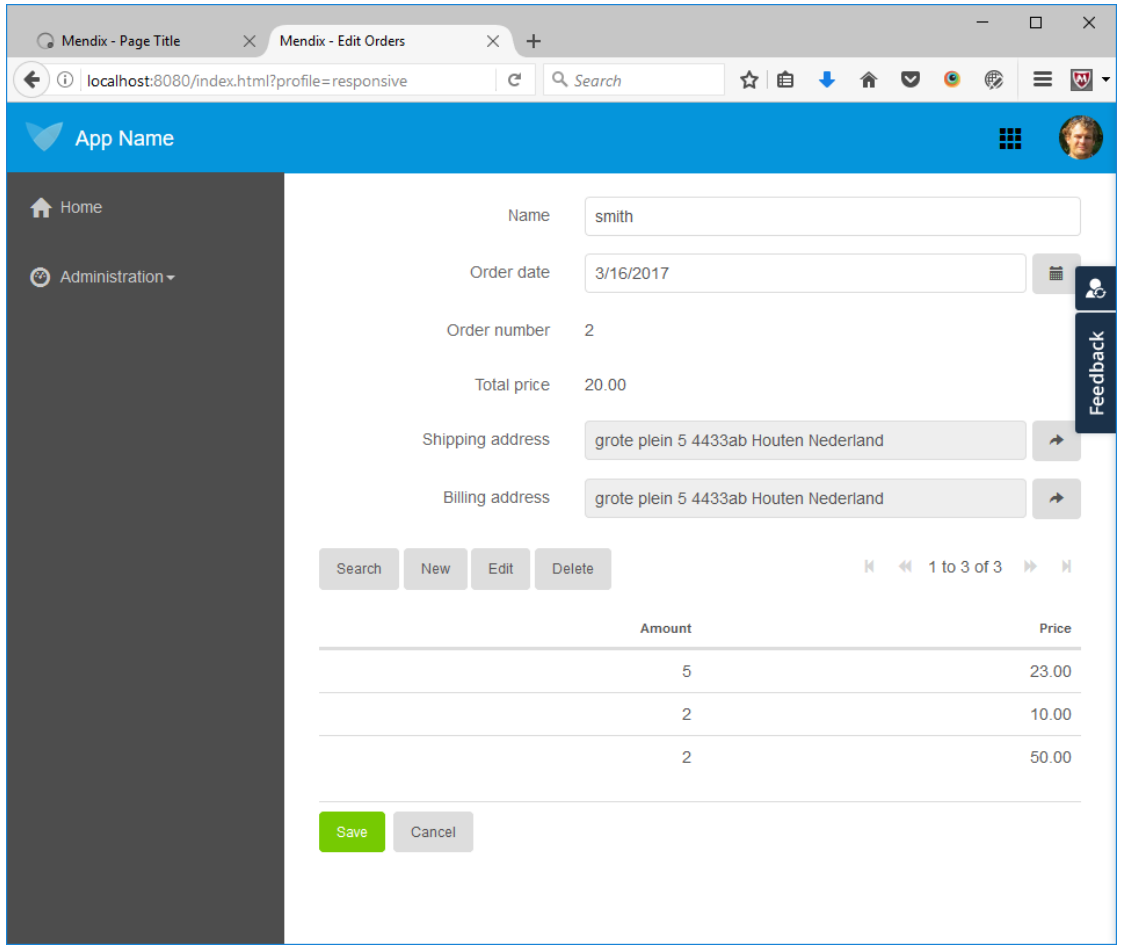
- Objects in layouts can be on the screen for a long time
- These will be sent back and forth between the client and Runtime very often



mx mendix

Browser State

- Browser refresh loses any unsaved changes
- Multiple browser tabs behave like separate browsers
  - Each browser tab has own client state



mx mendix

Custom Widgets: Use widget.subscribe

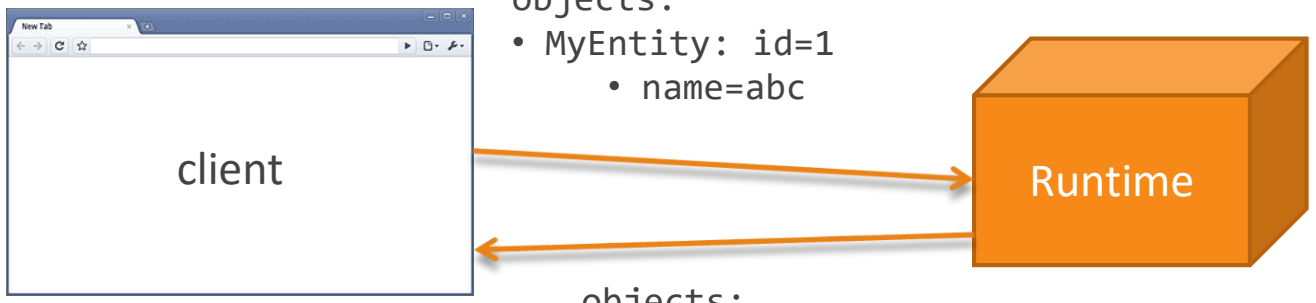
- Objects with subscribed widgets will not be garbage collected
  - widget.subscribe will automatically unsubscribe if no longer used
  - data.subscribe needs manual data.unsubscribe
- More info:
  - https://apidocs.mendix.com/7/client/mx.data.html
  - https://apidocs.mendix.com/7/client/mxui_widget__WidgetBase.html

mx mendix

Developer Support

mx mendix

Monitoring Request Size in Server Log

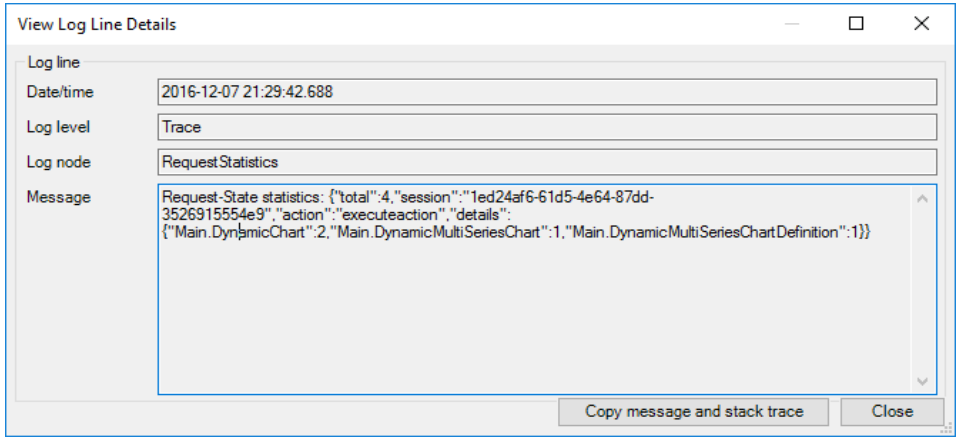



```
action: doSomething
param: id 1
objects:
• MyEntity: id=1
    • name=abc
```

```
objects:
• MyEntity: id=1
    • name=def
```

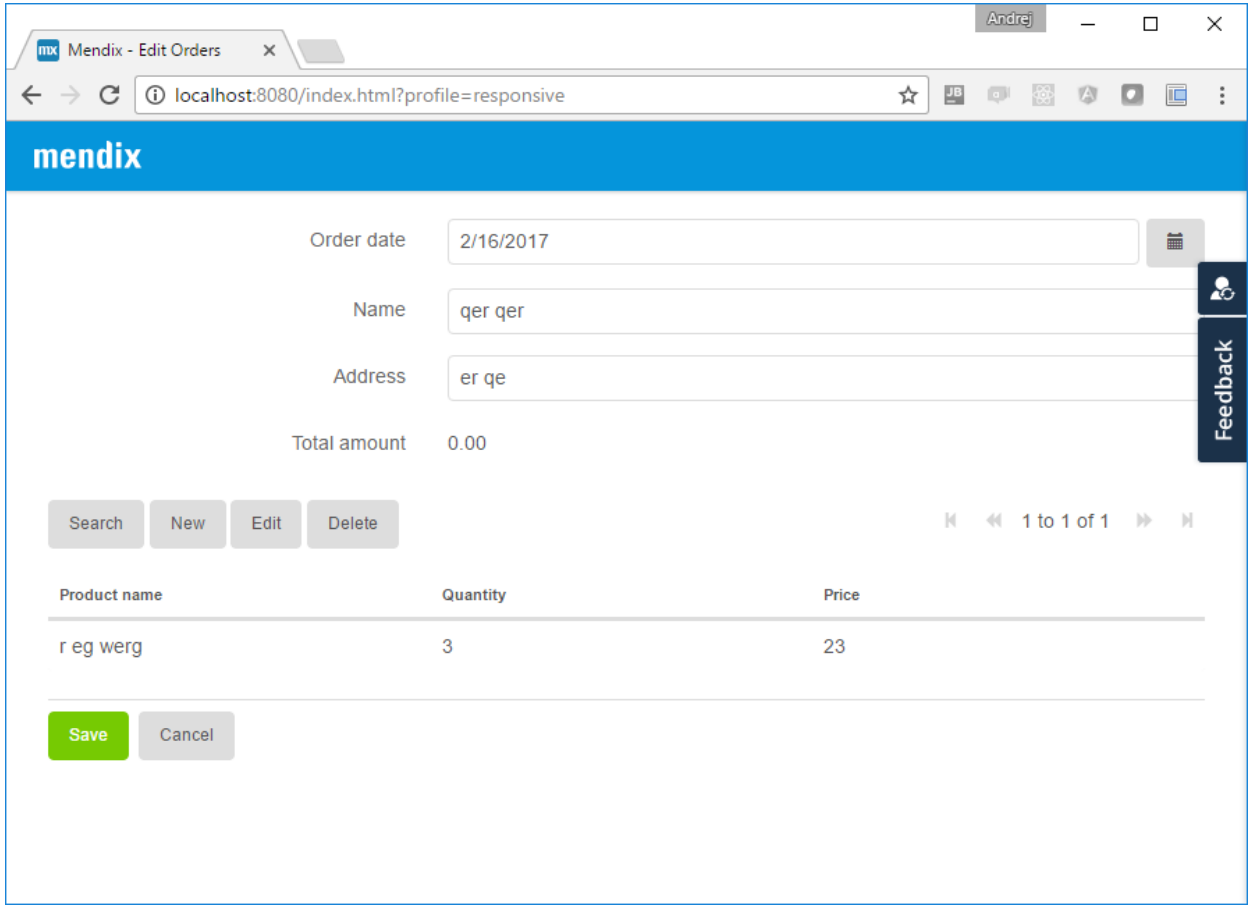
State:

- MyEntity
    - id: 1
    - name: def



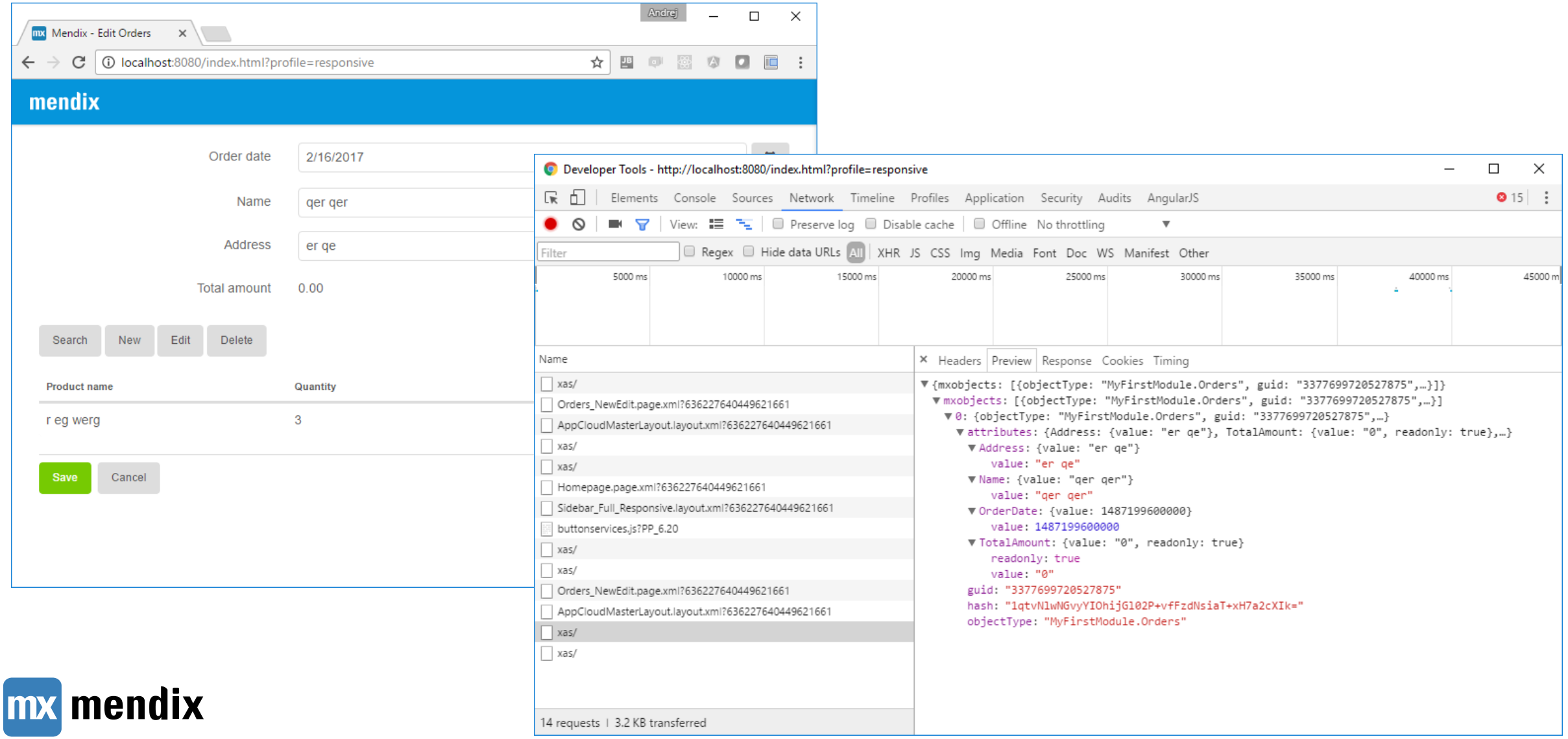
mx mendix

Browser Developer Tools



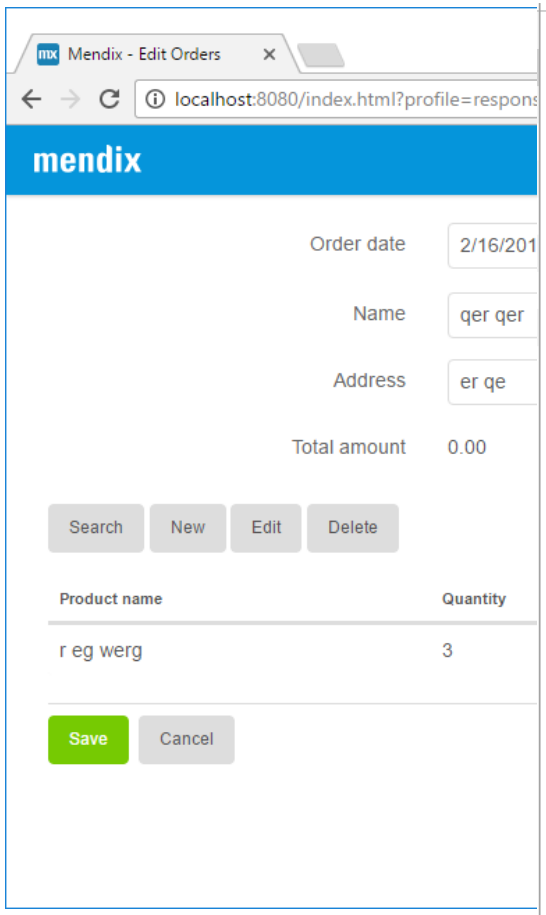
mx mendix

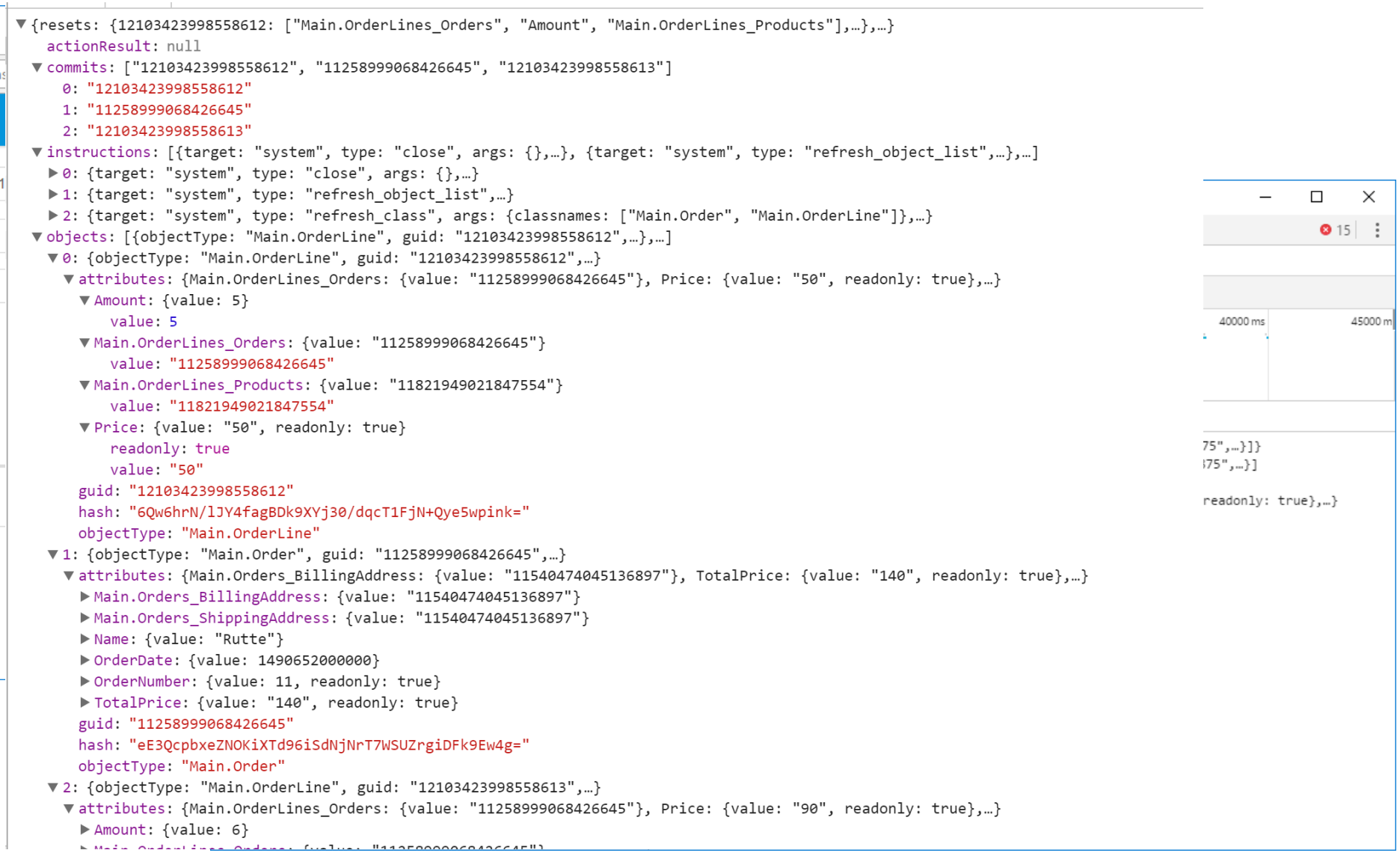
Browser Developer Tools



mx mendix

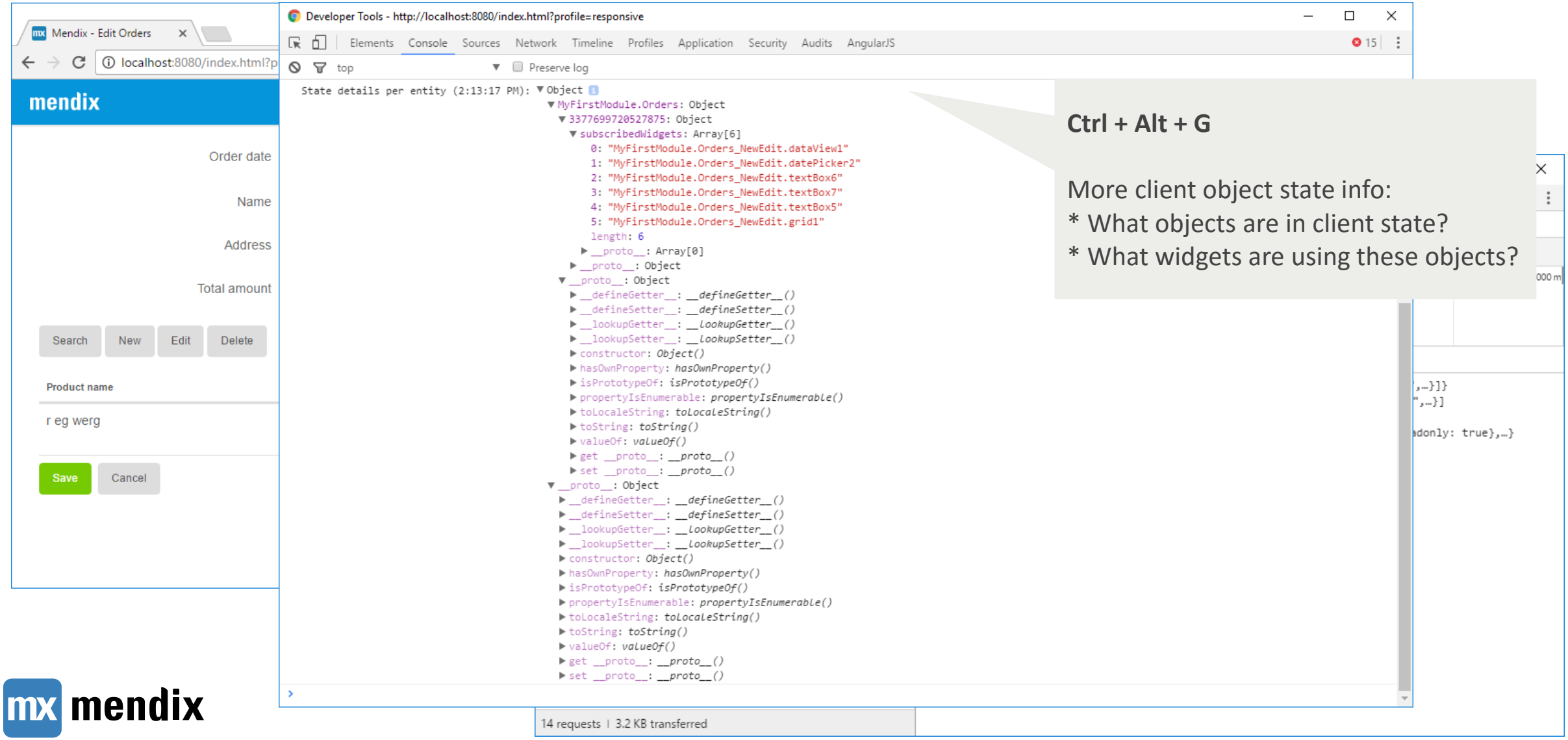
Browser Developer Tools





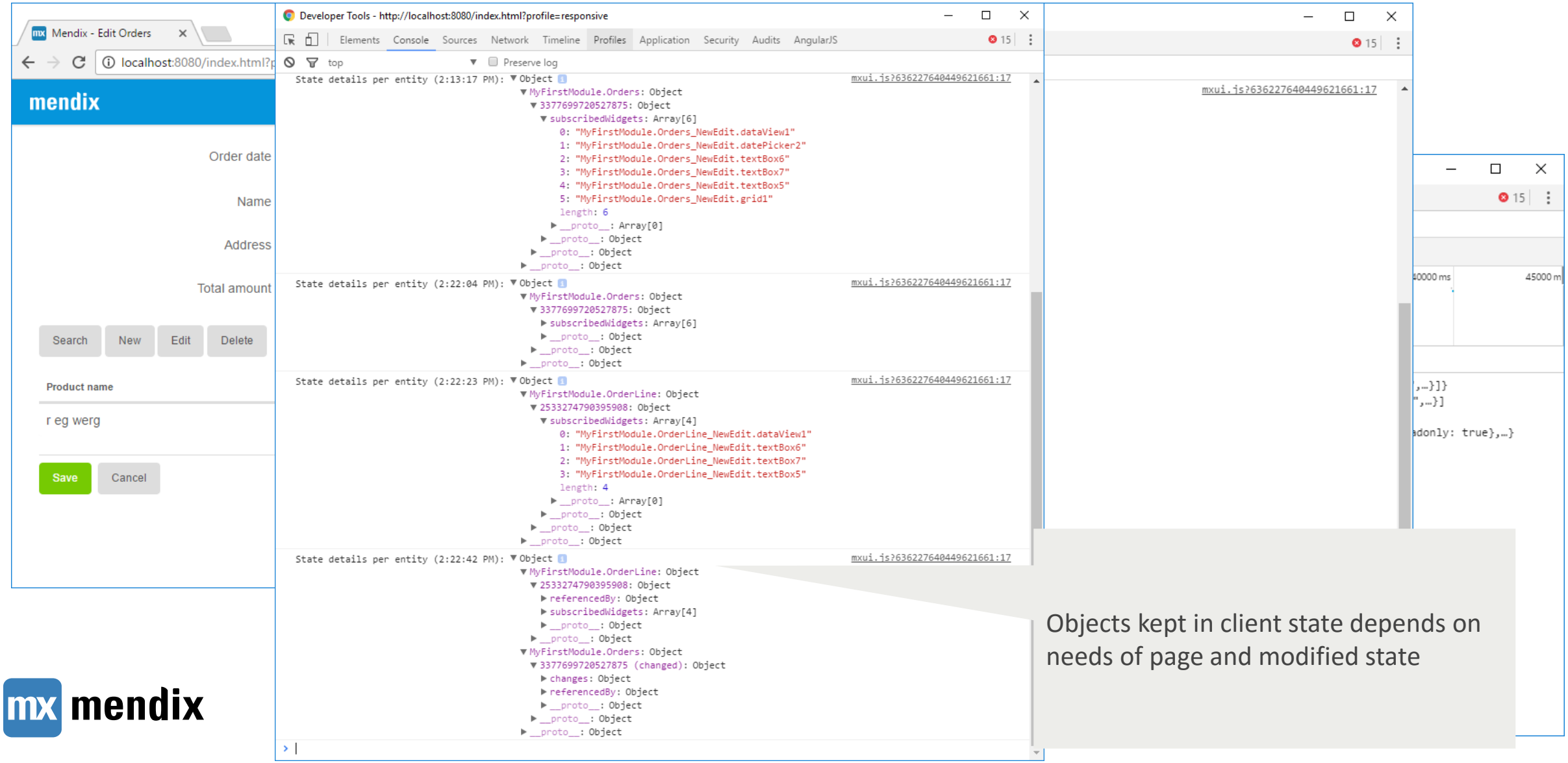
mx mendix

Browser Developer Tools



mx mendix

Browser developer tools



mendix

Migrating to Mendix 7

mx mendix

Migration

- Make sure you have a backup!

Migration

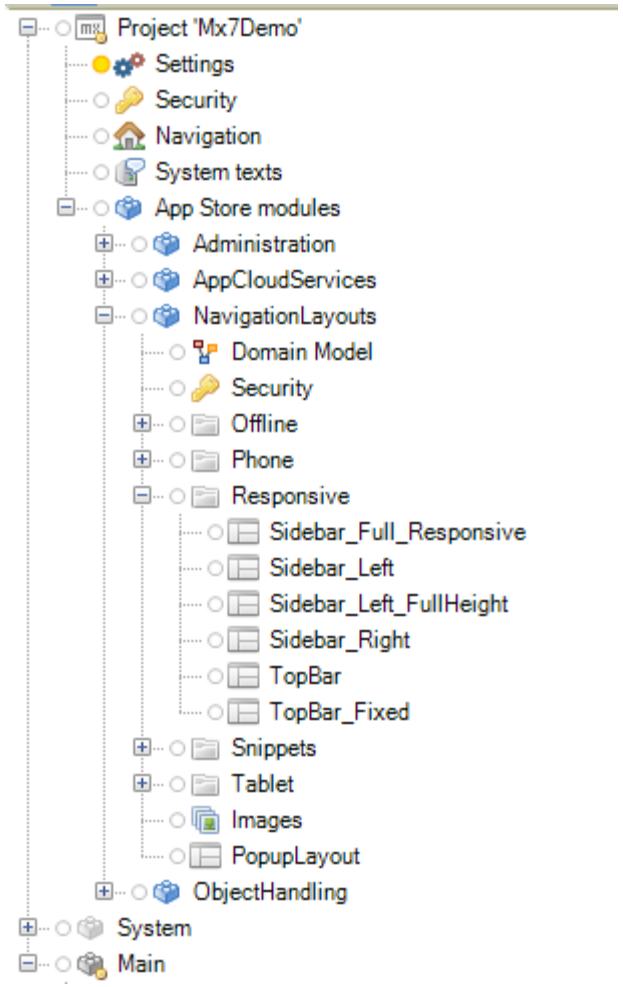
- Upgrade to the latest 6, i.e. 6.10.5
- Fix errors, warnings and deprecations
- Replace legacy layouts
- Upgrade App Store modules
  - Location of App Store modules in your project has moved to Project node
- Update your use of java APIs
  - Mendix API is more strict -> API may have been renamed or 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

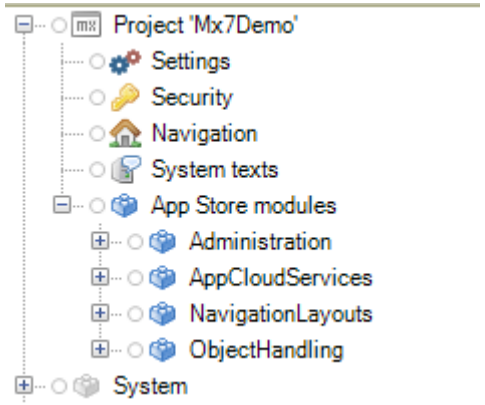
- Upgrade to the latest 6, i.e. 6.10.5
- Fix errors, warnings and deprecations
- Replace legacy layouts
- Upgrade App Store modules
  - Location of App Store modules in your project has moved to Project node
- Update your use of java APIs
  - Mendix API is more strict -> API may have been renamed or 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



mx mendix

Migration

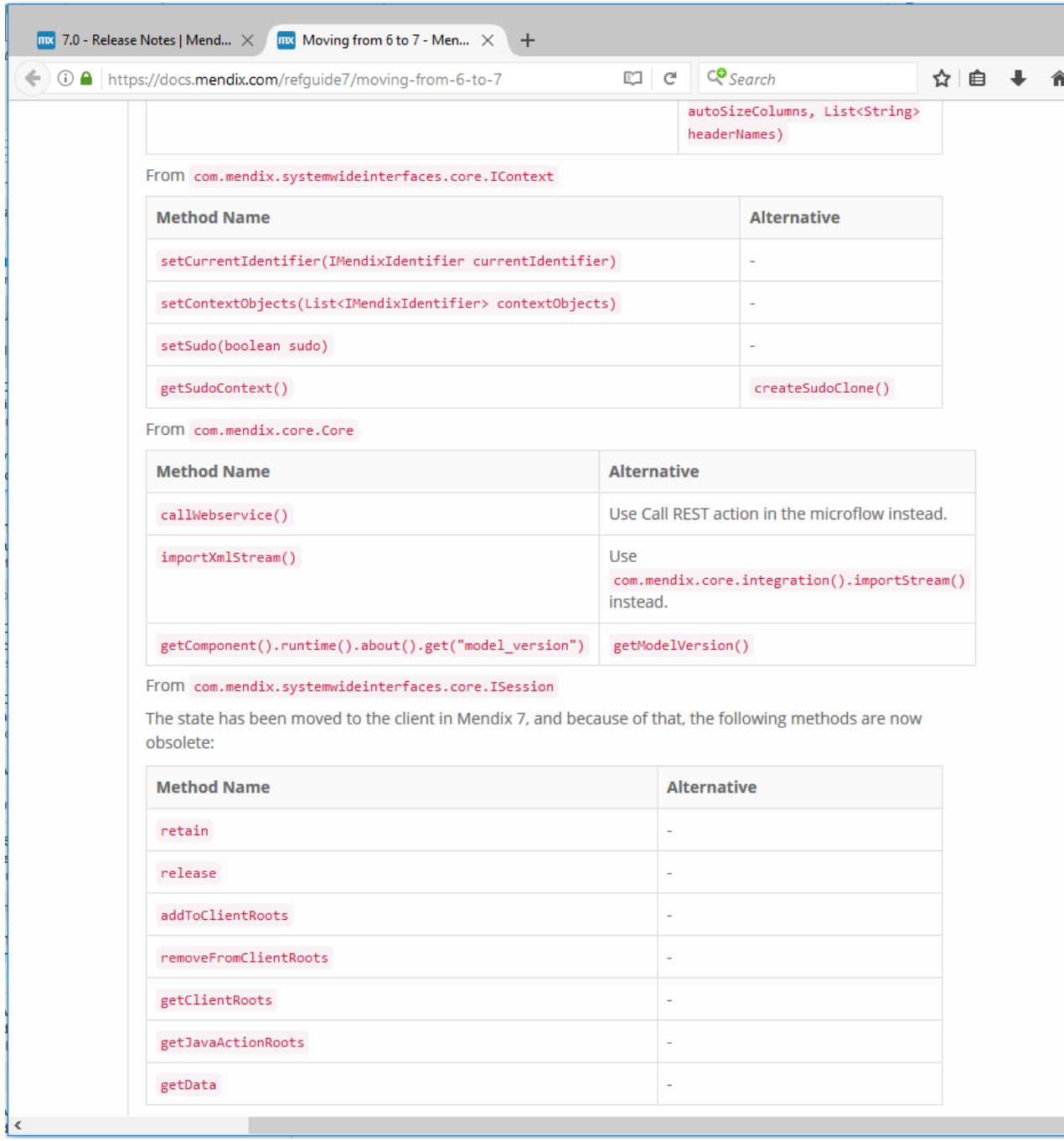
- Upgrade to the latest 6, i.e. 6.10.5
- Fix errors, warnings and deprecations
- Replace legacy layouts
- Upgrade App Store modules
  - Location of App Store modules in your project has moved to Project node
- Update your use of java APIs
  - Mendix API is more strict -> API may have been renamed or 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



mx mendix

Migration

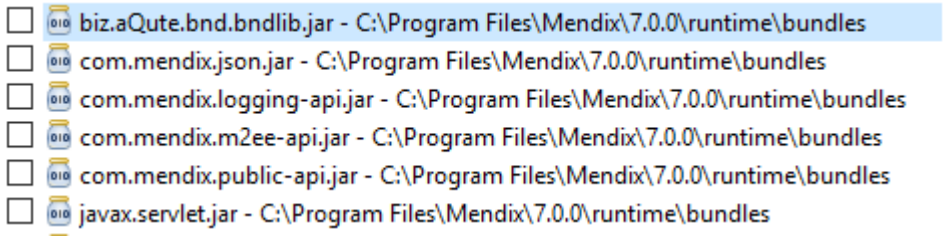
- Upgrade to the latest 6, i.e. 6.10.5
- Fix errors, warnings and deprecations
- Replace legacy layouts
- Upgrade App Store modules
  - Location of App Store modules in your project has moved to Project node
- Update your use of java APIs
  - Mendix API is more strict -> API may have been renamed or 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



mx mendix

Migration

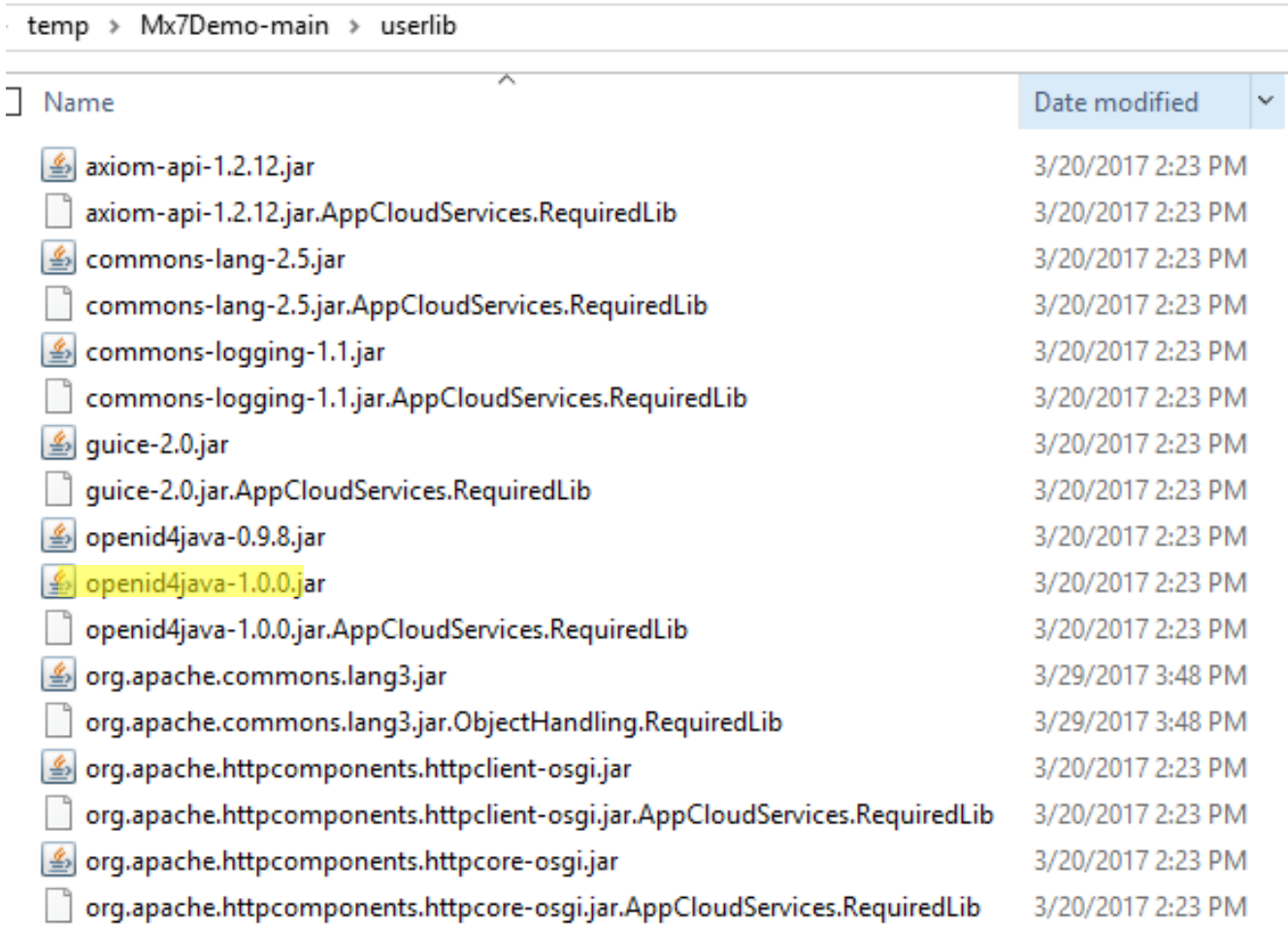
- Upgrade to the latest 6, i.e. 6.10.5
- Fix errors, warnings and deprecations
- Replace legacy layouts
- Upgrade App Store modules
  - Location of App Store modules in your project has moved to Project node
- Update your use of java APIs
  - Mendix API is more strict -> API may have been renamed or 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



mx mendix

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



mx mendix

Migration – Impact of Stateless Runtime

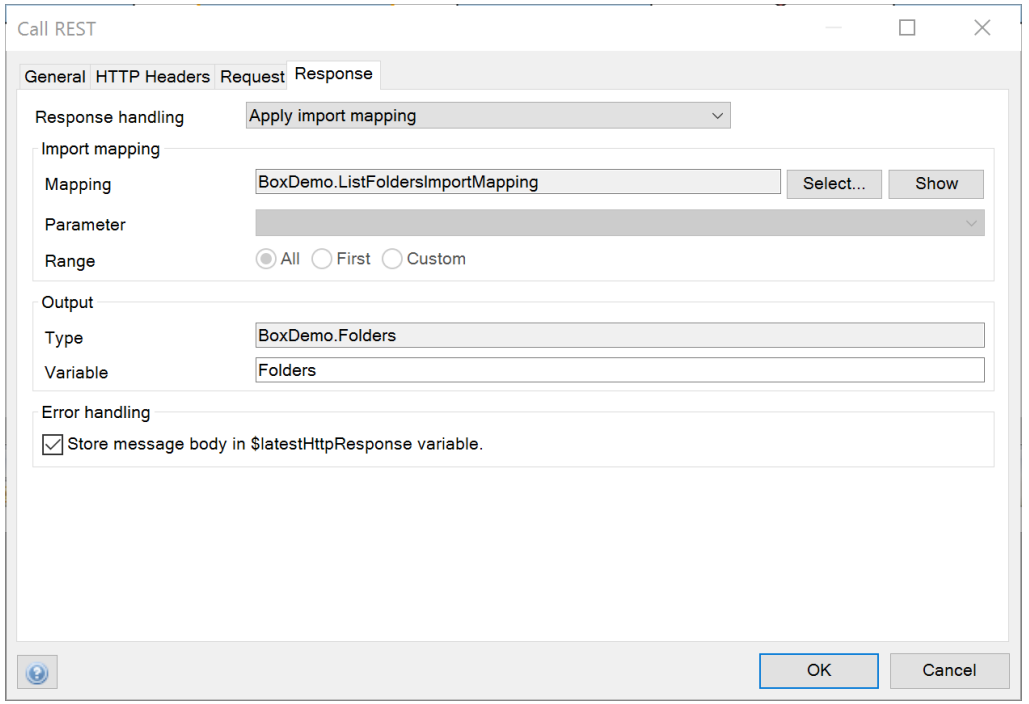
- Every session is a persistent session
  - After logout it may take up to 30 seconds before the logout is visible on all runtime instances
  - You can configure this using SessionValidationTimeout
- NPE attributes need to have read access
- Autocommitted objects not supported in system sessions
- Sign-in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

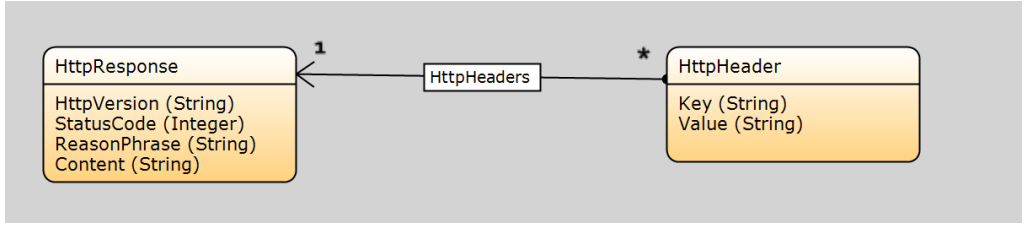
Rest

mx mendix

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

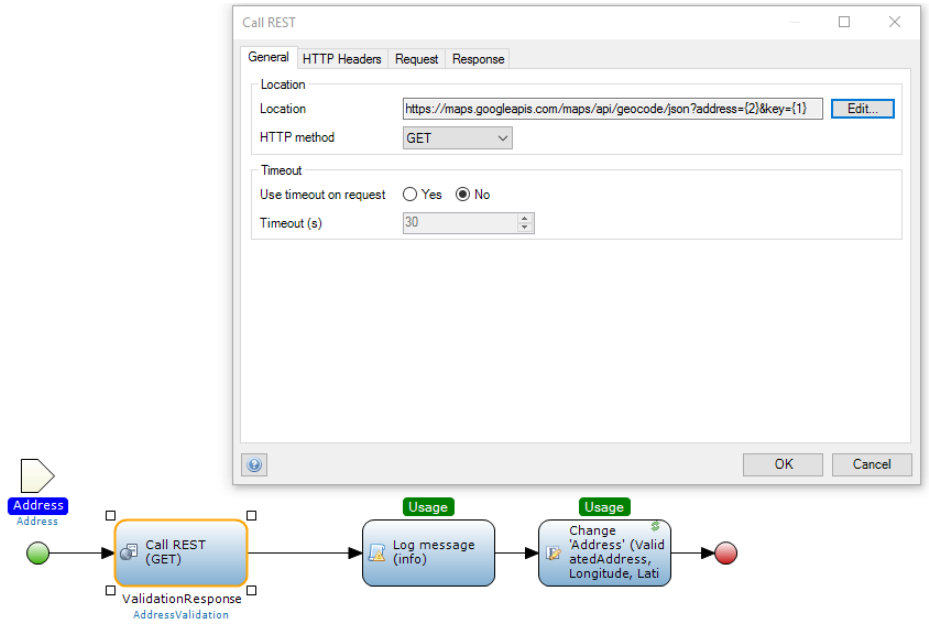


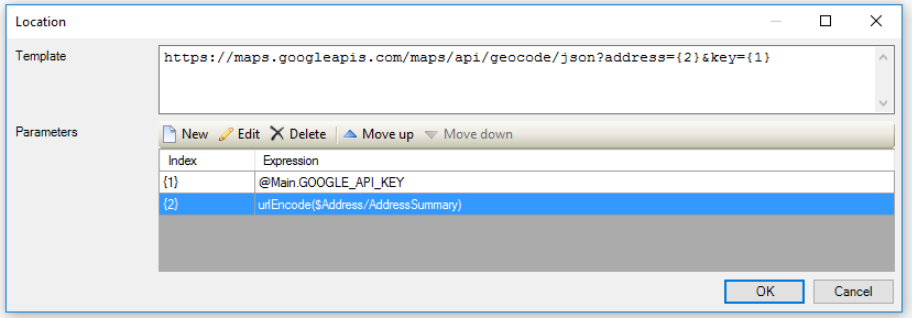


mx mendix

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**





mx mendix

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
- https://docs.mendix.com/howtogeneral/bestpractices/best-practices-for-app-performance-in-mendix-7

Thank You!

QUESTIONS?

mx mendix