# **MapStore Extension: Save Session to Local Storage**

## **Overview**

This extension allows users to save and manage map sessions (layers, annotations, groups, etc.) in local storage. Users can:

* Save the current map state to local storage
* Load sessions from local storage
* Apply saved sessions to the map
* Export/import sessions as JSON files
* Rename or delete saved sessions
* Select multiple sessions for bulk export
* Paginate through stored sessions

The extension integrates with the MapStore framework and is accessible via the toolbar or burger menu.

## **Components**

### **1. SaveSessionToLocalStorageExtension (Component.jsx)**

This is the main UI component responsible for rendering the session manager panel and handling interactions.

#### **Key Features:**

* **Toolbar Offset Handling:** Adjusts the right margin of the navigation bar when the panel is open.
* **Drag & Drop:** Supports reordering sessions via drag and drop.
* **Local Storage Management:** Retrieves, updates, and saves sessions in local storage.
* **Session Name Input:** Allows users to enter a name before saving a session.
* **Pagination:** Supports displaying sessions in batches of 3, 5, or 10 per page.
* **Session Deletion:** Enables users to remove sessions from storage.
* **Session Export:** Allows exporting multiple selected sessions as a JSON file.
* **Session Import:** Supports importing sessions from JSON files.
* **Session Rename:** Enables renaming stored sessions.

#### **Key Functions:**

* getAllSessionsFromLocalStorage(): Retrieves stored sessions from local storage.
* saveSessionToLocalStorage(): Saves the current map state as a session.
* removeSession(session): Deletes a session from storage.
* exportMultipleSessions(): Exports selected sessions as a JSON file.
* handleFileChange(event): Handles importing sessions from a JSON file.
* updateSessionName(oldName, newName): Updates the name of a saved session.

### **2. Session (Session.jsx)**

Handles individual session actions such as applying a session, exporting, renaming, or deleting it.

#### **Key Features:**

* **Apply Session:** Loads stored layers, groups, and annotations to the current map.
* **Export Session:** Saves an individual session as a JSON file.
* **Rename Session:** Supports renaming saved sessions.
* **Delete Session:** Removes a session from local storage.

#### **Key Functions:**

* compareMapToSession(sessionLayers, entireMapLayers): Determines if the stored session matches the current map state.
* extractAnnotation(layers): Filters annotation layers from the session.
* doesLayerExistOnCurrentMap(layer, entireMapLayers): Checks if a layer exists on the current map.
* ApplySessionToMap(): Applies a session to the map by loading its layers and updating visibility and opacity.
* exportSelectedSession(): Exports the session as a JSON file.
* handleRenameSubmit(event): Handles renaming a session.

### **3. Module.jsx**

Connects the extension to the MapStore framework and Redux.

#### **Key Features:**

* **Redux Integration:** Defines reducers for toggling the panel state and applying sessions.
* **Toolbar & Burger Menu Integration:** Adds the extension to the MapStore toolbar and menu.

#### **Key Redux Actions:**

* **Toggle Dialogue:** Opens and closes the extension panel.
* **Apply Session:** Handles loading a session’s data into the map.

#### **Key Objects:**

* currentSession: Extracts map state including layers, groups, annotations, projection, zoom, extent, and center.
* entireMap: Represents the full Redux store state.
* dialogueState: Determines if the session panel is open.

#### **Containers:**

* **Toolbar:** Adds a button in the toolbar to open the session panel.
* **Burger Menu:** Adds an entry in the menu to access the extension.

## **Installation & Usage**

### **1. Build the extension**

In your editor of choice, open a terminal and run the command *npm run ext:start*.

This will create a zip file containing all the necessary files for the extension to be imported in MapStore.

### **2. Use the extension**

After the extension was build, open MapStore and create a Context. While following the context setup dialogue, at step number 3 you can choose to import a custom extension. Do that and upload the .zip file created at step number 1.

### **3. Extension is now available**

Extension is now ready to use in your context.

### **4. Manage Sessions**

* **Save a Session:** Enter a name and click the save button.
* **Load a Session:** Click the load button and select a session.
* **Export a Session:** Click the export button next to a session.
* **Import Sessions:** Use the file upload button to load a JSON file.
* **Rename a Session:** Click the edit button and enter a new name.
* **Delete a Session:** Click the delete button.

This documentation provides an overview of the functionality, key components, and how to use the MapStore extension for saving and managing sessions in local storage.