/\*\*

\* V7.7.7 UI Renderer --- FIXED: Add state & detail:changed event

\* - Renders quick results to .quick-results-grid (and fallbacks)

\* - Updates detail panel fields

\* - FIXED: Maintains state.currentDetailItem

\* - FIXED: Dispatches 'detail:changed' for related-equipment.js

\*/

(function () {

'use strict';

const SELECTORS = {

quickListCandidates: [

'#quick-results-list',

'.quick-results-grid',

'#quick-results',

'[data-role="quick-results"]'

],

tableBodyCandidates: [

'#results-table-body',

'#all-results-body',

'.results-table-body',

'[data-role="results-body"]'

],

// Detail fields (IDs from V7.7.5 page)

detailCompany: '#detail-company',

detailRackId: '#detail-rack-id',

detailLayerNum: '#detail-layer-num',

detailRackLocation: '#detail-rack-location',

detailLayerNotes: '#detail-layer-notes',

detailCodeName: '#detail-code-name',

detailDimensions: '#detail-dimensions',

detailCutline: '#detail-cutline',

detailTeflon: '#detail-teflon',

detailPlastic: '#detail-plastic',

detailNotes: '#detail-notes',

detailProcessing: '#detail-processing',

};

const UiRenderer = {

// 🔥 THÊM: State object để lưu trữ item hiện tại

state: {

currentDetailItem: null,

selectedItemId: null,

isDetailPanelOpen: false

},

init() {

document.addEventListener('search:updated', (e) => {

const { results } = e.detail || {};

this.renderResults(results || []);

if (results && results.length) this.renderDetailInfo(results[0]);

else this.clearDetail();

});

// Click on quick cards

const quick = getFirst(SELECTORS.quickListCandidates);

if (quick) {

quick.addEventListener('click', (ev) => {

const card = ev.target.closest('.result-card');

if (!card || !quick.contains(card)) return;

const idx = Number(card.getAttribute('data-index'));

const items = window.SearchModule?.getResults?.() || [];

const item = items[idx];

if (item) {

this.renderDetailInfo(item);

highlightSelection(card);

}

});

}

// 🔥 THÊM: Click on table rows

const tbody = getFirst(SELECTORS.tableBodyCandidates);

if (tbody) {

tbody.addEventListener('click', (ev) => {

const tr = ev.target.closest('tr');

if (!tr) return;

const idx = Number(tr.getAttribute('data-index'));

const items = window.SearchModule?.getResults?.() || [];

const item = items[idx];

if (item) {

this.renderDetailInfo(item);

highlightSelection(tr);

}

});

}

},

renderResults(items) {

this.renderQuickCards(items);

this.renderTable(items);

},

renderQuickCards(items) {

const wrap = getFirst(SELECTORS.quickListCandidates);

if (!wrap) return;

wrap.innerHTML = '';

items.slice(0, 100).forEach((item, idx) => {

const isMold = item.itemType === 'mold';

const typeLabel = isMold ? '金型' : '抜型';

const code = esc(item.displayCode || item.MoldCode || item.CutterNo || '-');

const name = esc(item.displayName || '-');

const dim = esc(item.displayDimensions || item.cutlineSize || 'N/A');

const loc = esc(item.rackInfo?.RackLocation || '-');

const el = document.createElement('div');

el.className = 'result-card';

el.setAttribute('data-index', String(idx));

el.setAttribute('data-type', isMold ? 'mold' : 'cutter');

el.innerHTML = `

<div class="card-header">

<span class="card-type ${isMold ? 'mold' : 'cutter'}">${typeLabel}</span>

<span class="card-code">${code}</span>

</div>

<div class="card-name">${name}</div>

<div class="card-dimensions">${dim}</div>

`;

wrap.appendChild(el);

});

},

renderTable(items) {

const tbody = getFirst(SELECTORS.tableBodyCandidates);

if (!tbody) return;

tbody.innerHTML = '';

items.forEach((item, idx) => {

const tr = document.createElement('tr');

tr.setAttribute('data-index', String(idx));

tr.innerHTML = `

<td>${item.itemType === 'mold' ? '金型' : '抜型'}</td>

<td>${esc(item.displayCode || '-')}</td>

<td>${esc(item.displayName || '-')}</td>

<td>${esc(item.displayDimensions || item.cutlineSize || 'N/A')}</td>

<td>${esc(item.rackInfo?.RackLocation || '-')}</td>

<td>${esc(item.designInfo?.ManufactureDate || '-')}</td>

<td><span class="status-chip ${item.currentStatus?.class || ''}">${item.currentStatus?.text || '-'}</span></td>

<td><button class="btn-details" type="button">詳細</button></td>

`;

tbody.appendChild(tr);

});

},

renderDetailInfo(item) {

// 🔥 THÊM: Cập nhật state trước khi render

this.state.currentDetailItem = item;

this.state.selectedItemId = item.MoldID || item.CutterID || item.id;

this.state.isDetailPanelOpen = true;

// Render UI như cũ

setText(SELECTORS.detailCompany, displayCompany(item));

setText(SELECTORS.detailRackId, item.rackLayerInfo?.RackID || '-');

setText(SELECTORS.detailLayerNum, item.rackLayerInfo?.RackLayerNumber || '-');

const rackLoc = joinDash(item.rackInfo?.RackLocation, item.rackLayerInfo?.RackLayerNumber);

setText(SELECTORS.detailRackLocation, rackLoc || '-');

setText(SELECTORS.detailLayerNotes, item.rackLayerInfo?.RackLayerNotes || '');

setText(SELECTORS.detailCodeName, item.displayCode || '-');

setText(SELECTORS.detailDimensions, normalizeDim(item.displayDimensions));

setText(SELECTORS.detailCutline, normalizeDim(item.cutlineSize) || '-');

setText(SELECTORS.detailTeflon, item.TeflonCoating || '-');

setText(SELECTORS.detailPlastic, item.designInfo?.DesignForPlasticType || '-');

setText(SELECTORS.detailNotes, item.MoldNotes || item.CutterNote || '-');

let status = '-';

if (item.MoldReturning === 'TRUE') status = '返却済み';

else if (item.MoldDisposing === 'TRUE') status = '廃棄済み';

else status = item.currentStatus?.text || '利用可能';

setText(SELECTORS.detailProcessing, status);

// 🔥 THÊM: Dispatch event 'detail:changed' cho related-equipment.js

document.dispatchEvent(new CustomEvent('detail:changed', {

detail: { item }

}));

console.log('📡 [UIRenderer] Dispatched detail:changed:', item);

},

clearDetail() {

// 🔥 THÊM: Clear state khi xóa detail

this.state.currentDetailItem = null;

this.state.selectedItemId = null;

this.state.isDetailPanelOpen = false;

[

'detailCompany','detailRackId','detailLayerNum','detailRackLocation','detailLayerNotes',

'detailCodeName','detailDimensions','detailCutline','detailTeflon','detailPlastic','detailNotes','detailProcessing'

].forEach(k => setText(SELECTORS[k], '-'));

},

};

// Helpers

function getFirst(list) {

for (const sel of list) {

const el = document.querySelector(sel);

if (el) return el;

}

return null;

}

function setText(sel, val) { const el = document.querySelector(sel); if (el) el.textContent = (val ?? '-') || '-'; }

function esc(s) { return String(s || '').replace(/&/g,'&amp;').replace(/</g,'&lt;').replace(/>/g,'&gt;'); }

function joinDash(a,b){ const aa=(a||'').toString().trim(); const bb=(b||'').toString().trim(); return aa&&bb?`${aa}-${bb}`:aa||bb||''; }

function normalizeDim(s){ return s?String(s).replace(/x/gi,'×').replace(/\s+/g,'') : ''; }

function displayCompany(item){ return item?.storageCompanyInfo?.CompanyShortName || item?.storageCompanyInfo?.CompanyName || item?.storageCompany || 'YSD'; }

function highlightSelection(node){ document.querySelectorAll('.is-selected').forEach(n=>n.classList.remove('is-selected')); if (node) node.classList.add('is-selected'); }

window.UIRenderer = UiRenderer;

if (document.readyState === 'loading') {

document.addEventListener('DOMContentLoaded', () => UiRenderer.init());

} else {

UiRenderer.init();

}

})();