```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import { cloneDeep } from "lodash";
import RolePermitService from "services/RolePermitService";

export const getRoleByScope = createAsyncThunk(
  "role/getRoleByScope",
  async (id, { rejectWithValue }) => {
    try {
      const response = await RolePermitService.getRoleByScope(id);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getRoleDetail = createAsyncThunk(
  "role/getRoleDetail",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await RolePermitService.getRoleDetail(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const addRoleApi = createAsyncThunk(
  "role/addRoleApi",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await RolePermitService.addRole(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const updateRoleApi = createAsyncThunk(
  "role/updateRoleApi",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await RolePermitService.updateRole(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const updateRolePermitApi = createAsyncThunk(
  "role/updateRolePermitApi",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, tmpPermit } = data;
      const response = await RolePermitService.updateRolePermit(tmpPermit);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const deleteRoleApi = createAsyncThunk(
  "role/deleteRoleApi",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await RolePermitService.deleteRole(id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

const initialState = {
  loading: false,
  roleByScopeList: [],
  roleDetail: {},
  roleByPermitList: []
};

export const rolePermitSlice = createSlice({
  name: "role",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
    setRoleDetail: (state) => {
      state.roleDetail = state.payload;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(getRoleByScope.pending, (state) => {
        state.loading = true;
      })
      .addCase(getRoleByScope.fulfilled, (state, action) => {
        state.loading = false;
        state.roleByScopeList = action.payload;
      })
      .addCase(getRoleByScope.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getRoleDetail.pending, (state) => {
        state.loading = true;
      })
      .addCase(getRoleDetail.fulfilled, (state, action) => {
        state.loading = false;
        state.roleDetail = action.payload;
      })
      .addCase(getRoleDetail.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateRoleApi.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateRoleApi.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateRoleApi.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateRolePermitApi.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateRolePermitApi.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateRolePermitApi.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteRoleApi.pending, (state) => {
        state.loading = true;
      })
      .addCase(deleteRoleApi.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteRoleApi.rejected, (state, action) => {
        state.loading = false;
      });
  },
});

export const { showLoading, setRoleDetail } = rolePermitSlice.actions;

export default rolePermitSlice.reducer;
```