```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import employmentContractService from "services/EmploymentContractService";
import { cloneDeep } from "lodash";
// Async thunk to fetch contract types
export const getTypeContract = createAsyncThunk(
  "contractTypes/getTypeContract",
  async (_, { rejectWithValue }) => {
    try {
      const response = await employmentContractService.GetTypeContract();
      return response.data;
    } catch (err) {
      console.error("API call failed:", err);
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const updateContractType = createAsyncThunk(
  "contractTypes/updateContractType",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await employmentContractService.UpdateContractType(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);

export const createContractType = createAsyncThunk(
  "contractTypes/createContractType",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await employmentContractService.CreateContractType(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);

export const deleteContractType = createAsyncThunk(
  "contractTypes/deleteContractType",
  async (data, { rejectWithValue }) => {
    const { onSuccess, id } = data;
    try {
      const response = await employmentContractService.DeleteContractType(id);
      if (onSuccess) onSuccess(response);
      return id;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);

const initialState = {
  loading: false,
  contractTypeList: [],
  contractTypeListSetting: [],
};
```

```
const contractTypesSlice = createSlice({
  name: "contractTypes",
  initialState,
  reducers: {},
  extraReducers: (builder) => {
    builder
      .addCase(getTypeContract.pending, (state) => {
        state.loading = true;
      })
      .addCase(getTypeContract.fulfilled, (state, action) => {
        state.loading = false;
        state.contractTypeList = action.payload;
        state.contractTypeListSetting = [
          { action: "initial", isRequired: true },
          ...action.payload,
        ];
      })
      .addCase(getTypeContract.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateContractType.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateContractType.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateContractType.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(createContractType.pending, (state) => {
        state.loading = true;
      })
      .addCase(createContractType.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(createContractType.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteContractType.pending, (state) => {
        state.loading = true;
      })
      .addCase(deleteContractType.fulfilled, (state) => {
        state.loading = false;
      })
      .addCase(deleteContractType.rejected, (state, action) => {
        state.loading = false;
      });
  },
});

export default contractTypesSlice.reducer;
```