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

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

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

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

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

export const deleteDsBangPhuCapGiamTru = createAsyncThunk(
  "bangPhuCapGiamTru/deleteDsBangPhuCapGiamTru",
```

```
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await QuanLyPhuCapGiamTruService.deleteBangPhuCapGiamTru(
        id
      );
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

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

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

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

export const getBangPhuCapGiamTruSystem = createAsyncThunk(
  "bangPhuCapGiamTru/getBangPhuCapGiamTruSystem",
  async (branchId, { rejectWithValue }) => {
    try {
      const response =
        await QuanLyPhuCapGiamTruService.getBangPhuCapGiamTruSystem(branchId);
      return response.data;
```

```
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const DeleteBangPhuCapGiamTruSystem = createAsyncThunk(
  "bangPhuCapGiamTru/DeleteBangPhuCapGiamTruSystem",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response =
        await QuanLyPhuCapGiamTruService.delBangPhuCapGiamTruSystem(id);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const CopyBangPhuCapGiamTru = createAsyncThunk(
  "bangPhuCapGiamTru/CopyBangPhuCapGiamTru",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await QuanLyPhuCapGiamTruService.copyBangPhuCapGiamTru(
        id
      );
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const deleteNhanVienById = createAsyncThunk(
  "bangPhuCapGiamTru/deleteNhanVienById",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await QuanLyPhuCapGiamTruService.delNhanVienById(id);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

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

const initialState = {
  loading: false,
  bangPhuCapGiamTru: {},
  bangPhuCapGiamTruDetail: {},
  chiTietPhuCapGiamTruList: [],
  danhMucPhuCapGiamTruList: [],
```

```
  bangPhuCapGiamTruCate: [],
};

export const bangPhuCapGiamTruSlice = createSlice({
  name: "bangPhuCapGiamTru",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
    setChiTietPhuCapGiamTru: (state, action) => {
      state.chiTietPhuCapGiamTruList = action.payload;
    },
    setBangPhuCapGiamTruDetail: (state, action) => {
      state.bangPhuCapGiamTruDetail = action.payload;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(searchDsBangPhuCapGiamTru.fulfilled, (state, action) => {
        state.loading = false;
        state.bangPhuCapGiamTru = action.payload;
      })
      .addCase(searchDsBangPhuCapGiamTru.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(searchDsBangPhuCapGiamTru.pending, (state) => {
        state.loading = true;
      })

      .addCase(getByIdDsBangPhuCapGiamTru.fulfilled, (state, action) => {
        state.loading = false;
        state.bangPhuCapGiamTruDetail = action.payload;
      })
      .addCase(getByIdDsBangPhuCapGiamTru.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getByIdDsBangPhuCapGiamTru.pending, (state) => {
        state.loading = true;
      })

      .addCase(addNewDsBangPhuCapGiamTru.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(addNewDsBangPhuCapGiamTru.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(addNewDsBangPhuCapGiamTru.pending, (state) => {
        state.loading = true;
      })

      .addCase(CopyBangPhuCapGiamTru.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(CopyBangPhuCapGiamTru.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(CopyBangPhuCapGiamTru.pending, (state) => {
        state.loading = true;
      })

      .addCase(updateDsBangPhuCapGiamTru.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateDsBangPhuCapGiamTru.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateDsBangPhuCapGiamTru.pending, (state) => {
        state.loading = true;
      })

      .addCase(deleteDsBangPhuCapGiamTru.fulfilled, (state, action) => {
        state.loading = false;
      })
```

```
      .addCase(deleteDsBangPhuCapGiamTru.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteDsBangPhuCapGiamTru.pending, (state) => {
        state.loading = true;
      })

      .addCase(changeStateDsBangPhuCapGiamTru.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(changeStateDsBangPhuCapGiamTru.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(changeStateDsBangPhuCapGiamTru.pending, (state) => {
        state.loading = true;
      })

      .addCase(searchChiTietBangPhuCapGiamTru.fulfilled, (state, action) => {
        state.loading = false;
        state.chiTietPhuCapGiamTruList = action.payload.data;
        state.danhMucPhuCapGiamTruList = action.payload.danhSachPhuCapGiamTru;
        state.bangPhuCapGiamTruCate = action.payload;
      })
      .addCase(searchChiTietBangPhuCapGiamTru.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(searchChiTietBangPhuCapGiamTru.pending, (state) => {
        state.loading = true;
      })
      .addCase(deleteNhanVienById.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteNhanVienById.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteNhanVienById.pending, (state) => {
        state.loading = true;
      });
  },
});

export const {
  showLoading,
  setChiTietPhuCapGiamTru,
  setBangPhuCapGiamTruDetail,
} = bangPhuCapGiamTruSlice.actions;

export default bangPhuCapGiamTruSlice.reducer;
```