

blocked URL

```
public class FilterProvider extends AbstractCustomFilterWidgetProvider {
    /**
     *
     */
    @Override
    public String getName() {
        try {
            Locale currentLocale = null;
            // base name .properties
            // LocaleFinder base name
            String resourceBaseName = "ru/soundbi/filters/datesliderfilter/resource/locale/locale";
            try {
                InterProviderFactory.getProvider().addResource(resourceBaseName);
            } catch (Exception e) {
            };
            HttpServletRequest req = null;
            try {
                if (RequestContextHolder.getRequestAttributes() != null) {
                    req = ((ServletRequestAttributes) RequestContextHolder.getRequestAttributes()).getRequest();
                }
            } catch (Exception e) {
                //
            }
            if (req != null) {
                currentLocale = ProviderFactory.INSTANCE.getInternationalProvider().getClientLocale(req);
            }
            //
            if (currentLocale == null) {
                return "Plugin-Xml-I18n-Custom-Date-Slider-Plugin-Name";
            }

            // getLocText(String text, Locale locale)
            String key = "Plugin-Xml-I18n-Custom-Date-Slider-Plugin-Name";
            String localizedText = InterProviderFactory.getProvider().getLocText(key, currentLocale);
            return localizedText;
        } catch (Exception e) {
            //
        }
        return "Plugin-Xml-I18n-Custom-Date-Slider-Plugin-Name";
    }

    /**
     * xtype
     */
    @Override
    public String getType() {
        return "bi.custom.date.dateslider";
    }

    /**
     *
     */
    @Override
    public String getIcon() {
        return "http://webapi.amap.com/theme/v1.3/mapinfo_05.png";
    }

    /**
     * ()

```

```

    */
    @Override
    public String getCustomTool() {
        return "";//"bi.plugin.testwidget";
    }

    /**
     * html ()
     */
    @Override
    public String getPreviewPageHTML(OperationContext context) {
        return "<link rel=\"stylesheet\" type=\"text/css\" href=\"https://fanruan.design/fineui/2.0/fineui.min.
css\" />" +
            "<script src=\"https://fanruan.design/fineui/2.0/fineui.min.js\"></script>" +
            "<div>context: </div>" + context.getSystemInfo() + context.getUserInfo() +
            "<div id=\"container\"></div>";
    }

    /**
     * html ()
     */
    @Override
    public String getEditPageHTML(OperationContext context) {
        return "<link rel=\"stylesheet\" type=\"text/css\" href=\"https://fanruan.design/fineui/2.0/fineui.min.
css\" />" +
            "<script src=\"https://fanruan.design/fineui/2.0/fineui.min.js\"></script>" +
            "<div>context: </div>" + context.getSystemInfo().getServletURL() + context.getUserInfo().
getDisplayName() +
            "<div id=\"container\"></div>";
    }

    @Override
    public AssembleComponent previewClient(OperationContext context) {
        return FilterComponent.KEY;
    }

    /**
     *
     * For custom text filter widget, use BICommonConstants.COLUMN.STRING
     * For custom date range filter widget, use BICommonConstants.COLUMN.DATE
     * For custom number filter widget, use BICommonConstants.COLUMN.NUMBER
     */
    @Override
    public int getFilterFieldType() {
        return BICommonConstants.COLUMN.DATE;
    }
}

```

[blocked URL](#)

[blocked URL](#)

```

import { shortcut, store, extend } from '../core';
import { ControlFilterModel } from './control.filter.model';
import './HTMLComponents/date-picker.js';
import './HTMLComponents/scale-selector.js';
import './HTMLComponents/date-range-picker.js';
import './HTMLComponents/date-range-scale-selector.js';
import { ErrorCode, RES_STATUS } from '../contsant/index';

import { formatDate, getControlUsedMeasuresAndTableNames, setCssScale, setTransformScale } from '../utils';

```

```

export interface ControlStringSingleProps {
  baseCls: string;
  $testId: string;
  height: number;
  width?: number;
  isPreviewMode: boolean;
  cssScaleGetter: () => number;
}

@shortcut()
@store(ControlFilterModel)
export class ControlDateIntervalFilter extends BI.Widget {
  static xtype = 'bi.custom.date.dateslider';

  static EVENT = {
    EVENT_CHANGE: 'EVENT_CHANGE',
  };

  props: ControlStringSingleProps = {
    baseCls: 'bi-date-interval-control',
    $testId: 'bi-date-interval-control',
    isPreviewMode: false, //
    cssScaleGetter: () => 1,
  };

  model: ControlFilterModel['model'];
  store: ControlFilterModel['store'];

  watch = {
    disabled: () => {},
  };
  setMinMaxDate() {
    this.element[0].setAttribute('min-date',this.minData||'2022-01-01');
    this.element[0].setAttribute('max-date',this.maxData||'2022-12-01');

    console.log("setting minmax",this.minData,this.maxData);
  }

  mounted() {
    console.log('hello world1234',this);

    this.getMinMaxDate();
    //this.setValue(this.model.widgetHelper?.getResultWidgetValue(true, this.model.value) ?? this.model.value);
    try{
      this.element[0].closest('.bi-show-widget-factory.bi-control-widget,.bi-abs.bi-control-widget.bi-card').style.setProperty("overflow","visible");
    }catch(e){}
    try{
      this.element[0].closest('.bi-fit-widget.bi-export-widget').style.setProperty("z-index","100");
    }catch(e){}
    this.element[0].style.setProperty('height','24px');
    setTimeout(()=>{
      // this.element[0].style.setProperty('position',"absolute");
      this.element[0].style.setProperty('top',"0");
      this.element[0].style.setProperty('left',"0");
      this.element[0].style.setProperty('right',"0");
      this.element[0].style.setProperty('bottom',"0");
      this.element[0].style.setProperty('height','100%');

      this.element[0].style.setProperty('--scale-text-color','#2c60db');
      this.element[0].style.setProperty('--scale-border-color', '#2c60db');
      this.element[0].style.setProperty('--scale-tick-color', '#2c60db');
      this.element[0].style.setProperty('--collapse-btn-color', '#2c60db');
      this.element[0].style.setProperty('--collapse-icon-color', '#2c60db');
      this.element[0].style.setProperty('--collapse-btn-color-hover', '#2c60db');
      this.element[0].style.setProperty('--background-item-color', '#f2f7fe');
      this.element[0].style.setProperty('--higher-periods-active-color', '#cdelfc');
    }

```

```

    },200)

}

_defaultConfig() {
  console.log('default config 4');
  const conf = super._defaultConfig(...arguments);
  return extend(conf, {
    baseCls: '',
    el: null,
    tagName: 'date-range-scale-selector',

    listeners: [
      {
        eventName: 'range-changed',
        action(e) {
          // console.log("here we got message from ",e);
          // this.fireEvent(ControlDateInterval.EVENT.EVENT_CHANGE, {start:e.startDate,end:e.endDate});
        },
      },
    ],
  });
}

date_value_to_str(v){
  if(v==undefined)return undefined;
  let p0=(x=>`${x<10?'0':''}${x}`);
  return `${p0(v.year)}-${p0(v.month)}-${p0(v.day)}`;
}

//
private getMinMaxDate(): Promise<{minData: any, maxData: any}> {
  console.log('test', this.model)

  return BI.Utls.getControlDefaultValueByWidgetInfo([
    {...this.getRequestParams(), originDefaultValue: 1, measuresConfig: []}
  ], this.model.templateHelper).then((minRes) => {
    const minData = minRes.data;
    console.log('', minData);

    // Promise
    return BI.Utls.getControlDefaultValueByWidgetInfo(
      templateHelper).then((maxRes) => {
        const maxData = maxRes.data;
        this.minData=this.date_value_to_str(minData[Object.keys(minData)[0]].value);
        this.maxData=this.date_value_to_str(maxData[Object.keys(maxData)[0]].value);
        //console.log('min-max',this.date_value_to_str,this.date_value_to_str(minData[Object.keys
(minData)[0]].value), this.model, this.model.minData, maxData[Object.keys(maxData)[0]],maxData[Object.keys
(maxData)[0]].value, this.model.maxData,Object.keys(minData)[0],minData);
        if(this.element){
          this.element[0].setAttribute('min-date', this.minData);
          this.element[0].setAttribute('max-date', this.maxData);
        }
        return { minData, maxData };
      });
    }).catch((error) => {
      console.error(':', error);
    });
  }

//
private getRequestParams () {
  const measurePoolHelper = this.model.templateHelper.getMeasurePoolHelper();
  const measureHelper = measurePoolHelper.getMeasureHelper();
  // dimensions widget
  let fieldIds: string[] = [];
  BI.each(this.model.dimensions, (_, dimension) => {
    const { fieldId, sort } = dimension;

    fieldIds = [...fieldIds, fieldId, ...measureHelper.getAllUsedCalTargets(fieldId)];
  });
}

```

```

if (sort?.targetFieldId) {
  fieldIds = [
    ...fieldIds,
    sort.targetFieldId,
    ...measureHelper.getAllUsedCalTargets(sort.targetFieldId),
  ];
}
});

const { measures, generaTableNames } = getControlUsedMeasuresAndTableNames(fieldIds, measureHelper);
console.log('test11', measures, this.model);
const tableNames = this.model.selectedTable;
const relationTableNames = BI.flatten(
  this.model.templateHelper.getPoolHelper().getRelationModelByTableNames(tableNames)
);

return {
  wId: this.model.wId,
  dimensions: this.model.dimensions,
  view: this.model.view,
  value: this.model.value,
  type: this.model.type,
  tableName: BI.uniq(
    BI.concat(
      tableNames,
      generaTableNames.filter(tableName => relationTableNames.includes(tableName))
    )
  ),
  measuresConfig: [],
  dateIntervalValue: this.model.dateIntervalValue,
  useDateInterval: this.model.useDateInterval,
  showTime: this.model.showTime,
};
}

render() {
  // const { width, height, isPreviewMode } = this.options;
  // super.render();
  console.log("  this.element",this.element,JSON.stringify(this.model.value));

  this.element.attr({"min-date":'2020-01-01',
    "max-date":'2026-01-01',
    'granularity':"month",
    "start-date":this.date_value_to_str(this.model.value?.start?.value)||'2022-01-01',
    "end-date":this.date_value_to_str(this.model.value?.end?.value)||'2022-12-01',
    'week-numbering':'ISO', 'first-day-of-week':'1'});
  this.element.on("range-changed", (e:any)=>{
    console.log("got event e",e);
    let ev={start:e.originalEvent.detail.startDate,end:e.originalEvent.detail.endDate,};
    ev.start=ev.start.split("T")[0];
    ev.end=ev.end.split("T")[0];
    if((ev.start=="")||(ev.end=="")){
      console.log("fire null");
      this.fireEvent(ControlDateIntervalFilter.EVENT.EVENT_CHANGE, null);
      return;
    }
    ev.start={
      type:1,value:{
        day:parseInt(ev.start.split("-")[2]),
        month:parseInt(ev.start.split("-")[1]),
        year:parseInt(ev.start.split("-")[0])
      }
    }
    ev.end={
      type:1,value:{
        day:parseInt(ev.end.split("-")[2]),
        month:parseInt(ev.end.split("-")[1]),
        year:parseInt(ev.end.split("-")[0])
      }
    }
    this.model.collapse=e.originalEvent.detail.collapse;
    this.model.granularity=e.originalEvent.detail.granularity;
    console.log("fire ",ev);
  });
}

```

```

        this.fireEvent(ControlDateIntervalFilter.EVENT.EVENT_CHANGE, ev);
    });
    console.log("----->",this.element,this.options);

    this.element[0].style.overflow="visible";
    this.element[0].style.height="24px";
}

setValue(v?: any) {
    console.log("Somebody wants from us to set Value",v);
    if(v.start){
        this.element[0].setAttribute('start-date',this.date_value_to_str(v.start.value));
    }
    if(v.end){
        this.element[0].setAttribute('end-date',this.date_value_to_str(v.end.value));
    }
}

getValue() {}

reset() {
    this.setValue();
}

_itemsCreator(options: { type: number }, callback: Function) {
    console.log('hello this.model', this.model);

    if (this.model?.useCustom && BI.isNotEmptyArray(this.model?.customValue)) {
        //
        callback({
            items: this.createItemsByData(this.model?.customValue, options, true),
        });
    } else if (BI.size(this.model?.dimensions) === 0) {
        callback([]);
    } else {
        console.log('hello data');
        BI.Utils.getControlWidgetDataByWidgetConfig(
            this.model?.widgetConfig,
            (res: any) => {
                const { data = { value: [], hasNext: false }, errorCode } = res;
                console.log('data11@', data, res);
                if (errorCode === ErrorCode.STRING_SEARCH_OUT_OF_LIMIT) {
                    callback({
                        tipText: BI.i18nText('BI-Design_Search_Result_More_Than_10000'),
                    });

                    return;
                }
                if (errorCode === RES_STATUS.CACHE_MISS) {
                    callback({
                        tipText: BI.i18nText('BI-Conf_Cache_Loading_Retry_Later'),
                    });

                    return;
                }
                if (options.type === BI.MultiSelectCombo.REQ_GET_ALL_DATA) {
                    callback({
                        items: this.createItemsByData(data.value, options),
                    });

                    return;
                }
                if (options.type === BI.MultiSelectCombo.REQ_GET_DATA_LENGTH) {
                    callback({ count: data.value });

                    return;
                }
            }
        );
    }
}

```

```

        callback({
            items: this.createItemsByData(data.value, options),
            hasNext: data.hasNext,
        });
    },
    { textOptions: options },
    null,
    this.model.templateHelper
);
}
}

private setScale(el: any) {
    /*const { cssScaleGetter } = this.options;

    setCssScale(el, cssScaleGetter());
    setTransformScale(el, this.model.templateHelper.getReportScale());*/
}

private createItemsByData(valueArray: string[], options: Obj, isCustomValue = false) {
    console.log('valueArray', valueArray, options, isCustomValue);
    return createControlWidgetItemsByData(
        valueArray,
        options.keywords || [],
        options.selectedValues || [],
        isCustomValue
    );
}
}
}

```

<https://github.com/finereport-overseas/plugin-bi-custom-filter-widget>