1.在项目引用表格模块时，使用`registerPluginComponent`方法注册插件组件。

```
import cbbOnlineTable, { registerPluginComponent } from "@tiamaes/cbb-online-table";
Vue.use(cbbOnlineTable,{
  request,
  router,
  store,
  i18n,
});

import pluginComp1  from "@/components/plugin/address";
import pluginComp2  from "@/components/plugin/chooseMap";
// 单个注册
registerPluginComponent(pluginComp1)
// 批量注册
registerPluginComponent([
  pluginComp1,
  pluginComp2,
])
```

2.插件的组成

- json配置
- 设计器
- 解析器

2.1 json配置

```
export default {
  designer: configForm,   //  必填。 组件类型。设计器右侧配置表单，接收的 props: ['activeData'], 当前的配置
  render: render,     //  与 __config__.tag 二选一必填。组件类型。渲染器，接收的 props: ['value', ...自定义配置]
  //  必填。基础配置信息
  __config__: {
    plugin: true,                  // 必填。true标识插件控件
    widgetKey: "plugin1",          // 必填。组件标识，不能与内置重复，后期不能变更
    // tag: "el-input",               // 与 render 二选一必填。组件渲染的html标签名称。 如 el-input
    label: "省市区域",           // 必填。组件名称
    layout: "colFormItem",         // 必填。 默认 colFormItem
    span: 24,                      // 栅格宽度
    tagIcon: "icon-ym icon-ym-generator-Provinces",  // 控件在设计器中显示的图标 <i> 标签能被识别到的
    labelWidth: undefined,
    showLabel: true,              // 渲染时是否展示左侧label
    required: false,                // 是否必填
    defaultValue: '天津市/市辖区/和平区/新兴街道',  // 默认值
    dragDisabled: false,             // 是否可拖拽
    visibility: ["pc", "app"],       // 移动端是否也可用
    regList: [],                     // 正则
    noShow: false,                   // 是否隐藏
    trigger: "change",               // 校验触发时机
  },
  // 自定义事件监听。 非必填
  on: {
    change:
      "({ data, formData, setFormData, setShowOrHide, setRequired, setDisabled,
request }) => {\n    // 在此编写代码\n    \n}",
    xdj:
      "({ data, formData, setFormData, setShowOrHide, setRequired, setDisabled,
request }) => {\n    // 在此编写代码\n    \n}"
  },
  // 自定义事件的描述雨。非必填， 需和 on 属性里的 key 对应
  onDesc: {
    change: '我想在 change 时触发',
    xdj: '我想在 chanxdj 事件里触发',
  },
  // .... 标签上自定义属性，
  style: { width: "100%" },
  placeholder: "请选择",
  disabled: false,
  clearable: true,
  filterable: false,
  multiple: false,
  level: 2
}
```

2.2 设计器

接收的 props: ['activeData'], 当前的配置

```
<template>
  <el-row>
    <el-form-item label="控件标题">
      <el-input v-model="activeData.__config__.label" placeholder="请输入控件标题"
/>
    </el-form-item>
    <el-form-item label="占位提示">
      <el-input v-model="activeData.placeholder" placeholder="请输入占位提示" />
    </el-form-item>
    <el-form-item label="格式">
      <el-radio-group v-model="activeData.level" @change="levelChange"
class="level">
        <el-radio :label="0">省</el-radio>
        <el-radio :label="1">省市</el-radio>
        <el-radio :label="2">省市区</el-radio>
        <el-radio :label="3">省市区-街道</el-radio>
      </el-radio-group>
    </el-form-item>
    <el-form-item label="能否清空">
      <el-switch v-model="activeData.clearable" />
    </el-form-item>
    <el-form-item label="能否多选">
      <el-switch v-model="activeData.multiple" @change="levelChange" />
    </el-form-item>
    <el-form-item label="是否禁用">
      <el-switch v-model="activeData.disabled" />
    </el-form-item>
    <el-form-item label="是否必填">
      <el-switch v-model="activeData.__config__.required" />
    </el-form-item>
  </el-row>
</template>
<script>
export default {
  props: ['activeData'],    // 当前组件的 config 配置
  data() {
    return {}
  },
  created() { },
  methods: {
    levelChange() {
      this.$set(this.activeData.__config__, 'defaultValue', [])
      this.activeData.__config__.renderKey = +new Date() // 更新renderKey,重新渲染
该组件
    }
  }
}
</script>
<style lang="scss">
.level {
  .el-radio {
    display: block;
    margin: 10px 0;
  }
}
</style>
```

2.3 渲染器

接收的 props: ['value', ...自定义配置]

需要通过 `this.$emit("input", v);` 触发更新表单字段，来更新 v-model 的值。

```
<template>
  <div class="popupSelect-container">
    <div class="el-select" @click.stop="openDialog">
      <div
        class="el-select__tags"
        v-if="multiple"
        ref="tags"
        :style="{
          'max-width': inputWidth - 32 + 'px',
          width: '100%',
          cursor: 'pointer'
        }"
      >
        <span v-if="collapseTags && tagsList.length">
          <el-tag
            :closable="!selectDisabled"
            :size="collapseTagSize"
            type="info"
            @close="deleteTag($event, 0)"
            disable-transitions
          >
            <span class="el-select__tags-text">{{ tagsList[0] }}</span>
          </el-tag>
          <el-tag
            v-if="tagsList.length > 1"
            :closable="false"
            type="info"
            disable-transitions
          >
            <span class="el-select__tags-text"
              >+ {{ tagsList.length - 1 }}</span
            >
          </el-tag>
        </span>
        <transition-group @after-leave="resetInputHeight" v-if="!collapseTags">
          <el-tag
            v-for="(item, i) in tagsList"
            :key="item"
            :size="collapseTagSize"
            :closable="!selectDisabled"
            type="info"
            @close="deleteTag($event, i)"
            disable-transitions
          >
            <span class="el-select__tags-text">{{ item }}</span>
          </el-tag>
        </transition-group>
      </div>
      <el-input
        ref="reference"
        v-model="innerValue"
        type="text"
        :placeholder="currentPlaceholder"
```

```
          :disabled="selectDisabled"
          readonly
          :validate-event="false"
          :tabindex="multiple ? '-1' : null"
          @mouseenter.native="inputHovering = true"
          @mouseleave.native="inputHovering = false"
        >
          <template slot="suffix">
            <i
              v-show="!showClose"
              :class="['el-select__caret', 'el-input__icon', 'el-icon-arrow-up']"
            ></i>
            <i
              v-if="showClose"
              class="el-select__caret el-input__icon el-icon-circle-close"
              @click="handleClearClick"
            ></i>
          </template>
        </el-input>
      </div>
      <vxe-modal
        title="省市区"
        :close-on-click-modal="false"
        v-model="visible"
        class="dev-table-dialog transfer-dialog"
        lock-scroll
        append-to-body
        width="800px"
        :modal-append-to-body="false"
        showFooter
        @close="onClose"
      >
        <div class="transfer__body">
          <div class="transfer-pane">
            <div class="transfer-pane__tools">
              <span>全部数据</span>
            </div>
            <div class="transfer-pane__body">
              <el-tree
                :data="treeData"
                :props="props"
                check-on-click-node
                class="dev-table-common-el-tree"
                node-key="code"
                v-loading="loading"
                @node-click="handleNodeClick"
              >
                <!-- :load="loadNode" -->
                <!-- lazy -->
                <!-- <span class="custom-tree-node" slot-scope="{ node, data }">
                  <i :class="data.icon"></i>
                  <span class="text">{{ node.name }}</span>
                </span> -->
              </el-tree>
            </div>
          </div>
          <div class="transfer-pane">
            <div class="transfer-pane__tools">
              <span>已选</span>
              <el-button @click="removeAll" type="text" class="removeAllBtn"
                >清空列表</el-button
              >
            </div>
            <div class="transfer-pane__body shadow right-pane">
              <template>
                <div
                  v-for="(item, index) in selectedData"
                  :key="index"
                  class="selected-item"
                >
                  <span>{{ item }}</span>
                  <i class="el-icon-delete" @click="removeData(index)"></i>
                </div>
              </template>
            </div>
          </div>
        </div>
        <span slot="footer" class="dialog-footer">
          <el-button @click="visible = false">{{
            $t("common.cancelButton")
          }}</el-button>
          <el-button type="primary" @click="confirm">{{
            $t("common.confirmButton")
          }}</el-button>
        </span>
      </vxe-modal>
    </div>
  </template>

  <script>
  import {
    addResizeListener,
    removeResizeListener
  } from "element-ui/src/utils/resize-event";
  import data from "./pcas-code.json";

  export default {
    name: "JNPF-Address",
    inject: {
      elForm: {
        default: ""
      },
      elFormItem: {
        default: ""
      }
    },
    props: {
      // 内置
      value: {
        default: () => []
      },
      level: {
        type: Number,
        default: 2
      },
      placeholder: {
        type: String,
        default: "请选择"
      },
      disabled: {
        type: Boolean,
        default: false
      },
      multiple: {
        type: Boolean,
        default: false
      },
      collapseTags: {
        type: Boolean,
        default: false
      },
      clearable: {
        type: Boolean,
        default: false
      },
      size: String
    },
    data() {
      return {
        treeData: data,
        visible: false,
        loading: false,
        nodeId: "",
        innerValue: "",
        props: {
          children: "children",
          label: "name",
          isLeaf: "isLeaf"
        },
        selectedData: [],
        selectedIds: [],
        tagsList: [],
        inputHovering: false,
        inputWidth: 0,
        initialInputHeight: 0
      };
    },
    computed: {
      showClose() {
        let hasValue = Array.isArray(this.value) && this.value.length > 0;
        let criteria =
          this.clearable &&
          !this.selectDisabled &&
          this.inputHovering &&
          hasValue;
        return criteria;
      },
      currentPlaceholder() {
        if (this.multiple && Array.isArray(this.value) && this.value.length) {
          return "";
        } else {
          return this.placeholder;
        }
      },
      selectDisabled() {
        return this.disabled || (this.elForm || {}).disabled;
      },
      _elFormItemSize() {
        return (this.elFormItem || {}).elFormItemSize;
      },
      selectSize() {
        return this.size || this._elFormItemSize || (this.$ELEMENT || {}).size;
      },
      collapseTagSize() {
        return ["small", "mini"].indexOf(this.selectSize) > -1 ? "mini" : "small";
      }
    },
    created() {
      this.nodeId = "-1";
      this.initData();
    },
    mounted() {
      addResizeListener(this.$el, this.handleResize);

      const reference = this.$refs.reference;
      if (reference && reference.$el) {
        const sizeMap = {
          medium: 36,
          small: 32,
          mini: 28
        };
        const input = reference.$el.querySelector("input");
        this.initialInputHeight =
          input.getBoundingClientRect().height || sizeMap[this.selectSize];
      }
      if (this.multiple) {
        this.resetInputHeight();
      }
      this.$nextTick(() => {
        if (reference && reference.$el) {
          this.inputWidth = reference.$el.getBoundingClientRect().width;
        }
      });
      this.setDefault();
    },
    beforeDestroy() {
      if (this.$el && this.handleResize)
        removeResizeListener(this.$el, this.handleResize);
    },
    watch: {
      value(val) {
        this.setDefault();
      },
      selectDisabled() {
        this.$nextTick(() => {
          this.resetInputHeight();
        });
      }
    },
    methods: {
      onClose() {},
      clear() {
        if (this.selectDisabled) return;
        this.innerValue = "";
        this.selectedData = [];
        this.selectedIds = [];
        this.tagsList = [];
        this.$emit("input", []);
        this.$emit("change", [], []);
      },
      openDialog() {
        if (this.selectDisabled) return;
        this.visible = true;
        this.nodeId = "-1";
        this.setDefault();
      },
      initData() {
        // this.loading = true;
        // getProvinceSelector(this.nodeId).then((res) => {
        //   this.treeData = res.data.list.map((value) => ({
        //     ...value,
        //     isLeaf: 0 >= this.level ? true : value.isLeaf,
        //   }));
        //   this.loading = false;
        // });
      },
      getNodePath(node) {
        let fullPath = [];
        const loop = node => {
          if (node.level) fullPath.unshift(node.data);
          if (node.parent) loop(node.parent);
        };
        loop(node);
        return fullPath;
      },
      handleNodeClick(data, node) {
        if (!node.isLeaf) return;
        const nodePath = this.getNodePath(node);
        let currId = nodePath.map(o => o.code);
        let currData = nodePath.map(o => o.name).join("/");
        if (this.multiple) {
          const boo = this.selectedIds.some(
            o => o.join("/") === currId.join("/")
          );
          if (boo) return;
          this.selectedIds.push(currId);
          this.selectedData.push(currData);
        } else {
          this.selectedIds = [currId];
          this.selectedData = [currData];
        }
      },
      removeAll() {
        this.selectedData = [];
        this.selectedIds = [];
      },
      removeData(index) {
        this.selectedData.splice(index, 1);
        this.selectedIds.splice(index, 1);
      },
      confirm() {
        let selectedData = [];
        for (let i = 0; i < this.selectedIds.length; i++) {
          let item = [];
          let selectedNames = this.selectedData[i].split("/");
          for (let j = 0; j < this.selectedIds[i].length; j++) {
            item.push({
              code: this.selectedIds[i][j],
              name: selectedNames[j]
            });
          }
          selectedData.push(item);
        }
        if (this.multiple) {
          this.innerValue = "";
          this.tagsList = JSON.parse(JSON.stringify(this.selectedData));
          // this.$emit("input", this.selectedIds);
          this.$emit("input", this.innerValue);
          this.$emit("change", this.selectedIds, selectedData);
        } else {
          this.innerValue = this.selectedData.join(",");
          // this.$emit("input", this.selectedIds[0]);
          this.$emit("input", this.innerValue);
          this.$emit("change", this.selectedIds[0], selectedData[0]);
          this.$emit("xdj", this.selectedIds[0], selectedData[0]);
        }
        this.visible = false;
      },
      setDefault() {
        if (!this.value || !this.value.length) {
          this.innerValue = "";
          this.selectedIds = [];
          this.selectedData = [];
          this.tagsList = [];
          return;
        }
        let selectedIds = this.multiple ? this.value : [this.value];
        this.selectedIds = JSON.parse(JSON.stringify(selectedIds));
        this.selectedData = JSON.parse(JSON.stringify(selectedIds));
        if (this.multiple) {
          this.innerValue = "";
          this.tagsList = JSON.parse(JSON.stringify(this.selectedData));
        } else {
          this.innerValue = this.selectedData.join(",");
        }
        this.$nextTick(() => {
          if (this.multiple) {
            this.resetInputHeight();
          }
        });

        // GetAreaByIds(selectedIds).then((res) => {
        //   this.selectedIds = JSON.parse(JSON.stringify(selectedIds));
        //   this.selectedData = res.data.map((o) => o.join("/"));
        //   if (this.multiple) {
        //     this.innerValue = "";
        //     this.tagsList = JSON.parse(JSON.stringify(this.selectedData));
        //   } else {
        //     this.innerValue = this.selectedData.join(",");
        //   }
        //   this.$nextTick(() => {
        //     if (this.multiple) {
        //       this.resetInputHeight();
        //     }
        //   });
        // });
      },
      deleteTag(event, index) {
        this.selectedData.splice(index, 1);
        this.selectedIds.splice(index, 1);
        this.confirm();
        event.stopPropagation();
      },
      handleClearClick(event) {
        this.selectedData = [];
        this.selectedIds = [];
        this.confirm();
        event.stopPropagation();
      },
      resetInputWidth() {
        this.inputWidth = this.$refs.reference.$el.getBoundingClientRect().width;
      },
      handleResize() {
        this.resetInputWidth();
        if (this.multiple) this.resetInputHeight();
      },
      resetInputHeight() {
        if (this.collapseTags) return;
        this.$nextTick(() => {
          if (!this.$refs.reference) return;
          let inputChildNodes = this.$refs.reference.$el.childNodes;
          let input = [].filter.call(
            inputChildNodes,
            item => item.tagName === "INPUT"
          )[0];
          const tags = this.$refs.tags;
          const tagsHeight = tags
            ? Math.round(tags.getBoundingClientRect().height)
            : 0;
          const sizeInMap = this.initialInputHeight || 40;
          input.style.height =
            this.selectedData.length === 0
              ? sizeInMap + "px"
              : Math.max(
                  tags ? tagsHeight + (tagsHeight > sizeInMap ? 6 : 0) : 0,
                  sizeInMap
                ) + "px";
        });
      },
      resetInputWidth() {
        this.inputWidth = this.$refs.reference.$el.getBoundingClientRect().width;
      },
      handleResize() {
        this.resetInputWidth();
        if (this.multiple) this.resetInputHeight();
      }
    }
  };
  </script>
```