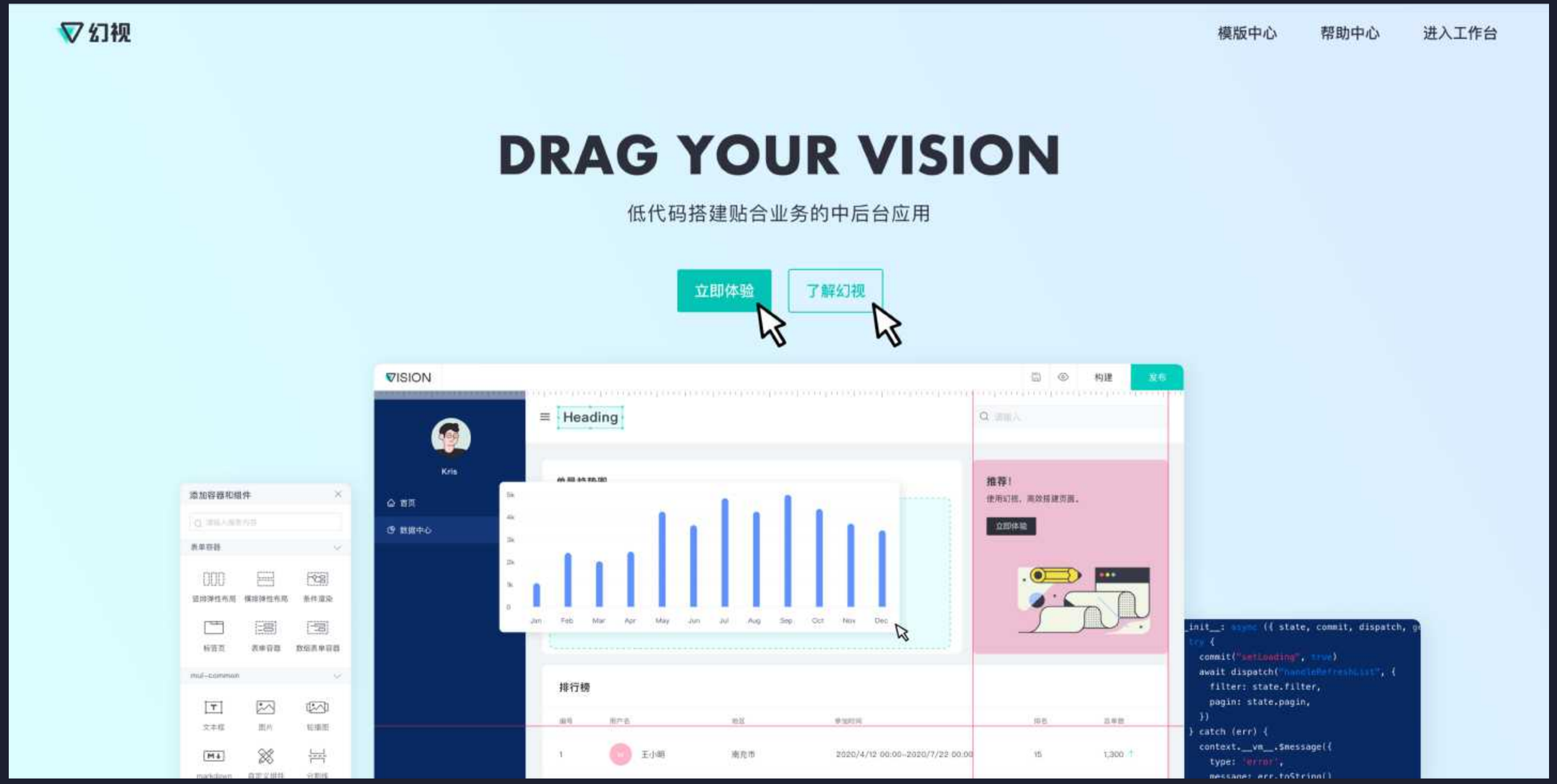
QCon+ 案例研习社

滴滴低代码平台建设及规模化探索

滴滴出行/刘宇

目录

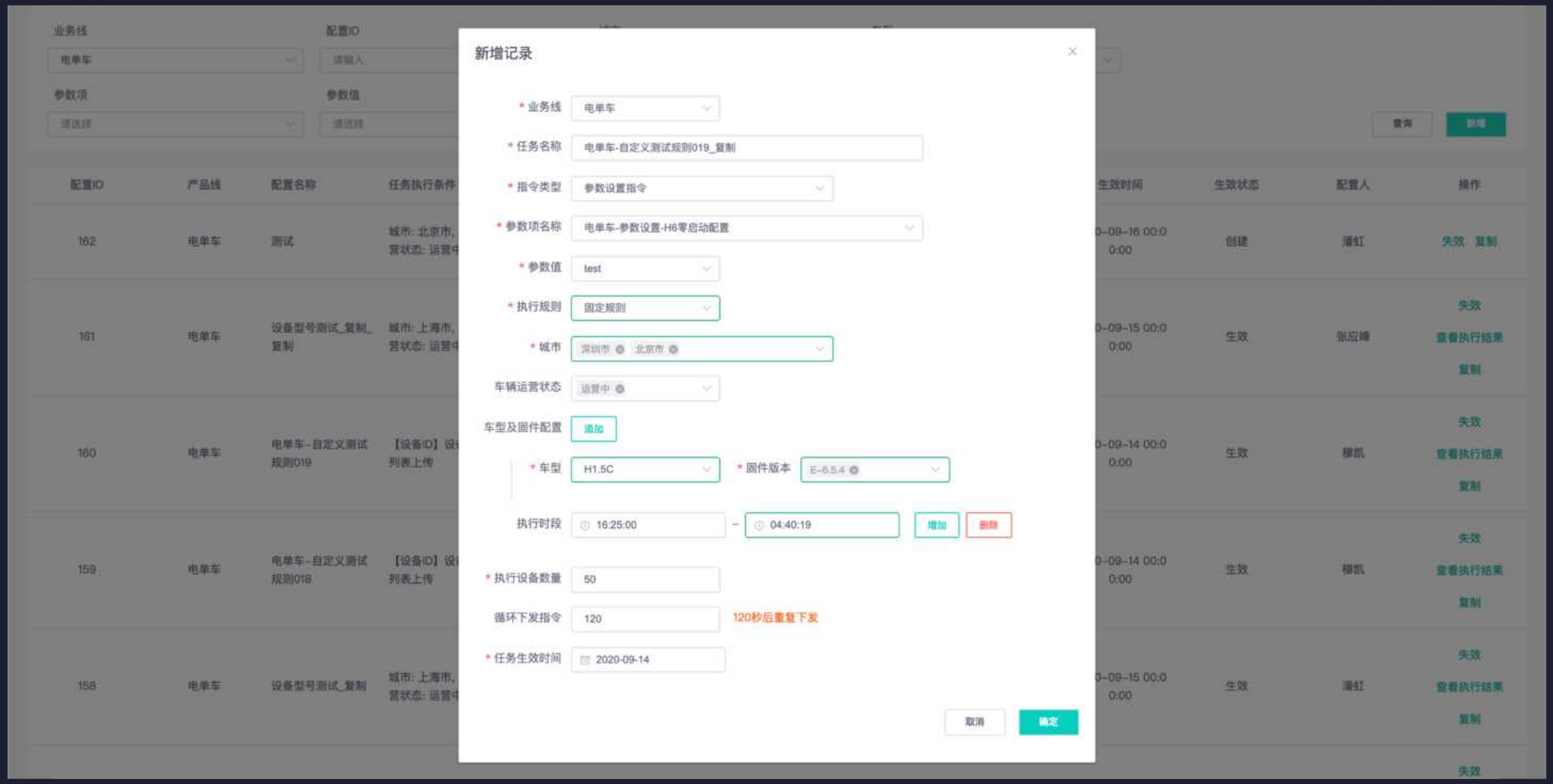
幻视业务现状



300+ 活跃用户、200+ 页面、300% 综合提效、L1 门槛降低

(2020/9 ~ 2020/12/31, 1/bu)

幻视核心产品特性



可视化布局编排、完备逻辑编写、一站式搭建

规模化的困难与阻力？

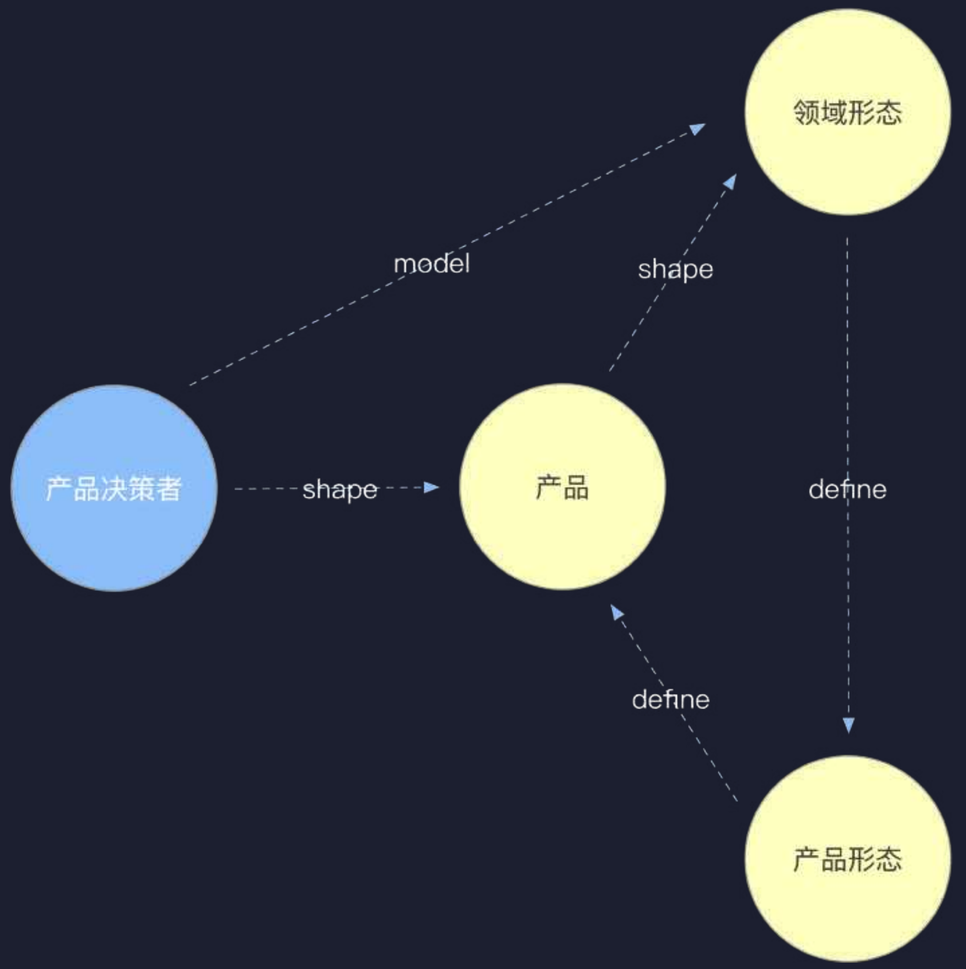
- ☑ 媲美 sketch 的画布
- ☑ 零代码表单/表格
- ☑ 可视化流程编排
- ☑ 丰富的物料/模板

《银河系漫游指南》, 2005

规模化

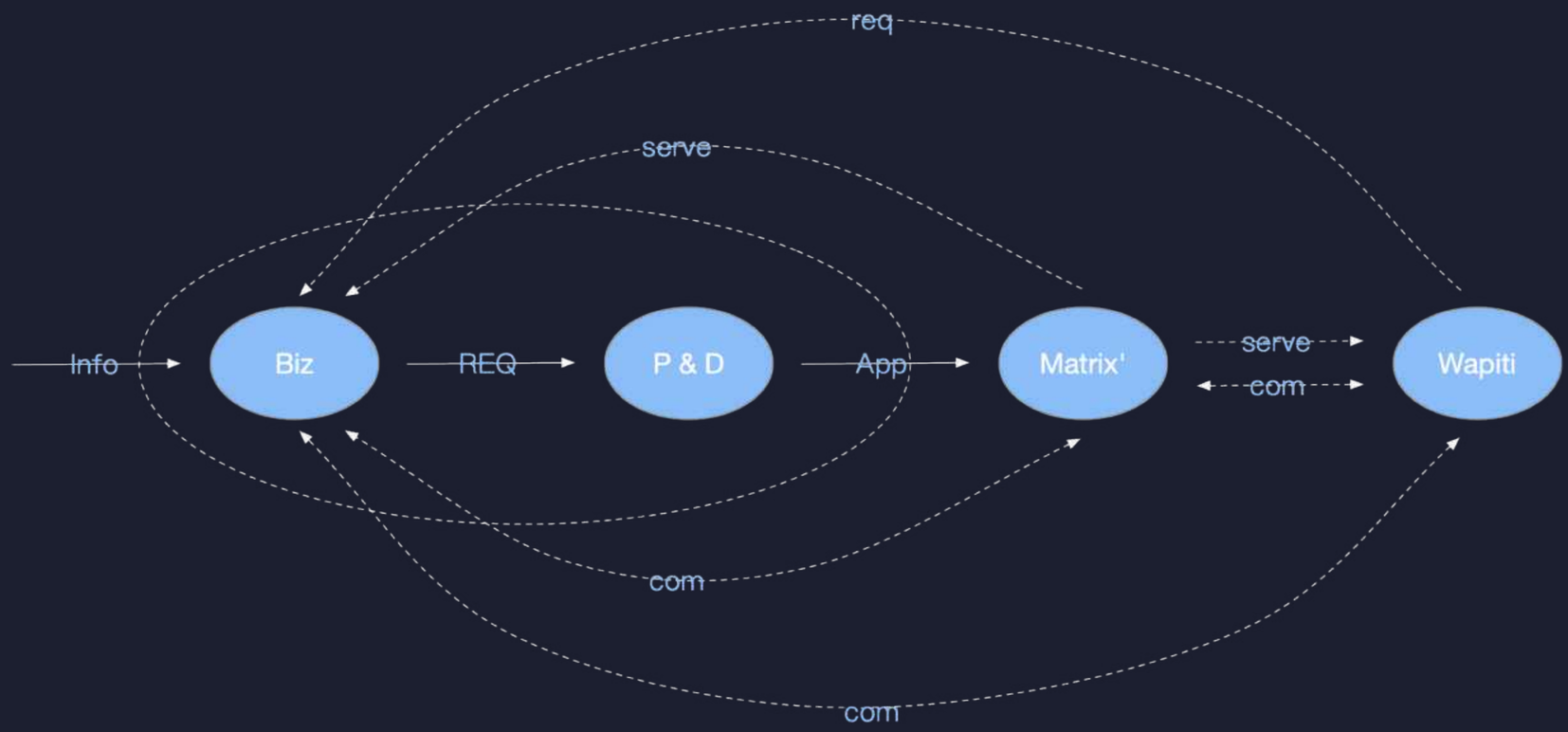
低代码开发平台到底解决什么问题？产品的核心竞争力是什么？
42 —— the answer to life, the universe, and everything

目录



传统软件/应用交付形态

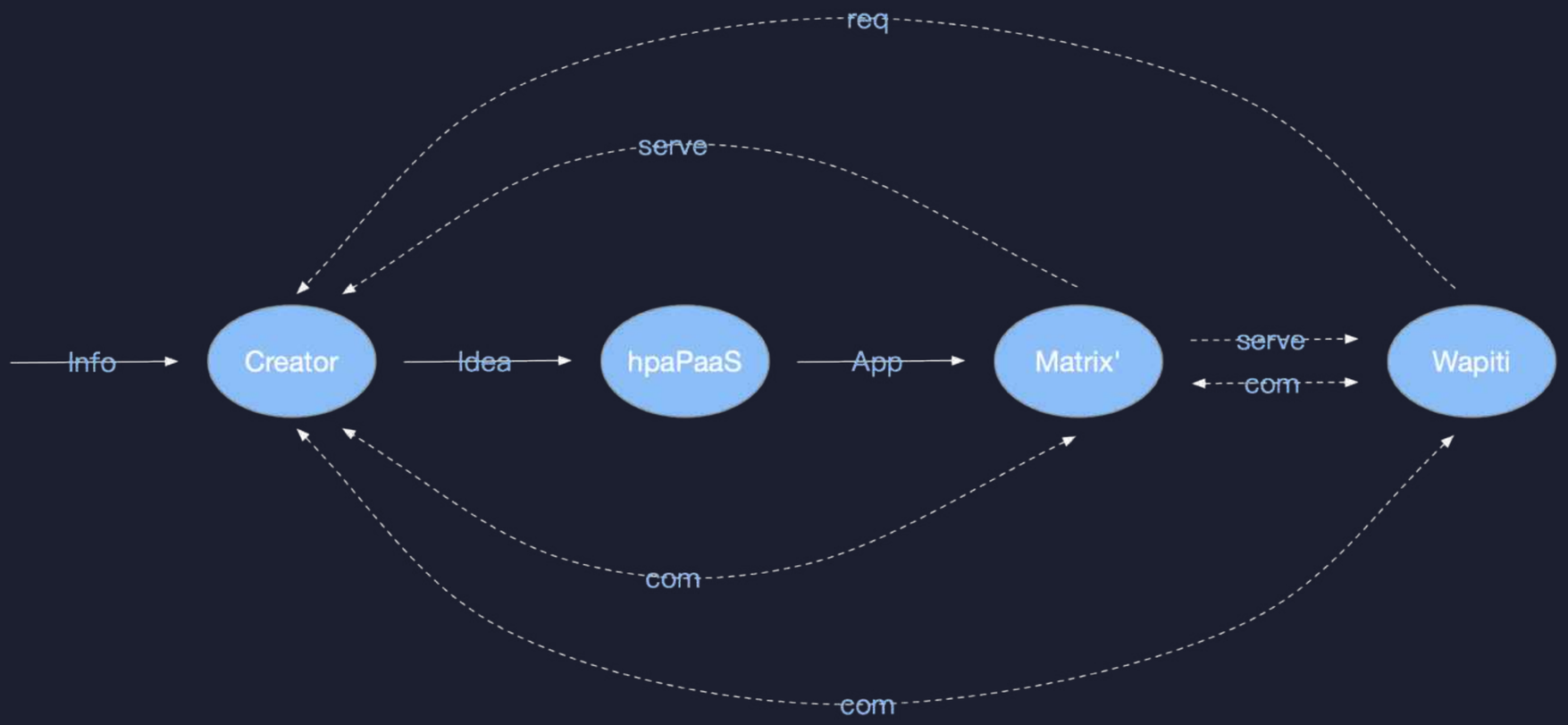
- 竖井效应
- 专业门槛



HPAPaaS 领域根本价值

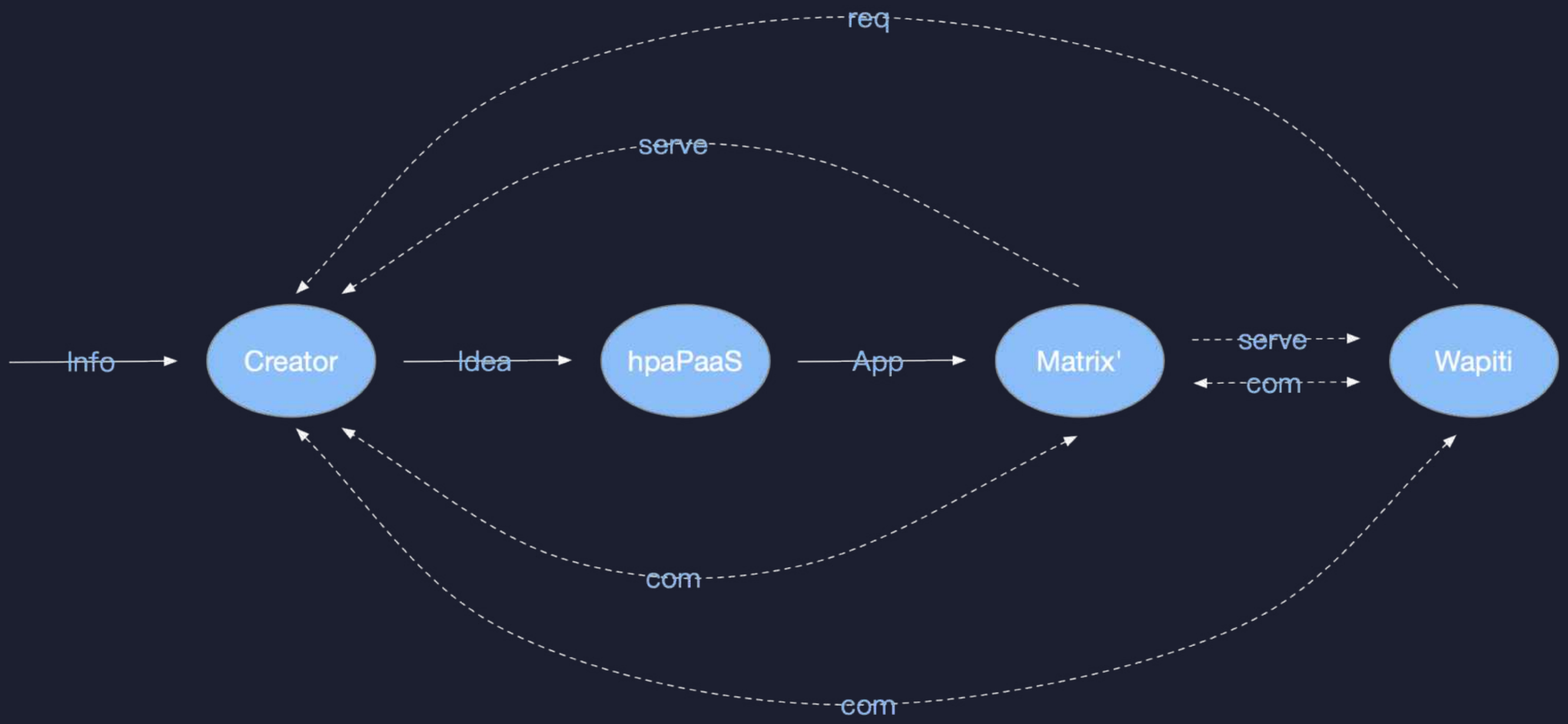
$$Info \overset{Creator}{=>} Idea$$

$$Idea \overset{hpaPaaS}{=>} App$$



帮助人们专注于创造性工作

HPAPaaS 领域根本驱动力



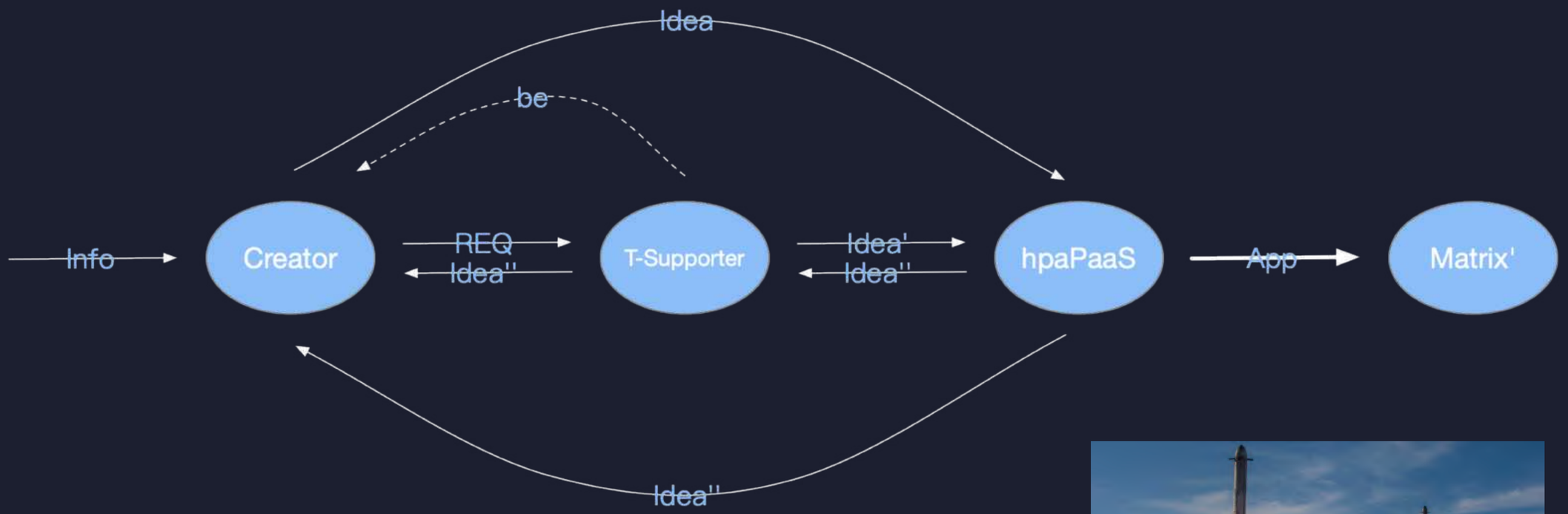
更优的需求结构化表达形式/应用模型 => 逐步降低最终消除技术复杂度

过渡阶段产品化建设

1 应用模型设计

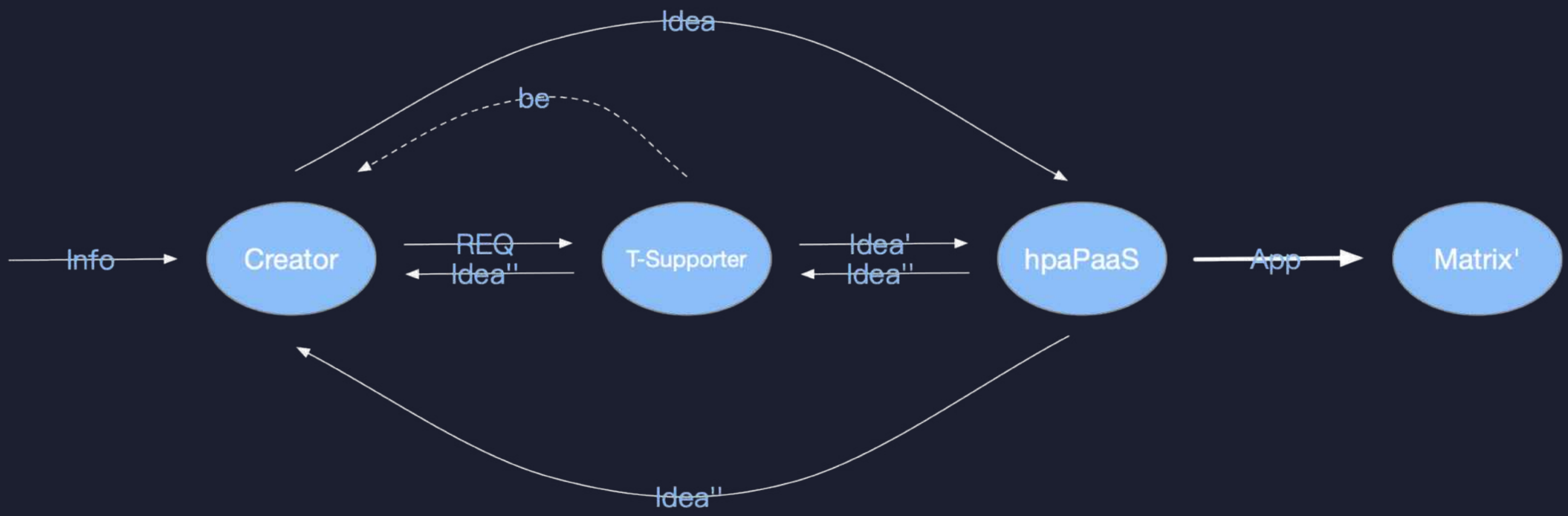
2 引擎工作流及其物料生态

过渡阶段领域形态及其核心产品特征



完备且支持规模化扩展的应用模型
逐步推动普通技术人员向创造者的角色迁移

应用模型设计 – 约束条件



完备可扩展、框架/语言无关

应用模型设计 – 理论基础 – 差量更新

```
class Example extends React.Component {
  constructor(props) {
    super(props);
  }

  componentDidMount() {
    this.handleDidMount && this.handleDidMount()
  }
  componentDidUpdate() {
    this.handleDidUpdate && this.handleDidUpdate()
  }

  render() {
    return (
      <div>
        <p>You clicked {this.count} times</p>
        <button onClick={this.handleClick}>
          Click me
        </button>
      </div>
    );
  }
}
```

```
class Example extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 0
    };
  }

  componentDidMount() {
    document.title = `You clicked ${this.state.count} times`;
  }
  componentDidUpdate() {
    document.title = `You clicked ${this.state.count} times`;
  }

  render() {
    return (
      <div>
        <p>You clicked {this.state.count} times</p>
        <button onClick={() => this.setState({ count: this.st
          Click me
        </button>
      </div>
    );
  }
}
```

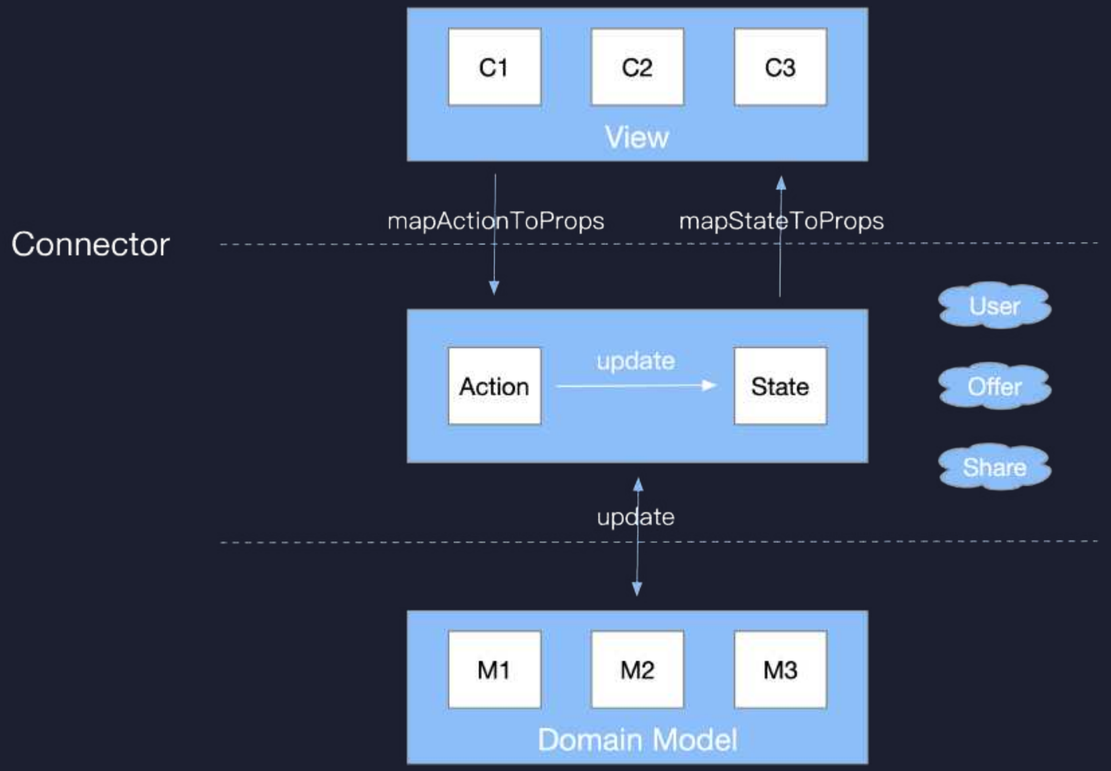
$App = App' + delta(Modular)$

$App = update(App')$

应用模型设计 – 工程实现（运行时）

```
import { createStore, connect } from 'markii'
import React from 'react'
import layout from './layout'
import store from './store'

const Page = connect({
  component: layout,
  store: createStore(store)
}, {
  mapStateToProps: (state, getters) => {
    return {
      content: state.content
    }
  },
  mapActionToProps: (dispatch) => {
    return {
      handleChange: (payload) => dispatch('handleChange', payload),
    }
  }
}, {
  __UI_FRAMEWORK__: {
    name: 'react',
    instance: React
  }
})
```

完备可扩展、框架无关



分离状态逻辑与UI

应用模型设计 – 原型示例

render

Mapper Config

```
{
  mapStateToProps: (state) => {
    return {
      formData: state.formData
    }
  },
  mapActionToProps: (actions) => {
    return {
      onValueChanged: actions.handleVa
```

Store Editor

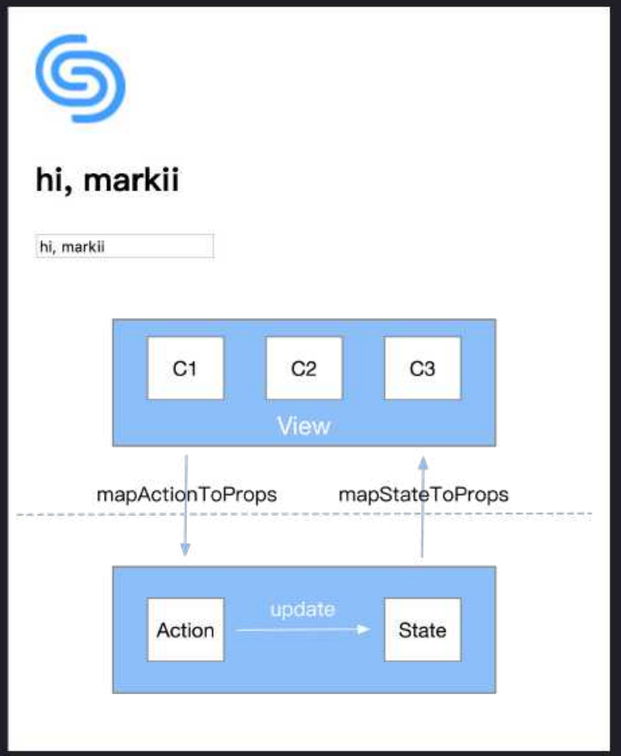
```
{
  state: {
    formData: {
      title: 'hi, markii'
    }
  },

  mutations: {
    commonUpdate (state, payload) {
```

template

```
<div class="root">
  <img src="//pt-starimg.didistatic.co
  <h2>{{title}}</h2>
  <div>
    <input v-model="title">
  </div>
</div>
```

script

```
{
  name: 'StarComponent',
  props: ['formData', 'onValueChanged'
  computed: {
    title: {
      get () {
        return this.formData.title
      },
      set (value) {
```



过渡阶段产品化建设

1 应用模型设计

2 引擎工作流及其物料生态

引擎工作流

- 视图布局排版
- 状态逻辑编写
- 视图/模型绑定

render

Mapper Config

```
{
  mapStateToProps: (state) => {
    return {
      formData: state.formData
    }
  },
  mapActionToProps: (actions) => {
    return {
      onValueChanged: actions.handleVa
```

Store Editor

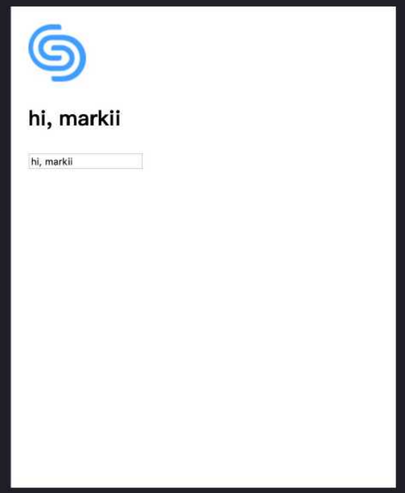
```
{
  state: {
    formData: {
      title: 'hi, markii'
    }
  },

  mutations: {
    commonUpdate (state, payload) {
```

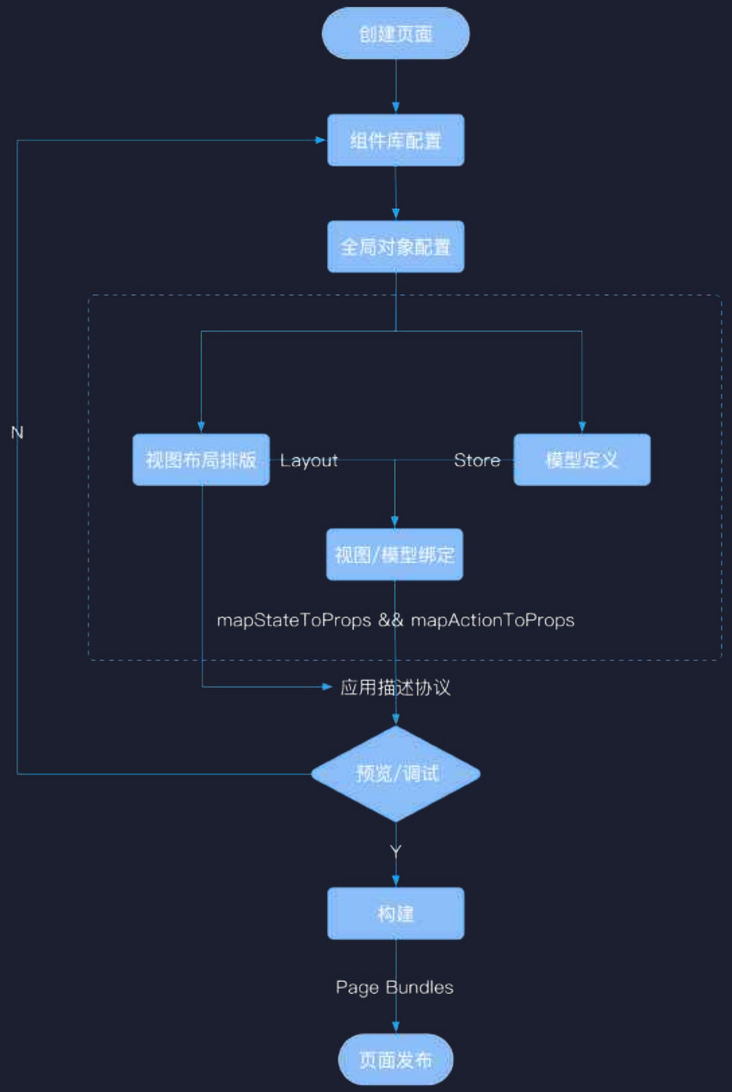
template

```
<div class="root">
  <img src="//pt-starimg.didistatic.co
  <h2>{{title}}</h2>
  <div>
    <input v-model="title">
  </div>
</div>
```

script

```
{
  name: 'StarComponent',
  props: ['formData', 'onValueChanged'
  computed: {
    title: {
      get () {
        return this.formData.title
      },
      set (value) {
```





UI描述协议及其等价工程产物

```
{
  "tag": "div",
  "attrs": {},
  "children": [{
    "tag": "el-row",
    "name": "栅格布局",
    "attrs": {
      ":gutter": 1,
      "type": "flex",
      "justify": "start",
      "align": "top"
    },
    "children": [{
        "tag": "el-col",
        "name": "栅格布局",
        "attrs": {
          ":span": 12
        },
        "children": [{
            "tag": "el-select",
            "name": "select",
            "attrs": {
              "__ckey__": "filter002",
              "key": "filter002",
              "label": "检索项002",
              "value.model": "选项2",
              "handleValueChange.handler": "handleFilterChange",
              "@change": "handleFilterChange",
              "options": [{
                "value": "选项1",
                "label": "黄金糕"
              }, {
                "value": "选项2",
                "label": "双皮奶"
              }]
            }
          }
        ]
      }
    ]
  }]
}
```

```
<div>
  <el-row :gutter="1" type="flex" justify="start" align="top">
    <el-col :span="12">
      <el-select
        __ckey__="filter002"
        :key="filter002.key"
        :label="filter002.label"
        v-model="filter002_value"
        :handlevaluechange="handleFilterChange"
        @change="handleFilterChange"
        :options="filter002.options"
      />
    </el-col>
  </el-row>
</div>
```

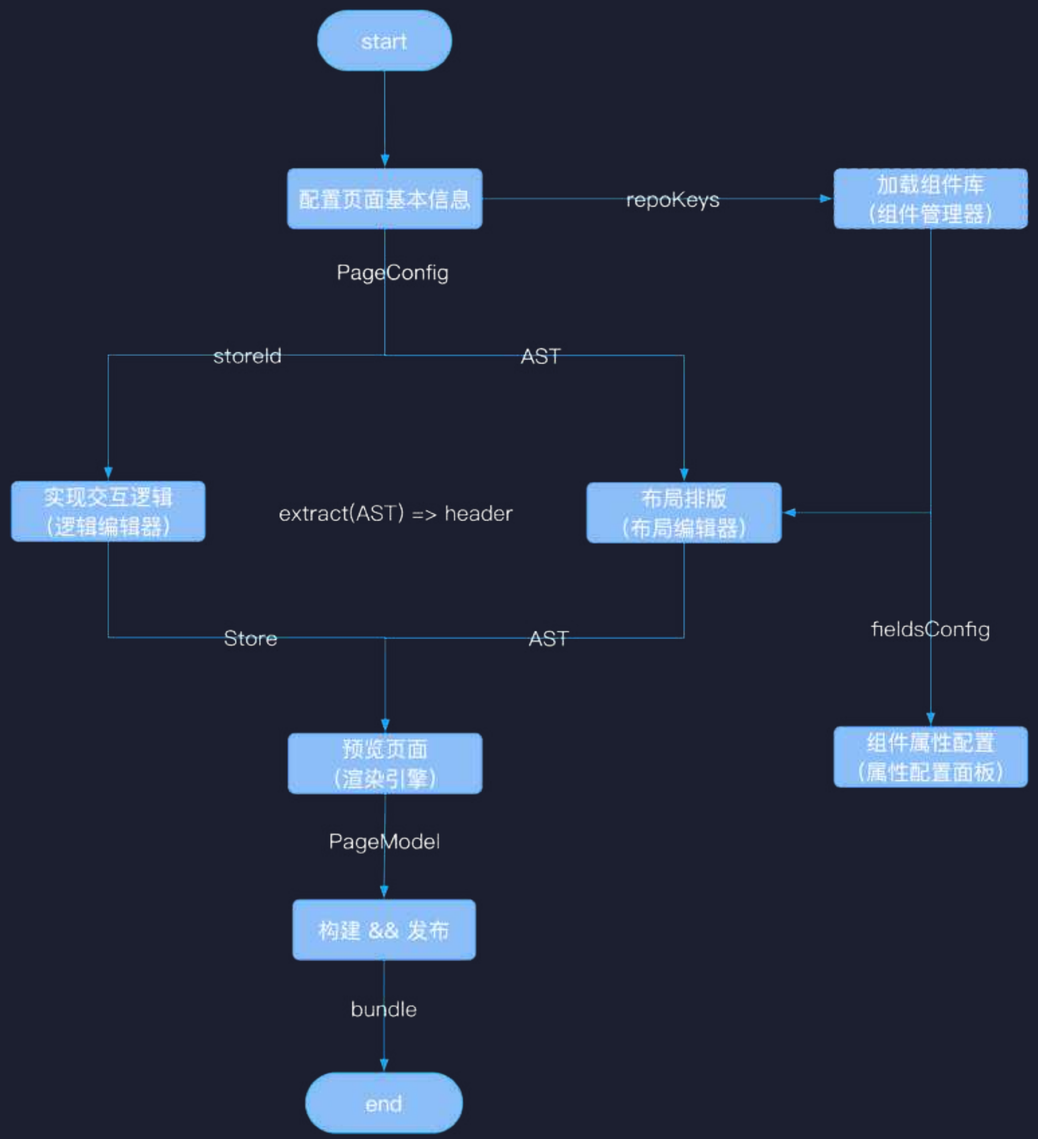
Store结构及其解析

```
const Store = {
  state: {
    content: 'don not panic.',
  },

  mutations: {
    setContent (state, payload) {
      state.content = payload
    }
  },

  actions: {
    handleChange ({state, commit}, payload) {
      return new Promise(async (resolve, reject) => {
        commit('setContent', payload)
        resolve(payload)
      })
    },
  },

  getters: {
  },

  modules: {
  },

  context: {
    __defined__: () => {
      return {
        VIEW_TYPE_A: 'A',
        VIEW_TYPE_B: 'B'
      }
    },

    _get: (context) => {
      return (params) => {
        console.log(8989, context, params)
      }
    }
  }
}

export default Store
```

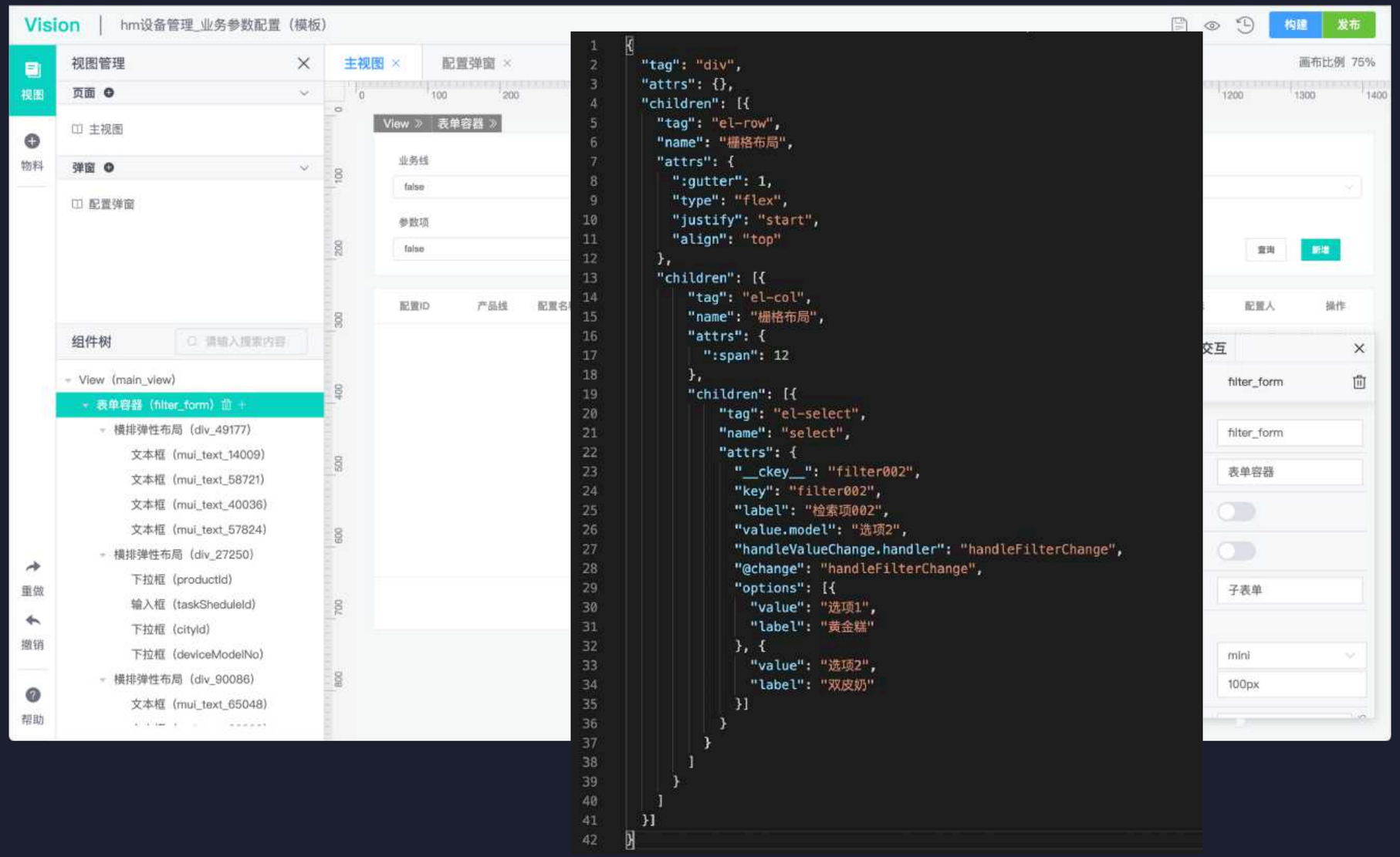

```
{
  "type": "Program",
  "start": 0,
  "end": 648,
  "body": [
    {
      "type": "VariableDeclaration",
      "start": 0,
      "end": 625,
      "declarations": [
        {
          "type": "VariableDeclarator",
          "start": 6,
          "end": 625,
          "id": {
            "type": "Identifier",
            "start": 6,
            "end": 11,
            "name": "Store"
          },
          "init": {
            "type": "ObjectExpression",
            "start": 14,
            "end": 625,
            "properties": [
              {↔},
              {↔},
              {↔},
              {↔},
              {↔},
              {↔}
            ]
          }
        }
      ],
      "kind": "const"
    },
    {↔}
  ],
  "sourceType": "module"
}
```

模型/视图绑定

```
interface model {
    key: string,
    type: string,
    defaultValue: any
}

interface handler {
    key: string
}

interface header {
    id: string,
    models: Array<model>,
    handlers: Array<handler>
}
```

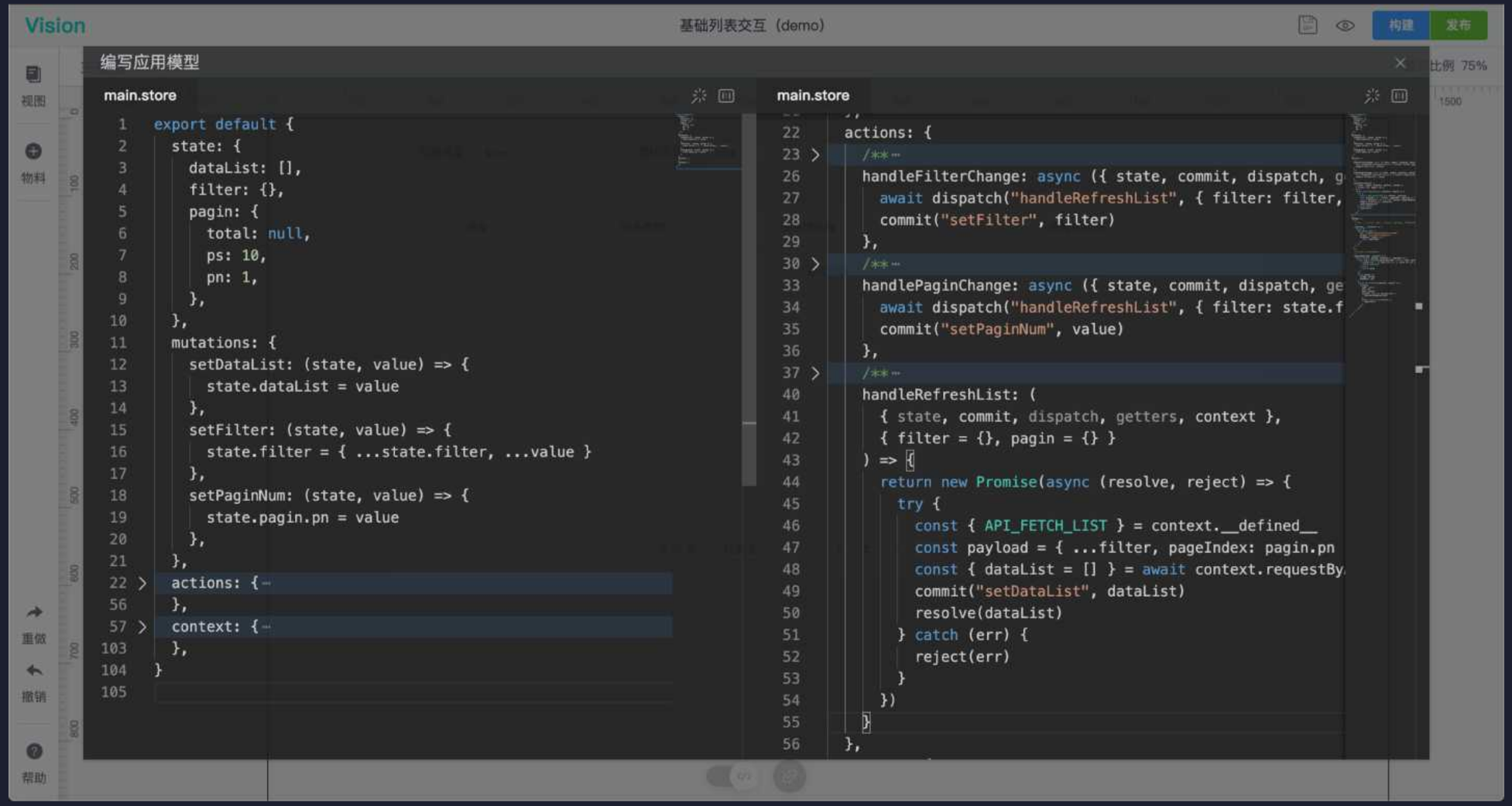


引擎核心工作流 - 布局排版




```
{
  "tag": "div",
  "attrs": {},
  "children": [{
    "tag": "el-row",
    "name": "栅格布局",
    "attrs": {
      ":gutter": 1,
      "type": "flex",
      "justify": "start",
      "align": "top"
    },
    "children": [{
        "tag": "el-col",
        "name": "栅格布局",
        "attrs": {
          ":span": 12
        },
        "children": [{
            "tag": "el-select",
            "name": "select",
            "attrs": {
              "__ckey__": "filter002",
              "key": "filter002",
              "label": "检索项002",
              "value.model": "选项2",
              "handleValueChange.handler": "handleFilterChange",
              "@change": "handleFilterChange",
              "options": [{
                "value": "选项1",
                "label": "黄金糕"
              }, {
                "value": "选项2",
                "label": "双皮奶"
              }]
            }
          }
        ]
      }
    ]
  }]
}
```

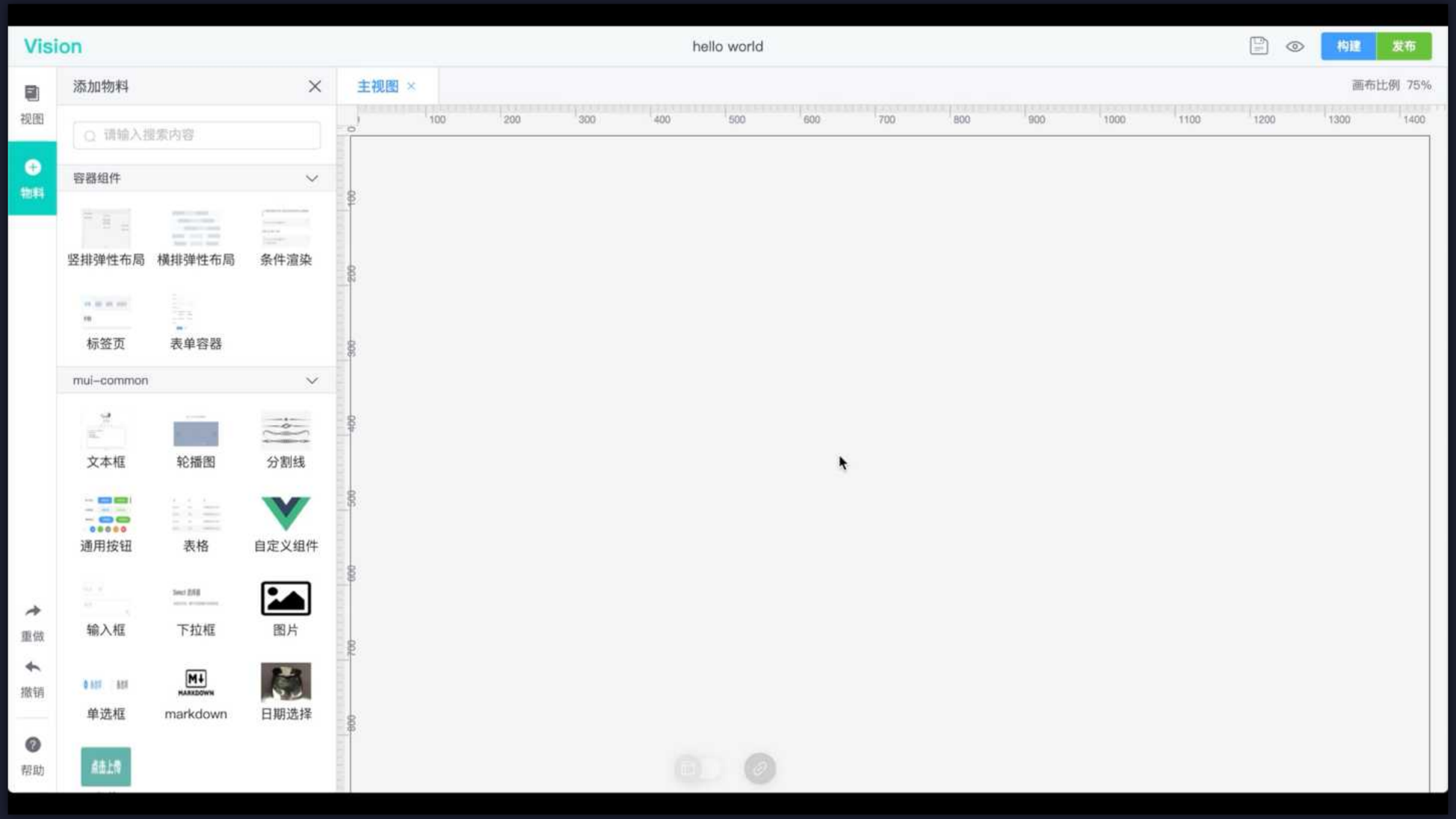
引擎核心工作流 – 模型定义




```
export default {
  state: {
    dataList: [],
    filter: {},
    pagin: {
      total: null,
      ps: 10,
      pn: 1,
    },
  },
  mutations: {
    setDataList: (state, value) => {
      state.dataList = value
    },
    setFilter: (state, value) => {
      state.filter = { ...state.filter, ...value }
    },
    setPaginNum: (state, value) => {
      state.pagin.pn = value
    },
  },
  actions: {…
  },
  context: {…
  },
}
```

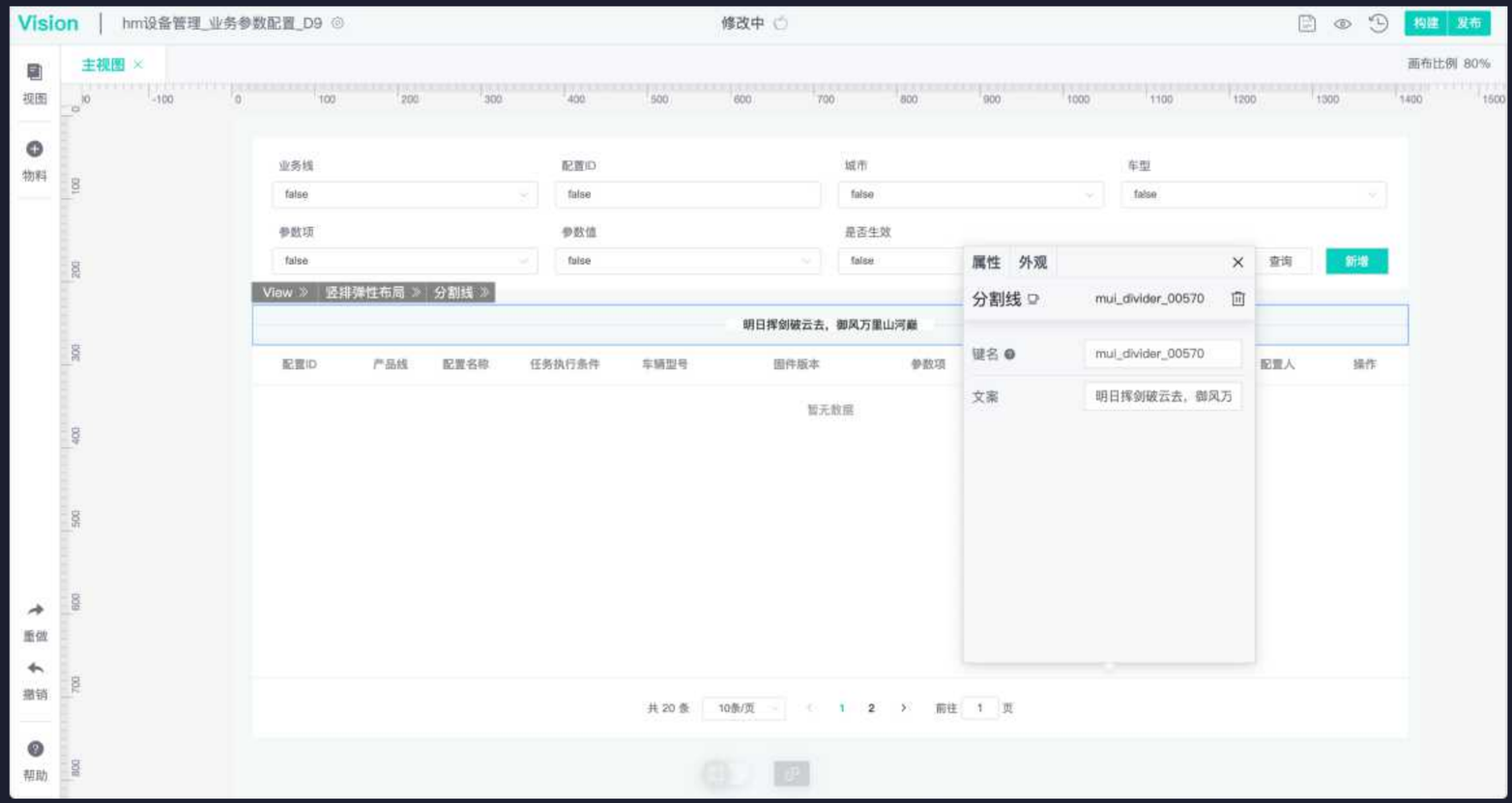
```
actions: {
  /**…
  handleFilterChange: async ({ state, commit, dispatch, g
    await dispatch("handleRefreshList", { filter: filter,
    commit("setFilter", filter)
  },
  /**…
  handlePaginChange: async ({ state, commit, dispatch, ge
    await dispatch("handleRefreshList", { filter: state.f
    commit("setPaginNum", value)
  },
  /**…
  handleRefreshList: (
    { state, commit, dispatch, getters, context },
    { filter = {}, pagin = {} }
  ) => {
    return new Promise(async (resolve, reject) => {
      try {
        const { API_FETCH_LIST } = context.__defined__
        const payload = { ...filter, pageIndex: pagin.pn
        const { dataList = [] } = await context.requestBy
        commit("setDataList", dataList)
        resolve(dataList)
      } catch (err) {
        reject(err)
      }
    })
  }
},
```

引擎核心工作流 – 视图/模型绑定

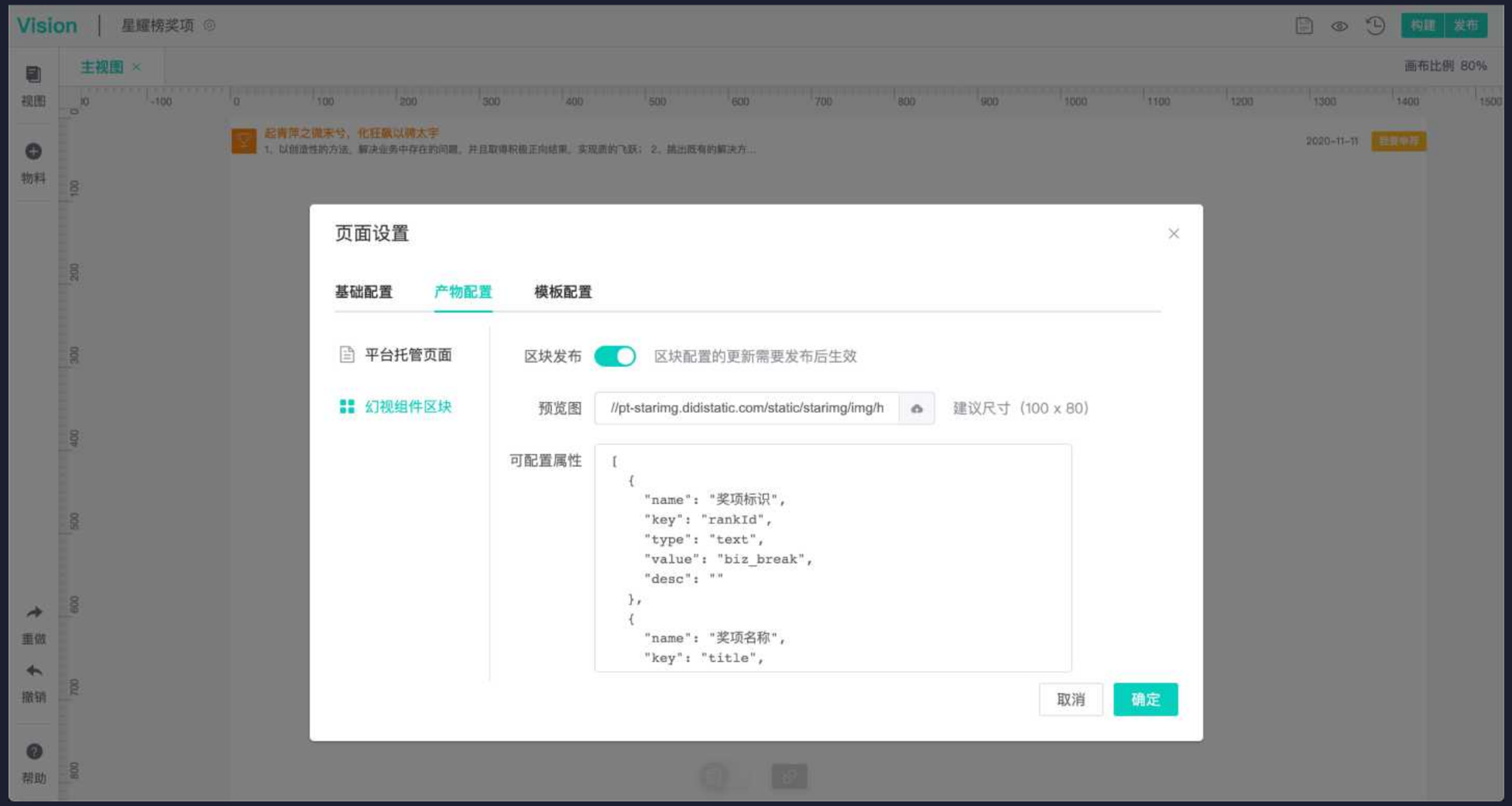


基础物料生态 – 标准化本地组件消费

```
[
    {
        "name": "组件样式",
        "key": "style",
        "type": "style",
        "category": "layout",
        "value": {
            "width": "100%"
        }
    },
    {
        "name": "文案",
        "key": "text",
        "type": "text",
        "value": “明日挥剑破云去 …”
    }
]
```



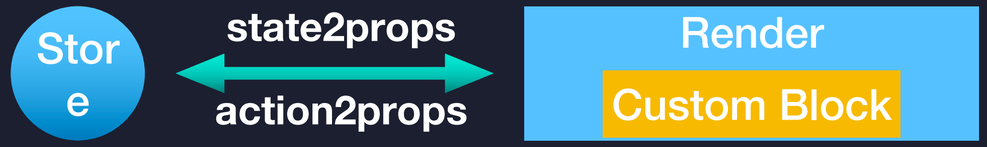
基础物料生态 – 线上物料生产（自定义区块）



消费方式与本地组件一致

基础物料生态 – 线上物料生产（自定义区块）

通过 HOC 属性/事件代理实现区块的状态/交互绑定



仅保留目标区块所有子组件的映射关系

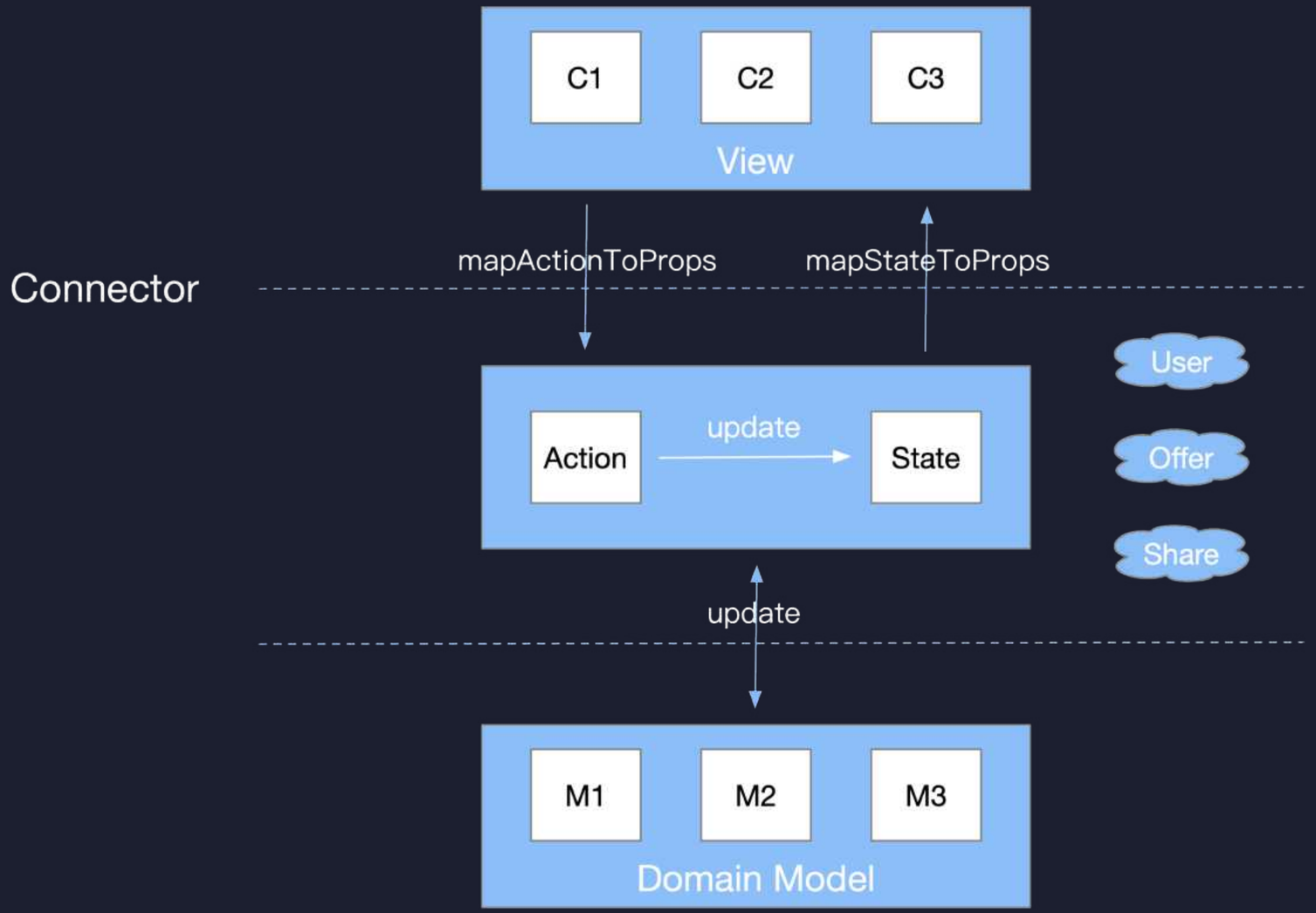
```
// 清理状态映射
Object.keys(blockSchema.mapper.state).map(widgetId => {
  if (!getLayoutFreeDomWidgetById(blockSchema.layout, widgetId)) {
    const widgetStateMapper = _.get(blockSchema, `mapper.state.${widgetId}`)
    if (widgetStateMapper) {
      delete blockSchema.mapper.state[widgetId]
    }
  }
})
// 清理动作映射
blockSchema.mapper.action = _.get(blockSchema, 'mapper.action', []).filter(item => {
  return getLayoutFreeDomWidgetById(blockSchema.layout, item.triggerId)
})
```

```
const renderV1X = ({
  root,
  key
}, {
  model, store
}) => {
  try {
    const storeInstance = MarkII.createStore(store)
    // 只渲染根视图
    // const pageSet = makeSPAPageSet(pages, window.__mstore__)

    // 不处理路由配置
    // const routerConfig = makeSPARouterConfig(model.router, pageSet)
    // const router = new Router({ routes: routerConfig })

    const { mapper, tpl, script } = _.cloneDeep(model.root)
    const layout = createComponent({template: tpl, options: script})
    const mapperObj = eval('(' + mapper + ')')
    const blockInstance = MarkII.connect({
      component: layout,
      store: storeInstance
    }, {
      ...mapperObj
    })
    return {
      storeInstance,
      blockInstance
    }
  } catch (e) {
    console.error(e)
  }
}
```


领域搭建产品
互动营销
流程 && 表单
泛h5
泛中后台
数据应用
问卷调研
模板库
应用/产物管理
工作流
工程管理
引擎/运行时
视图编辑器
视图模型编辑器
领域模型编辑器
表单
表格
垂直领域
引擎内核
Pro Code
视图模型
UI 组件
领域模型
非标准物料
基础物料生态
应用描述协议
运行时
基础设施

回顾

- 幻视产品现状及其核心特性
- **HPAPaaS** 领域根本价值
- 过渡阶段产品化建设



HPAPaaS价值规模化的根本驱动力在于

逐步消除软件生产过程中的技术复杂度

THANKS

QCon+ 案例研习社