拒绝千篇一律
低代码要效率也要个性

从理论到实践和你一起揭秘mendix前端组件开发

SIEMENS

联系方式

刘文高
西门子Mendix团队高级前端架构师
Street 123福田区锦堤安路3088号2401室
518040 珠海
中国

Phone -
Mobile +86 15819806324（微信同号）

E-mail wengao.liu@siemens.com

mendix中文论坛地址

SIEMENS

内容提要

Mendix前端组件的前世今生

实体的生命周期

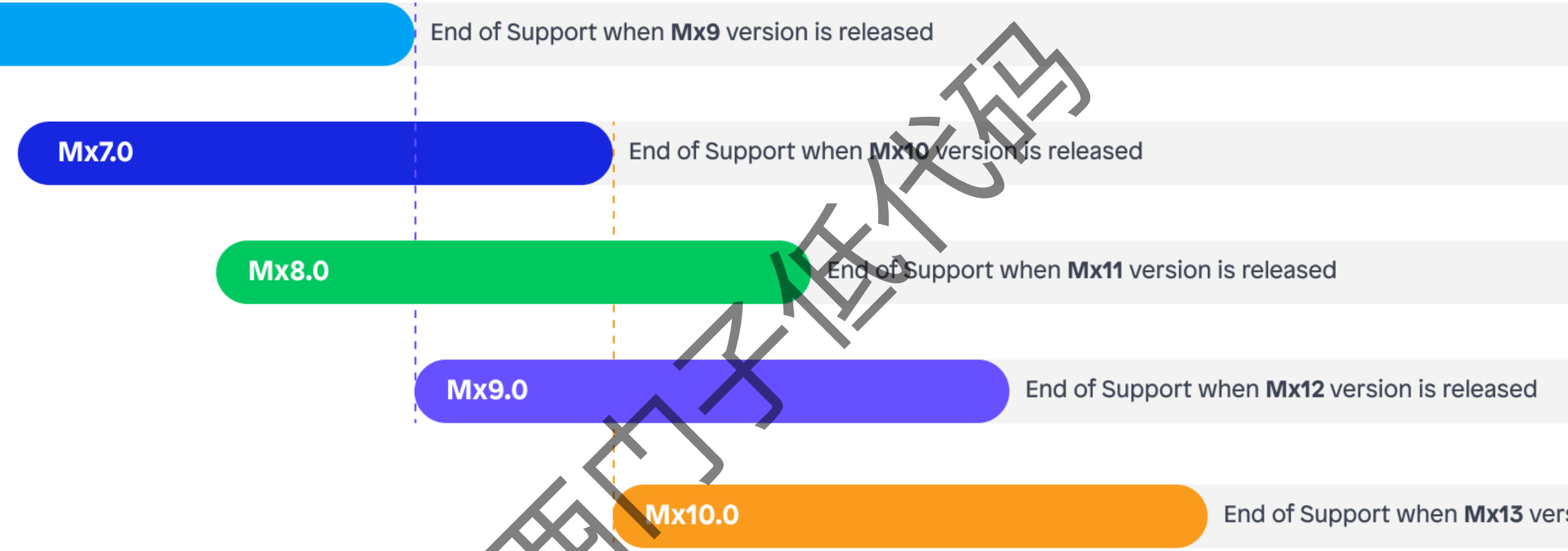
XPATH的通俗理解

一次典型的网络请求发生了什么

组件脚手架一键创建

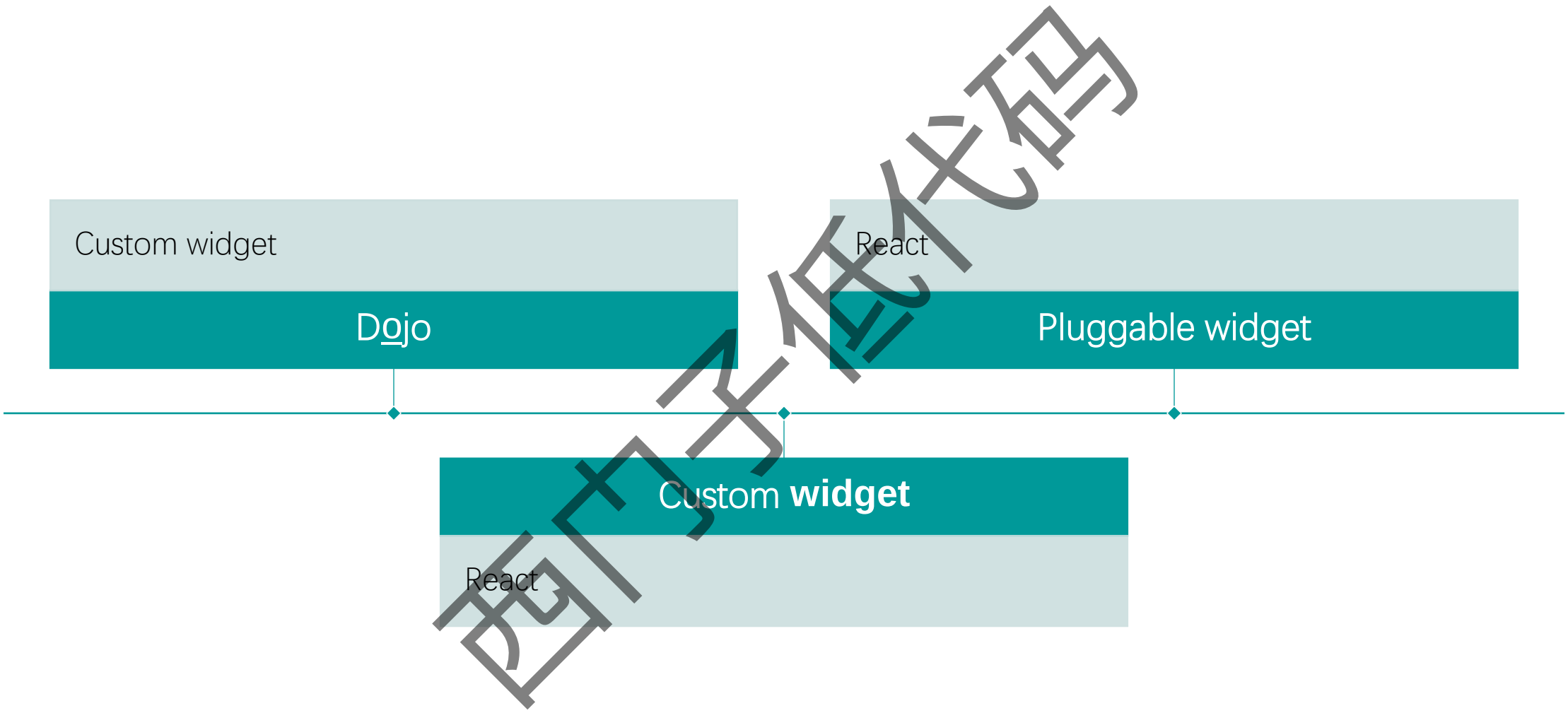
实战讲解组件开发的API设计

- 上下文属性集成
- 上下文数据源及其属性集成
- 动作集成



SIEMENS

组件开发技术演进历史



Mendix前端组件的前世今生

Dojo

Dojo 于 2004 年创建，使开发 DHTML 和 JavaScript web 应用程序开发流程更为容易，隐藏了很多现代 web 浏览器中普遍存在的跨浏览器矛盾。这使重点放在实现功能上，而不是调整代码使其在每个浏览器上运行。Dojo 属于 Dojo 基金会，该基金会是 Russell 和 Dylan Schiemann 于 2005 年创建的。Dojo 是一个开源软件（OSS），有双重许可，Academic Free License (AFL) 和一个修改的 BSD 许可，您可以选择遵守一个。

Mendix前端组件的前世今生

Custom widget 非可插拔

From mendix 8

- API丰富，但以字符串给出，需要开发都调用客户端API自行处理各种细节
- 微流参数只能有一个上下文实体

SIEMENS

Mendix前端组件的前世今生

Pluggable widget 可插拔

From mendix 9

- 声明式API，我们拿到的接口是经过运行时处理过的最终数据。极大简化开发难度，但对于特别复杂的数据交互逻辑不太适用。
- 微流传参除了上下文外，还支持祖先上下文

内容提要

Mendix前端组件的前世今生

实体的生命周期

XPATH的通俗理解

一次典型的网络请求发生了什么

组件脚手架一键创建

实战讲解组件开发的API设计

- 上下文属性集成
- 上下文数据源及其属性集成
- 动作集成

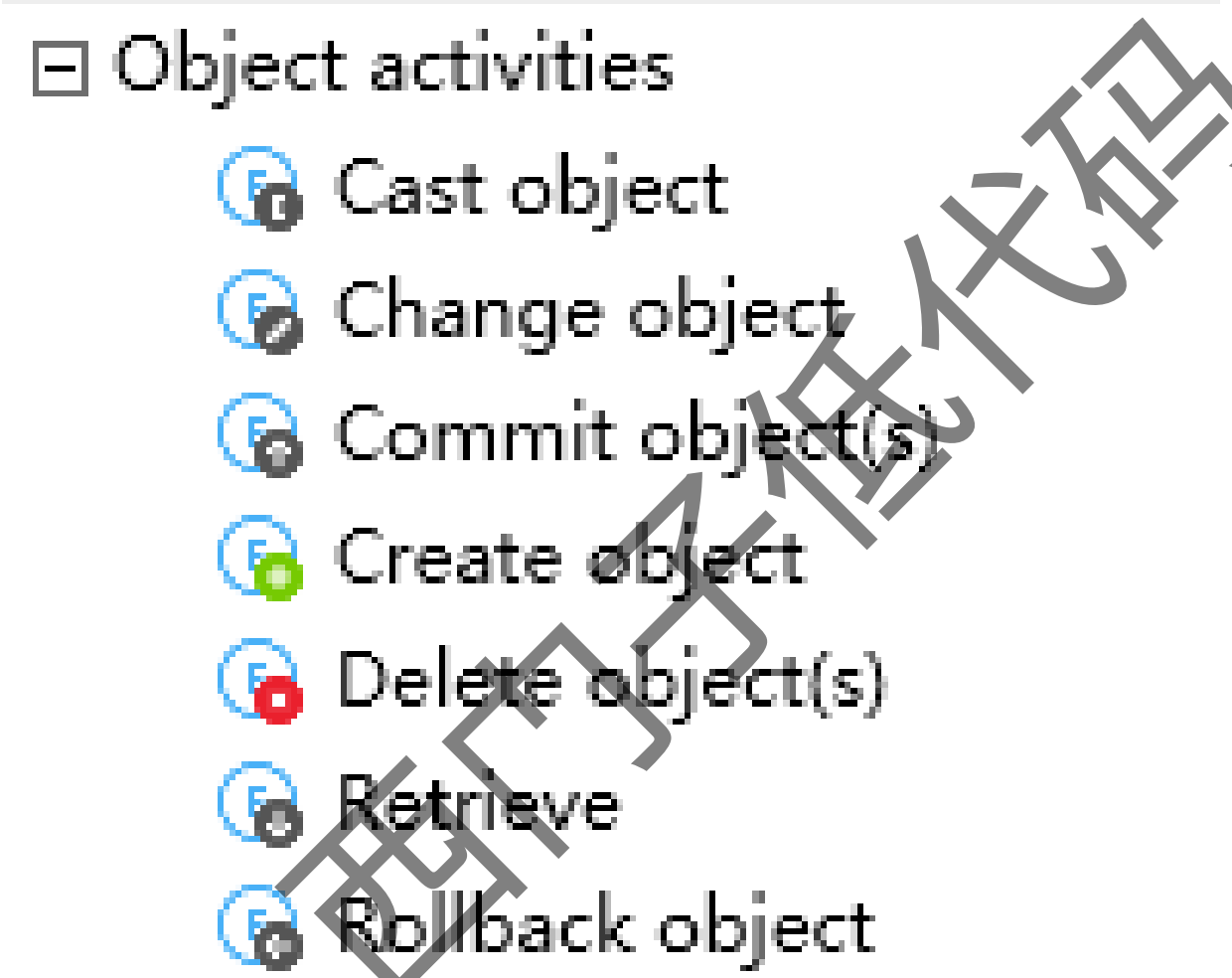
SIEMENS

实体生命周期
从日出到日落

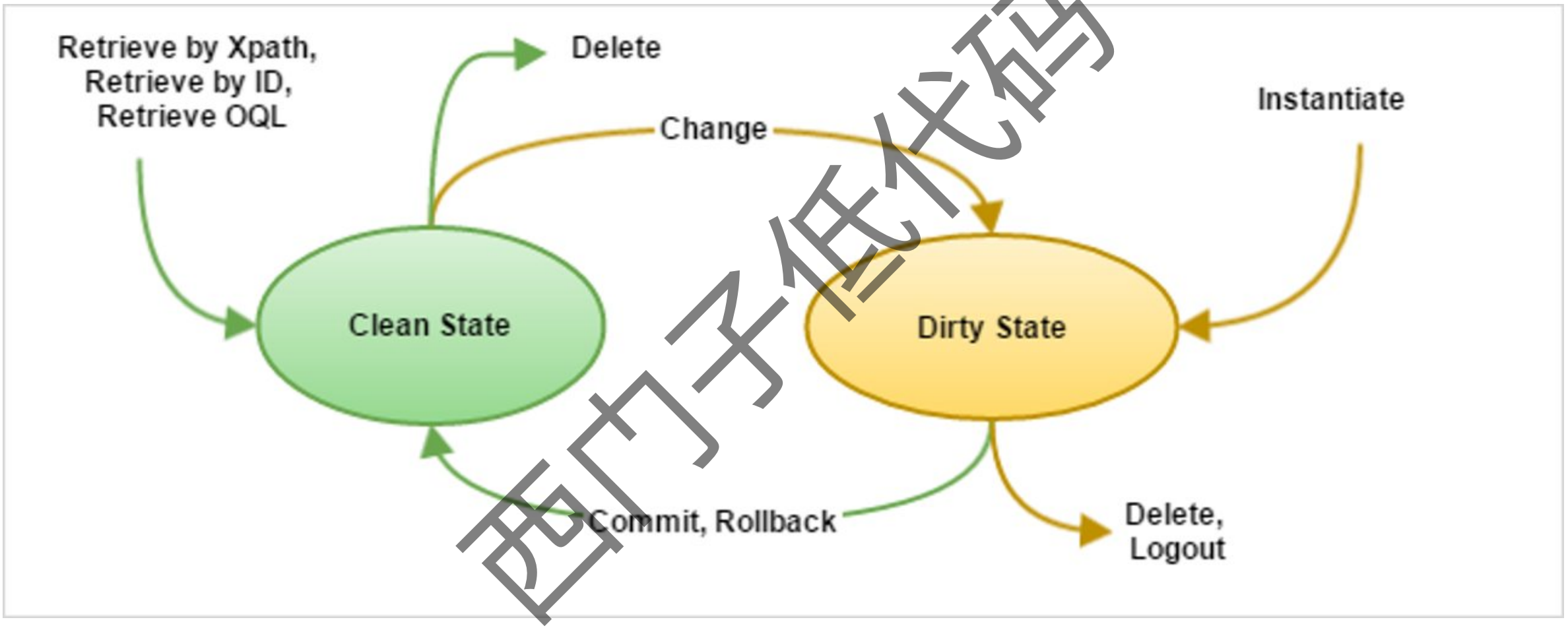
实体各个状态间的转换，以及垃圾回收

SIEMENS

改变实体状态相关活动

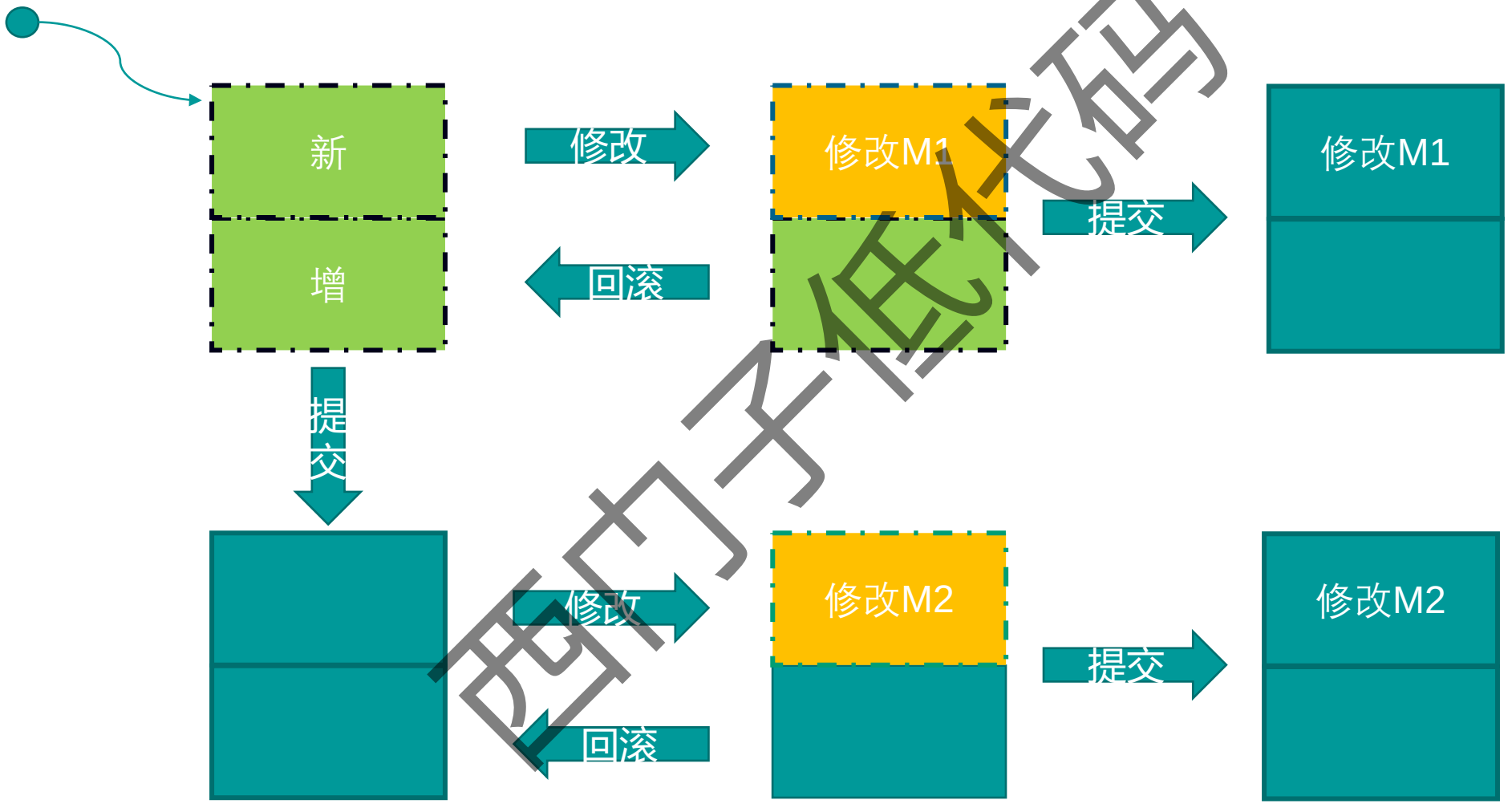


实体状态方程（二）



SIEMENS

实体状态方程（一）



SIEMENS

实体回收

- Mendix前端运行时每隔10秒进行垃圾回收
- 对于React 可插拔组件无须手动进行引用维护和垃圾回收
- 其它2种类型的组件需要开发者手动介入
- TL;DR transient-objects-garbage-collecting

内容提要

Mendix前端组件的前世今生

实体的生命周期

XPATH的通俗理解

一次典型的网络请求发生了什么

组件脚手架一键创建

实战讲解组件开发的API设计

- 上下文属性集成
- 上下文数据源及其属性集成
- 动作集成

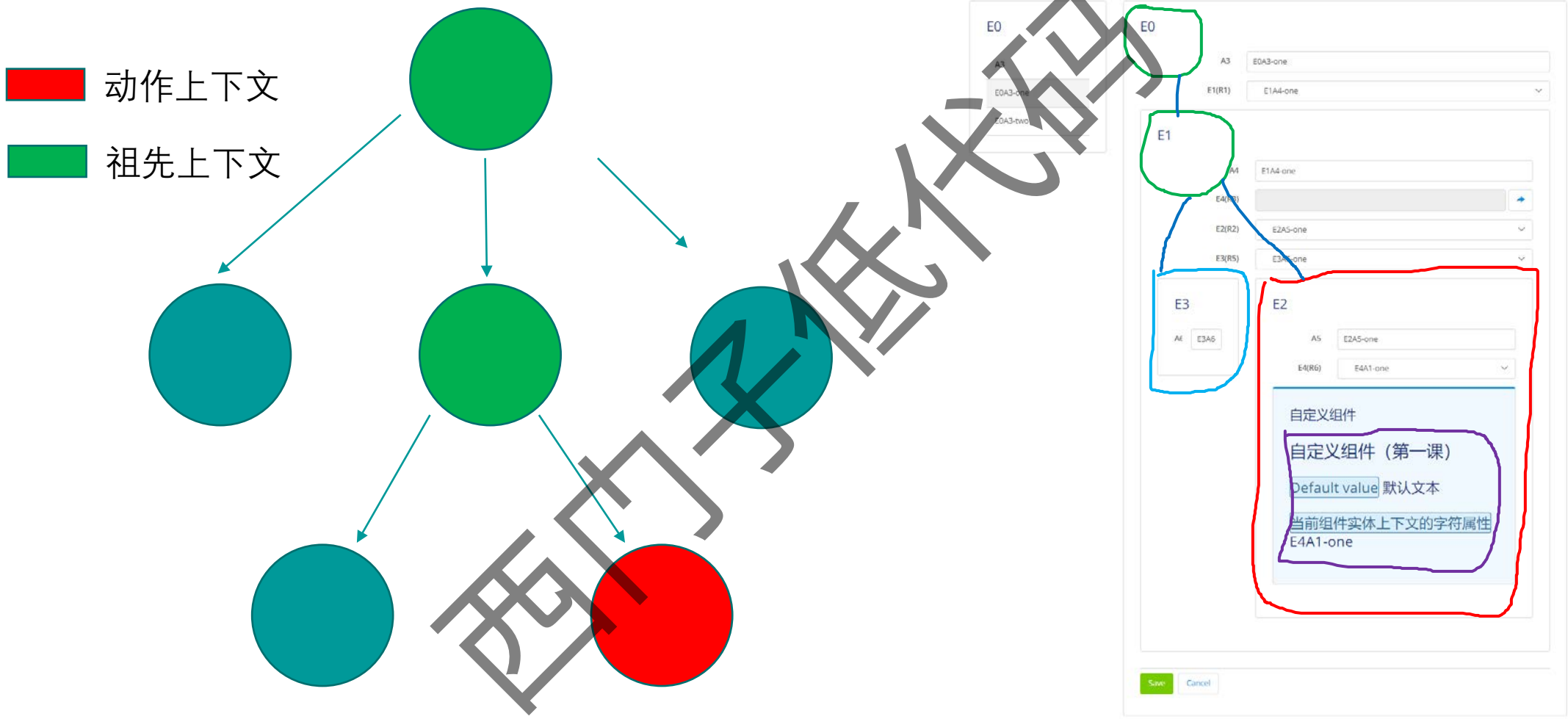
Digital layer images are now allowed in the new design **for presentations and web only**

XPATH
大表哥的小姨子的男朋友的名字

从一个实体出发通过实体架构图中的若干节点和边获得想要的一个目标属性或者一组目标实体对象

SIEMENS

上下文与祖先上下文

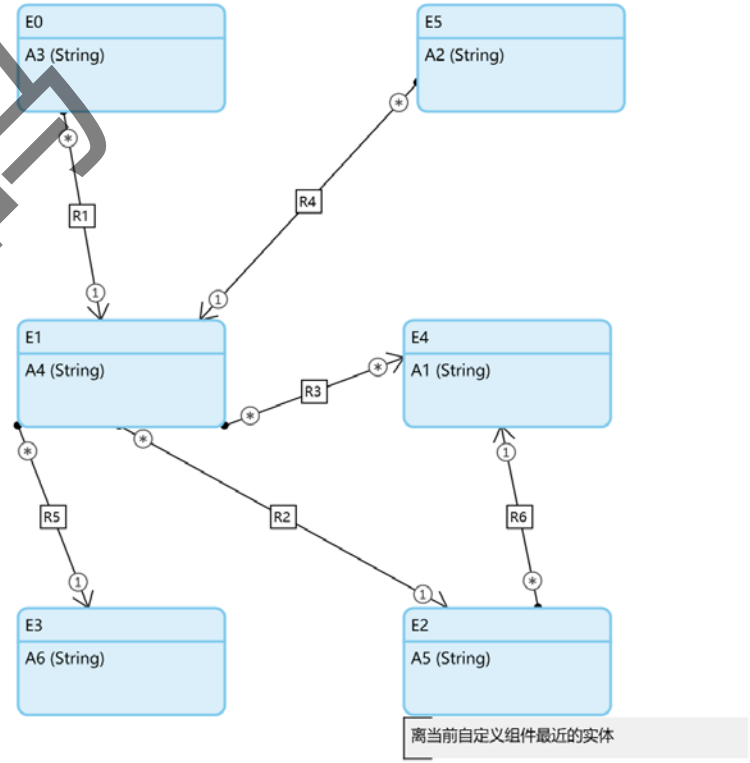


SIEMENS

XPATH（是一条路，一条从上下文实体出发获得单个实体或者实体的集合）

E0/R1/E1/R2/E2/A5

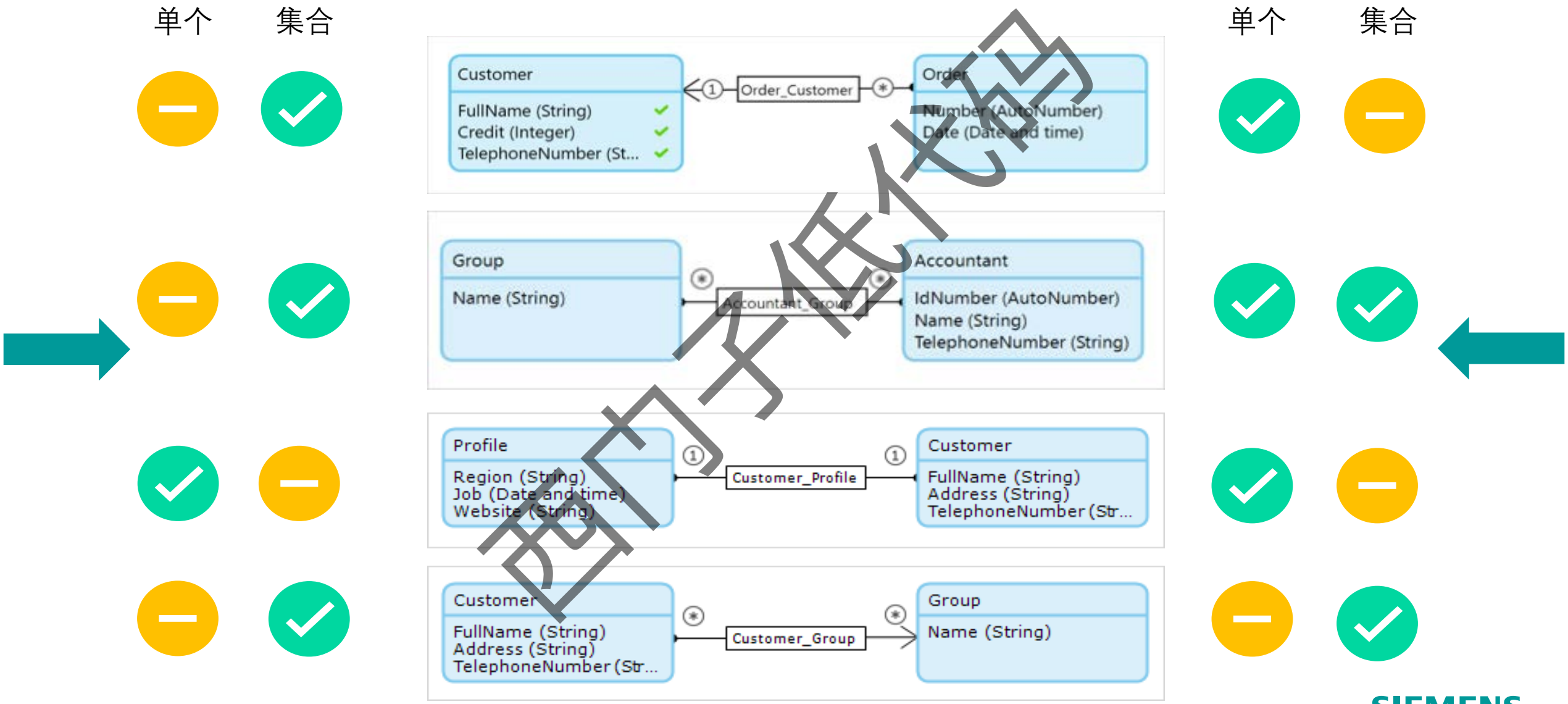
从E0这个上下文实体出发获得目标实体E2上的属性A5



E0/R1/E1/R3/E4

从E0这个上下文实体出发获得一组目标实体E4

XPATH连通性（路上的红绿灯）



SIEMENS

内容提要

Mendix前端组件的前世今生

实体的生命周期

XPATH的通俗理解

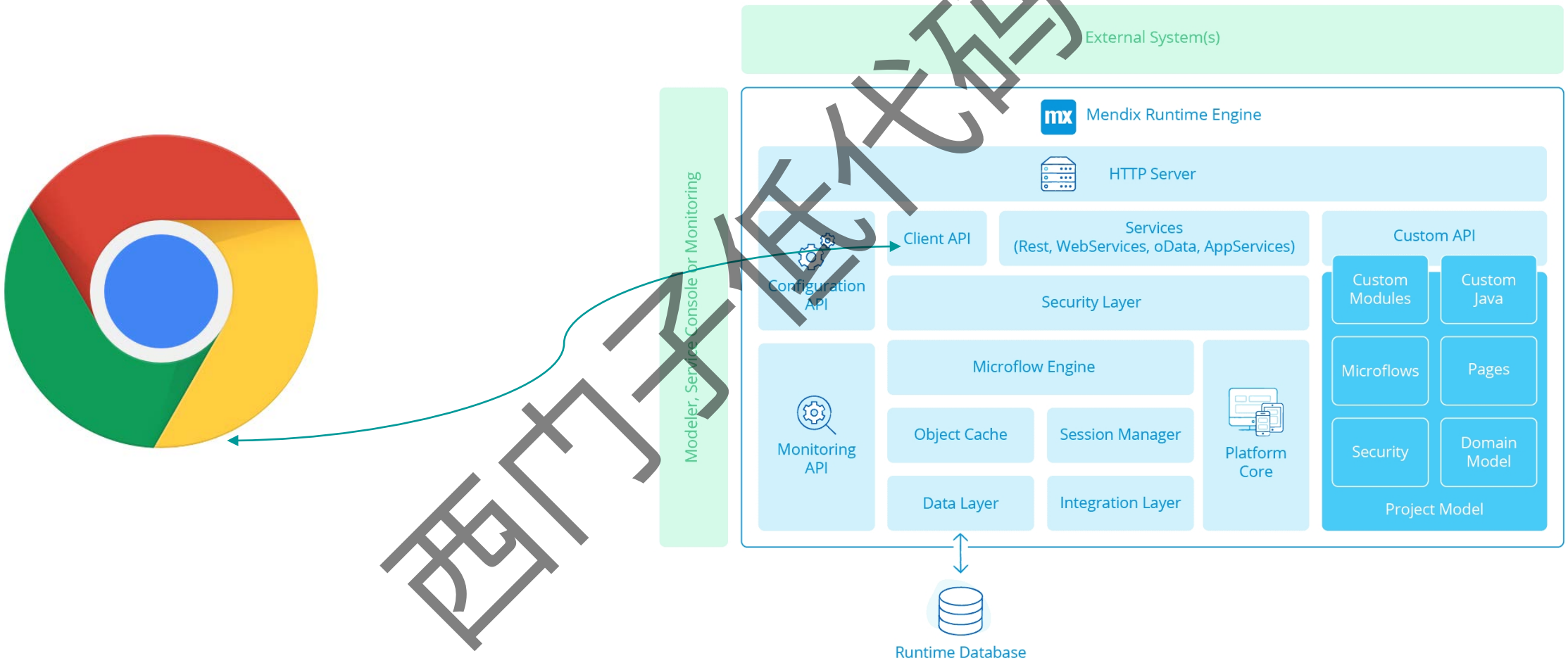
一次典型的网络请求发生了什么

组件脚手架一键创建

实战讲解组件开发的API设计

- 上下文属性集成
- 上下文数据源及其属性集成
- 动作集成

系统架构图

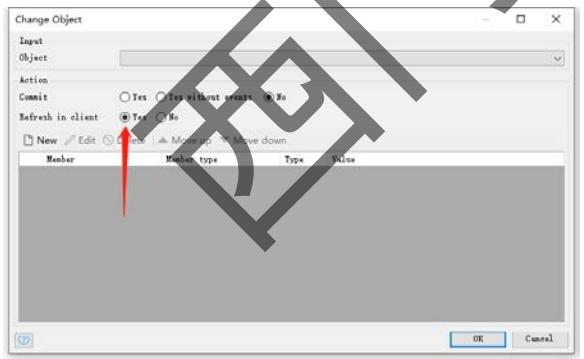


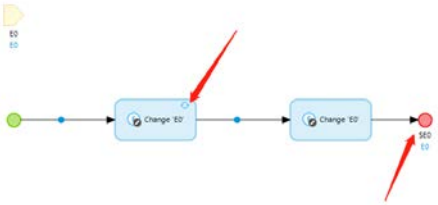
SIEMENS

实体在动作中传递

上下文实体
从此上下文实体出发，XPATH可达的非持久化实体和未提交的持久化实体

微流返回实体及其XPATH可达实体
Refresh in client 标记为True的实体





内容提要

Mendix前端组件的前世今生

实体的生命周期

XPATH的通俗理解

一次典型的网络请求发生了什么

组件脚手架一键创建

实战讲解组件开发的API设计

- 上下文属性集成
- 上下文数据源及其属性集成
- 动作集成

一键开发

下载并且执行批处理文件

https://gitee.com/engalar/mendix-pluggable-widget-template/blob/master/script/new_pw.bat

一键试玩

下载并且执行批处理文件

https://gitee.com/engalar/mendix-pluggable-widget-template/blob/master/script/try_pw.bat

内容提要

Mendix前端组件的前世今生

实体的生命周期

XPATH的通俗理解

一次典型的网络请求发生了什么

组件脚手架一键创建

实战讲解组件开发的API设计

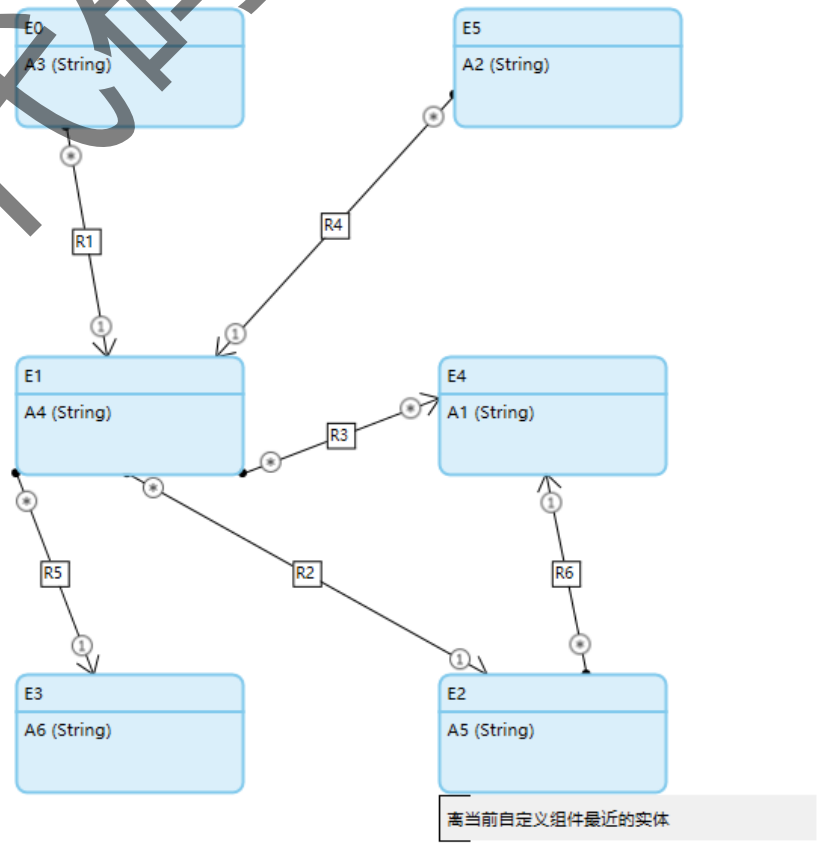
- **上下文属性集成**
- 上下文数据源及其属性集成
- 动作集成

SIEMENS

展示上下文单一实体（属性）

SIEMENS

展示上下文单一实体（属性）



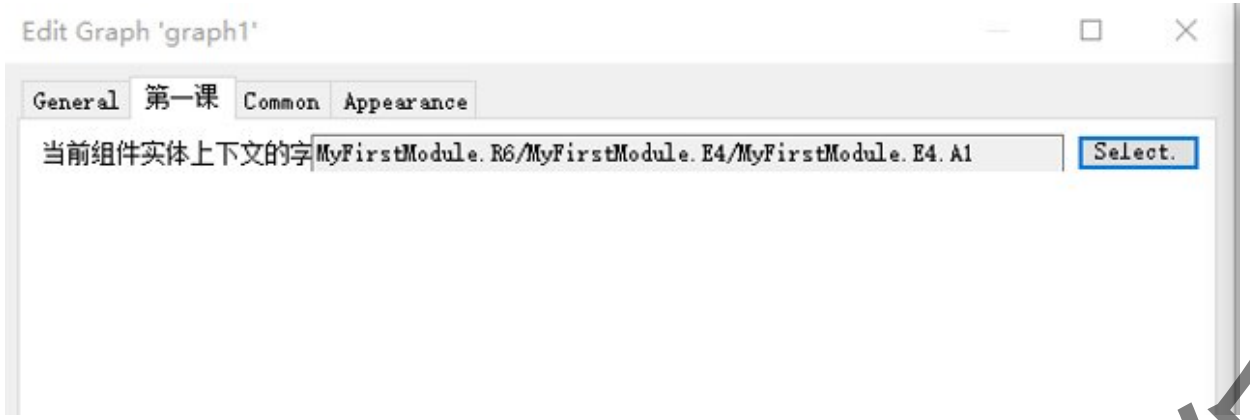
SIEMENS

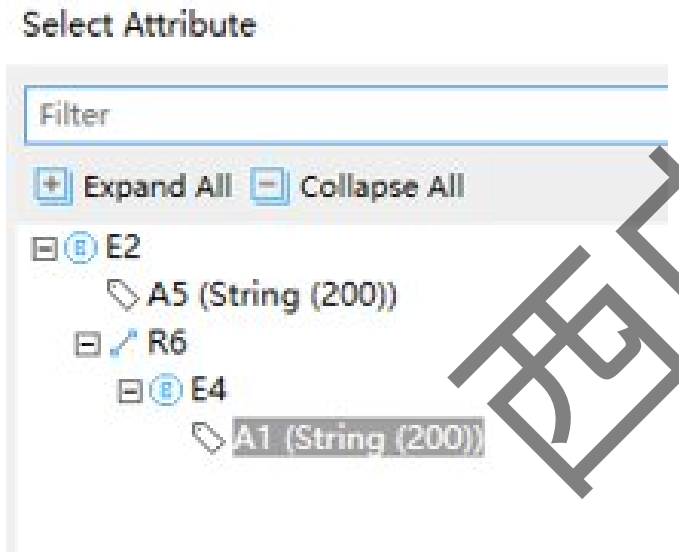
API

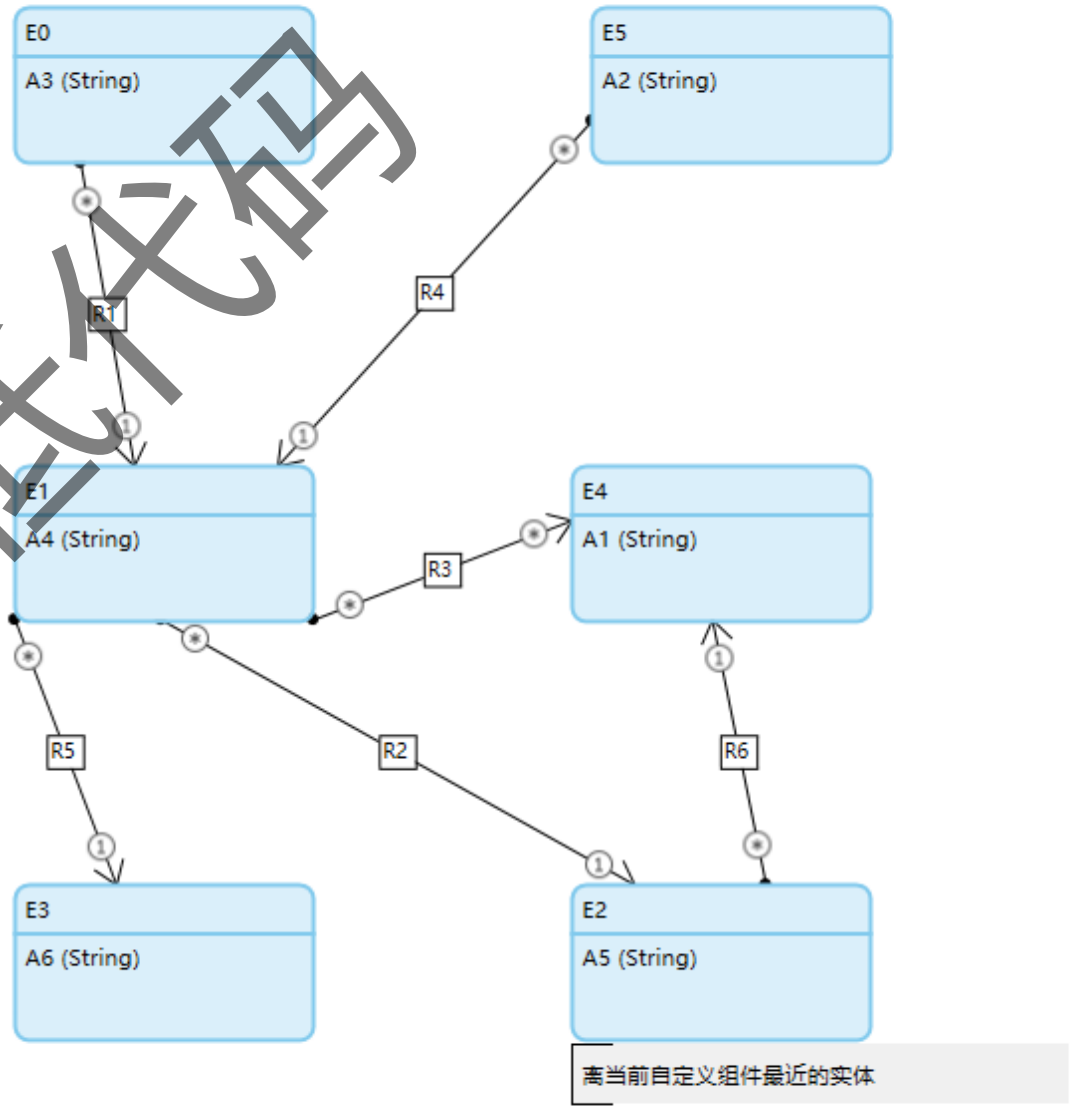
```xml
<?xml version="1.0" encoding="utf-8"?>
<widget id="mendixcn.graph.Graph" pluginWidget="true" needsEntityContext="true" offlineCapable="true"
supportedPlatform="Web" xmlns="http://www.mendix.com/widget/1.0/" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://www.mendix.com/widget/1.0/ ../node_modules/mendix/
custom_widget.xsd">
    <name>Graph</name>
    <description>My widget description</description>
    <icon/>
    <properties>
        <propertyGroup caption="General">
            <property key="sampleText" type="string" required="false">
                <caption>Default value</caption>
                <description>Sample text input</description>
            </property>
        </propertyGroup>
        <propertyGroup caption="第一课">      You, 3 weeks ago via PR #1 • 第一课: 展示实体属性
            <property key="myAttribute1" type="attribute" required="false">
              <caption>当前组件实体上下文的字符属性</caption>
              <description></description>
              <attributeTypes>
                <attributeType name="String" />
              </attributeTypes>
            </property>
        </propertyGroup>
    </properties>
</widget>
```

SIEMENS

展示上下文单一属性







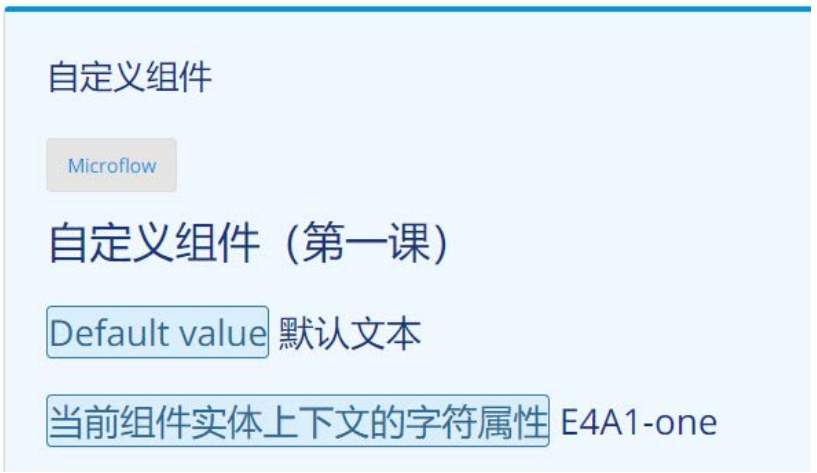
实现

```
import { createElement } from "react";

import { GraphContainerProps } from "../typings/
GraphProps";

export default function (props:
GraphContainerProps) {
    console.log(props);

    return <div></div>;
}
```

```
import { createElement, useMemo } from "react";
import { ValueStatus } from 'mendix';
import { GraphContainerProps } from "../typings/
GraphProps";

export default function (props:
GraphContainerProps) {
    const myAttribute1 = useMemo(() => {
        if (props.myAttribute1?.status ===
        ValueStatus.Available) {
            return props.myAttribute1.value;
        }
    }, [props.myAttribute1]);

    return <div>
        <h1>自定义组件（第一课）</h1>
        <h2><span className="alert-info
        panel">Default value</span>     {props.
        sampleText}</h2>
        <h2><span className="alert-info panel">当前
        组件实体上下文的字符属性</span>
        {myAttribute1 ? myAttribute1 : '--'}</h2>
    </div>;
}
```

提取上下文属性

渲染



SIEMENS

内容提要

Mendix前端组件的前世今生

实体的生命周期

XPATH的通俗理解

一次典型的网络请求发生了什么

组件脚手架一键创建

实战讲解组件开发的API设计

- 上下文属性集成
- **上下文数据源及其属性集成**
- 动作集成

上下文数据源及其属性集成

API

```
16                  <description></description>
17                  <attributeTypes>
18                      <attributeType name="String" />
19                  </attributeTypes>
20              </property>
21          </propertyGroup>
22+         <propertyGroup caption="第二课">
23+             <property key="myDataSource" type="datasource" required="false" isList="true">
24+                 <caption>Data Source</caption>
25+                 <description></description>
26+             </property>
27+             <property key="myAttribute2" type="attribute" required="false" dataSource="myDataSource">
28+                 <caption>Attribute</caption>
29+                 <description></description>
30+                 <attributeTypes>
31+                     <attributeType name="String" />
32+                     <attributeType name="Binary" />
33+                     <attributeType name="Boolean" />
34+                     <attributeType name="DateTime" />
35+                     <attributeType name="Decimal" />
36+                     <attributeType name="Enum" />
37+                     <attributeType name="HashString" />
38+                     <attributeType name="Integer" />
39+                     <attributeType name="Long" />
40+                     <attributeType name="AutoNumber" />
41+                 </attributeTypes>
42+             </property>
43+         </propertyGroup>
44      </properties>
45  </widget>
```

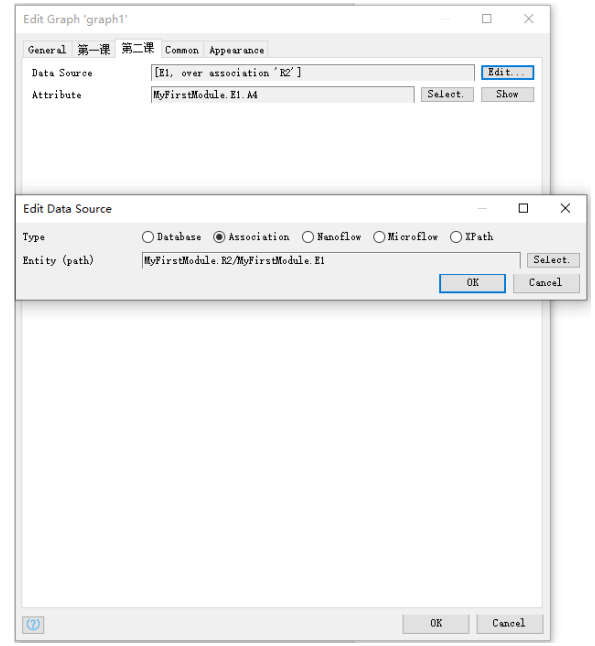
属性上下文

SIEMENS

实现

```
export default function (props: GraphContainerProps) {
    const myAttribute1 = useMemo(() => {
        if (props.myAttribute1?.status === ValueStatus.Available) {
            return props.myAttribute1.value;
        }
    }, [props.myAttribute1]);

    const data = useMemo(() => {        You, 2 hours ago • 渲染列表
        if (props.myDataSource?.status === ValueStatus.Available) {
            return props.myDataSource.items;
        }
    }, [props.myDataSource]);                    获取实体集

    return <div>
        <h1>自定义组件 (第一课) </h1>
        <h2><span className="alert-info panel">Default value</span>     {props.sampleText}</h2>
        <h2><span className="alert-info panel">当前组件实体上下文的字符属性</span>     {myAttribute1 ?
        myAttribute1 : '--'}</h2>
        <h2>
            <span className="alert-info panel">列表</span>
            <ul>
                {
                    data?.map(item => <li><button>{props.myAttribute2?.get(item).value}</button></li>)
                }
            </ul>
        </h2>
    </div>;
}
```

获取以实体集中特定实体为上下文的属性值



SIEMENS

效果

SIEMENS

内容提要

Mendix前端组件的前世今生

实体的生命周期

XPATH的通俗理解

一次典型的网络请求发生了什么

组件脚手架一键创建

实战讲解组件开发的API设计

- 上下文属性集成
- 上下文数据源及其属性集成
- **动作集成**

动作集成

API

```
        <propertyGroup caption="第二课">
            <property key="myDataSource" type="datasource" required="false" isList="true">
                <caption>Data Source</caption>
                <description></description>
            </property>
            <property key="myAttribute2" type="attribute" required="false" dataSource="myDataSource">
                <caption>Attribute</caption>
                <description></description>
                <attributeTypes>
                    <attributeType name="String" />
                    <attributeType name="Binary" />
                    <attributeType name="Boolean" />
                    <attributeType name="DateTime" />
                    <attributeType name="Decimal" />
                    <attributeType name="Enum" />
                    <attributeType name="HashString" />
                    <attributeType name="Integer" />
                    <attributeType name="Long" />
                    <attributeType name="AutoNumber" />
                </attributeTypes>
            </property>
        </propertyGroup>
        <propertyGroup caption="第三课">      You, 2 hours ago • 动作绑定
            <property key="mySelectAction" type="action" dataSource="myDataSource">
                <caption>选择动作</caption>
                <description></description>
            </property>
        </propertyGroup>
    </properties>
</widget>
```

动作上下文

动作

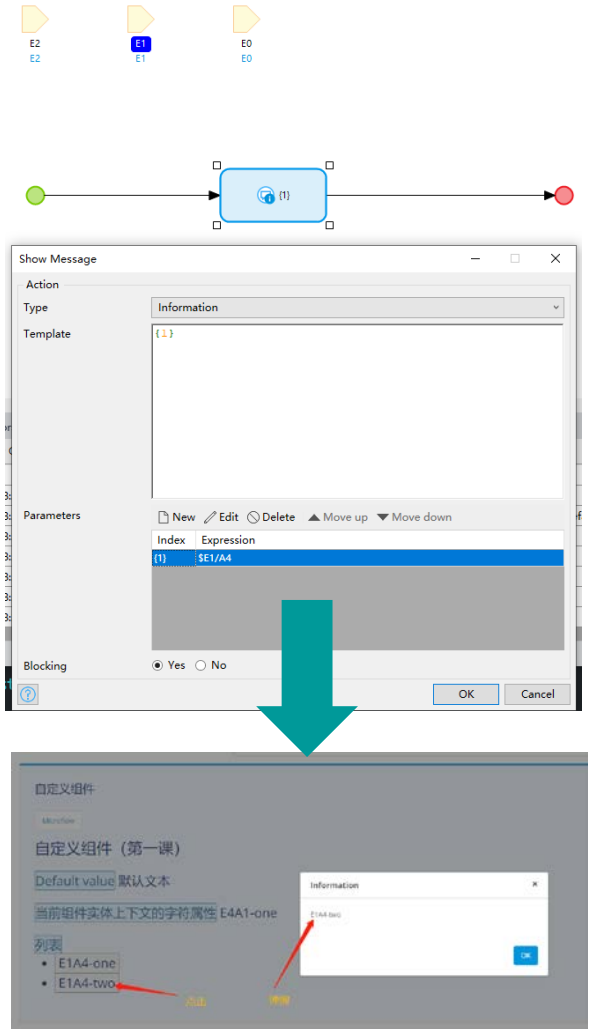
SIEMENS

实现

```tsx
export default function (props: GraphContainerProps) {
    const myAttribute1 = useMemo(() => {
        if (props.myAttribute1?.status === ValueStatus.Available) {
            return props.myAttribute1.value;
        }
    }, [props.myAttribute1]);

    const data = useMemo(() => {
        if (props.myDataSource?.status === ValueStatus.Available) {
            return props.myDataSource.items;
        }
    }, [props.myDataSource]);

    const onClick = useCallback(
        (item: ObjectItem) => {
            const myAction = props.mySelectAction?.get(item);
            if (myAction && myAction.canExecute) {
                myAction.execute();
            }
        },
        [props.mySelectAction],
    )

    return <div>
        <h1>自定义组件（第一课）</h1>
        <h2><span className="alert-info panel">Default valu      n>      {props.sampleText}</h2>

        <h2><span className="alert-info panel">当前组件实体上下文      性</span>      {myAttribute1 ? myAttribute1 : '--'}</h2>

        <h2>
            <span className="alert-info panel">列表</span>
            <ul>
                {
                    data?.map(item => <li><button onClick={onClick.bind(null, item)}>{props.myAttribute2?.get(item).value}</button></li>)

                }
            </ul>
        </h2>
    </div>;
}
```

执行动作

绑定事件



SIEMENS