컴포넌트 개발 가이드

> *tb_package.json, property_panel_info, WScript 마이그레이션 종합 문서*

1. 개요

RENOBIT에서 새 컴포넌트를 개발하려면 다음이 필요합니다:

1. **컴포넌트 클래스** - WVDOMComponent(2D) 또는 WV3DResourceComponent(3D) 상속
2. **tb_package.json** - 컴포넌트 등록 정보 (서버/에디터 연동)
3. **property_panel_info** - 속성 패널 UI 정의 (에디터 내 속성 편집)

---

2. tb_package.json

2.1 전체 구조

```
{
  "UPDATE": {
    "componentInstallService.insertData": [
      {
        "TB_PACKAGES": [ /* 컴포넌트 배열 */ ]
      }
    ]
  },
  "DELETE": {
    "adminService.deletePack": [
      { "pack_name": "PackName" }
    ]
  }
}
```

2.2 NAME, 폴더명, 파일명, 클래스명 규칙

> ***중요***: `NAME` *, 폴더명, 파일명, 클래스명은* ***반드시 동일****해야 합니다.*

```
NAME: "MyComponent"
        ↓
FILE_PATH: "custom/packs/{PACK}/components/{NAME}/{NAME}.js"
                                            ↑       ↑
                                          폴더명   파일명
        ↓
JS 파일 내: class MyComponent extends WVDOMComponent { ... }
                  ↑
              클래스명
```

폴더 구조:

```
components/
└── MyComponent/          ← NAME과 동일
    ├── MyComponent.js    ← NAME과 동일, 내부 class MyComponent
    ├── preview.png       ← 컴포넌트 패널 미리보기 이미지
    └── popup.vue         ← 속성 편집 팝업 (선택사항)
```

2.3 TB_PACKAGES 필드 명세

필드	필수	타입	역할	사용처
FILE_ID	✅	string	컴포넌트 고유 식별자	componentPanelInfoMap의 키
LABEL	✅	string	에디터 UI 표시명	컴포넌트 패널 목록
NAME	✅	string	컴포넌트 클래스명	eval(name)으로 클래스 로드, 폴더/파일명과 동일
INSTANCE_NAME	✅	string	인스턴스 기본 접두사	인스턴스명 생성 시 사용
TYPE	✅	string	"component" / "extension"	로딩 분기 처리
PACK_NAME	✅	string	소속 팩 이름	그룹핑, 삭제 시 사용
LAYER	✅	string	"2D" / "3D"	레이어 분류
CATEGORY	✅	string	컴포넌트 카테고리	패널 트리 구조
FILE_PATH	✅	string	JS 파일 경로	동적 스크립트 로딩
STYLE_PATH	❌	string	CSS 파일 경로	스타일시트 로딩
PROPS	✅	object	컴포넌트 기본 속성	드래그&드롭 시 전달
MAJOR	✅	string	메이저 버전	버전 관리
MINOR	✅	string	마이너 버전	버전 관리

2.4 PROPS 내부 구조

```
{
  "id": "MyComponent",        // FILE_ID와 동일
  "label": "My Component",    // LABEL과 동일
  "name": "MyComponent",      // NAME과 동일
  "show": true,               // 컴포넌트 패널 표시 여부
  "initProperties": {         // 초기 속성 오버라이드 (선택)
    "props": {
      "setter": {}
    }
  }
}
```

2.5 예시

2D 컴포넌트:

```
{
  "FILE_ID": "FreeCode",
  "LABEL": "FreeCode",
  "NAME": "FreeCode",
  "INSTANCE_NAME": "FreeCode",
  "TYPE": "component",
  "PACK_NAME": "Template",
  "LAYER": "2D",
  "CATEGORY": "FreeCode",
  "FILE_PATH": "custom/packs/Template/components/FreeCode/FreeCode.js",
  "PROPS": {
    "id": "FreeCode",
```

```
    "label": "FreeCode",
    "name": "FreeCode",
    "show": true
  },
  "MAJOR": "3.2",
  "MINOR": "0"
}
```

3D 컴포넌트:

```
{
  "FILE_ID": "Model Loader",
  "LABEL": "ModelLoader",
  "NAME": "ModelLoaderComponent",
  "INSTANCE_NAME": "3d_Model",
  "TYPE": "component",
  "PACK_NAME": "Three",
  "LAYER": "3D",
  "CATEGORY": "Modeling",
  "FILE_PATH": "custom/packs/Three/components/basic/ModelLoaderComponent/ModelLoaderComponent.js",
  "PROPS": {
    "id": "Model Loader",
    "label": "Model Loader",
    "name": "ModelLoaderComponent",
    "show": true,
    "initProperties": {
      "props": {
        "setter": {}
      }
    }
  },
  "MAJOR": "174",
  "MINOR": "0"
}
```

---

3. property_panel_info

```
WVPropertyManager.attach_default_component_infos(FreeCode, { // FreeCode (2D)
  info: {
    componentName: 'FreeCode',
    componentType: 'htmlCssJsEditable',
    version: '3.3.0',
  },
  setter: {
    width: 200,
    height: 200,
    widthUnit: 'px',
    heightUnit: 'px',
  },
  publishCode: {
    htmlCode: '',
    cssCode: '',
```

```
  },
});
```

```
WV3DPropertyManager.attach_default_component_infos(ModelLoaderComponent, { // ModelLoaderComponent (3D)
  info: {
    componentName: 'ModelLoaderComponent',
    componentType: 'htmlCssJsEditable',
    version: '174.0',
  },
  setter: {
    selectItem: '',
  },
  label: {
    label_text: 'ModelLoaderComponent',
  },
  publishCode: {
    htmlCode: '',
    cssCode: '',
  },
});
```

3.2 추가 속성 정의 (add_property_panel_group_info)

```
WV3DPropertyManager.add_property_panel_group_info(ModelLoaderComponent, { // ModelLoaderComponent (3D)
  label: 'Resource',
  template: 'resource',
  children: [
    {
      owner: 'setter',
      name: 'selectItem',
      type: 'resource',
      label: 'Resource',
      show: true,
      writable: true,
      description: '3D Object',
      options: { type: 'gltf' }
    }
  ]
});
```

4. 컴포넌트 생명주기

4.1 실행 순서

등록 시점 (컴포넌트 초기화):

```
1. _onViewerReady()        ← 컴포넌트 클래스 내부
2. WScript REGISTER        ← 사용자 스크립트
```

소멸 시점 (컴포넌트 제거):

```
1. WScript BEFORE_DESTROY  ← 사용자 스크립트
2. _onViewerDestroy()      ← 컴포넌트 클래스 내부
3. WScript DESTROY         ← 사용자 스크립트
```

4.2 주요 생명주기 메서드

메서드	실행 시점	용도
`_onViewerReady()`	뷰어에서 컴포넌트 준비 완료	초기화 로직
`_onViewerDestroy()`	뷰어에서 컴포넌트 제거 전	정리 로직
`_onBeforeRegister()`	컴포넌트 등록 직전	속성 업데이트
`_onDestroy()`	컴포넌트 완전 제거	최종 정리

> ***Note**: `_onViewerReady()`, `_onViewerDestroy()` 는 **뷰어 모드에서만** 실행됩니다.*

---

5. WScript에서 마이그레이션

5.1 register → 클래스 메서드 + _onViewerReady

WScript:

```
// register 이벤트
function renderChart(data) {
  const option = { /* ECharts 옵션 */ };
  this.chart.setOption(option);
}
this.renderChart = renderChart.bind(this);

const chart = echarts.init(this.element.querySelector('#echarts'));
this.chart = chart;
```

컴포넌트 클래스:

```
class MyChart extends WVDOMComponent {
  // WScript의 function 정의 → 클래스 메서드로 변환
  renderChart(data) {
    const option = { /* ECharts 옵션 */ };
    this.chart.setOption(option);
  }

  _onViewerReady() {
    // 초기화 로직만 작성 (메서드 정의는 클래스 본문에)
    const chart = echarts.init(this.element.querySelector('#echarts'));
    this.chart = chart;
  }
}
```

> ***핵심 포인트***
>
> - *WScript의 `function` 정의 → **클래스 메서드**로 변환*
> - *`_onViewerReady()` 에는 **초기화 로직만** 작성*
> - *register의 모든 코드를 `_onViewerReady()` 에 넣지 않음*

5.2 beforeDestroy → _onViewerDestroy

WScript:

```
// beforeDestroy 이벤트
if (this.chart) {
  this.chart.dispose();
  this.chart = null;
}
```

컴포넌트 클래스:

```
class MyChart extends WVDOMComponent {
  _onViewerDestroy() {
    if (this.chart) {
      this.chart.dispose();
      this.chart = null;
    }
  }
}
```

---

6. 컴포넌트 개발 템플릿

6.1 2D 컴포넌트 (WVDOMComponent)

```
class MyComponent extends WVDOMComponent {
  constructor() {
    super();
  }

  // 클래스 메서드 (WScript function 정의를 여기로)
  myMethod(data) {
    // 비즈니스 로직
  }

  _onBeforeRegister() {
    super._onBeforeRegister();
  }

  _onViewerReady() {
    // 뷰어 전용 초기화 로직
  }

  _onViewerDestroy() {
    // 뷰어 전용 정리
  }

  _onDestroy() {
    // 에디터 포함 컴포넌트 정리 로직 (E.g. Preview Tab)
    super._onDestroy();
  }
}

// 기본 속성 등록 (property_panel_info, event_info 자동 포함)
WVPropertyManager.attach_default_component_infos(MyComponent, {
  info: {
    componentName: 'MyComponent',
```

```
    componentType: 'htmlCssJsEditable',
    version: '1.0.0',
  },
  setter: {
    width: 200,
    height: 200,
    widthUnit: 'px',
    heightUnit: 'px',
  },
  style: {
    backgroundColor: 'rgba(255,255,255,0.3)',
  },
  publishCode: {
    htmlCode: '<div class="my-component">Hello</div>',
    cssCode: '.my-component { color: blue; }',
  },
});
```

6.2 3D 컴포넌트 (WV3DResourceComponent)

```
class My3DComponent extends WV3DResourceComponent {
  constructor() {
    super();
    ComponentMixin.applyModelLoaderMixin(this);
  }

  // 클래스 메서드 (WScript function 정의를 여기로)
  myMethod(data) {
    // 비즈니스 로직
  }

  _onBeforeRegister() {
    super._onBeforeRegister();
    // 업데이트 로직
  }

  _onViewerReady() {
    // 뷰어 전용 초기화 로직
  }

  _onViewerDestroy() {
    // 뷰어 전용 정리
  }

  _onDestroy() {
    // 정리 로직
    super._onDestroy();
  }
}

// 기본 속성 등록
WV3DPropertyManager.attach_default_component_infos(My3DComponent, {
  info: {
    componentName: 'My3DComponent',
    componentType: 'htmlCssJsEditable',
```

```
    version: '1.0.0',
  },
  setter: {
    selectItem: '',
  },
  label: {
    label_text: 'My3DComponent',
  },
  publishCode: {
    htmlCode: '',
    cssCode: '',
  }
});

// 리소스 속성 추가
WV3DPropertyManager.add_property_panel_group_info(My3DComponent, {
  label: 'Resource',
  template: 'resource',
  children: [
    {
      owner: 'setter',
      name: 'selectItem',
      type: 'resource',
      label: 'Resource',
      show: true,
      writable: true,
      description: '3D Object',
      options: { type: 'gltf' },
    },
  ],
});
```

---

8. tb_package.json 등록

8.1 2D 컴포넌트

```
{
  "FILE_ID": "MyComponent",
  "LABEL": "My Component",
  "NAME": "MyComponent",
  "INSTANCE_NAME": "my_component",
  "TYPE": "component",
  "PACK_NAME": "CustomPack",
  "LAYER": "2D",
  "CATEGORY": "Custom",
  "FILE_PATH": "custom/packs/CustomPack/components/MyComponent/MyComponent.js",
  "PROPS": {
    "id": "MyComponent",
    "label": "My Component",
    "name": "MyComponent",
    "show": true
  },
  "MAJOR": "1.0",
```

```
  "MINOR": "0"
}
```


8.2 3D 컴포넌트


```
{
  "FILE_ID": "My3DComponent",
  "LABEL": "My 3D Component",
  "NAME": "My3DComponent",
  "INSTANCE_NAME": "3d_my_component",
  "TYPE": "component",
  "PACK_NAME": "CustomPack",
  "LAYER": "3D",
  "CATEGORY": "Custom",
  "FILE_PATH": "custom/packs/CustomPack/components/basic/My3DComponent/My3DComponent.js",
  "PROPS": {
    "id": "My3DComponent",
    "label": "My 3D Component",
    "name": "My3DComponent",
    "show": true,
    "initProperties": {
      "props": {
        "setter": {}
      }
    }
  },
  "MAJOR": "1.0",
  "MINOR": "0"
}
```


최종 업데이트: 2026-01-18