(O) Modelo de renderizado JS 2.0

Cobertura

Objetivos

- Para conocer los fundamentos de un objeto **de modelo JS Model**
- Para aprender cómo integrar JS Models en su entorno Magnolia
- Para aprender cómo ejecutar la lógica backend usando solo archivos javascript

Abstracto

- Proporciona una capacidad de módulo ligero para modelos de renderizado basados en Java sin la necesidad de volver a implementar Magnolia.
- Definir la lógica ejecutada por el backend usando Javascript
- Accesible a cualquier script de Freemarker a través del atributo de contexto **del modelo**
- Consulte Modelos Javascript para obtener más información sobre los modelos de renderizado JS y la documentación del modelo para obtener más información sobre los modelos de renderizado Java.

Conceptos de modelos de renderizado de Javascript

- Proporciona la capacidad de ejecutar lógica de backend y exponer la lógica a scripts de Freemarker.
- La lógica de backend se puede reutilizar en varias plantillas usando la propiedad **modelClass** y se puede hacer referencia a ella en Freemarker usando el objeto de contexto ${ **model**} .

Ventajas de utilizar modelos de renderizado Javascript

- A diferencia de los modelos de renderizado basados en Java, no es necesario reiniciar el servidor. Una vez que el archivo se guarda en un módulo de luz, su lógica se ejecutará en la siguiente invocación del objeto **modelo** .
- La lógica empresarial se puede modificar fácilmente sin tener que cambiar la lógica de representación del script Freemarker.
- La lógica empresarial puede estandarizarse y compartirse entre múltiples scripts.

Ejemplos de código

- Creando un nuevo modelo de renderizado Javascript
- (*requerido*) Definir la ejecución
- Crea funciones y accede a ellas en Freemarker

```
this.execute = function () {
    if (!cmsfn.isEditMode()) {
        let redirectToPage = this.getRedirectLinkManuallyDefined()
        if (StringUtils.isBlank(redirectToPage)) {
            redirectToPage = this.getRedirectLinkToFirstChild();
        }

        if (!StringUtils.isEmpty(redirectToPage)) {
            ctx.getResponse().sendRedirect(redirectToPage);

        } else {
            ctx.getResponse().sendError(HttpServletResponse.SC_NOT_FOUND);
        }

        if (cmsfn.isPublicInstance() || !StringUtils.isEmpty(redirectToPage)) {
            return RenderingModel.SKIP_RENDERING;
        }
    }
    return "";
}

this.getRedirectLinkToFirstChild = function () {
    this.log.info("redirectLinkPathToFirstChild")
    const allChildren = NodeUtil.getNodes(content.getJCRNode(), NodeTypes.Page.NAME);
    const iterator = allChildren.iterator();
    if (iterator.hasNext()) {
        return cmsfn.link(iterator.next());
    }
    return null;
}
```

Implementación

- Crear un nuevo modelo de renderizado Javascript

```
// Declare your Javascript object
var RedirectModel = function () {
}
// Instantiate the object
new RedirectModel()
```

ⓘ Necesitará usar `var` y no `const` o `let` para que esto funcione correctamente.

- Haga referencia a su modelo de representación de Javascript en la definición de plantilla

```
title: Training - Redirect Template Javascript Models
class: info.magnolia.module.jsmodels.rendering.JavascriptTemplateDefinition
renderType: freemarker
dialog: training-backend-js-magic:pages/redirect
templateScript: /training-backend-js-magic/templates/pages/redirect.ftl
modelClass: info.magnolia.module.jsmodels.rendering.JavascriptRenderingModel
modelPath: /training-backend-js-magic/templates/pages/redirect-with-js.js
```

- Accediendo a objetos Java

```
const NodeUtil = Java.type("info.magnolia.jcr.util.NodeUtil")
const NodeTypes = Java.type("info.magnolia.jcr.util.NodeTypes")
const StringUtils = Java.type("org.apache.commons.lang3.StringUtils")
```

- Definir la función de ejecución

```
this.execute = function () {
    if (!cmsfn.isEditMode()) {
        let redirectToPage = this.getRedirectLinkManuallyDefined()
        if (StringUtils.isBlank(redirectToPage)) {
            redirectToPage = this.getRedirectLinkToFirstChild();
        }

        if (!StringUtils.isEmpty(redirectToPage)) {
            ctx.getResponse().sendRedirect(redirectToPage);

        } else {
            ctx.getResponse().sendError(HttpServletResponse.SC_NOT_FOUND);
        }

        if (cmsfn.isPublicInstance() || !StringUtils.isEmpty(redirectToPage)) {
            return RenderingModel.SKIP_RENDERING;
        }
    }
    return "";
}
```

- Definir un método accesible

```
this.getRedirectLinkManuallyDefined = function () {
    this.log.info("redirectLinkPathManuallyDefined")
    const contentNode = content.getJCRNode()
    if (contentNode.hasProperty(TARGET_LINK_PROPERTY_NAME) && StringUtils.isNoneBlank(PropertyUtil.getString(contentNode, TARGET_LINK_PROPERTY_NAME))) {
        const pageId = contentNode.getProperty(TARGET_LINK_PROPERTY_NAME).getString();
        return cmsfn.link(RepositoryConstants.WEBSITE, pageId);
    }
    return null;
}
```

- Utilice el método en Freemarker

```
[#assign targetPageLink = model.getRedirectLinkManuallyDefined()!]
[#if targetPageLink?has_content]
    <div class="panel-heading">
        <h1 class="panel-title">Success: Redirect to manually defined page.</h1>
    </div>
    <div class="panel-body">
        <p>This page will redirect to the manually defined target page in the page's dialog.</p>
        <p><a href="${targetPageLink}">${targetPageLink}</a></p>
    </div>

[/#if]
```