Выступление: Classic UI Dynamic Dialogs

слайд 2:

Добрый день, меня зовут Павел Носов и я хотел бы представить вам презентацию на тему Classic UI Dynamic Dialogs.

слайд 2:

Я из кампании Hight Torque Software. Мы занимаемся разработкой AEM/CQ проектов уже больше 3 лет. Начали с CQ5.2, сейчас работаем с CQ5.5, AEM6.0

слaйд 3:

За это враме я несколько раз сталкивался с необходимость реализовывать динамические диалоги. И по своему опыту могу сказать, что те примеры которой присутствуют в документации CQ не в полной мере покрывают процесс создания динамических диалогов и описаны поверхностно. Также документаций по Widget API, с которой приходиться работать, не первых стадиях больше мешает чем помогает. Она обширна, но информации слишком много, и нет описанного критического пути, по которому можно было бы идти в процессе создания динамического диалога.

По этому задачей данного доклада является:

1) Предоставить критический путь создания динамических диалогов

2) Рассмотреть на примере процесс создания динамических диалогов

3) Выделить "BEST PRACTICE"

Для этого предлагаю рассмотреть следующие темы: ...перечислить пункты со слайда...

слайд 4:

Начнём с рассмотрения что же такое Classic UI диалог, как его создать, каким образом он появляется на странице.

слайд 5:

Classic UI диалог - это нод с именем «dialog» типа cq:Dialog. Дочерние ноды dialog образуют структуру будущей формы диалога. Они могут быть типов cq:Widget, cq:WidgetCollection, cq:Panel, nt:unstructured и других.

слайд 6:

В конечном случае за контент отвечают cq:Widget. Основное поле cq:Widget, отвечающее за то в какой элемент управления будет интерпретирован нод - это xtype. Каждому значению xtype в Widgets API сопоставлен класс, написанный на базе ExtJS. Этот класс отвечает за построение элемента управления из нода и за поведение элемента управления

слайд 7:

Конечный диалог, как собственно и вся диалоговая система Classic UI, строиться на клиенте.

слайд 8:

Давайте посмотрим как. В edit mode CQ добавляет ,после каждого HTML компонента, блок который вы видите на слайде.

слайд 9:

Как вы видите, блок состоит из вызова функции CQ.WCM.edit(...) в которую передаётся конфигурационный JSON. JSON содержит путь к диалогу, путь к контенту, resourceType и почую информацию необходимую для построения диалоговой системы данного компонента.

слайд 10:

Затем в момент построения страницы функция из данного блока выполняется, в результате строиться диалоговая система компонента. Частью этой диалоговой системы является и сам диалог.

слайд 11:

CQ предоставляет набор xtypes который можно найти по ссылке представленной на слайде. Этого набора и статических диалогов достаточно для организации взаимодействия с компонентами в 95% случаев. Особенно для проектов, который находиться в активной стадии разработки. Так как любой сложный компонент с замысловатым поведение можно переразбить на более простые компоненты, и тем самым понизить сложность как самого компонента, так и его диалога.

Но бывают случаи, особенно на старых проектах, когда нужно доработать функционал в уже существующий компонент, при этом не меняя его поведения на уже существующих страницах. Это приводит к разрастанию компонента, и к усложнению его диалоговой системы. Диалог обвешивается вкладками, кладки полями. И в конечном итоге взаимодействие автора с диалогом превращается а муку. Это в свою очередь вызывает кучу ошибок и необходимость писать подробные инструкции...

слайд 12:

В общем страдает User experience.

Для решения данной проблемы необходимо реализовывать динамическое поведение, которое призвано разгрузить диалог и соответственно улучшить User experience

сдайд 13:

Есть много способов как можно реализовать это динамическое поведение. Но остановимся на одном из подходов - на построении динамического поведения при помощи обработки событий генерируемых стандартными Widgets.

слайд 14:

Для начала давайте разберёмся, как повесить event listener на событие, что такое event listener, и где взять список доступных событий.

Для того чтобы повесить обработчик на событие, под нодом виджета необходимо создать нод с именем "listeners" типа "nt:unstructured".

слайд 15:

Затем под нодом listeners необходимо создать поле именем которого будет являться имя события, а значение - обычная js функция

слайд 16:

Для того чтобы найти доступные события необходимо взять xtype поля, и найти в Widgets API документации сопоставленный ему класс.

слйд 17:

Затем, в Widgets API документации соответствующего класса необходимо найти раздел с именем "Public Events". В этом разделе перечисленны все доступные события а также описан формат обработчиков.

слайд 18:

Для поиска классов соответствующих xtype удобно использовать страницу из CQ документации "Using xtypes". Так же можно использовать поиск или же дерево контента на странице Wigdets API документации.

Вот собственно и вся теория, а теперь давайте перейдём к практике.

слайд 19:

Давайте реализуем динамический диалог следующего вида (показать на страничке динамический диалог).

слайд 20:

Пройдём шаги представленные на слайде один за одним

Описание кода:

1 init) Начнем с создания тестового проекта

git checkout init

mvn clean install -PauthInstallPackage

2 step1) Теперь создадим диалог со всеми табами и прочими элементами управления необходимыми для будущего динамического диалога.

git checkout step1

mvn clean install -PauthInstallPackage

открыть dialog.xml, показать структуру.

перейти на страницу, посмотреть структуру диалога на странице

3 step2) Выберем события, которые будем использовать для реализации динамического поведения. Для этого создадим обработчики для всех возможных событий для Selection поля на Config табе.

git checkout step2

mvn clean install -PauthInstallPackage

Показать dialog.xml

показать лог событий на странице

4 step3) Создадим clientlibs для будущих обработчиков

git checkout step3

mvn clean install -PauthInstallPackage

показать папку с clientlibs

показать head.html

5 step4) Спрячем все табы при помоoи обработчика, реализуем код для обработки изменения значения управляющего поля

git checkout step4

mvn clean install -PauthInstallPackage

показать tabs.js

показать dialog.xml

показать на странице спрятанные табы

6 step5) повесим обработчики на события управляющего поля

git checkout step5

mvn clean install -PauthInstallPackage

показать tabs.js

показать dialog.xml

показать на странице меняющееся табы и баг с валидацией

7 done) исправить проблему с валидацией, добавить обработчики для каждого таба

git checkout done

mvn clean install -PauthInstallPackage

показать tabs.js

показать dialog.xml

показать на странице конечный результат

слайды 21-40

Описать BEST PRACTICE