0.1 Что такое динамическая форма?

0.2 Режимы открытия динамических форм

Динамические формы поддерживают несколько режимов работы:

1. `MODE_CREATE_INTERACTION` – режим создания нового обращения из шаблона
2. `MODE_OPEN_INTERACTION` – режим чтения обращения
3. `MODE_OPEN_INCIDENT` – режим работы с инцидентом
4. `MODE_COPY_TEMPLATE` – режим копирования обращения

0.2.1 MODE_CREATE_INTERACTION

В режиме создания обращения возможность редактирования полей устанавливается флагом `sbmodify` в элементах шаблона: если он равен `true` – то динамическое поле доступно для редактирования, если такую возможность подразумевает типа данного поля. При открытии шаблона выполняются динамические команды, указанные в свойствах `sbcommand` динамических полей последовательно, в порядке их объявления в шаблоне. Кнопка «Создать обращение» отображается по умолчанию

0.2.2 MODE_OPEN_INTERACTION

В режиме чтения обращения все поля по умолчанию закрыты для редактирования, флаг `sbmodify` игнорируется. Для «Разблокирования» некоторых полей существует механизм `unreadEditableFields` – данный атрибут обращения соддержит массив идентификаторов динамических полей, которые будут доступны для редактирования в данном режиме. Команды `sbcommand` выполняются только в разблокированных полях, вместо них отрабатывают команды команды из атрибутов `sbonload` динамических полей. Кнопка «Сохранить обращение» отображается только в том случае, если массив `unreadEditableFields` непустой.

0.2.3 MODE_OPEN_INCIDENT

Режим работы с инцидентом аналогичен с режиму чтения обращения – все поля закрыты для редактирования, однако, механизм `unreadEditableFields` недоступен. При этом в интерфейсе доступны дополнительные действия для управления инцидентом: «Решить», «Назначить на другую группу» и «Вернуть диспетчеру» Поддерживается только в среде выполнения domcore.

0.2.4 MODE_COPY_TEMPLATE

Режим копирования обращения.

0.3 Общее описания динамического поля

Каждое динамическое поле представляет собой набор стандартных и дополнительных параметров отвечающих за визуальное отображение и функциональное поеведение элемента. Определение элемента задается при формировании шаблонов в HPSM, разнообразие типов достигается комбинацией значений параметров **sbtype**, **type**, **logicalType**, **objectType**, **logicalType** В качестве примера приведем простое текстовое поле:

```
{
"objectType": "text",
"logicalType": "text",
"id": "editmarriagenumauto",
"label": "Свидетельство о браке",
"mandatory": false,
"sbcommand": "",
"sbmask": "",
"sbmodify": true,
"sbtask": "",
"sbtitle": "Введите серию и номер свидетельства о браке как указано в докум
"sbtype": "",
"sbstyle": "width:33%",
"style": "text",
"visible": false,
```

```
"width": "",
"sbobject": null,
"sbdbfield": null,
"sbaction": null,
"sbmode": null,
"sbonload": null,
"sbcopyinfo": false,
"childs": null,
"groupid": "",
"text": null,
"options": null,
"type": "2",
"multiline": null,
"button": null,
"matchTable": null,
"matchField": null,
"query": null,
"hpcGroupByFields": null,
"sbfield": ""
}
```


и элемент типа select:


```
{
"objectType": "select",
"logicalType": "combo",
"id": "purpose",
"label": "Цель поездки",
"mandatory": true,
"sbcommand": "setvalue([something],[purpose])",
"sbmask": "",
"sbmodify": true,
"sbtask": "setvalue([something],[purpose])",
"sbtitle": "",
"sbtype": "",
"sbstyle": "",
"style": "combo",
"visible": true,
```

```
"width": "",
"sbobject": null,
"sbdbfield": "purpose",
"sbaction": null,
"sbmode": null,
"sbonload": "setvalue([something],[purpose])",
"sbcopyinfo": false,
"children": null,
"groupid": "",
"text": "Встречи с клиентами",
"options": [
{
"text": "Выезды на аварии",
"label": "id02",
"id": "id02"
},
{
"text": "Встречи с клиентами",
"label": "Встречи с клиентами что-т там",
"id": "id03"
},
{
"text": "Выезды в государственные органы",
"label": "Выезды в государственные органы",
"id": "id04"
}
],
"type": "2",
"multiline": null,
"button": null,
"matchTable": null,
"matchField": null,
"query": null,
"hpcGroupByFields": null,
"sbfield": ""
}
```

0.3.1 Параметры динамического поля

1. `objectType` - параметр отвечающий за определение типа поля
2. `logicalType` - параметр отвечающий за определение типа поля
3. `sbtype` - параметр отвечающий за определение типа поля
4. `type` - параметр отвечающий за определение типа поля
5. `style` - параметр отвечающий за определение типа поля
6. `id` - уникальный идентификатор поля используется в командах динамического языка для поиска элемента
7. `text` - значение поля по умолчанию
8. `label` - заголовок поля, отображаемое значение для информации и обозначения поля
9. `sbtitle` - подсказка для поля, содержит дополнительную информацию о поле, выводиться в виде вопроса
10. `groupid` - параметр сортировки и формирования для плоских и legacy групп
11. `sbcommand` - содержит команды динамического языка выполняемые при инициализации `MODE_CREATE_INTERACTION`
12. `sbtask` - содержит команды динамического языка выполняемые при изменении поля через пользовательский ввода
13. `sbonload` - содержит команды динамического языка выполняемые при инициализации `MODE_OPEN_INTERACTION` или `MODE_OPEN_INCIDENT`
14. `mandatory` - флаг отвечающий за обязательность поля
15. `sbmask` - маска поля ввода(используется для текстовых полей и дат)
16. `sbmodify` - флаг определяющий можно ли изменять поле

17. `sbstyle` - параметр отвечающий за визуальные стили элемента вызывает команду setStyle над элементом

18. `visible` - флаг определяющий видимость поля в пользовательском интерфейсе

19. `width` - значение в % ширины элемента относительно блока

20. `sbobject` - HP SM property, не изменяется в рамках обработки

21. `sbdbfield` - HP SM property, не изменяется в рамках обработки

22. `sbaction` - HP SM property, может быть изменено командой setAction

23. `sbmode` - HP SM property, может быть изменено командой setMode

24. `sbcopyinfo` - флаг отвечающий за использовани

25. `children` - дочерние элементы используется в элементе типа group

26. `options` - варианты значений для поля для полей с выбором значений

27. `multiline` - флаг отвечающий за возможность многострочного ввода для текстовых полей

28. `button` - HP SM property, не изменяется в рамках обработки

29. `matchTable` - HP SM property, не изменяется в рамках обработки

30. `matchField` - HP SM property, не изменяется в рамках обработки

31. `query` - HP SM property, не изменяется в рамках обработки

32. `hpcGroupByFields` - HP SM property, не изменяется в рамках обработки

33. `sbfield` - HP SM property, не изменяется в рамках обработки

0.3.2 Управление состоянием

0.4 Текстовые поля

Текстовые поля могут быть четырех разных визуальных типов + все поля для которых не определены правила тоже преобразуются в обычное текстовое поле

1. Обычное текстовое поле
2. Неизменяемое текстовое поле(Label) - текстовое поле с флагом **modify** = "false"
3. Многострочное текстовое поле - текстовое поле с флагом **multiline** = "true"и параметром **type** = "2"
4. Числовое поле - текстовое поле с параметром **type** = "1"или **sbtype** = `NUMBER`

Текстовые поля поддерживают ввод по маске(параметр **mask** не пустой)

0.4.1 Маски ввода

Для ограничения ввода в текстовые поля используются обобщенные маски. Маски представляют собой собственные элементы по регулярным выражениям:

\# - число аналог стандартного [0-9]

9 - число или пробел [0-9]

A - символ в верхнем регистре [A-ZA-Я]

a - символ в нижнем регистре [a-za-я]

B - символ в любом регистре [A-ZA-Яa-za-я]

C - символ в люом регистре или число [A-ZA-Яa-za-я]

Остальные символы не из набра масок соответствует статическим символам, которые не участвуют в валидации маски

0.5 Поля с выбором значения

Динамическая форма поддерживает различные комбинации элементов с возможность выбрать одно из предлагаемых значений, ниже мы рассмотрим подробнее каждый из элементов

0.5.1 Select

Есть два варианта определния стандартного поле с выбором значения, которые определяются по одному из параметров **sbtype** = `COMBOAREA` или **style** = `COMBO` Варианты выбора могут быть записаны в поле в параметре **options**, а так же заполняться из команд динамического языка. Вариантами выбора могут быть текстовые значения, ссылки, изображения, а также их комбинация.

0.5.2 Suggest

Suggest или поле с подсказкой - расширение стандартного поля Select с возможностью поиска и фильтрации. Элемент определяется параметром **sbtype** = `SUGGEST` Частными случаями поля типа Suggest являются поля со значениями **sbtype** = `SEARCH` и `ADDRESS` предназначенные для поиска(запросы к Автоисполнятору) и получение адреса по вводу города/улицы/дома Для поля suggest есть дополнительный параметр `longSuggestItems` - отвечает за визуальное отображение элементов с длинными названиями в вариантах.

0.5.3 Чекбоксы

Чекбокс - элементы позволяющие сделать выбор из вариантов Да/Нет, Выбрано/Не выбрано. Определяется по параметрам **style** = `CHECKBOX` или **objectType** = `CHECKBOX`

0.5.4 Радиокнопки

Элемент типа радиокнопка определяется параметром **style** = `RADIO` и представляет собой выбор одно из предлагаемых значение. Элемент логически не отличается от стандартного Select, но визуальное и функциональное

поведение в рамках выполнения команд динамического языка отличается(подробнее в разд. 0.5.6)

0.5.5 Мультиэлементы

Все типы полей кроме радиокнопок имею свое представление в виде полей со множественным выбором - `multiSelect`, `multiSuggest`, `multiAddress`, `multiCheckbox`. Все мультиэлементы определяются параметром **sbtype** соответствующим типу поля. Визуально `multiSelect` и `multiSuggest` аналогичны своим единичным типам, но добавляют возможность добавлять несколько значений. При выборе значения сохраняются в поле **text** через символ ";". Так как не все системы поддерживают элементы со множественным выбором - дополнительно придуман метод сохранения таких полей, когда исходное поле визуально скрывается, а после него вставляется текстовое поле с многострочным вводом, каждая строка которого соответствует выбранному значению в исходном поле с индексом значения в начале строки.

0.5.6 Осбенности элементов с выбором значения

Все элементы с выбором значения кроме радиокнопок, чекбоксов и мультичекбоксов имею важную особенность при выполнении команд динамического языка: Если командой динамического языка вызвать установку значения элемента через функции `setValue`, `setValues` или `setSelectedValue` при этом передать одно единственное значение, то у элемента происходит запуск `sbtask` таким образом эмулируя действия пользователя. Примечание: функция `setSelectedValue` всегда вызывает запуск `sbtask` не зависимо от количества переданных значений для установки.

0.6