Учреждение образования
“Белорусский государственный университет
информатики и радиоэлектроники”

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине «Проектирование интеллектуальных геоинформационных систем»

на тему

«Написание примера (документации) по тому, как писать компоненты интерфейса при помощи sc-web »

Выполнили ст. гр. 921703: Козел А. С.

Мелешко А. С.

Суринович А. А.

Проверил: Самодумкин С. А.

Минск 2022

Тема:

Написание компонентов интерфейса с помощью sc-web.

Цель:

Проанализировать материал по написанию компонентов интерфейса с помощью sc-web, задокументировать информацию.

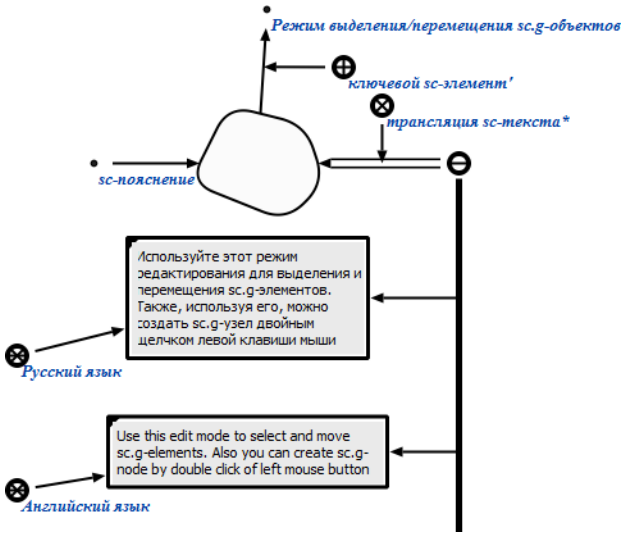
Введение:

1. база знаний (БЗ) системы редактируется множеством разработчиков, что можно даже делать во время эксплуатации системы;
2. разграничение доступа к различным частям базы знаний;
3. возможность отображать (редактировать) части базы знаний с использованием различных внешних языков (SCg-код , SCn-код, чертежи, ЕЯ и т. д.);
4. изменения в базе знаний в режиме реального времени должны отображаться у всех пользователей системы.

- обеспечивает взаимодействие между компонентами пользовательского интерфейса. Именно ядро принимает решение, какой компонент использовать для отображения информации в том или ином виде. Пользователь лишь указывает внешний язык, на котором он хочет увидеть ответ на свой запрос. Взаимодействие между компонентом и ядром ПИ осуществляется через песочницу - component sandbox. Компонент в своей работе может работать **лишь *со своей песочницей, не используя функции ядра напрямую. Это позволяет значительно сократить временные затраты на поддержку большого числа***

Компоненты ядра пользовательского интерфейса реализованы на клиентской части приложения с использованием JavaScript. По сути, они отвечают за весь базовый функционал клиентской части приложения и обеспечивают корректную работу компонентов. Вкратце опишем функции, которые выполняют компоненты ядра ПИ:

- отображение и инициирование команд ПИ, которые располагаются в главном меню ;
- логика работы с аргументами команд. В качестве аргумента команды может быть указан любой элемент на странице, у которого установлен атрибут *sc_addr* (адрес sc-элемента в памяти);
- навигация по истории диалога пользователя с системой;
- переключение между режимами идентификации (русский, английский). Ядро автоматически запрашивает идентификаторы на необходимом языке и меняет их для всех элементов на странице с атрибутом *sc_addr*. Если же идентификатор для указанного языка не найден, то устанавливается системный идентификатор sc-элемента;
- ***компонентов, когда происходят изменения в ядре ПИ;***
- обеспечивает поиск по идентификатору. При введении пользователем 3-х или более символов в строку поиска, ядро делает запрос на сервер, где происходит поиск всех идентификаторов, которые содержат указанную подстроку символов. На сервере имеется sc-агент, который реагирует на появление новых идентификаторов в системе, и добавляет их в базу ключ значений redis (в которой их ищет сервер по запросу клиента);
- отображение всплывающих подсказок. Для каждого элемента на странице с атрибутом sc_addr, при наведении, формируется всплывающая подсказка. Эта подсказка описана в базе знаний для разных языков



Компоненты пользовательского интерфейса реализуются также с использованием JavaScript. Раньше мы рассматривали лишь один вид

компонентов – компоненты, которые позволяют просматривать (редактировать) содержимое sc- ссылок (файлы). Но для того чтобы изменения в базе знаний отображались у пользователей в режиме реального времени этого было недостаточно. Чтобы решить данную проблему были добавлены компоненты, которые позволяют просматривать (редактировать) структуры напрямую из базы знаний.

При регистрации в ядре, компоненту ставится в соответствие некоторое его описание, где указано, может ли он отображать (редактировать) содержимое sc-ссылок и/или структур. В JavaScript коде это выглядит следующим образом:

```
SCgComponent = { ext_lang: 'scg_code',
  formats: ['format_scg_json'], struct_support: true,
  factory: function(sandbox) {
    return new scgWindow(sandbox);
  }
```

Рассмотрим поля объекта *SCgComponent* подробнее:

5. *ext_lang* – в этом поле указывается системный идентификатор sc-узла, который обозначает внешний язык с которым работает компонент;
6. *fomats* – в этом поле указывается список системных идентификаторов sc-узлов, которые обозначают форматы файлов (sc-ссылок) поддерживаемые компонентом. Если данное поле пусто, то компонент не может отображать (редактировать) sc-ссылки;
7. *sturct_support* – наличие этого поля со значением true, указывает на то, что данный компонент умеет отображать (редактировать) структуры напрямую из памяти;
8. *factory* - это функция, которая вызывается, когда ядру ПИ необходимо создать экземпляр данного компонента. В качестве параметра передается экземпляр объекта component sandbox, через который будет осуществляться взаимодействие между компонентом и ядром. Возвращает функция экземпляр созданного компонента.

В момент создания экземпляра компонента доступна следующая информация:

- *sandbox.container* – это строка, которая содержит id элемента на странице, куда должен быть встроен компонент;

- *sandbox.canEdit()* – функция, которая возвращает true, если требуется создать редактор, а не просмотрщик;
- *sandbox.is_struct* – поле, которое равно true, если необходимо просмтаривать (редактировать) структуры, а не sc-ссылки;
- *sandbox.addr* – это адрес структуры или sc-
- ссылки с которой работает компонент.
- В момент создания компонент должен подписаться на необходимые ему события (установить в соответствующие поля sandbox функции обработчики):
- *sandbox.eventDataAppend* - это событие, которое вызывается, когда необходимо отобразить содержимое sc-ссылки. В качестве аргумента функция обработчика принимает данные (содержимое sc-ссылки);
- *sandbox.getObjectsToTranslate* – данное событие вызывается ядром, когда происходит переключение между режимами идентификации. Обработчик данного события – это функция, которая возвращает список всех sc-адресов отображаемых (редактируемых) элементов;
- *sandbox.eventApplyTranslation* – это событие, которое происходит, когда необходимые идентификаторы для нового режима идентификации найдены. Обработчик данного события принимает в качестве параметра объект, полями которого являются адреса sc-элементов, а значениями – новые идентификаторы. Если же для какого-то элемента нет поля, то идентификатор для него не найден;
- *sandbox.eventStructUpdate* – это событие инициируется, когда изменяется отображаемая структура (появляются или удаляются выходящие дуги из sc-узла обозначающего эту структуру).

  Таким образом, если нам необходимо реализовать компонент, который отображает содержимое sc-ссылок, необходимо:

- описать формат данных в базе знаний;
- зарегистрировать компонент для отображения нового формата;
- реализовать функцию
- *sandbox.eventDataAppend*, которая на вход примет данные из sc-ссылки. Эти данные уже необходимо обрабатывать в компоненте и как-то визуализировать.