|  |
| --- |
| ИнтерТраст |
| Архитектура GUI  Business Universe |
|  |

|  |
| --- |
| Митавский Д. В. |

Содержание

[Архитектура GUI 2](#_Toc374034134)

[Компоненты GUI 3](#_Toc374034135)

[Плагины 5](#_Toc374034136)

[Business Universe – разметка 6](#_Toc374034137)

[Глобальная шина сообщений 7](#_Toc374034138)

[Плагин «Domain Object Surfer» 8](#_Toc374034139)

[Локальная шина сообщений 9](#_Toc374034140)

[Формы 10](#_Toc374034141)

[Конфигурация 11](#_Toc374034142)

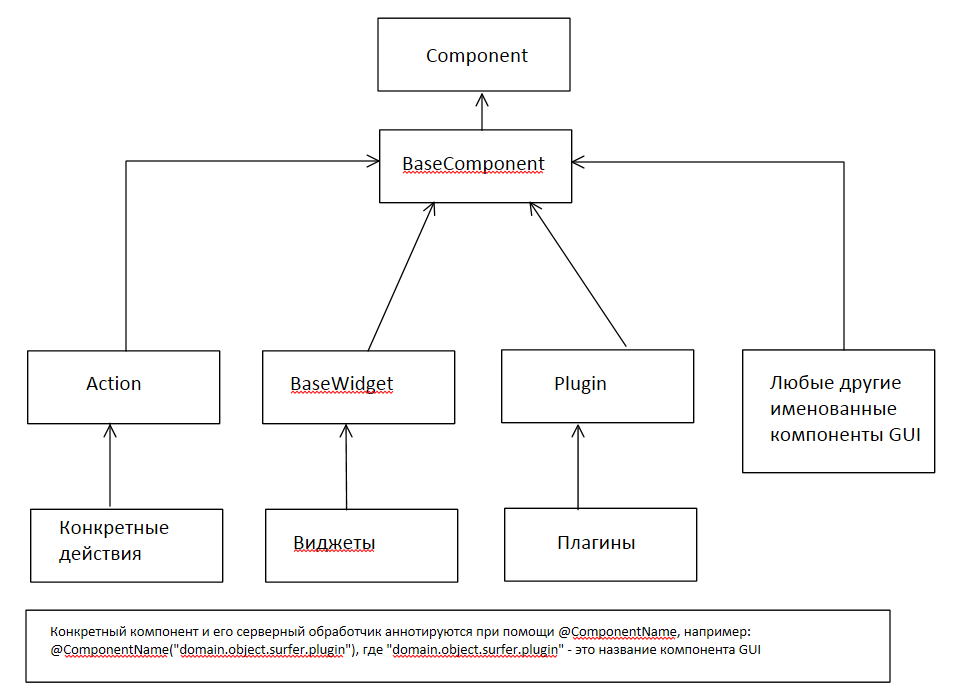
[Панель навигации 11](#_Toc374034143)

# Архитектура GUI

Документ описывает основные понятия, принципы построения и архитектуру веб-приложения «Business Universe». Приложение построено по принципам расширяемости и заменяемости (pluggability) и обладает очень широкими возможностями по конфигурируемости. Все основные свойства архитектуры останутся практически неизменными при реализации новой версии пользовательского интерфейса, таким образом, применение новых дизайнов к системе существенно упрощается.

# Компоненты GUI

Пользовательский интерфейс построен при помощи именованных компонентов. Это обеспечивает возможность замены одного компонента другим и возможность ссылаться на компоненты из конфигурации, не затрагивая при этом исходный код. Любой компонент должен реализовывать интерфейс Component. Для удобства разработчиков и возложения некоторых утилитарных функций на архитектуру существует абстрактный класс BaseComponent (реализующий Component), наследниками которого все компоненты системы и являются.



Если компонент использует взаимодействие с серверной частью системы, то для него должен быть создан соответствующий обработчик на стороне сервера, реализующий интерфейс ComponentHandler. Интерфейс маркерный и обязательных к реализации методов не содержит. Серверный обработчик и клиентский класс компонента должны быть аннотированы следующим образом:

@ComponentName("domain.object.surfer.plugin")

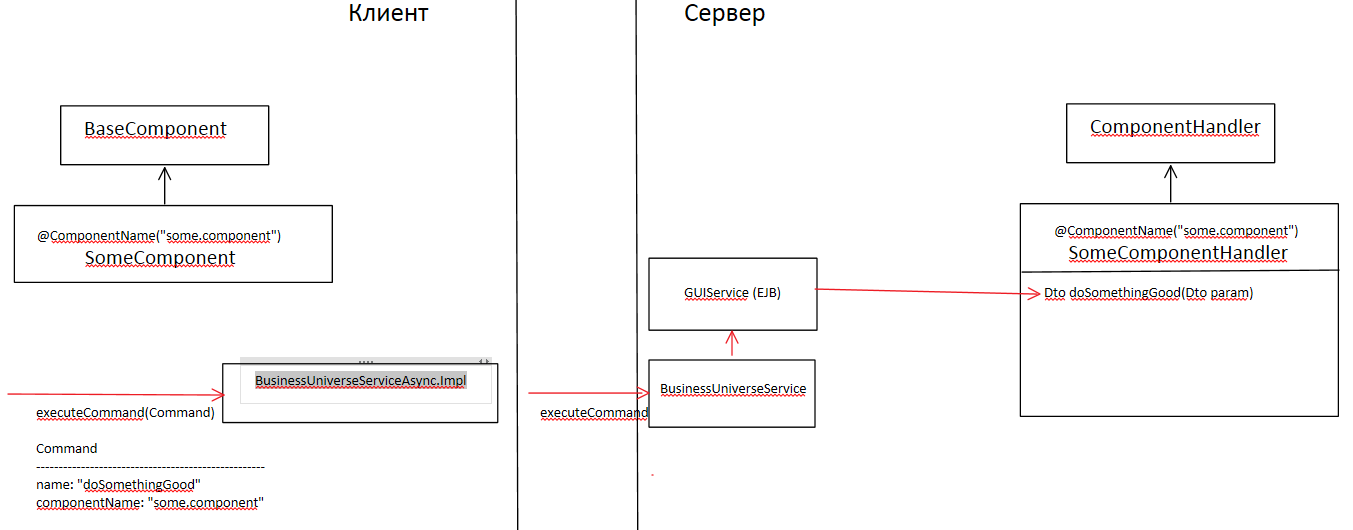
где в кавычках фигурирует название компонентов.

Схема взаимодействия серверной и клиентской частей компонента представлина на рисунке ниже.

Для выполнения команды достаточно получить экзмепляр класса BusinessUniverseServiceAsync.Impl и вызвать асинхронный метод executeCommand с параметром, описывающим команду. Описание команды содержит :

1. Название компонента
2. Название команды. Оно совпадает с названием метода на стороне сервера в реализации ComponentHandler
3. Параметр команды (любой Dto)

Серверные команды также возвращают Dto, который доступен в методе обратного вызова (callback) асинхронного запроса.



Получить экземпляр именованного компонента пользовательского интерфейса можно используя так называемый «Реестр компонентов»: ComponentRegistry.get().

Реализацию любого именованного компонента можно заменить. Для этого необходимо альтернативную реализацию компонента снабдить аннотацией, определяющей то же имя, как у компонента, который требуется заместить, и в файле gui.properties определить соответствие между названием компонента и классом, который будет при этом использоваться. Например, для того, чтобы использовать «окно логина для разработчиков», которое автоматически заполняется логином и паролей и авторизуется, достаточно поменять реализацию компонента «login.window»:

login.window.a = ru.intertrust.cm.core.gui.impl.authentication.tempdev.DevelopmentLoginWindow

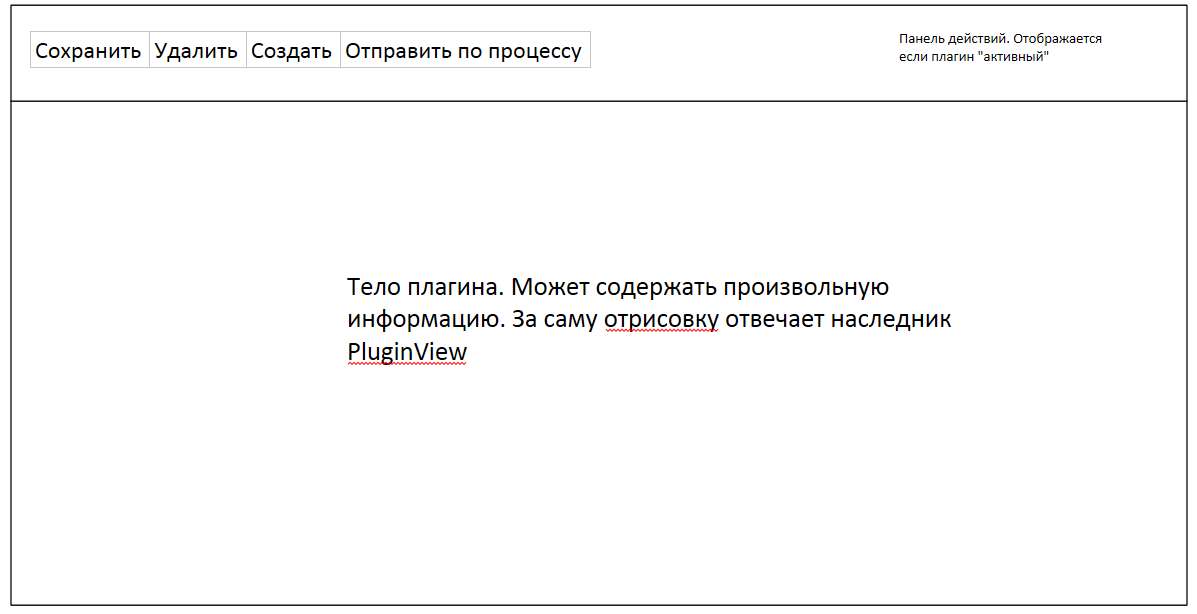
**package** ru.intertrust.cm.core.business.api;

# Плагины

Плагин – это некоторая, часто, самостоятельная часть пользовательского интерфейса, которую можно отобразить в тех частях экрана, в которых это предусмотрено (в этих частях должна быть расположена панель плагина, которую реализует класс PluginPanel).

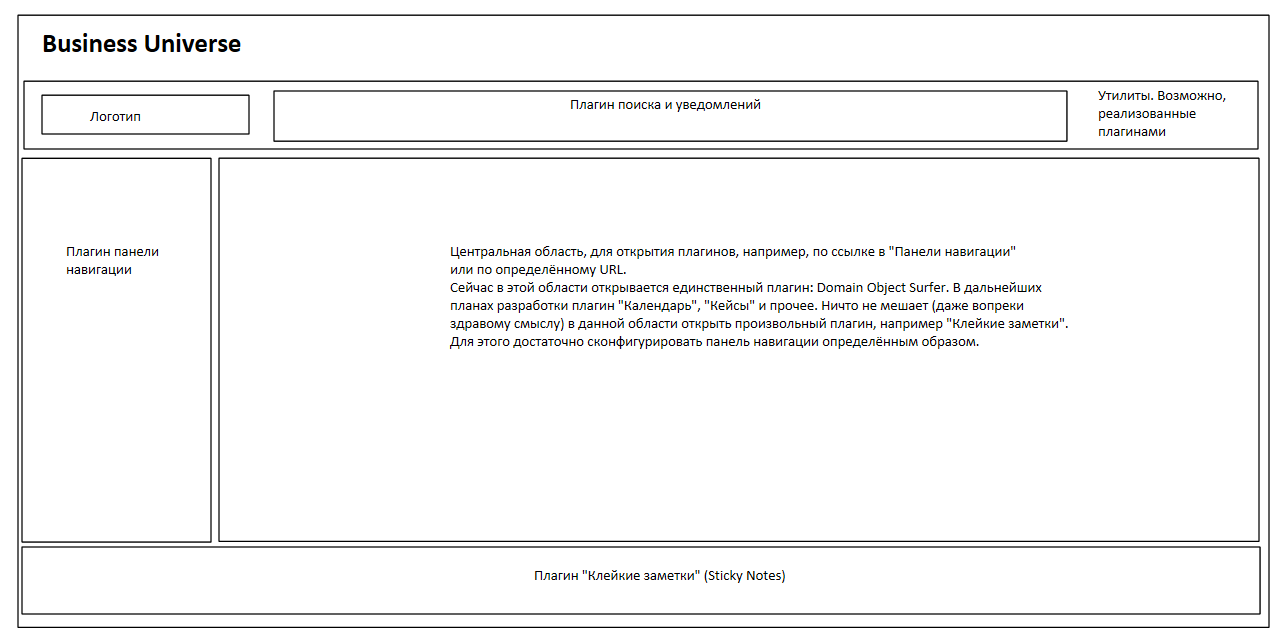
В качестве примера плагинов можно привести: Панель навигации, Форма, Коллекция, Календарь. Плагины являются именованными компонентами.

Плагин может быть как «активным» так и «неактивным». «Активный» плагин позволяет пользователю выполнять действия, расположенные в панели действий в верхней части плагина. Клиентский класс «активного» плагина должен реализовывать интерфейс IsActive, а серверный обработчик должен быть наследником ActivePluginHandler. Структура плагина показана на следующем рисунке.



Конфигурация действий, отображаемых «активным» плагином, формируется на стороне сервера при инициализации плагина в методе ActivePluginHandler.initialize(). Чтобы открыть плагин в GUI, необходимо создать его экземпляр, назначить конфигурацию и вызвать метод PluginPanel.open(Plugin plugin).

# Business Universe – разметка

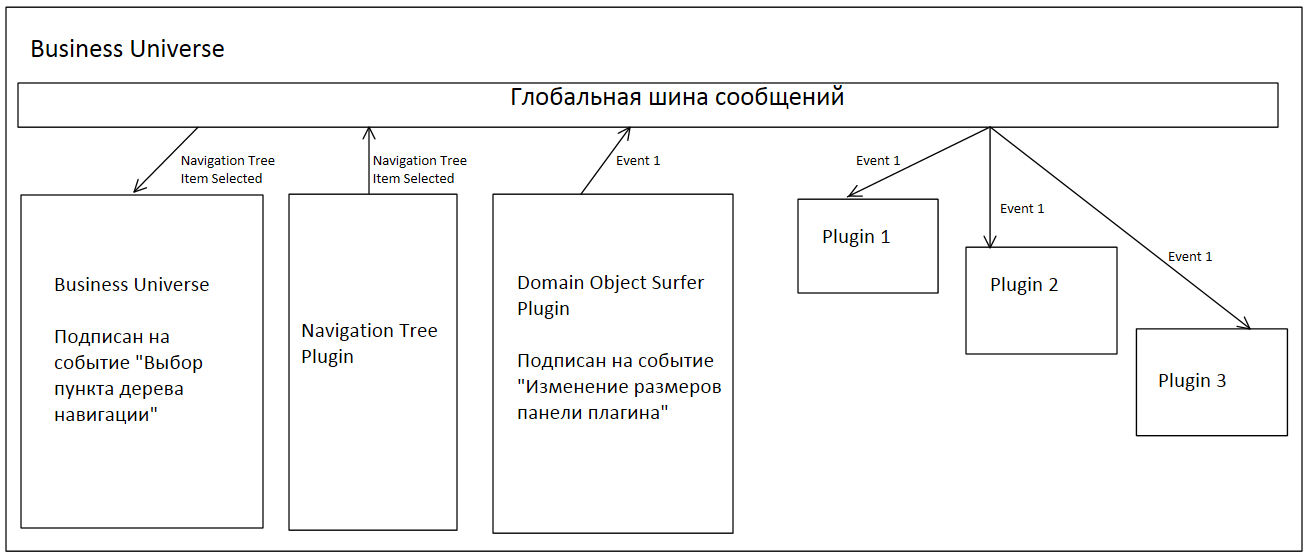


Business Universe состоит из следующих частей, показанных на рисунке.

1. Верхняя панель, которая в себя включает:
   1. Логотип
   2. Плагин поиска и уведомлений
   3. Утилитарные области (настройки, «Выход», информация о пользователе. На данный момент они реализованы классическим способом, впоследствии, вполне вероятно, что они будут трансформированы в плагины (например, для того, чтобы в другом варианте пользовательского интерфейса иметь возможность быть размещёнными в произвольной (настраиваемой) области экрана.
2. Плагин панели навигации
3. Центральная область плагина, открываемого по требованию (например, по ссылке в "Панели навигации" или по определённому URL). Сейчас в этой области открывается единственный плагин: Domain Object Surfer. В дальнейших планах разработки плагины "Календарь", "Кейсы" и другие. Ничто не мешает (даже вопреки здравому смыслу) в данной области открыть произвольный плагин, например "Клейкие заметки". Для этого достаточно сконфигурировать панель навигации определённым образом.
4. Нижняя область, предназначенная в данной реализации исключительно для плагина «Клейкие заметки».

## Глобальная шина сообщений

Для обмена сообщениями между компонентами в BusinessUniverse используется стандартный механизм GWT – Event Bus.



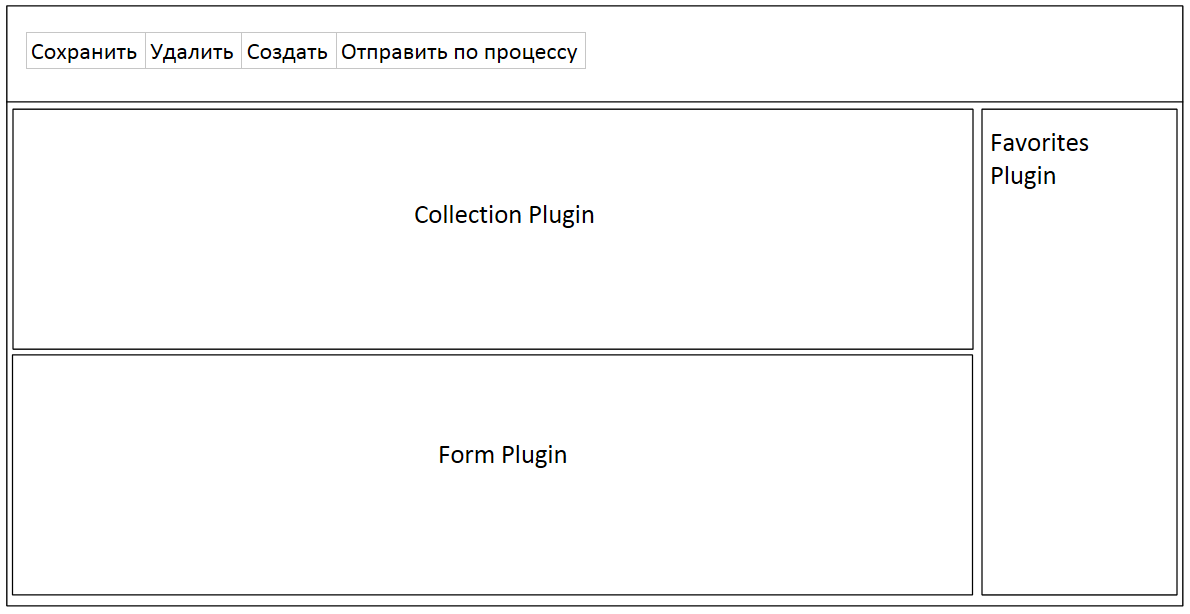
Глобальная шина сообщений доступна абсолютно всем компонентам. Получить её экземпляр можно следующим образом:

Application.getInstance().getEventBus();

В данный момент глобальная шина используется только плагинами, которые могут подписываться на интересующие сообщения. Для подписки на глобальные сообщения достаточно переопределить абстрактный метод Plugin.getEventTypesToHandle().

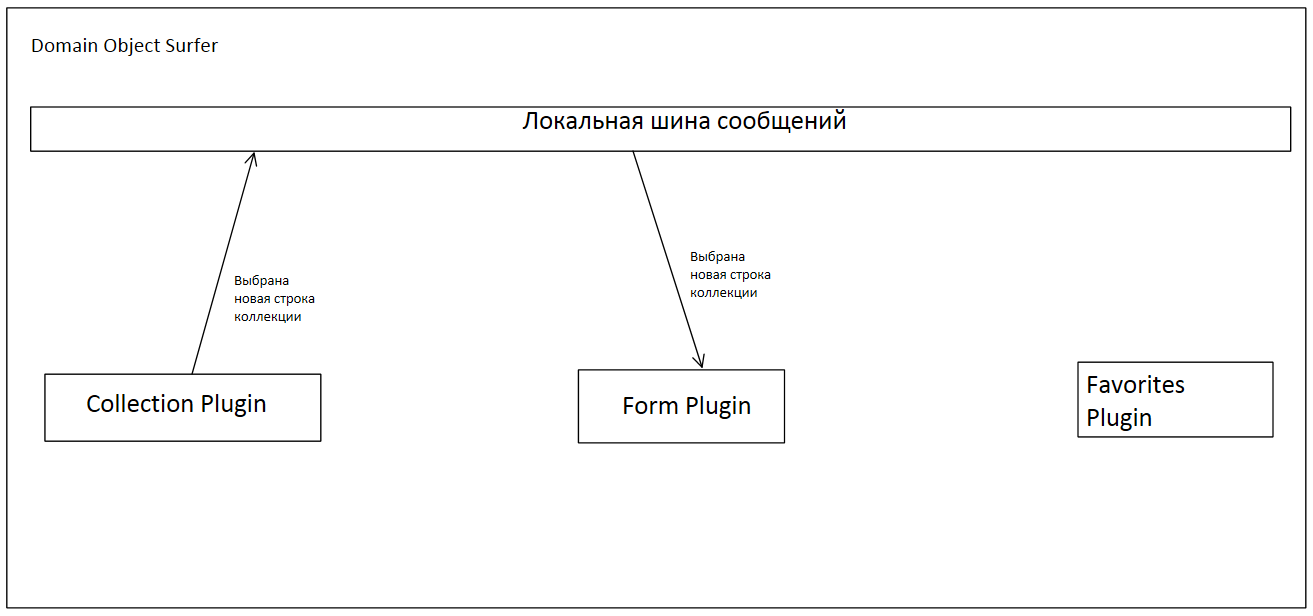
## Плагин «Domain Object Surfer»

Самым часто используемым плагином в системах администрирования является «Domain Object Surfer» - плагин, совмещающий плагины «Collection Viewer» и «Form», а также плагин «Favorites». В верхней части этого плагина показывается коллекция, в нижней – форма, соответствующая выбранной строке, в правой части плагина присутствует панель плагина «Избранное», которая может быть свёрнута или развёрнута.



## Локальная шина сообщений

Плагины и другие компоненты системы часто используют в своём локальном контексте шину сообщений (GWT Event Bus) вместо классических наблюдателей и слушателей. Например, Domain Object Surfer, выступая в качестве «контроллера» между тремя плагинами, использует локальную шину сообщений для организации связывания компонентов.



# Формы и виджеты

Одной из ключевых особенностей GUI является механизм «конфигурируемых форм». Данный механизм обеспечивает возможность определить современные, удобные в использовании формы и связать их с данными в хранилище декларативно – не написав ни строчки Java-кода.

Форма состоит из виджетов – элементов управления, при помощи которых пользователь вводит необходимые данные. Примерами виджетов являются: текстовое поле (Text Box), выпадающий список с авто-подсказками (Suggest Box), загрузчик вложений (Attachment Box) и т.д. Виджеты формы размещаются в определённой разметке, о которой будет рассказано далее.

# Конфигурация

Данный раздел описывает конфигурирование пользовательского интерфейса, в частности следующие его составные части: панель навигации, плагин коллекций, формы и действия.

## Панель навигации

Панель навигации настраивается конфигурационно. Пример конфигурации панели навигации:

<navigation name="panel" is-default="true">

<link name="Geography" display-text="География"

image="images/inbox.png" child-to-open="Countries">

<child-links>

<link name="Countries" display-text="Страны">

<plugin>

<domain-object-surfer

domain-object-type-to-create="country">

<collection-viewer>

<collection-ref name="Countries"/>

<sort-criterion field="name" order="asc"/>

</collection-viewer>

</domain-object-surfer>

</plugin>

</link>

<link name="Cities" display-text="Города">

<plugin>

<domain-object-surfer

domain-object-type-to-create="city">

<collection-viewer>

<collection-ref name="Cities"/>

<sort-criterion field="name" order="asc"/>

</collection-viewer>

</domain-object-surfer>

</plugin>

<decorations>

<collection-counter/>

</decorations>

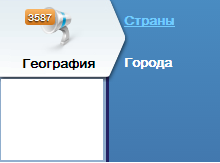
</link>

</child-links>

</link>

</navigation>

Такая конфигурация сформирует следующую панель навигации:



Панель навигации можно определить для каждой роли и для конкретных пользователей (прим.: *ещё не реализовано*).