|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [=](https://mail.yandex.ru/message_part/?hid=1.1.2&ids=2530000000187447428&name=) |  | |  | | --- | |  | | РЕАЛИЗАЦИЯ ПОЛЬЗОВАТЕЛЬСКОГО ДЕЙСТВИЯ | |  | | | | | |
| **Код:** | | |  | **Версия:** |  |
| **Дата создания:** | | | 02.09.2014 | **Дата изменения:** |  |
| **Автор:** | | | Абдулхаиров Равиль | **Редактор:** |  |
| **Статус:** | | | В работе | | |
| **Назначение ТК:** | | | Реализация функционала пользовательского действия в приложении с использованием платформы | | |
| **Входящие записи:** | | | Спецификация требования на разработку ПО | | |
| **Исходящие записи:** | | | Функционирующий программный код реализующий пользовательское действие, описанное в спецификации | | |

# [TM120.01] Определение целевых модулей

|  |  |
| --- | --- |
|  | На основании документа <ARC_Пользовательские_действия.pdf> определите модули проекта в которых будет реализован функционал пользовательского действия. В реализации будут участвовать четыре модуля: модуль клиентских классов, модуль серверных классов, модуль модели (общедоступный для первых двух) и целевой модуль, из которого будет производится вызов действия. |

# [TM120.02] объявление действия в файле конфигурации целевого модуля

|  |  |
| --- | --- |
|  | Добавить объявление вызова пользовательского действия в файле конфигурации целевого модуля. Формат и правила работы с файлами конфигурации модуля описаны в карте (TM0XXX). Добавим следующий код в файл конфигурации в необходимом нам месте.  **<act:action componentName="имя действия.action" weight="10" merged="true" text="Текст для отображения"/>** |
|  | Вместо «имя действия» впишите имя компонента, которое будет использовано в дальнейшем при аннотировании классов. Это уникальное имя под которым компонент будет зарегистрирован в системе. Замените «**Текст для отображения»** на значение, которое должно отображаться на кнопке. |

# [TM120.03] реализация класса вызова события в клиентском модуле

|  |  |
| --- | --- |
|  | Создайте в исходных кодах клиентского модуля новый Java класс с кодом, указанным ниже.  @ComponentName("**имя действия**.action")  public class **ИмяДействия** extends SimpleServerAction {  @Override  public Component createNew() {  return new **ИмяДействия** ();  }  } |
|  | Вместо «**имя действия**» впишите имя компонента, присвоенное вами на Шаге 2.  Вместо «**ИмяДействия**» задайте имя класса. |

# [TM120.04] реализация класса обработчика события в серверном модуле

|  |  |
| --- | --- |
|  | Создайте в исходных кодах серверного модуля новый Java класс с кодом указанным ниже.  @ComponentName("**имя действия**.action")  public class **ИмяДействия**Handler extends ActionHandler<ActionContext, **ИмяДействия**Data> {  @Override  public **ИмяДействия**ActionData executeAction(ActionContext context) {  **ИмяДействия**ActionData actionData = new **ИмяДействия**ActionData();  /\* Какие нибудь полезные действия \*/  return actionData;  }  @Override  public ActionContext getActionContext() {  return new ActionContext();  }  } |
|  | Вместо «**имя действия**» впишите имя компонента, присвоенное вами на Шаге 2.  Вместо «**ИмяДействия**» задайте имя класса, присвоенное на Шаге 3. |

# [TM120.05] реализация объекта передачи данных в модуле модели

|  |  |
| --- | --- |
|  | Создайте в исходных кодах модуля модели новый Java класс с кодом, указанным ниже.  public class **ИмяДействия**ActionData extends ActionData {}  **Данный класс должен быть доступен для использования и в клиентском и в серверном модуле.** |
|  | Вместо «**ИмяДействия**» задайте имя класса, присвоенное на Шаге 3. |

# [TM120.06] сборка и тестирование

|  |  |
| --- | --- |
|  | Проведите сборку и деплоймент приложения на сервер. Войдите в приложение и вызовите действие с той страницы на которой оно было сконфигурировано во втором шаге. |

|  |
| --- |
| Пример создания пользовательского действия вы можете найти в файле [EXM\_Пример\_создания\_пользовательского\_действия.pdf](EXM_Пример_пользовательского_действия.pdf) |