РЕАЛИЗАЦИЯ ПОЛЬЗОВАТЕЛЬСКОГО ДЕЙСТВИЯ

Код:	[TM120]	**Версия:**	0.01
Дата создания:	02.09.2014	**Дата изменения:**	02.09.2014
Автор:	Абдулхаиров Равиль	**Редактор:**	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		

ШАГ 1. [TM120.01] ОПРЕДЕЛЕНИЕ ЦЕЛЕВЫХ МОДУЛЕЙ

☒ На основании документа ARC_Custom_Actions.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.

ШАГ 2. [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.

ШАГ 3. [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.

ШАГ 4. [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 ИмяДействияHandler extends ActionHandler<ActionContext, ИмяДействияData> {


   @Override
   public ИмяДействияActionData executeAction(ActionContext context) {
      ИмяДействияActionData actionData = new ИмяДействияActionData();
      /* 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.

ШАГ 5. [TM120.05] РЕАЛИЗАЦИЯ ОБЪЕКТА ПЕРЕДАЧИ ДАННЫХ В МОДУЛЕ МОДЕЛИ

☐ Создайте в исходных кодах модуля модели новый Java класс с кодом, указанным ниже.

```
public class ИмяДействияActionData 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.

ШАГ 6. [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_Example_Custom_Actions.pdf