ПРИМЕР СОЗДАНИЯ ПОЛЬЗОВАТЕЛЬСКОГО ДЕЙСТВИЯ

В данном примере мы реализуем пользовательское действие с помощью которого произведем какие-то действия с доменным обьектом выбраным в списке коллекции.

1. Объявляем действие в файле конфигурации целевого модуля

```
<act:action componentName="example.action" weight="10" merged="true" text="Тестовый вызов"
imageClass="actionBar-delete" dirtySensitivity="true" order="300" >
    <act:before-execution>
        <act:confirmation-message text="Выполнить действие?"/>
    </act:before-execution>
    <act:after-execution>
        <act:on-success-message text="Действие успешно выполнено."/>
    </act:after-execution>
</act:action>
```

2. Реализация класса вызова действия в клиентском модуле

```
import com.google.gwt.user.client.Window;
import ru.intertrust.cm.core.gui.api.client.Component;
import ru.intertrust.cm.core.gui.impl.client.action.Action;
import ru.intertrust.cm.core.gui.impl.client.action.SimpleServerAction;
import ru.intertrust.cm.core.gui.model.ComponentName;

@ComponentName("example.action")
public class DeactivateServiceAction extends SimpleServerAction {

    @Override
    public Component createNew() {
        return new DeactivateServiceAction();
    }

    @Override
    protected ActionContext appendCurrentContext(ActionContext context) {
        final IsDomainObjectEditor editor = (IsDomainObjectEditor) getPlugin();
        FormState formState = editor.getFormState();
        context.setRootObjectId(formState.getObjects().getRootNode().getDomainObject().getId());
        return context;
    }

    @Override
    protected void onSuccess(ActionData result) {
        DeactivateServiceActionData changeServiceActionData = (DeactivateServiceActionData) result;
        IdentifiableObject rootDomainObject = changeServiceActionData.getRootDomainObject();
        if (rootDomainObject != null) {
            plugin.getLocalEventBus().fireEvent(new UpdateCollectionEvent(rootDomainObject));
        }
    }
}
```

Методы appendCurrentContext и onSuccess не являются обязательными к реализации когда мы не передаем обьект серверному обработчику и не нужно выполнять дополнительных действий на ГУИ при успешном выполнении вызова. В данном случае, мы передаем доменный объект с которым нужно произвести какие-то действия на стороне сервера и обновляем коллекцию после успешного вызова. Не забудьте указать правильное имя пакета в котором расположен класс.

3. Реализация класса вызова действия в серверном модуле

```
import ru.intertrust.cm.core.gui.api.server.action.ActionHandler;
import ru.intertrust.cm.core.gui.model.ComponentName;
import ru.intertrust.cm.core.gui.model.action.ActionContext;
import ru.intertrust.sed.gui.model.action.DeactivateServiceActionData;
```

```
@ComponentName("example.action")
public class DeactivateServiceActionHandler extends ActionHandler<ActionContext,DeactivateServiceActionData> {

    private static Logger logger = LoggerFactory.getLogger(DeactivateServiceActionHandler.class);

    @Autowired
    private CrudService crudService;

    @Override
    public DeactivateServiceActionData executeAction(ActionContext context) {
        Id rootObjectId = context.getRootObjectId();
        DeactivateServiceActionData actionData = new DeactivateServiceActionData();
        if (rootObjectId != null) {
            DomainObject rootDomainObject = crudService.find(rootObjectId);
            logger.info("DEACTIVATION");
            actionData.setRootDomainObject(rootDomainObject);
        }
        return actionData;
    }

    @Override
    public ActionContext getActionContext() {
        return new ActionContext();
    }
}
```

В обработчике мы получаем доменный объект, переданный нам в контексте, и можем произвести какие-либо действия с ним.

4. Реализация класса объекта для передачи данных в модуле модели

```
import ru.intertrust.cm.core.business.api.dto.DomainObject;
import ru.intertrust.cm.core.gui.model.action.ActionData;

public class DeactivateServiceActionData extends ActionData {
    private DomainObject rootDomainObject;

    public DomainObject getRootDomainObject() {
        return rootDomainObject;
    }

    public void setRootDomainObject(DomainObject rootDomainObject) {
        this.rootDomainObject = rootDomainObject;
    }
}
```

Проведите сборку и деплоймент приложения на сервер. Войдите в приложение и вызовите действие с той страницы на которой оно было сконфигурировано. При вызове действия вы должны получить окно подтверждения вызова, после того как действие выполнено, появиться окно об успешном выполнении. В логе сервера появиться запись "DEACTIVATION" означающая что обработчик события на серверной стороне успешно отработал.