Table of Contents

A server action is a method/function that is defined in a life cycle object at the server. Execution is initiated either by the client or directly at the server.

In short, it is any method at the business level of the application.

Server actions are used for functions that are not/should not be executed at the client, including business logic such as mail transmission, interface requests, calculations, etc.

Example

We want to develop an application that manages purchase orders. These orders are provided by SAP via a web services interface and shown in a separate form in our application. In addition, cancellation of individual orders via SAP web services should be possible.

Cancellations are initiated by the client via a button. The order number, as well as a PIN/confirmation code, are required for execution.

We'll show snippets of client- and server-side implementation.

```
...
...

/** the communication connection to the server. */
private AbstractConnection connection;

/** the orders table. */
private RemoteDataBook rdbOrder = new RemoteDataBook();

/**
 * Initializes the UI.
 */
private void init()
{
    connection = ((MasterConnection)application.getConnection()).
                 createSubConnection("apps.firstapp.frames.Orders");
    connection.open();

    ...
    ...

    rdbOrder.setDataSource(dataSource);
    rdbOrder.setName("orders");
    rdbOrder.open();

    UIButton butStorno = new UIButton("Cancel");
    butCancel.eventAction().addListener(this, "doCancel");
}

/**
 * Performs the cancellation of an order.
 *
```

```
 * @throws Throwable if the cancellation is not possible or the remote
system has errors
 */
public void doCancel() throws Throwable
{
    connection.callAction("cancel", rdbOrder.getValue("ID"),
editPin.getText());
}
```

The action call is done with the following line:

```
connection.callAction("cancel", rdbOrder.getValue("ID"), editPin.getText());
```

The action “cancel” is called via the server connection connection. The parameter ID and PIN are first entered by the user and passed on to the call.

Server

Orders.java

```
package apps.firstapp.frames;

...
...

/**
 * The LCO for the Orders Workscreen.
 * <p/>
 * @author René Jahn
 */
Public Class Orders Extends Session
{
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   // User-defined methods
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   /**
    * Returns the orders storage.
    *
    * @return the orders storage
    * @throws Exception if the initialization throws an error
    */
   public DBStorage getOrders() throws Exception
   {
            ...
   }

   /**
    * Performs the cancellation of an order via SAP webservice.
```

```
     *
     * @param pOrderId the order ID
     * @param pPin the storno PIN
     */
    public void cancel(BigDecimal pOrderId, String pPin)
    {
       //call SAP webservice with ID and PIN
    }

}   // Orders
```

If exceptions occur during execution, they can easily be passed on using the throws clause, as the application independently handles errors.