Table of Contents

Eine Server Action ist eine Methode/Funktion die in einem Lifecycle Objekt am Server definiert wird. Die Ausführung wird entweder vom Client oder direkt am Server initiiert.

Kurz gesagt handelt es sich um eine beliebige Methode in der Business Schicht der Applikation.

Wir verwenden Server Actions für Funktionen die nicht am Client durchgeführt werden sollen/dürfen, sprich unsere Business Logik wie z.B. Mailversand, Schnittstellenabfragen, Berechnungen, usw.

Anwendungsbeispiel

Wir verwenden eine Applikation die Bestellungen verwaltet. Diese Bestellungen werden von SAP über eine Webservice Schnittstelle bereitgestellt und von unserer Applikation in einer eigenen Maske angezeigt. Die Stornierung, via SAP Webservice, einzelner Bestellungen soll zusätzlich implementiert werden.

Ein Storno wird vom Client über einen Button angestoßen. Für die Durchfürung ist die Bestellnr sowie ein PIN/Bestätigungs Code notwendig.

Wir zeigen die Ausschnitte aus Client- und Serverseitiger Implementierung.

Client

```
...
...

/** 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("Storno");
    butStorno.eventAction().addListener(this, "doStorno");
}
```

```
/**
 * Performs the storno of an order.
 *
 * @throws Throwable if the storno is not possible or the remote system has
errors
 */
public void doStorno() throws Throwable
{
    connection.callAction("storno", rdbOrder.getValue("ID"),
editPin.getText());
}
```

Der Action Aufruf passiert mit folgender Zeile

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

Über die Server Verbindung connection wird die Action storno aufgerufen. Die Parameter ID und PIN werden zuvor vom User erfasst und an den Aufruf übergeben.

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

}   // Orders
```

Sollten Exceptions bei der Ausführung auftreten, können diese ohne weiteres mit der throws Klausel weitergegeben werden, da sich die Applikation selbständig um die Fehlerbehandlung kümmert.