Table of Contents

Any function at the client´s business logic can be addressed using Server Actions. These are usually actions that do not take up a lot of time, such as, input validation.

For this reason, communication between client and server is synchronized, i.e., the client makes a request to the server and waits for a response. During the wait time, the mouse cursor in the application is displayed as an hourglass, and the user can perform no further actions.

When, for example, a report needs to be created that requires time-intensive calculations and data analysis, then long wait times are undesirable.

For time-consuming tasks, asynchronous communication is possible without additional effort. This does not mean, however, that the entire communication has to work asynchronously. It is up to the developer to decide which tasks should be executed asynchronously.

Results are transmitted automatically; the developer does, therefore, not need to worry about particulars.

Example

Our application allows the creation of reports that are based on millions of datasets. However, the datasets have to be evaluated and compressed. Processing time for an individual report is approximately three minutes.

The user initiates the report by clicking a button (see Client Actions) and is notified when the report has been created. In the meantime, the user can continue to work as usual.

As a first step, a server action is required:

```java
/**
 * Creates a statistic report.
 *
 * @param pId the master data ID
 * @param pName the report name entered from the user
 */
public String startReport(BigDecimal pId, String pName)
{
    ReportGenerator rgen = new ReportGenerator();

    rgen.createReport(pId);

    return pName;
}
```

The server action is now called asynchronously. This is accomplished through the use of a CallBack listener:

```java
/**
 * Starts the creation of the statistic report.
 *
 * @throws Throwable if it's not possible to start report creation
 */
```

```
public void doStartReport() throws Throwable
{
    getConnection().callAction(new ICallBackListener()
                               {
                                    public void callBack(CallBackEvent
pEvent)
                                    {
                                       try
                                       {
                                          showInformation(Showcase.this,
(String)pEvent.getObject());
                                       }
                                       catch (Exception e)
                                       {
                                          ExceptionHandler.raise(e);
                                       }
                                    }
                               },
                               "startReport",
                               bdId,
                               "Report: " + bdId);
}
```

Once the report is created, the client is notified. In this example, an information dialog is displayed.