Table of Contents

Als Client Action definieren wir eine Methode/Funktion die am Client zu einem vordefinierten Zeitpunkt, automatisch aufgerufen wird. Ein vordefinierter Zeitpunkt wäre z.B. ein Button klick, der Zeilenwechsel in einer Tabelle, Einfügen in einer Tabelle, ...

Das Action Konzept setzt auf das Listener Konzept von Java auf und vereinfacht die Listener Verwaltung und vor allem die Verwendung.

Die Methoden/Funktionen mit denen Actions definiert werden, beginnen in JVx mit event z.B. eventAction. Das erleichtert die Suche nach möglichen Actions in der javadoc und mit der IDE.

Verwendung

Die Definition einer Action kann auf zwei unterschiedliche Arten durchgeführt werden. Einerseits durch die Kombination von Objekt und Methodenbezeichnung:

```
/**
 * Initializes the UI components.
 */
private void init()
{
    UIButton butCancel = new UIButton();
    butCancel.setText("Cancel");

    butCancel.eventAction().addListener(this, "doCancel");
}

/**
 * Cancel action.
 */
public void doCancel()
{
    dispose();
}
```

Die Zeile

```
butCancel.eventAction().addListener(this, "doCancel");
```

besagt, das die Methode doCancel im Objekt this beim Klick auf den Cancel Button ausgeführt wird.

Die Methode

```
public void doCancel()
```

wird in unserem Beispiel ohne Parameter und ohne throws Klausel definiert, da wir keine speziellen Anforderungen haben. Es wäre allerdings auch möglich die Methode so zu definieren:

```
public void doCancel(UIActionEvent pEvent) throws Exception
```

Der Parameter wird von JVx nur übergeben wenn dieser auch in der Parameterbeschreibung enthalten

ist. Wenn die throws Klausel angegeben wurde, werden die Exceptions von JVx abgefangen und an den Fehlerdialog weitergeleitet. Man muss sich also um die Fehler- Behandlung und Darstellung nicht mehr gesondert kümmern. Wenn jedoch eine spezielle Fehlerbehandlung nötig ist, muss ein try/catch Block implementert und die throws Klausel entfernt werden.

Da unser Action Konzept auf dem Java Listener Konzept aufsetzt, ist es auch ohne weiteres möglich mit Listenern zu arbeiten:

```
butCancel.eventAction().addListener(new IActionListener()
{
    public void action(UIActionEvent pActionEvent)
    {
        dispose();
    }
});
```

oder

```
public class MyFrame implements IActionListener
{
    public void action(UIActionEvent pActionEvent)
}
```