Manipulationsgeschützte Formulare mit Secureform

Vom System generierte Werte in Formularen sind leicht veränderbar. Falls in den Aktionen keine Überprüfungen dieser Werte vorhanden sind, entstehen schnell Sicherheitslecks. Parameter können dann beliebig manipuliert werden und die Aktionsbausteine verrichten unbedarft ihren Dienst mit gefälschten Daten. So können z.B. IDs ausgetauscht werden und somit fremde Datensätze überschrieben werden.

Damit nicht in jeder Aktion selbst eine Überprüfung in JSP oder Groovy geschrieben werden muss, gibt es das Tag `bx:secureform` und die Aktion Sicheres Formular überprüfen (SecureformAction). Im Zusammenspiel stellen diese beiden Komponenten sicher, dass ausgewählte Parameter nicht manipuliert werden können.

Dieser Artikel bietet eine detaillierte Erklärung der Funktion und Vorgehensweise. Es sollten zusätzlich die Seiten des Tags und der Aktion durchgelesen werden.

Parameter sammeln

Im Formular

Im ersten Schritt werden die zu sichernden Formularelemente gesammelt. Das sind jene Parameter, die später an das Action geschickt werden. Es müssen nicht alle Parameter abgesichert werden, nur jene die im Action als zu sichern angegeben wurden.

Zunächst wird um das gesamte Formular ein `<bx:secureform>` Tag gesetzt. Dies definiert einen Bereich, in dem Parameter gesammelt werden. Dadurch können auf einer Seite mehrere solche Bereiche definiert werden, falls es mehrere Formulare gibt.

Bereich definieren

```
<bx:secureform>
  <form action="...">
  </form>
</bx:secureform>
```

Danach werden die zu schützenden Parameterwerte gesammelt. In den meisten Fällen wird der `value` eines `<input type="hidden">` einfach mit `<bx:secureform.field>` umschlossen.

Parameter sammeln

```
<bx:secureform>
  <form action="...">
    <input type="hidden" name="Besitzer" value="<bx:secureform.field
name="Besitzer"><bx:userdata.id /></bx:secureform.field>">
  </form>
</bx:secureform>
```

Falls ein Parameter im Formular geschützt wird, muss er auf jeden Fall auch in den Aktionseinstellungen angegeben werden (s.u.) - da die Prüfung im Baustein sonst fehlschlägt.

Parameter, die nicht mitgesendet werden, falls sie keinen Wert haben, dürfen nicht mit zur Hashbildung herangezogen werden. Das sind z.B. Datensatz-ID Felder, die nur bei fehlerhaften Validierungen gefüllt werden, damit beim erneuten Absenden des Formulars kein neuer Datensatz erzeugt wird. Meist kommt dabei `<bx:containerfilter dummy>` oder `<bx:record dummy>` zum Einsatz. Der Parameter `hidden-when-empty` bewirkt, dass ein Feld mit leerem Wert von `<bx:secureform.field>` ignoriert wird.

hidden-when-empty

```
 <bx:secureform>
  <form action="...">
    <bx:if><input type="hidden" name="kid" value="<bx:secureform.field name="kid" hidden-when-
empty><bx:recorddata.id /></bx:secureform.field>"></bx:if>
    <input type="hidden" name="Besitzer" value="<bx:secureform.field
name="Besitzer"><bx:userdata.id /></bx:secureform.field>">
  </form>
</bx:secureform>
```

Zum Schluss wird der Hash generiert und in das Formular geschrieben. Die einfachste Methode ist vom System direkt ein `<input type="hidden">` generieren zu lassen.

Hash ausgeben

```
<bx:secureform>
  <form action="...">
    <bx:if><input type="hidden" name="kid" value="<bx:secureform.field name="kid" hidden-when-
empty><bx:recorddata.id /></bx:secureform.field>"></bx:if>
    <input type="hidden" name="Besitzer" value="<bx:secureform.field
name="Besitzer"><bx:userdata.id /></bx:secureform.field>">
    <bx:secureform.hash />
```

```
  </form>
</bx:secureform>
```

Dies erzeugt ein verstecktes Formularfeld mit passendem Name und Wert. Alternativ kann der Name oder Wert auch einzeln ausgegeben werden, siehe dazu die Tag Doku.

Im Link

Das Tag kann auch benutzt werden, um einen Link / Button abzusichern:

```
<bx:secureform>
  <a href="delete.act?kid=<bx:secureform.field name="kid"><bx:recorddata.id
/></bx:secureform.field>&<bx:secureform.hash name />=<bx:secureform.hash value />">löschen</a>
</bx:secureform>
```

Hier werden die Parameter `name` und `value` von `bx:securefom.hash` benutzt, denn beide Werte sind dynamisch.

Falls URL-Parameter mit z.B. `bx:if` umschlossen sind, muss auch wieder `hidden-when-empty` benutzt werden (s.o.).

Erlaubte Werte festlegen

Falls ein Parameter mehrere, bestimmte Werte annehmen darf, ist dies mit `bx:secureform.values` abbildbar. In diesem Fall wird nicht das Feld mit dem Wert gehasht (da es vom Nutzer ja geändert werden kann), sondern ein zweites Feld eingefügt. Es enthält die erlaubten Werte und wird anstelle des veränderlichen Feldes mitgehasht.

So kann sich der Inhalt des Feldes (z.B. ein `<select>`) ändern, es wird aber trotzdem eine abgesicherte Information über die erlaubten Werte mitgeschickt.

Der Parametername wird normal in die Aktion eingetragen, diese unterscheidet aber, ob ein Info-Parameter mit den erlaubten Werten übergeben wurde oder nur der Normale und zieht dann den entsprechenden Parameter zur Hashbildung heran.

Im Formular

mehrere erlaubte Werte (Formular)

```
<select name="Anrede">
  <option value="<bx:secureform.values name="Anrede">m</bx:secureform.values>">Herr</option>
```

```
  <option value="<bx:secureform.values name="Anrede">w</bx:secureform.values>">Frau</option>
</select>
<bx:secureform.values name="Anrede" />
```

Das Tag in der geschlossenen Variante erzeugt automatisch ein `<input type="hidden">` mit passenden `name` und `value` Attributen.

Im Link

mehrere erlaubte Werte (Link)

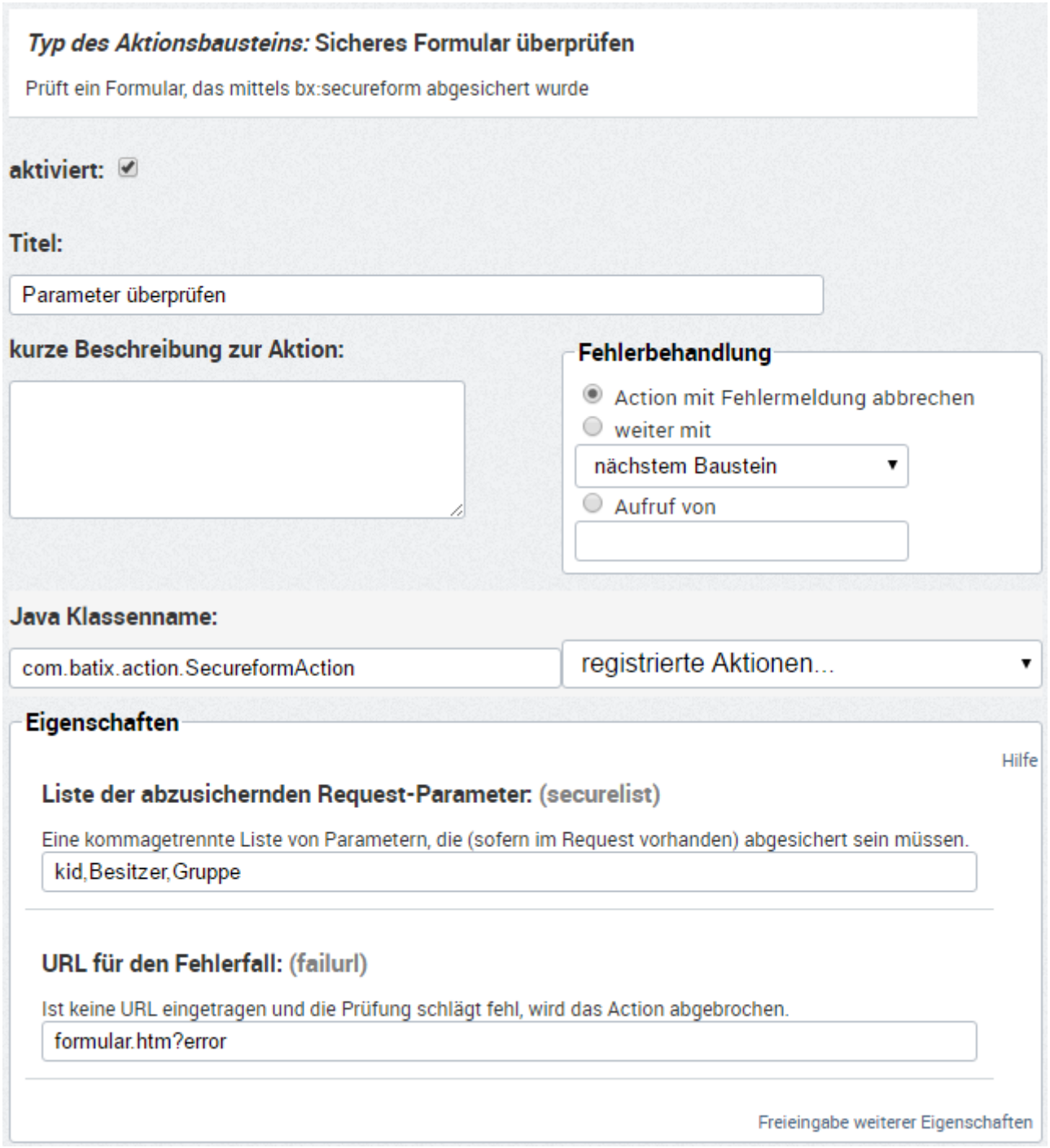
```
var url = "save.act?";
// ...andere (ggf. abgesicherte) Parameter, wie z.B. ID...
url += "&Anrede=" + encodeURIComponent($("[name=Anrede]").val());
url += "&<bx:secureform.values name="Anrede" param encode="javascript"
/>=<bx:secureform.values name="Anrede" values encode="javascript" />"
```

Aktion absichern

Um eine Aktion abzusichern, wird ein neuer Sicheres Formular überprüfen (SecureformAction) Baustein eingefügt. **Dieser muss an erster Stelle stehen**, damit bei Manipulation die Aktion direkt abgebrochen wird und nachgelagerte Bausteine nicht ausgeführt werden.

In der *Liste der abzusichernden Request-Parameter* werden all die Parameter aufgezählt, die (falls übergeben) abgesichert sein müssen - d.h. diese Parameter werden zur Hashbildung im Baustein herangezogen. Das bedingt natürlich, dass diese Parameter auch im Formular eingesammelt wurden. Umgedreht gilt dasselbe: Parameter, die im Formular gesammelt wurden, müssen auch im Aktionsbaustein gelistet sein.

Im unteren Feld kann eine URL für den Fehlerfall spezifiziert werden. Hierhin wird weitergeleitet, sobald die Prüfung fehlschlägt. Wurde keine URL angegeben und die Prüfung schlägt fehl, so wird die gesamte Aktion mit einer Fehlermeldung abgebrochen.



Steht ein Parameter in der Liste, wurde aber nicht übergeben, so wird er vom Baustein nicht zu Hashbildung herangezogen.

> Die Liste kann auch für Parameter verwendet werden, die der Aktion niemals übergeben werden dürfen (falls z.B. andere Bausteine schlecht darauf reagieren).
>
> Da diese nicht im Formular gesammelt werden und somit nicht in den Hash einfließen, wird vom Formular ein anderer Hash erzeugt, als von der Aktion (da diese den Parameter mit in die Berechnung integriert, da er in der Liste steht).

Seite absichern

Analog zum Absichern einer Aktion existiert die Möglichkeit ein Template abzusichern. Dies kann z.B. verwendet werden, um bestimmte Parameter einer Listen- oder Detail-Seite abzusichern ("Kontakttyp darf nur Kunde sein" oder "Gruppe muss 123ABC oder 456DEF sein").

Hierfür gibt es das Tag `<bx:secureform.check>`, welches ganz oben in der Seite eingebaut wird und die Abarbeitung der Seite abbricht, sobald etwas manipuliert wurde. Optional kann auch eine URL angegeben werden, auf die weitergeleitet wird.

einfache Prüfung in Template

```
<bx:secureform.check securelist="Kategorie" />
<!DOCTYPE html>
<html>
...
```

Funktionsweise

Das Verfahren beruht auf kryptografischen Hashes. Dies sind Einwegfunktionen, die einen beliebigen Input in einen Output bestimmter Länge transformieren. Diese Berechnung ist nicht umkehrbar und somit ist der Input nicht rekonstruierbar.

Es fließen verschiedene Sachen in die Berechnung des Hashes ein. Allein die Parameternamen und -werte genügen nicht, da sonst der Hash von einem Angreifer selbst gebildet werden könnte. Der Algorithmus ist im Allgemeinen bekannt oder kann rekonstruiert werden.

Daher werden zusätzlich serverseitig generierte bzw. gespeicherte Werte für die Berechnung des Hashes benutzt. Diese Werte werden dem Client nicht mitgeteilt und er kann diese auch nicht auslesen. Nur das Tag und das Action kennen diese und können damit den korrekten Hash bilden.

Alle Parameter, die im Formular zum Hashen gesammelt und im Request mitgeschickt wurden, müssen auch in den Aktioneinstellungen angegeben werden. Umgekehrt müssen aber nicht alle Parameter, die im Aktionsbaustein stehen, auch im Request vorhanden sein. Falls sie vorhanden sind, müssen sie allerdings auch abgesichert sein. Das Formular und die Aktion müssen im Grunde denselben Satz an Parametern zur Hashbildung benutzen, sonst sind die Ergebnisse verschieden.