1. Änderungen in Version 2.2.0

Tabelle 1. Änderungen in Version 2.2.0

Ticket-Nr	Beschreibung
3	Validatoren konfigurierbar Die DbUnit-Toolbox verwendet jetzt vier Standard-Validatoren: Time-, Date-, Number- und String-Validator. Validatoren können für alle oder für einzelne Tests konfiguriert werden. Dies umfasst die Aktivierung/Deaktivierung und das Festlegen der Ausführungsreihenfolge der Validatoren. Darüber hinaus können neue Validatoren registriert werden.
4	Annotation zum Löschen der Datenbank Um die Tabellen der Datenbank initial zu löschen, musste bisher eine leere XML definiert werden. Jetzt gibt es die ClearTables Annotation, mit der dieses Verhalten direkt an der Klasse angegeben werden kann.
8	Ausschluss von cachable Tabellen aus dem Diff Report Möchte man einen übersichtlicheren DiffReport generieren, so können die cacheable Tabellen jetzt ausgeschlossen werden.
22	DTD im Wurzelverzeichnis Wenn die DTD im Wurzelverzeichnis liegt, wurde der Pfad bisher auf 'null' gesetzt. Jetzt wird der Pfad leer gelassen, sodass DTDs auch im Wurzelverzeichnis definiert werden können.
24	Konstanten für INCLUDE-Tabelle Für die Attribute des Include-Tags werden intern jetzt überall Konstanten verwendet.
25	INCLUDE-Tag sollte konfigurierbar sein Für den Fall, dass eine Tabelle der zu testenden Datenbank "INCLUDE" heißt, kann der Bezeichner des INCLUDE-Tags auf einen anderen Namen konfiguriert werden.
28	Neue Methode assertEqualsInitial Mit der Methode assertEqualsInitial kann nun überprüft werden, ob die Tabellen genau den initialen Daten entsprechen. Im Gegensatz zu assertInitialDataUnchanged wird nicht nur überprüft, ob die initialen Daten nicht geändert wurden, sondern auch ob keine neuen hinzugefügt wurden.
29	Leere Tabellen prüfen Um sicherzustellen, dass eine Tabelle initial leer ist und auch zum Vergleich ob eine Tabelle nach Testausführung keine Daten enthält, kann man nun das EMPTY_TABLE Tag in den XML Dateien nutzen. Auch hierfür ist der Bezeichner konfigurierbar.
33	Konfiguration Test-URL Für die Selenium Toolbox gibt es jetzt eine TestUrl Annotation, mit der die zu testende URL direkt an der Testklasse oder –methode annotiert werden kann.
34	Der Tabellen-Cache sollte global über die Konfiguration ausgeschaltet werden können Möchte man das Caching z.B. zu Analysezwecken ausschalten, kann dies nun auch global für alle Tests über das Configuration Callback erfolgen.
35	Bei NoCache werden gar keine Tabellen in den Cache geschrieben Der NoCache Annotation kann optional eine Liste von Tabellen übergeben werden, die nicht gecacht werden sollen. Bisher wurden in dieser Situation keine neuen Daten in den Cache aufgenommen. Das Verhalten wurde dahingehend geändert, dass cacheable Tabellen, die

Ticket-Nr	Beschreibung
	noch nicht im Cache stehen und die nicht über die NoCache-Annotation ausgeschlossen wurden, in den Cache aufgenommen werden.
36	Bei dumpDatabase mit auszuschließenden Tabellen Möchte man einen übersichtlicheren Datenbank Dump erstellen, so können die cacheable Tabellen jetzt ausgeschlossen werden.
58	Globaler Tabellenfilter Tabellen, die von der DbUnit-Toolbox nicht berührt werden sollen, können nun in dem Configuration Callback ausgeschlossen werden. Diese Tabellen verhalten sich für die Toolbox, als seien sie nicht vorhanden. Das betrifft das Löschen und das Einspielen von Tabellen, das Erstellen von DTDs, Reports und Dumps etc.
61	Inkludierte Testdaten werden nicht gefunden Der Fall, dass die Testdaten nicht im Klassenpfad relativ zu den referenzierten Testdaten wurde in die FAQ aufgenommen.
96	Mehrere DbUnit-Toolbox-Umgebungen unter JUnit4 und TestNG ermöglichen Die Verwendung der Test-Toolbox über Annotationen war nur eingeschränkt möglich. Da die Komplexität der Annotationen immer stärker zunehmen würde und die Annotationen keinen sichtbaren Mehrwert liefern, wurde auf die Konfiguration der Test-Toolbox durch Annotationen vollständig verzichtet.
103	FAQ: Warum ändert sich die Reihenfolge der Tabellen in der DTD? Existieren Fremdschlüsselbeziehungen zwischen den Tabellen, werden diese aufgelöst, indem die Reihenfolge der Tabellen so angepasst wird, dass bei dem Einspielen der Testdaten in der festgelegten Reihenfolge keine Fremdschlüsselverletzungen auftreten können.
126	Integration JUnit4 und TestNG Bei der Integration von JUnit4 und TestNG wurden sehr viele Annotationen verwendet. Dadurch unterscheidet sich die Initialisierung der Test-Toolbox je nachdem, welches Testframework eingesetzt wird. Die Integration der Test-Toolbox sollte jedoch unter allen Testframeworks möglichst einheitlich aussehen. Aus diesem Grund wird die gesamte Initialisierung der Test-Toolbox wieder in den setUp- und tearDown-Methoden durchgeführt. Es gibt lediglich zwei Execution-Listener, die die Initialisierung unterstützen. Zum Einen kann durch den TestMethodNameResolvingExecutionListener der Name der aktuellen Testmethode ermittelt werden. Der Name wird in der Klasse ToolboxMethodNameResolver zur Verfügung gestellt. Dieser Execution-Listener wird für JUnit in der Version 4 bis 4.7 benötigt. In diesen Versionen gibt es keine andere Möglichkeit, den Namen der aktuellen Testmethode zu ermitteln. Zum Anderen kann durch den Execution-Listener ApplicationContextResolvingExecutionListener der aktuelle Application-Kontext ermittelt werden. Der Kontext steht dann in der Klasse ToolboxApplicationContextResolver zur Verfügung.
127	ATDD-Report als Hudson-Plugin Über ein Maven- und ein Hudson-Plugin wird die Verbindung zwischen den Akzeptanztests der User-Stories und den Ergebnissen der Test-Ausführung hergestellt. Auf diese Art und Weise erhält man einen fachlichen Überblick über den Fortschritt der Entwicklung innerhalb eines Sprints.
128	Eigener Test-Toolbox ClasspathResourceLoader

Ticket-Nr	Beschreibung
	Die Toolbox stellt jetzt den zusätzlichen CoreClassLoader zur Verfügung, der ohne einen SpringContext instanziiert werden kann. Die bisher obligatorische Angabe einer ClasspathResourceLoader-Instanz ist jetzt somit optional.
132	Diff-Report braucht Metadaten für alle Tabellen Aktuell braucht der Diff-Report die Metadaten für alle Tabellen der Datenbank. Dies führt zu Problemen, wenn man viele Tabellen in der Datenbank hat, die für die aktuellen Tests noch keine Metadaten benötigen. Der Diff-Report wurde dahingehend geändert, dass bei Tabellen ohne Metadaten implizit alle Spalten als Metadaten verwendet werden.
133	Explizite Fehlermeldung bei fehlendem Include Wenn ein Include nicht vorhanden ist, erschien bisher eine NullPointerException. Jetzt wird an dieser Stelle eine explizite Exception geworfen.
135	Maskierung bei Datenbankdumps Wenn Datenbankdumps erstellt werden, werden jetzt Schlüsselbegriffe aus der Datenbank maskiert. Steht in der Datenbank z.B. der Wert 'Hans -> Heidi' so führt das beim Einlesen der Daten zu Problemen, da '->' das Starttag für Funktionen ist. Bei der Erstellung eines Dumps wird daher der Wert der Datenbank ausmaskiert 'Hans ^->Heidi'. Natürlich wird bei der Prüfung die aktuell konfigurierte Syntax berücksichtigt.
137	Mehr Kontext in Fehlermeldungen Die Fehlermeldungen zum Parsen der Spaltenwerte enthält jetzt immer den Wert der Spalte, in der der Fehler aufgetreten ist.
139	Groß- und Kleinschreibungsproblem Innerhalb des XML-Converters werden die Namen der Attribute immer in Großbuchstaben umgewandelt. Das führt zu Problemen, wenn die verwendete Tabelle zwischen Groß- und Kleinschreibung unterscheidet. Die Test-Toolbox wurde dahingehend angepasst, dass die exakte Schreibweise der Tabellen und Spalten berücksichtigt wird.
140	Diff-Report sollte nur MetaDaten von gefüllten Tabellen benötigen Der Diff-Report ließ sich bisher nicht generieren, wenn Tabellen in der Datenbank existieren, für die keine MetaDaten angegeben wurden. Jetzt werden diese Tabellen ebenfalls in dem Diff-Report angezeigt, wobei ein Vergleich der Daten nicht möglich ist.
141	Groß- und Kleinschreibungsproblem (Cache) Die Verwendung der Tabellennamen ist jetzt überall konsistent und case-sensitiv, d.h. für die Test-Toolbox gehören die Tabellennamen "USERS", "users" und "Users" zu drei unterschiedlichen Tabellen.
142	Firebug in Benutzerhandbuch Firebug wurde als Tool für die Ermittlung von HTML-Lokatoren aufgenommen.
144	EJB3-Toolbox Für die Verwendung der Test-Toolbox im EJB3-Umfeld wurde eine eigene Toolbox-Komponente entwickelt, die einen Embedded Container im Test startet und die Beans in diesem Container ausführt.
145	Dynamisches Linken Bisher mussten die optionalen Abhängigkeiten der Test-Toolbox zum Test-Connector explizit in der pom des Projekts beim Kunden angegeben werden. In der neuen Version werden diese Abhängigkeiten dynamisch aufgelöst. Das bedeutet, dass immer alle Ausprägungen (JUnit3,

Ticket-Nr	Beschreibung
	JUnit4 oder TestNG) im Classpath vorhanden sind und ein genereller Test-Connector zur Laufzeit feststellt, welche Klassen eingebunden sind und dann den konkreten Test-Connector verwendet.
147	FAQ: Fehlermeldung trotz korrekter Testdaten Die Fehlermeldung "java.lang.RuntimeException: org.dbunit.dataset.DataSetException: Line XX: The content of element type "dataset" must match "(INCLUDE*,TABLE1*, TABLE2*, ..., TABLEn*)" tritt auf, wenn die Reihenfolge der Tabellen in den Testdaten nicht mit der Reihenfolge in der DTD übereinstimmt. Diesen Laufzeitfehler kann man verhindern, wenn man die XML-Testdaten bereits in der IDE gegen die DTD validiert.
157	Best-Practice: Datenbank-Sequenz mit Puffer Oftmals werden Sequenzen in der Datenbank verwendet, um neue IDs zu erzeugen. Dies kann zu Konflikten führen, wenn z.B. die Testdaten die gleiche ID verwenden, wie die Sequenz. In diesem Fall kann ein Datensatz u.U. nicht angelegt werden, da die ID bereits vorhanden ist. Um dieses Problem zu umgehen, kann die Sequanz mit einem Offset initialisiert werden, d.h. anstatt bei 1 fängt die Sequenz dann bei 100, 1.000 oder 10.000 an. Der dadurch geschützte ID-Raum kann für die IDs der Testdaten verwendet werden.
161	Kapseln der Toolbox API Die API der DbUnit-Toolbox wurde dahingehend geändert, dass nur noch Interfaces verwendet werden. Dies ermöglicht eine bessere Trennung zwischen der Schnittstelle und der Implementierung der Test-Toolbox. Innerhalb der Selenium-Toolbox sind einige Klassen nach wie vor zugreifbar, da sie als Basisklassen zur Verfügung stehen müssen.
166	FAQ: Was sind Lookup-Keys? Das Konzept der Lookup-Keys ähnelt dem der Primärschlüssel. Für jede Tabelle wird eine Menge von Spalten definiert, die den Lookup-Key für diese Tabelle bilden. Durch die Spaltenwerte sind die einzelnen Datensätze eindeutig identifizierbar. Dadurch ist eine Zuordnung von Testdaten auf Datenbankinhalte möglich.
168	FAQ: Warum brauche ich eine DTD? DbUnit braucht nicht zwingend eine DTD, dann ist aber sehr intransparent, wie die Vergleiche aussehen. Teilweise geht das gar nicht wegen der fehlenden Unterscheidung zwischen null und nicht zu prüfenden Spalten. Weitere Vorteile: Validierung der Testdaten in der IDE, Autovervollständigung in der IDE.