Kapitel 6. Fragen und Antworten

6.1. Checkerberry test center

6.1.1. Wie kann ich die Versionsnummer einer externen Bibliothek ändern?

Die Versionsangaben in den Maven-Konfigurationen des checkerberry test centers beziehen sich nie auf feste Versionen, sondern geben immer die empfohlene Versionsnummer an. Wird das checkerberry test center in ein Maven-Projekt des Kunden eingebunden, können die Versionsnummern einfach in der Maven-Konfiguration überschrieben werden. Maven richtet sich bei der Auswahl der Versionsnummer an der Entfernung zum Wurzelprojekt, welches in diesem Beispiel das Kundenprojekt ist. Das bedeutet, dass die Definition der Versionen in den `pom.xml` Dateien des Kundenprojekts (Distanz 1) eine höhere Priorität haben, als die Versionen des checkerberry test center (Distanz 2). Dies liegt daran, dass die Maven-Konfigurationen des checkerberry test centers in dem Kundenprojekt lediglich referenziert werden.

Neben dem Standardverfahren von Maven zur Ermittlung der besten Version kann für jede externe Bibliothek die verwendete Versionsnummer geändert werden. Für alle externen Bibliotheken existiert eine Property, die den Wert der Version enthält. Das Property kann durch eine Konfiguration in der `settings.xml` überschrieben werden. Folgendes Code-Beispiel beschreibt die erforderliche Konfiguration der `settings.xml`.

```
…
<profiles>
  <profile>
    <activation>
      <activeByDefault>true</activeByDefault>
    </activation>
    <properties>
      <org.dbunit.checkerberry.version>2.4.7</org.dbunit.checkerberry.version>
    </properties>
  </profile>
</profiles>
…
```

Beispiel 6.1. Änderung einer Versionsnummer in der settings.xml

Um ein Property in der `settings.xml` zu setzen, muss ein neues Profil angelegt werden. In dem angegebenen Code-Beispiel wird das Profil standardmäßig aktiviert (`<activeByDefault>true</activeByDefault>`). Dann wird die Versionsnummer von DbUnit auf den Wert 2.4.7 gesetzt. Diese Einstellung führt dazu, dass der in dem checkerberry test center konfigurierte Wert überschrieben wird.

Es ist darauf zu achten, dass die `settings.xml` immer dem aktuellen Benutzer zugeordnet ist. Daher muss diese Einstellung bei allen Nutzern angepasst werden. Dies betrifft insbesondere auch die Benutzer, unter deren Namen die Continuous Integration Systeme gestartet werden.

6.1.2. Wie konfiguriere ich die Logging-Einstellungen?

Das checkerberry test center verwendet `slf4j` [slf4j Homepage, 2013] als Logging-Api. Dadurch kann der Benutzer durch eine Bindung (z.B. `slf4j-log4j12-1.7.5.jar`) die konkrete Implementierung des Logging Frameworks wählen, z.B. Log4j [Log4j Homepage, 2010]. Das folgende Code-Beispiel enthält eine mögliche Konfiguration in Form einer `log4j.properties` Datei.

```
log4j.rootLogger=INFO, MyConsApp, MyFileApp
log4j.appender.MyConsApp=org.apache.log4j.ConsoleAppender
log4j.appender.MyConsApp.layout=org.apache.log4j.PatternLayout
log4j.appender.MyConsApp.layout.ConversionPattern=%d{ISO8601} %-5p [%t] %c:
%m%n

log4j.appender.MyFileApp=org.apache.log4j.DailyRollingFileAppender
log4j.appender.MyFileApp.datePattern='.'yyyy-MM-dd_HH-mm
log4j.appender.MyFileApp.file=checkerberry.log
log4j.appender.MyFileApp.layout=org.apache.log4j.PatternLayout
log4j.appender.MyFileApp.layout.ConversionPattern=%d{ISO8601} %-5p [%t] %c:
%m%n

log4j.logger.de.conceptpeople=DEBUG
```

Beispiel 6.2. log4j.properties

Das Code-Beispiel enthält eine mögliche log4j-Konfiguration. Die Konfiguration aktiviert einen Appender, der die Ausgabe auf die Konsole steuert (`MyConsApp`). Des Weiteren wird ein Appender definiert, der Log-Informationen in die Datei checkerberry.log schreibt (`MyFileApp`).

Über den `rootLogger` wird der Standard-Log-Level definiert, der bei der Logging-Ausgabe berücksichtigt wird. In dem Beispiel werden nur Logging-Ausgaben auf INFO-Level und darüber ausgegeben. Als Ausnahme wird für alle Packages, die mit `de.conceptpeople` beginnen, der Log-Level auf DEBUG gesetzt.

Eine detaillierte Auflistung der Konfigurationsmöglichkeiten ist auf der Log4j-Homepage vorhanden.

6.1.2.1. Konfiguration des Logging von Selenium

Selenium verwendet die Java Logging API. Die Konfigurationsdatei von der der Java Logging API liegt unter `JAVA_HOME/jre/lib`. Um eine andere Konfiguration zu verwenden kann die Datei über das Property `java.util.logging.config.file` angegeben werden. Der entsprechende Aufruf mit Maven wäre:

```
mvn clean install -Djava.util.logging.conf.file=C:\settings\logging.properties
```

Beispiel 6.3. Maven Aufruf mit Angabe einer alternativen logging.properties

Ein Konfigurationsbeispiel für die Datei logging.properties ist in Beispiel 6.4, „logging.properties“ angegeben.

```
handlers=java.util.logging.ConsoleHandler

# Limit the message that are printed on the console to INFO and above.
java.util.logging.ConsoleHandler.level=INFO
java.util.logging.ConsoleHandler.formatter=java.util.logging.SimpleFormatter
```

Beispiel 6.4. logging.properties

6.1.3. Welche generellen Fehlermeldungen gibt es?

Tabelle 6.1. Fehlermeldungen des checkerberry test centers

CB-1000	Das Verzeichnis 'XXX' konnte nicht erstellt werden.
Um eine Datei zu speichern, z.B. eine neue DTD, einen Datenbankdump, einen Diff-Report, o.ä., sollte ein neues Verzeichnis erstellt werden. Das Erstellen dieses Verzeichnisses ist fehlgeschlagen.	
CB-1001	Spring hat eine Exception geworfen.

Das checkerberry test center wird in Verbindung mit Spring genutzt. Diese Fehlermeldung kapselt eine Exception aus dem Springframework und tritt z.B. auf, wenn der ApplicationContext nicht geladen werden konnte.	
CB-1002	Unbekanntes Testframework. Bitte lesen sie die Dokumentation dieser Exception (entweder im Benutzerhandbuch oder im Quelltext) für weitere Hinweise zu möglichen Lösungswegen.
Das Testframework konnte nicht ermittelt werden. Wie Sie das checkerberry test center für verschiedene Testframeworks konfigurieren, können Sie unter Abschnitt 2.5.3, „Erzeugen der checkerberry db-Umgebung" nachlesen.	
CB-1004	Unerwarteter Programmzustand. Diese Exception sollte niemals auftreten. Bitte wenden Sie sich an die ConceptPeople consulting gmbh.
Wir entschuldigen uns für die Unannehmlichkeiten. Bitte teilen Sie uns mit, dass diese Exception aufgetreten ist.	
CB-1005	Der TestConnectorCreator 'XXX' konnte nicht instanziiert werden.
Der über die TestConnectorConfiguration Annotation angegebene *`TestConnectorCreator`* konnte nicht instanziiert werden.	
CB-1006	Der Name der aktuellen Testmethode konnte nicht bestimmt werden. Bitte lesen sie die Dokumentation dieser Exception (entweder im Benutzerhandbuch oder im Quelltext) für weitere Hinweise zu möglichen Lösungswegen.

Das checkerberry test center konnte den Namen der aktuellen Testmethode nicht bestimmen. Dies hängt damit zusammen, dass einige Testframeworks den Zugriff auf den Methodennamen nur über Umwege ermöglichen. Das checkerberry test center bietet daher für unterschiedliche Frameworks jeweils passende Methoden zur Bestimmung des Namens:

- **JUnit3** erlaubt den direkten Zugriff auf den Methodennamen. Es sind keinerlei andere Methoden nötig.
- **JUnit4** bietet ab der Version 4.7 mit der `@Rule`-Annotation eine einfache Möglichkeit, den Methodennamen zu bestimmen. Um dem checkerberry test center dies zu ermöglichen, fügen Sie bitte folgenden Code in Ihre Testklasse ein:

```
@Rule
public MethodNameDeterminationRule rule = new MethodNameDeterminationRule();
```

  In den Versionen zwischen 4.0 bis 4.7 bietet JUnit4 leider keinerlei Möglichkeit, die Testmethode zu bestimmen. Es wird daher empfohlen auf das Spring-Framework zurückzugreifen (s.u.) oder auf eine aktuellere Version von JUnit umzusteigen.
- **JUnit4+Spring:** Das Spring Framework stellt ebenfalls einen Mechanismus zur Verfügung, mit dem sich der Methodenname bestimmen lässt. Um diesen Weg unter JUnit4 zu nutzen, annotieren Sie Ihre Testklasse bitte folgendermassen:

```
@RunWith(SpringJUnit4ClassRunner.class)
@TestExecutionListeners({ TestMethodNameResolvingExecutionListener.class })
```

- **TestNG:** In TestNG lässt sich die Testmethode einfach bei der Verwendung der Annotation `@BeforeMethod` angeben:

```
@BeforeMethod
public void setUp(Method testMethod) {
...
environment.setUp(this, method);
}
```

Bei jeder dieser Methoden reicht es aus, sie im abstrakten Basistestfall anzuwenden. Auf diese Weise können alle abgeleiteten Testfälle die Methode ohne Code-Duplizierung nutzen. (siehe Abschnitt 2.5.3, „Erzeugen der checkerberry db-Umgebung“)

CB-1007	Die Testmethode 'XXX' existiert nicht.
Die Testmethode mit dem Namen XXX konnte nicht gefunden werden. Möglicherweise gibt es ein Problem mit der Einstellung des Testframeworks (siehe Abschnitt 2.5.3, „Erzeugen der checkerberry db-Umgebung“).	
CB-1008	Der Bezeichner 'XXX' enthält das XML Steuerzeichen 'YYY' und kann deshalb nicht verwendet werden.
Es wurde der String XXX für einen konfigurierbaren Bezeichner gewählt. Dieser wird in XML Dateien verwendet und darf deshalb kein XML Steuerzeichen enthalten. Bitte entfernen Sie den Substring YYY aus dem Bezeichner.	
CB-1009	Der Wert 'XXX' konnte nicht in den Typ 'YYY' konvertiert werden.
Bei der Verwendung der Validatoren werden aus String-Representationen spezielle Java-Klassen z.B. java.util.Date erzeugt. Wenn eine Zeichenkette XXX nicht in einen Typ YYY konvertiert werden konnte, wird diese Fehlermeldung erzeugt.	
CB-1010	Der Wert "XXX" konnte nicht in eine BCD-Darstellung konvertiert werden, da die oberen 4 Bit ("Y") oder die unteren 4 Bit ("Z") nicht durch eine Ziffer darstellbar sind.
Dieser Fehler tritt auf, wenn in checkerberry db die Verwendung des BCD-Formats aktiviert ist und der Wert XXX nicht im BCD-Format dargestellt werden kann. Dies ist der Fall, wenn der Wert mindestens ein Byte enthält, bei dem das obere (Y) oder untere Nibble (Z) einen Wert enthält der nicht zwischen 0 und 9 liegt. Die Werte Y und Z werden als Zahl angegeben z.B. 9 oder 11.	
CB-1011	Die BCD-Darstellung >XXX< konnte nicht in ein Byte-Array konvertiert werden, da sie ein ungültiges Zeichen ("Y") enthält.
Dieser Fehler tritt auf, wenn ein String im BCD-Format ein ungültiges Zeichen Y enthält z.B. „X'1ö'“ und daher nicht in ein Byte-Array konvertiert werden konnte.	

CB-1012	Die BCD-Darstellung >XXX< hat eine ungültige Länge. Die Anzahl der Ziffern muss ein Vielfaches von 2 sein.
Dieser Fehler tritt auf, wenn ein String im BCD-Format mit ungültiger Länge angegeben wird z.B. „X'123'".	
CB-1013	Die BCD-Darstellung >XXX< hat ein ungültiges Präfix. Das gültige Präfix lautet >Y<.
Dieser Fehler tritt auf, wenn ein String im BCD-Format das falsche Präfix verwendet z.B. „X123". Das erwartete Präfix („X'") wird in der Meldung ebenfalls angezeigt.	
CB-1014	Die BCD-Darstellung >XXX< hat ein ungültiges Suffix. Das gültige Suffix lautet >Y<.
Dieser Fehler tritt auf, wenn ein String im BCD-Format das falsche Suffix verwendet z.B. „X'123". Das erwartete Präfix („'") wird in der Meldung ebenfalls angezeigt.	

6.2. Checkerberry db

Dieses Kapitel listet eine Reihe von häufig gestellten Fragen und deren Antworten auf.

6.2.1. Wie erstelle ich eine neue DTD?

Die Erzeugung einer neuen DTD erfolgt über die Methode `createDtd` der Klasse `DbTestHandler`. Die Methode kann temporär innerhalb eines Testfalls ausgeführt werden und erzeugt dann eine neue DTD anhand der aktuellen Datenbankstruktur.

Zusätzlich erzeugt checkerberry db automatisch eine neue DTD, wenn initiale Testdaten nicht in die Datenbank eingespielt werden können. Der Name der neuen DTD hat das Präfix „`new-`" und wird im Klassenpfad abgelegt. Danach muss die erstellte DTD umbenannt und in den Source-Baum kopiert werden. Weitere Informationen sind in Abschnitt 2.4.1.5, „Anlegen einer neuen Testdaten-DTD"beschrieben.

6.2.2. Wie kann ich die Datenbank-Statements loggen?

Die Konfiguration zum Loggen der Batch-Statements ist in Abschnitt 2.4.17, „Aktivieren des Datenbank-Loggings" beschrieben.

6.2.3. Warum bekomme ich bei einem Datenbank-Dump einen OutOfMemoryError?

Bei der Erstellung eines Datenbank-Dumps werden die Daten aus der Datenbank in den Speicher geladen. Wenn die Datenbank viele Daten beinhaltet, reicht der reservierte Speicherplatz nicht aus, sodass es zu einem `OutOfMemoryError` kommt. Das Problem kann durch eine Konfigurationsänderung in DbUnit gelöst werden (siehe Abschnitt 2.4.20, „Anpassen von DbUnit-Properties und Features").

6.2.4. Meine Testdatei ist da, wird aber nicht gefunden?

Dieses Verhalten kann mehrere Ursachen haben. Zum einen muss geprüft werden, ob die Datei tatsächlich mit dem korrekten Namen im Klassenpfad liegt. Wurde ggf. etwas an der Namenskonvention zum Auffinden der Testdaten geändert? Die Log-Ausgaben sollten auf diese Punkte entsprechende Hinweise geben. Eine weitere Ursache kann in der Länge des absoluten Dateinamens liegen. Wenn der Dateiname inklusive der Pfadangaben über 256 Zeichen lang ist, kann das Betriebssystem die Datei nicht mehr lesen, sodass die Datei nicht gefunden wird. Die einzige Lösung besteht darin, den Namen der Testdatei zu kürzen.

6.2.5. Meine inkludierten Testdaten (LOCATION=RELATIVE) werden nicht gefunden?

Checkerberry db sucht die inkludierten Testdaten relativ zu den einbindenden Testdaten. Es ist zu beachten, dass die einzubindenden Testdaten aus dem Klassenpfad verwendet werden. Die inkludierten Testdaten müssen somit relativ zu der Testdatendatei in dem Klassenpfad vorhanden sein. Bitte prüfen Sie unbedingt die Log-Ausgaben.

6.2.6. Wie wird die Verbindung zwischen Datenbank und erwarteten Daten hergestellt?

Bei der Überprüfung der erwarteten Ergebnisse muss checkerberry db eine Zuordnung zwischen erwarteten und tatsächlichen Daten herstellen. Auf diese Art und Weise werden neue, fehlende und geänderte Datensätze erkannt. Zu diesem Zweck muss für jede zu überprüfende Tabelle ein Eintrag in den Tabellenbeschreibungen vorgenommen werden. Pro Tabelle werden Lookup-Keys definiert, die für die Identifizierung einzelner Zeilen verwendet werden. Bei den Lookup-Keys handelt es sich um Namen der Spalten der Tabelle, die für die Identifizierung herangezogen werden. Lookup-Keys sind konzeptionell somit identisch zu Primary Keys.

Es wird empfohlen fachliche Schlüssel für die Lookup-Keys zu verwenden (siehe Abschnitt 7.2.7, „Verwende fachliche Lookup-Keys"). Da Primary Keys häufig technische Schlüssel sind, verzichtet checkerberry db darauf, diese generell als Lookup-Keys zu verwenden.

6.2.7. Warum liest checkerberry db nicht die korrekten Werte aus der Datenbank?

Wenn checkerberry db bei der Überprüfung von erwarteten Testdaten unerwartete Werte aus der Datenbank liest, liegt dies mit hoher Wahrscheinlichkeit an einem Fehler im Transaktionsverhalten. Checkerberry db kommuniziert über eine eigene JDBC-Verbindung mit der Datenbank, sodass diese Statements nicht Teil der Transaktion der zu testenden Komponenten sind. Wenn die Transaktion noch geöffnet ist, während checkerberry db die erwarteten Testdaten prüft, sind die Änderungen durch die zu testenden Komponenten für checkerberry db nicht sichtbar. Diese Situation tritt auf, wenn die Transaktion der zu testenden Komponenten in der Setup-Phase geöffnet und erst nach dem Vergleich z.B. in der Teardown-Phase geschlossen wird. In dieser Konstellation muss die Transaktion vor der Überprüfung durch checkerberry db manuell geschlossen werden, damit die Änderungen in der Datenbank sichtbar werden.

Eine detaillierte Beschreibung der Problematik ist in Kapitel Abschnitt 2.5.3.2, „Einstellen der Transaktionsverwaltung" aufgeführt.

6.2.8. Was sind Lookup-Keys?

Bei der Überprüfung von Testdaten mit Datenbankinhalten ist die Zuordnung von Datensätzen aus den Testdaten zu Datensätzen aus der Datenbank erforderlich. Zu diesem Zweck werden die sogenannten Lookup-Keys verwendet. 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.

Im Gegensatz zu Primärschlüsseln werden für Lookup-Keys in der Regel fachliche und keine technischen Schlüssel verwendet. Insbesondere bedeutet dies, dass der Lookup-Key von dem Primärschlüssel einer Tabelle abweichen kann. Dies ist der wesentliche Grund, warum die neue Bezeichnung „Lookup-Key" eingeführt wurde.

Darüber hinaus können Lookup-Keys einer Tabelle variable pro Testmethode angepasst werden. Dies kann sinnvoll sein, wenn der standardmäßig definierte Lookup-Key in speziellen Testsituationen nicht eindeutig ist.

6.2.9. Muss ich immer eine DTD für die Testdaten verwenden?

Bei der Verwendung von checkerberry db ist die Verwendung einer DTD vorgeschrieben. Checkerberry ist an dieser Stelle somit restriktiver als DbUnit. Warum ist das so?

Die DTD definiert die Struktur der zu testenden Datenbank. Sie enthält die Namen der Tabellen in der Reihenfolge ihrer Abhängigkeiten, definiert für jede Tabelle die vorhanden Spalten und legt fest, welche Spalten Pflichtangaben (`not nullable`) enthalten.

DbUnit unterstützt bei der Überprüfung von Testdaten mit der Datenbank zwei verschiedene Vorgehensweisen. Zum einen kann eine DTD verwendet werden. DbUnit prüft dann für jede angegebene Tabelle die Gleichheit der Werte für alle Spalten.

Zum anderen kann DbUnit ohne DTD verwendet werden. In dieser Situation werden die zu überprüfenden Spalten einer Tabelle direkt aus den Testdaten ermittelt. Der erste Eintrag der Testdaten legt fest, welche Spalten bei der Überprüfung berücksichtigt werden sollen. Enthalten die Testdaten zum Beispiel vier Datensätze für die Tabelle `USERS` und der erste Eintrag beinhaltet die Spalten `ID`, `NAME` und `SURNAME`, dann werden nur diese drei Spalten überprüft – auch für die drei folgenden Datensätze. Der Vorteil dieses Vorgehens besteht darin, dass dynamisch festgelegt werden kann, welche Spalten überprüft werden sollen. Der Nachteil liegt darin, dass dieses Verhalten sehr intransparent ist.

Es existieren im Wesentlichen zwei Probleme bei dem Verzicht auf eine DTD. Das erste Problem besteht darin, dass durch die Definition des ersten Testdatensatzes aus Versehen Spalten von der Überprüfung ausgeschlossen werden. In dem obigen Beispiel würden die Werte der Spalte `BIRTHDATE` nicht geprüft werden, wenn sie in den drei folgenden Datensätzen verwendet werden würden. In diesem Fall würde der Test erfolgreich durchlaufen, obwohl Abweichungen in Spalten vorhanden sind. Das zweite Problem betrifft die Behandlung von `null`-Werten. Innerhalb der Testdaten werden `null`-Werte dadurch definiert, dass die zugehörigen Spalten in dem Datensatz nicht aufgenommen werden. Dies führt zu einer doppelten Bedeutung nicht vorhandener Spalten. In dem ersten Datensatz einer Tabelle bedeuten nicht vorhandene Spalten, dass diese nicht überprüft werden sollen. In den folgenden Datensätzen bedeuten sie, dass die Werte `null` sind. Diese Situation führt schnell zu Problemen, wenn in dem ersten Datensatz eine Spalte einen `null`-Wert enthält, der geprüft werden soll. Diese Situation tritt sehr häufig ein und kann ohne DTD nicht gelöst werden.

Aus der Erfahrung heraus kann DbUnit aus den oben beschriebenen Gründen sinnvoll nur mit einer DTD verwendet werden. Anderenfalls ergeben sich große Fehlerpotentiale oder wilde Work-Arounds, die der Verständlichkeit der Tests und der Testdaten schaden. In der DbUnit-Dokumentation wird im Übrigen ebenfalls die Verwendung einer DTD empfohlen.

Darüber hinaus ergeben sich durch die Verwendung einer DTD weitere Vorteile. Die Testdaten können bereits während der Entwicklung auf Gültigkeit validiert werden. Des Weiteren bieten die gängigen Entwicklungsumgebungen Auto-Vervollständigungen bei der Erstellung der XML-Testdaten an, sodass die Erstellung der Testdaten deutlich beschleunigt wird.

6.2.10. Warum bekomme ich die Fehlermeldung trotz korrekter Testdaten?

Bei der Verwendung von Testdaten kann es zu folgender Fehlermeldung kommen:

java.lang.RuntimeException: org.dbunit.dataset.DataSetException: Line XX: The content of element type "dataset" must match "(INCLUDE*,TABLE1*, TABLE2*, ..., TABLEn*)"

Diese Fehlermeldung erscheint, wenn die Reihenfolge der Tabellen innerhalb der Testdaten nicht der vorgegebenen Reihenfolge aus der DTD entspricht. Die Reihenfolge ist jedoch wichtig, da z.B. beim Einspielen von Testdaten die Fremdschlüsselbeziehungen berücksichtigt werden müssen. Anderenfalls kann es passieren, dass ein Testdatensatz in die Datenbank eingespielt wird, der auf einen noch nicht vorhandenen Datensatz verweist. In diesem Fall würde die Datenbank eine Fehlermeldung wegen einer Constraint-Verletzung werfen. In der DTD berücksichtigt die Reihenfolge der Tabellen diese Abhängigkeiten, sodass diese Reihenfolge als bindend für alle Testdaten verwendet wird.

Das Auftreten dieses Laufzeitfehlers kann bereits während der Entwicklung verhindert werden, wenn die Testdaten bereits in der Entwicklungsumgebung gegen die DTD validiert werden.

6.2.11. Warum ändert sich die Reihenfolge der Tabellen in der DTD?

In der DTD wird die Struktur der Datenbank beschrieben. Dies umfasst u.a. die Liste der Tabellennamen, die in der Datenbank vorhanden sind. Die Reihenfolge ist dabei festgelegt. Bei der Generierung einer neuen DTD aus der Datenbank kann es sein, dass sich die Reihenfolge der Tabellen ändert. Um diesen Sachverhalt zu verstehen, muss man sich anschauen, wie die Reihenfolge ermittelt wird.

Zunächst werden die Namen der Tabellen alphabetisch sortiert. Wenn die Tabellen keine Fremdschlüsselbeziehungen haben, ist das 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. Die Reihenfolge der Tabellen ist somit abhängig von den vorhandenen Fremdschlüsselbeziehungen. Ändert sich etwas an diesen Beziehungen, kann sich dadurch ebenfalls eine neue Reihenfolge der Tabellen ergeben.

6.2.12. Welche Fehlermeldungen gibt es in checkerberry db?

Tabelle 6.2. Fehlermeldungen von checkerberry db

CB-DB-1000	Der Ausführungskontext ist nicht definiert.
Diese Fehlermeldung erscheint, wenn innerhalb von checkerberry db auf den Ausführungskontext zugegriffen wird, dieser aber nicht vorhanden sind. Der Ausführungskontext ist dabei ein Container, der verschiedene konkrete Kontexte z.B. den ParameterContext enthält.	
CB-DB-1001	Die Tabellenbeschreibung für die Tabelle 'XXX' ist nicht definiert.
Die Ermittlung der Lookup-Keys für die Tabelle `XXX` konnte nicht durchgeführt werden, da für die Tabelle `XXX` keine Tabellenbeschreibung definiert ist. Um dieses Problem zu beheben, muss in der Implementierung des Interfaces `DatabaseDescriptionCallback` eine Tabellenbeschreibung für die Tabelle `XXX` angelegt werden.	
CB-DB-1002	Die Lookup-Keys für die Tabelle 'XXX' identifizieren nicht eindeutig die Zeilen der Tabelle.
Diese Fehlermeldung tritt bei der Erzeugung eines Diff-Reports auf, wenn die definierten Lookup-Keys der Tabelle `XXX` die Zeilen der Tabelle nicht eindeutig identifizieren. Das Ergebnis des Diff-Reports wäre dann verfälscht, da die Zuordnung von erwarteten und vorhandenen Daten nicht eindeutig möglich ist.	

CB-DB-1003	Der Kontext für die Testklasse 'XXX' und die Testmethode 'YYY' konnte nicht ermittelt werden.
In der Setup-Phase wird für die zu startende Testmethode ein Kontext aufgebaut, der den Namen der Testklasse und –methode beinhaltet und vorhandene Annotationen auswertet. Diese Fehlermeldung wird geworfen, wenn bei der Erstellung dieses Kontextes ein Fehler auftritt.	
CB-DB-1004	Für die Konfiguration der Datenbankbeschreibung ist kein DatabaseDescriptionCallback definiert.
Diese Fehlermeldung erscheint, wenn innerhalb der checkerberry db-Bridge kein Callback für die Angabe der Datenbankbeschreibung definiert wurde.	
CB-DB-1005	Die initialen Testdaten konnten nicht gelesen werden.
Beim Einlesen der initialen Testdaten ist ein Fehler aufgetreten.	
CB-DB-1006	Die Testdaten mit dem Suffix 'XXX' konnten nicht gelesen werden.
Beim Einlesen der Testdaten mit dem Suffix `XXX` ist ein Fehler aufgetreten.	
CB-DB-1007	Die Daten aus der Datenbank konnten nicht gedumpt werden.
Bei der Erstellung eines Datenbank-Dumps ist ein Fehler aufgetreten, sodass die Daten nicht in einer XML-Datei gespeichert werden konnten.	
CB-DB-1008	Für das Include 'XXX' ('YYY') wurde keine Datei gefunden.
Diese Meldung erscheint, wenn eine Testdatendatei eine andere Datei XXX einbindet, für die keine Testdaten unter der Location `YYY` (`RELATIVE`, `ABSOLUTE`, `CLASSPATH`) gefunden werden konnten.	
CB-DB-1009	Für das Include 'XXX' ('YYY') konnte die Datei 'ZZZ' nicht eingelesen werden.
Diese Meldung erscheint, wenn eine Testdatendatei eine andere Datei XXX einbindet, für die unter der Location `YYY` (`RELATIVE`, `ABSOLUTE`, `CLASSPATH`) die Datei `ZZZ` gefunden wurde, die aber nicht eingelesen werden konnte.	
CB-DB-1010	Die Konfiguration der DTD-Informationen ist nicht vorhanden.
Diese Meldung wird ausgegeben, wenn die DTD-Informationen in der checkerberry db-Konfiguration nicht angegeben wurden. Zur Behebung dieses Fehlers müssen die DTD-Informationen (`public identifier` und relativer Dateiname zum Klassenpfad) in der Konfiguration eingetragen werden (siehe Abschnitt 2.5.2.3, „Konfiguration von checkerberry db mit dem DbConfigurationCallback“).	
CB-DB-1011	Die DTD-Datei 'XXX.dtd' ist nicht vorhanden.

Diese Meldung wird ausgegeben, wenn die konfigurierte DTD mit dem Namen `XXX.dtd` nicht gefunden werden konnte.	
CB-DB-1012	Der Zugriff auf die Datenbank-URL 'jdbc:XXX' wurde nicht erlaubt. Bitte überprüfen Sie die konfigurierten White- und Blacklist-Einstellungen.
Diese Meldung erscheint, wenn die aktuelle JDBC-URL (`jdbc:XXX`) nicht freigeschaltet wurde. Wenn der Zugriff für die angegebene JDBC-URL erlaubt werden soll, muss die Whitelist-Konfiguration angepasst werden.	
CB-DB-1015	Die checkerberry db-Umgebung (CheckerberryDbEnvironment) fehlt. Bitte erstellen Sie die Umgebung in der Setup-Phase.
Diese Meldung erscheint bei dem Zugriff auf die checkerberry db-Umgebung, wenn diese nicht erzeugt wurde. Das bedeutet, dass die Umgebung in der Setup-Phase nicht erzeugt wurde, da ggf. die Testklasse nicht von der korrekten Oberklasse erbt. Bitte achten Sie auch darauf, dass sie die Umgebung in der Teardown-Phase durch den Aufruf der Methode tearDown wieder freigeben.	
CB-DB-1016	Ungültiger Index 'XXX' beim Zugriff auf die Liste der Tabellen mit der Größe 'YYY'.
Innerhalb von checkerberry db wird häufig über Listen von Tabellen iteriert. Diese Fehlermeldung erscheint, wenn der Iterator auf einen ungültigen Index verweist.	
CB-DB-1017	In der Spalte 'XXX' ist ein leerer Parameter enthalten.
Diese Meldung erscheint, wenn ein leerer Parameter `${}` in der Spalte `XXX` in den erwarteten Testdaten definiert wurde. Parameternamen müssen mindestens ein Zeichen beinhalten.	
CB-DB-1018	Für den Parameter 'XXX' in Tabelle A, Zeile B, Spalte C wurde keine Ersetzung angegeben.
Diese Meldung erscheint, wenn in den erwarteten Testdaten der Parameter `${XXX}` definiert wurde und in dem dazugehörigen Test kein Wert für diesen Parameter definiert wurde. Das Problem lässt sich dadurch beheben, dass ein Wert für den Parameter definiert wird. Alternativ kann der Parameter auch aus den Testdaten entfernt werden oder die Spalte zu der Liste der auszuschließenden Spalten hinzugefügt werden. Es ist eine fachliche Entscheidung, welcher Lösungsweg in dieser Situation eingeschlagen wird.	
CB-DB-1020	Es wurden keine initialen Testdaten gefunden.
Checkerberry db ermöglicht die Überprüfung des aktuellen Datenbankinhalts gegen die initialen Testdaten. Diese Fehlermeldung wird geworfen, wenn bei der Überprüfung der Datenbank gegen die initialen Testdaten keine initialen Testdaten vorhanden sind.	
CB-DB-1022	Der Report konnte nicht erstellt werden.

Diese Meldung wird verwendet, wenn während der Erstellung eines Reports ein Fehler auftritt. Die Ursache des Fehlers wird dieser Meldung angehängt.	
CB-DB-1023	Das Template 'XXX' wurde nicht gefunden.
Die Reports werden intern durch die Verwendung von Templates erzeugt. Diese Meldung tritt auf, wenn das benötigte Template `XXX` zur Erstellung eines Reports nicht vorhanden ist.	
CB-DB-1025	Uninterpretierbarer Funktionsaufruf in Tabelle A, Zeile B, Spalte C bei Indizes D
Ein (nicht-maskierter) Funktionsaufruf in dem angegebenen Feld konnte nicht interpretiert werden. Die häufigsten Ursachen hierfür sind Syntaxfehler, das Maskieren nur eines Teils des Funktionsaufrufes oder Fehler beim Schachteln von Funktionsaufrufen. Die Fehlermeldung gibt eine Liste mit den Indizes der einzelnen, nicht interpretierbaren Funktionsaufrufs-Teile an.	
CB-DB-1026	Unbekannte Funktion XXX in Tabelle A, Zeile B, Spalte C bei Index D
Es wurde versucht, eine für checkerberry db unbekannte Funktion aufzurufen. Entweder der Name der Funktion ist falsch geschrieben oder es handelt sich um eine nutzerdefinierte Funktion, die noch nicht registriert wurde (siehe Beispiel 2.31, „Registrieren der AddFunction“).	
CB-DB-1027	Ungültige Konfiguration: XXX darf kein Prefix von YYY enthalten.
Bei der ersten Verwendung einer neuen Syntax für Parameter oder Funktionsaufrufe wird diese auf Konsistenz hin überprüft. Diese Fehlermeldung besagt, dass dabei ein Fehler festgestellt wurde und gibt die beiden kollidierenden Syntax-Elemente an, um eine Rekonfiguration zu ermöglichen.	
CB-DB-1030	Validierung des Trennzeichens für Funktionsargumente fehlgeschlagen (X): muss ein einzelnes Zeichen sein.
Das Trennzeichen für Funktionsargumente muss ein einzelnes Zeichen sein.	
CB-DB-1031	Validierung des Maskierungszeichens fehlgeschlagen (X): muss ein einzelnes Zeichen sein.
Das Maskierungszeichen muss ein einzelnes Zeichen sein.	
CB-DB-1032	Validierung der Funktion fehlgeschlagen: Darf nicht ‚null‘ sein.
Die zu registrierende Funktion darf nicht ‚null‘ sein.	
CB-DB-1033	Validierung des Parameter-Prefix fehlgeschlagen: Darf nicht ‚null‘ und nicht leer sein.
Der Parameter-Prefix darf nicht ‚null‘ und nicht leer sein.	

CB-DB-1034	Validierung des Parameter-Suffix fehlgeschlagen: Darf nicht ‚null' und nicht leer sein.
Der Parameter-Suffix darf nicht ‚null' und nicht leer sein.	
CB-DB-1035	Validierung des Funktionsaufruf-Prefixes fehlgeschlagen: Darf nicht ‚null' und nicht leer sein.
Der Funktionsaufruf-Prefix darf nicht ‚null' und nicht leer sein.	
CB-DB-1036	Validierung der öffnenden Funktionsaufruf-Klammer fehlgeschlagen: Darf nicht ‚null' und nicht leer sein.
Die öffnende Klammer für Funktionsaufrufe darf nicht ‚null' und nicht leer sein.	
CB-DB-1037	Validierung der schließenden Funktionsaufruf-Klammer fehlgeschlagen: Darf nicht ‚null' und nicht leer sein.
Die schließende Klammer für Funktionsaufrufe darf nicht ‚null' und nicht leer sein.	
CB-DB-1038	Argumente der Funktion sollten dem Format „+3 days" oder „-2 years" entsprechen. XXX ist kein gültiges Argument.
Die Funktion `->now()` oder `->today()` wurde mit einem ungültigen Argument aufgerufen. Ein häufiger Fehler ist das Einfügen von Leerzeichen vor oder nach den Argumenten.	
CB-DB-1039	Die Funktion XXX in Tabelle A, Zeile B, Spalte C benötigt Argumente.
Die Funktion XXX muss mit mindestens einem Argument aufgerufen werden.	
CB-DB-1040	DbUnit hat eine Exception geworfen.
Kapselt eine Exception des DbUnit Frameworks.	
CB-DB-1041	Der Bezeichner XXX kann nicht für die Tabelle YYY verwendet werden.
Es wurde ein ungültiger Bezeichner für eine konfigurierbare Tabelle gewählt. Dies muss in der DbConfigurationCallback geändert werden.	
CB-DB-1042	Wenn für eine Klasse Initialdaten definiert sind, darf die Annotation ClearTables nicht an dieser Klasse gesetzt werden.
Wenn Initialdaten für eine Klasse vorliegen, darf diese Klasse nicht mit ClearTables annotiert werden. Es ist hingegen zulässig, eine Methode mit ClearTables zu annotieren, wenn nur für die Klasse Initialdaten vorliegen und umgekehrt.	

CB-DB-1043	Wenn für eine Methode Initialdaten definiert sind, darf die Annotation ClearTables nicht an die Methode gesetzt werden.
Wenn Initialdaten für eine Methode vorliegen, darf diese Methode nicht mit ClearTables annotiert werden. Es ist hingegen zulässig, eine Methode mit ClearTables zu annotieren, wenn nur für die Klasse Initialdaten vorliegen und umgekehrt.	
CB-DB-1044	Für die Tabelle XXX sind sowohl Daten als auch ein EMPTY_TABLE Tag definiert.
Das EMPTY_TABLE Tag soll genutzt werden, um explizit anzugeben, dass eine Tabelle leer ist. Wenn zusätzlich Daten für diese Tabelle definiert wurden, deutet das auf einen fachlichen Fehler hin. Wenn Dateien inkludiert werden, ist es möglich, dass in der einen Datei ein Datensatz für eine Tabelle definiert wurde, die in einer anderen Datei als leer markiert ist.	
CB-DB-1048	Die DTD Datei 'XXX' konnte nicht gelesen werden.
Diese Meldung erscheint, wenn beim Lesen der angegebenen DTD Datei ein Fehler aufgetreten ist.	
CB-DB-1049	Der Auto-Parameter "XXX" hat bereits den Wert "YYY". Der neue Wert "ZZZ" kann nicht zugeordnet werden.
Diese Meldung erscheint, wenn der Wert eines Autoparameters nicht eindeutig ermittelt werden kann.	
CB-DB-1051	Für die Tabelle "XXX" und die Spalte "YYY" wurde kein aktiver Validator gefunden.
Diese Meldung erscheint, wenn für eine Spalte kein aktiver Validator gefunden wurde.	
CB-DB-1052	Für die Tabelle "XXX" und die Spalte "YYY" wurde ein unbekannter Validator definiert: "ZZZ".
Diese Meldung erscheint, wenn für die Spalte YYY der Tabelle XXX ein Validator mit der Id ZZZ verwendet werden soll, der an der Konfiguration nicht registriert wurde.	
CB-DB-1053	Die Reihenfolge der Tabellen innerhalb der XML-Testdaten entspricht nicht der Reihenfolge aus der DTD. Die Reihenfolge muss lauten [XXX-1, XXX-2, ..., XXX-n].
Diese Fehlermeldung erscheint, wenn die Reihenfolge der Tabellen in den Testdaten nicht der Reihenfolge der Tabellen in der DTD entspricht. Die Tabellenreihenfolge in der DTD legt fest, in welcher Reihenfolge die Tabellen in die Datenbank eingespielt werden. Diese Reihenfolge ist wichtig, da sie Fremdschlüsselbeziehungen berücksichtigt und somit verhindert, dass Constraint-Verletzungen beim Einspielen der Testdaten auftreten. Diese Fehlermeldung listet die Tabellen in der Reihenfolge auf, in der sie angegeben werden müssen. Die Liste enthält dabei nur Tabellen, die tatsächlich in den fehlerhaften Testdaten verwendet wurden.	

CB-DB-1054	Es wurde versucht innerhalb einer Read-Only-Verbindung schreibend auf die Datenbank zuzugreifen.
Diese Fehlermeldung erscheint, wenn schreibend auf eine JDBC-URL zugegriffen wurde, die als Read-Only konfiguriert ist.	
CB-DB-1055	Bitte geben Sie in der Konfiguration eine JDBC-URL (z.B. "jdbc:hsqldb:mem:sample-test") ein, wenn sie den Standard JDBC Database-Connector verwenden.
Diese Fehlermeldung erscheint, wenn die Standard-Implementation des `DatabaseConnectors` verwendet wird, ohne dass in der Konfiguration die JDBC-URL der zu verwendenden Datenbank angegeben wurde. Ohne diese URL ist checkerberry db nicht in der Lage, eine Verbindung zu der gewünschten Datenbank herzustellen.	
CB-DB-1056	Bitte geben Sie in der Konfiguration einen JDBC-Treibernamen (z.B. "org.hsqldb.jdbcDriver") ein, wenn sie den Standard JDBC Database-Connector verwenden.
Diese Fehlermeldung erscheint, wenn die Standard-Implementation des `DatabaseConnectors` verwendet wird, ohne dass in der Konfiguration der zu verwendenden Datenbanktreiber angegeben wurde. Ohne den Treiber ist checkerberry db nicht in der Lage, eine Verbindung zu der gewünschten Datenbank herzustellen.	
CB-DB-1057	Der Datenbankwert "XXX" konnte nicht in einen String konvertiert werden.
Dieser Fehler tritt auf, wenn die Konvertierung eines Datenbankinhalt in einen String fehlschlägt, z.B. bei der Konvertierung von Binärdaten zur Erstellung eines Datenbank-Dumps.	
CB-DB-1058	Die folgenden Autoparameter konnten nicht aufgelöst werden: XXX.
Dieser Fehler tritt auf, wenn eine Reihe von Autoparametern nicht ermittelt werden konnten. Die nicht ermittelbaren Autoparameter werden als Liste XXX angegeben.	
CB-DB-1059	Die Reihenfolge der Datenbanktabellen konnte aufgrund ungültiger Abhängigkeiten nicht berechnet werden (siehe LOG(error)).
Dieser Fehler tritt auf, wenn eine neue DTD berechnet werden soll und die Datenbanktabellen ungültige, in der Regel zyklische, Abhängigkeiten haben. In der Log-Datei finden sich weitere Informationen zu den Tabellen und ihren Abhängigkeiten.	
CB-DB-1060	Die Anzahl der Spalten ('TABLE') für den Primärschlüssel ist unterschiedlich OLD! =NEW
Der Fehler tritt auf, wenn beim Zusammenfügen von Tabellendaten unterschiedliche Metadaten für eine Tabelle ermittelt wurden. Die Anzahl der Spalten (OLD und NEW) für den Primärschlüssel unterscheidet sich bei der Tabelle 'TABLE'.	

CB-DB-1061	Die Spalten ('TABLE') passen nicht zueinander 'OLD'!='NEW'
Der Fehler tritt auf, wenn beim Zusammenfügen von Tabellendaten unterschiedliche Metadaten für eine Tabelle ermittelt wurden. Die Informationen der vorhanden Spalten der Tabelle 'TABLE' passen nicht zueinander. Die Liste der Spaltennamen (OLD und NEW) werden als Liste ausgegeben.	
CB-DB-1062	Die Spalten ('TABLE') passen nicht zueinander. Das Attribut 'COLUMN.ATTR' hat unterschiedliche Werte: 'OLD'!='NEW'
Der Fehler tritt auf, wenn beim Zusammenfügen von Tabellendaten unterschiedliche Metadaten für eine Tabelle ermittelt wurden. Dabei ist es zu Abweichungen des Attributes ATTR der Spalte COLUMN in der Tabelle TABLE gekommen.	
CB-DB-1063	Der Vorgang wurde nach XXX Durchläufen abgebrochen.
Der Fehler tritt auf, wenn die Anzahl der Durchläufe den eingestellten Maximalwert (XXX) überschritten haben. In diesem Fall sollte man entweder die Auswahl einschränken oder die erlaubte Anzahl erhöhen (siehe Beispiel 2.43, „Erhöhen der maximalen Anzahl der Durchläufe“).	

6.3. Checkerberry web

Dieses Kapitel listet eine Reihe von häufig gestellten Fragen und deren Antworten auf.

6.3.1. Wie kann ich checkerberry web in Hudson integrieren?

Checkerberry web benötigt für die Ausführung der Tests einen Browser und somit auch eine grafische Oberfläche. Diese Vorbedingung ist bei einer Hudson-Installation auf Windows-Systemen automatisch gegeben. Bei der Installation auf Linux-Systemen sieht dies anders aus. Für dieses Problem gibt es unterschiedliche Lösungen. Eine Möglichkeit besteht in der Registrierung verschiedener Windows-Rechner als Hudson-Slaves, die die Tests durchführen. Dadurch wird zusätzlich die Last verteilt. Des Weiteren können verschiedene Windows-Betriebssysteme verwendet werden.

Eine andere Möglichkeit besteht in der Bereitstellung eines X-Servers. Für den Betrieb auf Linux-Servern empfiehlt sich der Einsatz von `xvfb` (siehe auch [Blog alittlemadness.com, 2010]) oder `vncserver`.

Das beschriebene Vorgehen, lässt sich entsprechend auf den Jenkins übertragen.

6.3.2. Wie kann ich Portal-Anwendungen testen?

In Portal-Anwendungen ist es möglich, mehrere Portlets auf einer Webseite anzuzeigen. Da insbesondere auch das gleiche Portlet mehrfach auf der Webseite angezeigt werden kann, muss die Portal-Anwendung die Eindeutigkeit der HTML-IDs sicherstellen. Zu diesem Zweck wird jeder Portlet-Instanz in der Regel eine dynamische ID zugeordnet, die als Präfix für die HTML-IDs verwendet wird. Die Vergabe der Portlet-ID kann während des Deployments oder während des Starts des Application-Servers erfolgen. Insbesondere ist diese Portlet-ID zum Zeitpunkt der Testerstellung nicht verfügbar. Wie kann man in diesem Umfeld GUI-Tests programmieren?

Checkerberry web bietet den großen Vorteil des Modellansatzes, über den die erforderliche Implementierung realisiert werden kann. Die Lösung ist dabei von der konkreten Anwendung und Portal-Implementierung abhängig. Dennoch lässt sich ein allgemeiner Lösungsweg beschreiben.

Die wesentliche Aufgabe bei dem Test von Portal-Anwendung besteht darin, die Zuordnung zwischen Portlet und Portlet-ID herzustellen. Zu diesem Zweck bietet checkerberry web die abstrakte Klasse `AbstractRemoteControlPortlet`. Für jedes Portlet der Anwendung wird eine eigene Portlet-Klasse implementiert z.B. `LoginPortlet`. In dieser Klasse implementiert der Entwickler Getter-Methode für alle Webseiten, die zu diesem Portlet gehören z.B. `public LoginPage getLoginPage()`. Die abstrakte Portlet-Klasse verfügt über das Property `portletId`. Dieses Property kann über die enthaltenen Page-Modellklassen an die Komponenten weitergereicht werden.

Das folgende Beispiel zeigt eine Implementierung der `LoginPage` im Portal-Umfeld. Im Gegensatz zu dem vorherigen Beispiel enthält die `LoginPage`-Klasse nun eine Referenz auf das Portlet, in dem die Seite angezeigt wird. Über die Portlet-ID kann dann die HTML-ID der enthaltenen Komponenten dynamisch erzeugt werden.

```
public class LoginPage extends AbstractRemoteControlPage {

  private AbstractRemoteControlPortlet portlet;

  // Konstruktor mit Contextprovider und Portlet aufrufen
  public LoginPage(ContextProvider contextProvider, AbstractRemoteControlPortlet portlet) {
      super(contextProvider);
      this.portlet = portlet;
  }

  public RemoteControlComponent getUserTextField() {
    // Portlet Präfix bei der HTML-ID berücksichtigen.
    return createComponentProxy(portlet.getPortletId() + ":userId");
  }
}
```

Beispiel 6.5. Login-Page-Modell im Portal-Umfeld

Da das LoginPage-Modell über das Portlet erzeugt wird, kann die Referenz auf das Portlet leicht übergeben werden. Das folgende Beispiel zeigt eine mögliche Implementierung.

```
public class LoginPortlet extends AbstractRemoteControlPortlet {

  // Konstruktor mit Fernsteuerung und ContextProvider aufrufen
  public LoginPortlet(ContextProvider contextProvider) {
    super(contextProvider);
  }

  public LoginPage getLoginPage() {
    // LoginPage mit Referenz auf dieses Portlet aufrufen. (Das
    // Portlet ist auch ein ContextProvider...)
    return new LoginPage(this);
  }
}
```

Beispiel 6.6. Beispiel-Implementierung Login-Portlet-Modell

Das Portlet-Modell wird mit dem aktuellen ContextProvider erzeugt. Über Getter-Methoden werden die Modelle der Webseiten zur Verfügung gestellt, die innerhalb des Portlets verwendet werden. Bei der Erzeugung dieser Modelle wird die Referenz auf das entsprechende Portlet einfach übergeben.

Das vorgestellte Verfahren bietet eine gute Möglichkeit, Modelle in Portal-Umgebungen zu definieren. Die entscheidende Frage ist bisher jedoch unbeantwortet geblieben: Wie kann die Portlet-ID für ein Portlet ermittelt werden?

Eine Möglichkeit für die Ermittlung der Zuordnung von Portlet und Portlet-ID besteht in der Erweiterung der HTML-Datei. Über ein zusätzliches oder bestehendes HTML-Tag kann die Information in die Webseite eingebunden werden. Es ist z.B. möglich, ein `DIV`-Element um jedes Portlet

zu definieren, das einen Portlet-Namen und die Portlet-ID in der HTML-ID enthält z.B. `<DIV ID="PORTLET:NAME=Login;ID=XYZ4711">`. Der vollständige Inhalt der HTML-Seite kann über die Methode `RemoteControl.getHtmlSource()` ermittelt werden, sodass die Zuordnung ausgelesen und interpretiert werden kann.

Es ist abhängig von der Portal-Anwendung, wann die Zuordnung von Portlet und Portlet-ID ausgelesen und verarbeitet wird. Aus Performance-Gründen kann es sinnvoll sein, die HTML-Seite einmal einzulesen und in die ermittelten Zuordnungen in der Klasse `WebContext` zu speichern. In anderen Portal-Anwendungen kann es sinnvoller sein, die Zuordnung direkt bei der Erzeugung des Portal-Modells vorzunehmen.

6.3.3. Warum erscheint bei clickAndWaitForPage ein Timeout?

Warum tritt bei dem Aufruf von `clickAndWaitForPage` immer ein Timeout auf, obwohl die Seite angezeigt wird? Die Methode `clickAndWaitForPage` wird verwendet, wenn nach dem Klicken einer Komponente eine komplette Seite neu geladen wird. Der Browser versendet einen HTTP-Request und empfängt eine HTTP-Response. Nachdem die Seite vollständig geladen wurde, wird der Aufruf der Methode `clickAndWaitForPage` beendet. Die Methode erkennt jedoch nicht, wenn die Seite durch einen Ajax-Request geladen wurde. Dynamische Änderungen des DOM-Baums durch JavaScript werden ebenfalls nicht als Neuladen der Seite erkannt. In diesen Situationen muss die Methode `clickAndWait` verwendet werden.

Alternativ kann auch explizit auf die Existenz konkreter Felder gewartet werden z.B. `userField.waitUntilAvailable()`. Dieses Vorgehen hat zum einen den Vorteil, dass die Tests schneller durchlaufen, da keine unnötigen Wartezeiten verwendet werden. Zum anderen sind die Tests auch robuster. Dies liegt daran, dass die Verwendung von Timeouts auf unterschiedlichen Plattformen zu Problemen führt. Die Timeouts müssen so klein wie möglich sein, um die Testlaufzeiten nicht zu stark zu verzögern. Auf der anderen Seite müssen die Timeouts groß genug sein, damit die Tests auf allen Umgebungen korrekt ausgeführt werden. Es kommt daher häufig vor, dass die Timeouts zu gering gewählt werden und Tests fehlschlagen, obwohl die Ausführung langsamer ist. Das passiert bei dem expliziten Warten auf Komponenten nicht.

6.3.4. Warum laufen die Tests im Internet Explorer nicht?

Wenn GUI-Tests im Firefox Browser erfolgreich verlaufen und im Internet Explorer nicht, kann das verschiedene Ursachen haben. In der Regel hat Selenium beim Internet Explorer Schwierigkeiten mit Funktionen, die das native Key-Handling betreffen. Wenn also der Aufruf `typeKeys()` an einer Komponente des Page Models fehlschlägt, ersetzen Sie ihn durch `type()`. Selenium arbeitet mit JavaScript (siehe Abschnitt 3.2, „Funktionsweise"). Der Internet Explorer blockiert diese in der Standard Einstellung. Wenn Sie den geschützen Modus deaktivieren, kann Selenium per JavaScript auf die Application under Test zugreifen. Alternativ ist es auch ausreichend, die zu testende URL als vertrauenswürdige Seite in den Einstellungen des Internet Explorers aufzunehmen.

6.3.5. Welche Fehlermeldungen gibt es in checkerberry web?

Tabelle 6.3. Fehlermeldungen in checkerberry web

CB-WEB-1000	Die Durchführung der Aktion ist nicht möglich, da der Bildschirmschoner aktiv ist.
Diese Fehlermeldung erscheint, wenn eine Aktion durchgeführt werden soll, die bei aktivem Bildschirmschoner nicht ausgeführt werden kann. Dies ist z. B. der Fall, wenn ein nativer Tastendruck ausgeführt werden soll. Bei einem aktiven Bildschirmschoner erreicht der Tastendruck nie die zu testende Webseite, sodass diese Fehlermeldung die Ursache für den fehlschlagenden Test angibt.	

CB-WEB-1001	Die Testklasse 'XXX' ist unbekannt.
Diese Fehlermeldung erscheint, wenn bei der Erzeugung der checkerberry web-Umgebung auf einen Testklassenname `XXX` zugegriffen wird, der unbekannt ist.	
CB-WEB-1002	Die Testmethode 'XXX' der Testklasse 'YYY' ist unbekannt.
Diese Fehlermeldung erscheint, wenn auf eine Testmethode `YYY` der Testklasse `XXX` zugegriffen wird und die Testklasse diese Methode nicht enthält oder der Zugriff auf diese Methode nicht möglich ist.	
CB-WEB-1004	Selenium hat eine Exception geworfen.
Kapselt eine Exception des Selenium Frameworks.	
CB-WEB-1007	Das checkerberry web-Environment ist bereits aktiv. Bitte prüfen Sie, ob die `setUp`-Methode zweimal aufgerufen wurde oder ein `tearDown`-Aufruf fehlt.
Ein checkerberry web-Environment muss vor jeder Testmethode über den Aufruf der Methode `setUp` initialisiert werden und nach jeder Testmethode über den Aufruf `tearDown` runtergefahren werden. Anderenfalls erscheint die angegebene Fehlermeldung.	
CB-WEB-1008	Für den Browsertyp XXX wurde kein WebDriver-Creator registriert.
Für die Verwendung von WebDriver in checkerberry web ist es erforderlich, dass ein Creator zur Erzeugung der entsprechenden WebDriver-Instanz konfiguriert ist. Für Firefox, Internet Explorer und Google Chrome sind die entsprechenden Creator-Instanzen schon registriert. Wenn ein Browser mit WebDriver verwendet werden soll, der über keine Creator-Instanz verfügt, wird diese Fehlermeldung ausgegeben.	

6.3.6. Warum Selenium RC als Default-Fernsteuerung?

Bei der Verwendung von Selenium gibt es zwei Möglichkeiten, die Kommunikation (Fernsteuerung) mit dem Browser umzusetzen: Selenium RC (Remote Control) und WebDriver. Während Selenium RC die ursprüngliche Implementierung von Selenium ist, wurde WebDriver über Google in das Projekt eingebracht. Es wird vermutet, dass WebDriver die einzige zukünftige Lösung sein wird. Aktuell ist es jedoch so, dass Selenium RC und WebDriver sich sehr gut ergänzen. Dies sieht man gerade auch bei der Browser-Unterstützung (siehe [Selenium Platforms, 2012]). Da Selenium RC weitere Vorteile hat (mehrere Browser-Instanzen pro Test, umfangreichere API, Unterstützung älterer Browser) wurde die Umstellung der internen Fernsteuerung nicht vorgenommen.

6.3.7. Kann ich mein Modelle für Selenium RC und WebDriver verwenden?

Ja. Theoretisch ist es problemlos möglich, die verwendeten Modellklassen sowohl mit Selenium RC und WebDriver zu verwenden. In der Praxis ist es jedoch so, dass WebDriver eine eingeschränktere API als Selenium RC hat. Aus diesem Grund kann es passieren, dass Funktionen verwendet werden, die in Selenium RC zu dem gewünschten Ergebnis führen, während sie unter WebDriver nicht oder anders funktionieren. In der Konfiguration von checkerberry web kann eingestellt werden, dass Selenium RC und WebDriver sich möglichst gleich verhalten sollen (Kompatibilitätsmodus), damit das Umschalten der Fernsteuerung möglichst einfach ist. Per Default ist der Kompatibilitätsmodus aktiviert. Wenn das ursprüngliche Verhalten

der ausgewählten Implementierung gewünscht ist, muss dieser Kompatibilitätsmodus explizit deaktiviert werden.