



Benutzer-Handbuch

Release 1.6.1

Inhaltsverzeichnis

1. Benutzer-Handbuch	3
1.1 Test-Editor starten	4
1.2 Test-Editor bedienen	5
1.3 Demo-Projekte erzeugen	8
1.4 Projekte anlegen und konfigurieren	14
1.5 Mit Testsuiten gruppieren	18
1.6 Testfälle anlegen und bearbeiten	22
1.7 Mit Szenariensuiten arbeiten	31
1.8 Szenarien in Testfällen verwenden	35
1.9 Besonderheiten bei der Verwendung von Szenarien in Szenarien	41
1.10 Tests ausführen	42
1.11 Testhistorie anzeigen	46
1.12 Projekte im Team teilen	50
2. Glossar	56
2.1 Akzeptanztest	58
2.2 AUT	59
2.3 Element-Liste	60
2.4 Fachsprache (DSL)	61
2.5 FitNesse	62
2.6 Fixture	63
2.7 Masken und Testschritte	64
2.8 Port	65
2.9 RAP	66
2.10 REST	67
2.11 SOAP	68
2.12 Szenario	69
2.13 Szenariosuite	70
2.14 Test	71
2.15 Testfall	72
2.16 Testlog	73
2.17 Testsuite	74
2.18 Test-Treiber	75
2.19 Widget	76
2.20 XML	77
3. Release Notes	78

Benutzer-Handbuch

Der *Test-Editor* ist ein Open Source Projekt zur Erfassung und automatischen Ausführung von [Akzeptanztests](#). Die Anwendung stellt eine intuitiv zu bedienende Oberfläche bereit, so dass [Testfälle](#) auch ohne Entwickler Know-how erfasst werden können. Als Unterbau (Backend) wird das Open-Source Framework [FitNesse](#) genutzt. Der *Test-Editor* ist ein Open-Source Projekt und wird u.a. von der [akquinet](#) und der [Signal Iduna](#) entwickelt.

Das Benutzer-Handbuch beschreibt die Funktionsweise der grafischen Oberfläche, während das Administrator-Handbuch technische Details beinhaltet, um den Test-Editor an individuelle Projekte anzupassen.

Da der *Test-Editor* auf verschiedenen Systemen entwickelt wird, können die eingesetzten Screenshots vom Aussehen aus voneinander abweichen. Inhaltlich wird die Anwendung auf Windows, Mac und Linux aber identisch reagieren, was durch die unterschiedlichen Entwicklungssysteme sichergestellt wird.

Test-Editor starten

Der *Test-Editor* stellt eine intuitive Anwendung bereit, die das Erfassen und Ausführen von [Akzeptanztests](#) ermöglicht. Es können also Oberflächen aus einer fachlichen Sichtweise beschrieben und getestet werden. Ein Wissen über die Implementierung der jeweiligen Anwendung ist dabei nicht notwendig.

Starten des Test-Editors

Gestartet wird der Test-Editor über die **Startdatei (Windows: `testeditor.exe`, Linux: `testeditor`, Mac OS: `testeditor.app`)**. Wird der *Test-Editor* das erste Mal gestartet, wird automatisch ein Workspace im Homeverzeichnis des Benutzers angelegt. Der Workspace ist unter dem Namen "**test editor**" zu finden und liegt in Windows Betriebssystemen z.B. unter

C:\Users\MaxMustermann\.testeditor

Browserkonfiguration bei Tests gegen eine Web-Anwendung

Für die Ausführung von [Testfällen](#), die einen Browser benötigen (z.B. im Projekt DemoWebTests), wird ein bestimmter Browser benötigt. Der *Test-Editor* startet auch ohne diesen Browser, es können dann aber keine Tests, die einen Browser fernsteuern, ausgeführt werden. Es ergeben sich folgende Möglichkeiten:

- **Nutzung des Internet Explorers (Windows):** Die Beispiele erwarten derzeit den Firefox Browser, dies kann auf den Windows Internet Explorer (IE) umgestellt werden, so dass die lokale Installation des IE genutzt werden kann. Hierfür muss der jeweilige Testschritt "*starte Browser Firefox*" in "*starte Browser IE*" geändert werden (im Fall der DemoWebTests ist diese Zeile in dem Test-Szenario *ApplikationStartSzenario* zu finden).
- **Nutzung des Firefox Portable (alle Plattformen):** Der Firefox Portable muss in einer bestimmten Version vorliegen, es ist keine Installation notwendig. Ausgeliefert wird ein ZIP-File neben dem *Test-Editor*, welches die richtige Version für alle drei Plattformen beinhaltet. Der Pfad zu dem entpackten ZIP muss über die Einstellungen des *Test-Editors* angepasst werden.

Die notwendige Firefox-Version kann [hier](#) heruntergeladen werden.

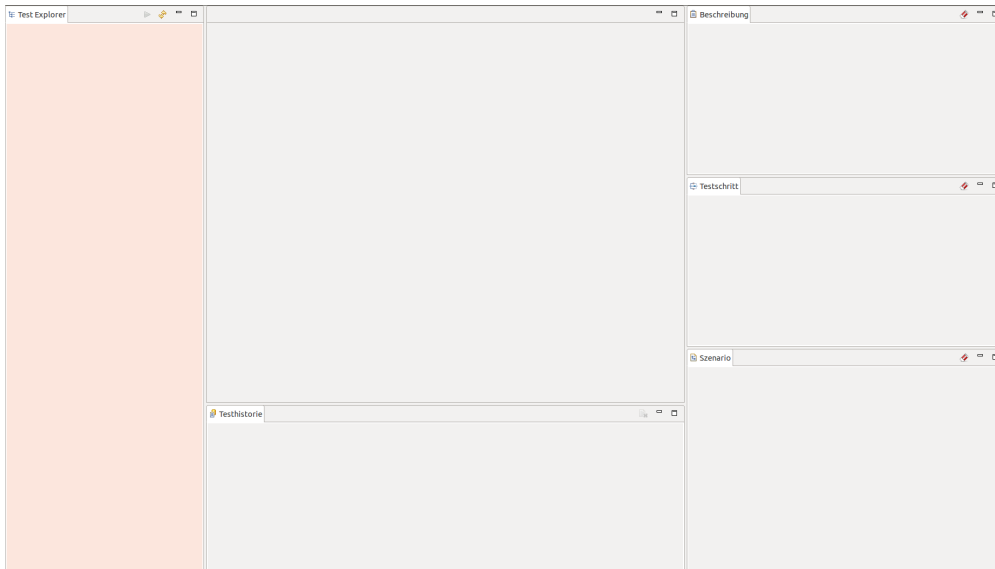


Installation auf iOS

Der Pfad führt für Windows und Linux zur ausführbaren Datei (unter Windows z. B. `Firefox.exe`). Beim Mac muss man tiefer verzweigen, z. B. `../Firefox.app/Contents/MacOS/firefox-bin` (Firefox.app ist die heruntergeladene Datei).

Startbildschirm

Der *Test-Editor* öffnet sich und es wird folgender Startbildschirm angezeigt:



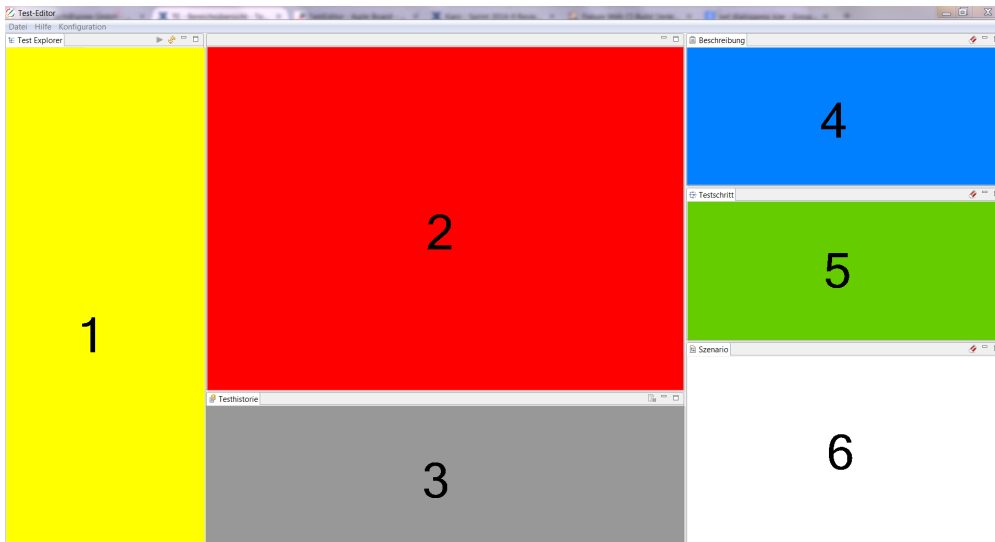
Test-Editor bedienen

Der *Test-Editor* startet nach dem ersten Aufruf ohne jegliche Projekte und Testfälle. Um die Applikation besser kennenzulernen, bietet es sich an zunächst die Demo-Projekte zu erzeugen oder ein eigenes Projekt anzulegen (vgl. das folgende Kapitel).

Mit Bereichen arbeiten

Der *Test-Editor* ist dabei in folgende fünf Bereiche eingeteilt:

- **Test-Explorer (Bereich 1):** Der Explorer bildet die Auswahlmöglichkeit der einzelnen Projekte, [Szenarien](#), [Testfälle](#) und [Testsuiten](#) in einer hierarchischen Struktur ab. Im Kontext zu dem jeweiligen Element, können verschiedene Aktionen ausgeführt werden (z.B. Kind-Elemente erzeugen, [Tests](#) ausführen, Einträge löschen usw.).
- **Editor (Bereich 2):** Durch einen Doppelklick im Test-Explorer wird das entsprechende Element im mittleren Bereich angezeigt. Der Screenshot zeigt den Testfall *LoginValidTest*. Durch die unterschiedliche Text hervorhebung werden die jeweiligen Testschritte dargestellt. Der Testfall kann entsprechend editiert werden (eine einzelne Zeile wird z.B. durch einen Doppelklick editiert). Wird ein Projekt im Test-Explorer ausgewählt, entsteht im Editor Bereich eine andere Sicht, nämlich die um ein Projekt zu konfigurieren.
- **Testhistorie (Bereich 3):** In diesem Bereich wird eine Historie bzgl. der Testausführung zu einem Testfall oder einer Testsuite angezeigt.
- **Beschreibung (Bereich 4):** Über diesen Bereich wird eine bestehende Beschreibung innerhalb eines Testfalls editiert oder eine neue Beschreibung hinzugefügt.
- **Testschritt (Bereich 5):** Analog zu der Beschreibung, kann in diesem Bereich ein Testschritt editiert bzw. zum Testfall hinzugefügt werden.
- **Szenario (Bereich 6):** Über diesen Bereich können Szenarien in einem Testfall verwendet werden.



Folgende **Elemente** werden im Test-Explorer unterstützt:

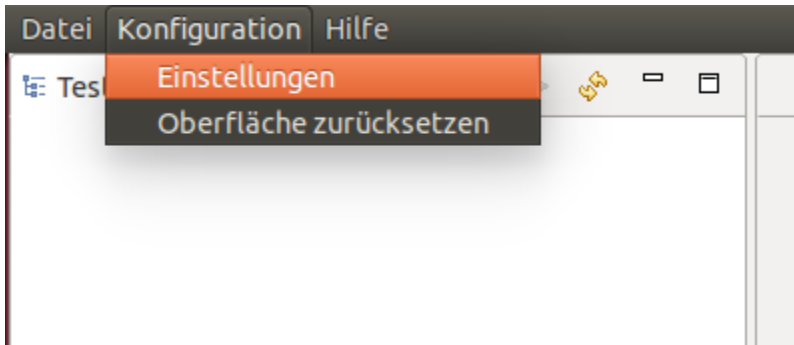
- **Projekte** beinhalten in der Regel mehrere Testsuiten und die Projekteinstellungen, sie sind die oberste Ebene im Test-Explorer
- **Testsuiten** gruppieren Testfälle bzw. untergeordnete Test-Suiten
- **Testfälle** können keine weiteren untergeordneten Elemente (Kindelemente) enthalten und beinhalten die eigentlichen Testabläufe
- **Szenariosuite** ist ein Gruppierungsobjekt, um Szenarien zu gruppieren
- **Szenario** sind wiederverwendbare Abschnitte von [Testschritten](#), die in Testfällen universell wiederverwendet werden können und können somit auch als Vorlage verwendet werden

In den folgenden Kapiteln werden die einzelnen Funktionen detailliert beschrieben, das Projekt *DemoWebTests* wird dabei als Beispiel verwendet.

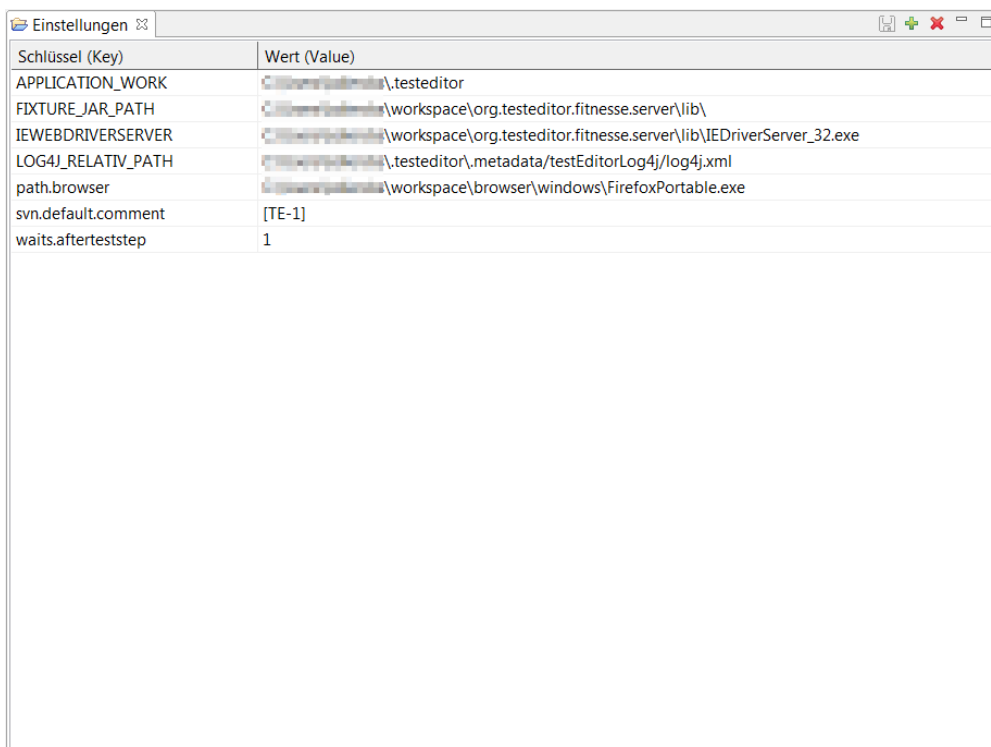
Globale Einstellungen vornehmen

Die globalen Einstellungen ermöglichen es, den Test-Editor auf ein bestimmtes Projekt anzupassen. Hier werden z. B. der Pfad zum individuellen Projekt-Java-Archiv hinterlegt, der Pfad zum Browser, der in den Tests aufgerufen werden soll usw.

Die globalen Einstellungen können im Test-Editor im Menü Konfiguration unter Einstellungen geöffnet werden.




Die globalen Einstellungen werden dann in Form einer Liste angezeigt. Die Liste kann bearbeitet werden, in dem man in die Tabellenzelle klickt, die man ändern möchte. Neue Einträge können mit Hilfe des Hinzufügen-Buttons (grünes Kreuz) hinzugefügt werden.



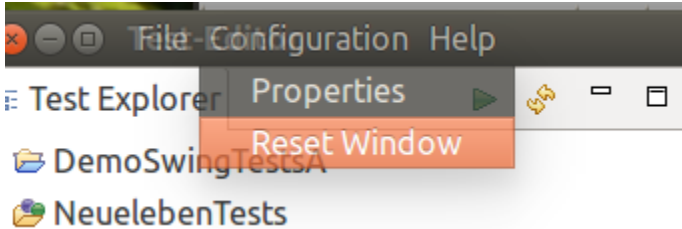
Schlüssel (Key)	Wert (Value)
APPLICATION_WORK	C:\Users\Andreas\workspace\org.testeditor
FIXTURE_JAR_PATH	C:\Users\Andreas\workspace\org.testeditor.fitness.server\lib\
IEWEBDRIVERSERVER	C:\Users\Andreas\workspace\org.testeditor.fitness.server\lib\IEDriverServer_32.exe
LOG4J_RELATIV_PATH	C:\Users\Andreas\workspace\org.testeditor\metadata\testEditorLog4j\log4j.xml
path.browser	C:\Users\Andreas\workspace\browser\windows\FirefoxPortable.exe
svn.default.comment	[TE-1]
waits.underteststep	1

Test-Editor Oberfläche zurücksetzen

Die Oberflächen Elemente, wie der Test-Explorer, Test-Historie und Testschritte editieren und andere können flexibel angeordnet werden.

 Beim Starten des Test-Editors werden alle im Editor-Bereich geöffneten Reiter aufgebaut. Bei vielen Reitern oder großen Testfällen und Szenarien kann dies die Startzeit des Test-Editors negativ beeinflussen. Dies lässt sich durch schließen von nicht mehr benötigten Reitern verbessern.

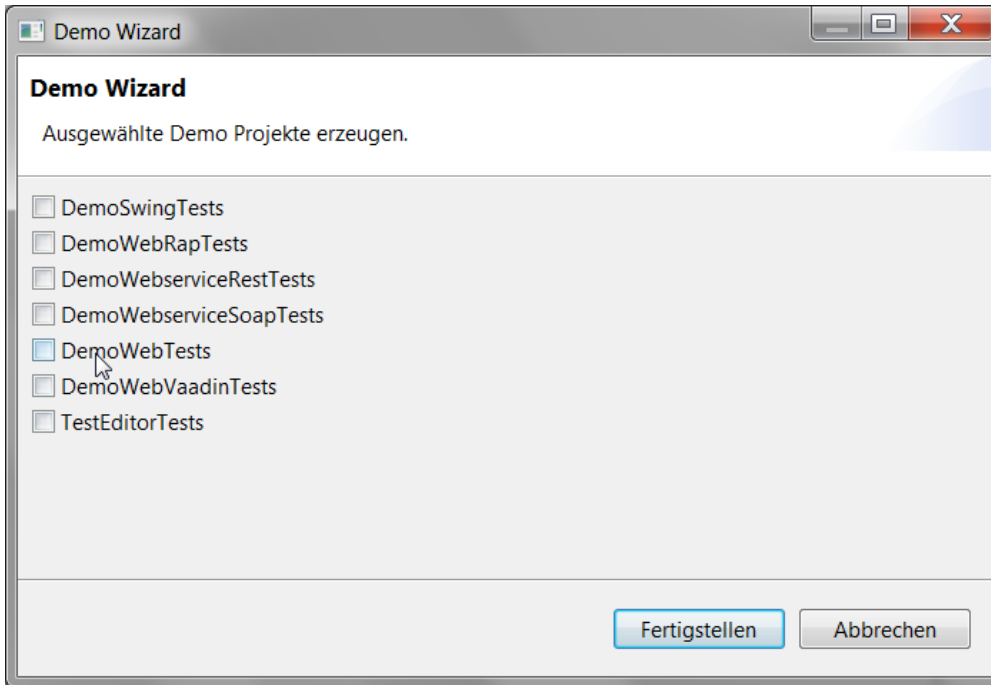
Der Benutzer kann den Initialzustand wieder herstellen, indem er die Aktion Oberfläche zurücksetzen im Menü Konfiguration aufruft.



Der erforderliche Neustart der Anwendung wird daraufhin automatisch vom System ausgeführt. Sollten noch geänderte Objekte, wie z. B. Testfälle, geöffnet sein, erscheint ein Dialog, in dem die zu speichernden Objekte ausgewählt werden können. Die Speicherung erfolgt dann vor dem Neustart des Test-Editors.

Demo-Projekte erzeugen

Über das Hauptmenü **Datei -> Demo-Projekte erzeugen** wird der Wizard zum Erzeugen von Beispiel-Projekten geöffnet. Im Wizard kann man durch das Anklicken der jeweiligen Checkboxes auswählen, welche Demo-Projekte erzeugt werden sollen. Die nicht erzeugten Projekte können beim erneuten Aufruf des Wizards ggf. nachträglich erzeugt werden. Bereits erzeugte Projekte werden für die Auswahl deaktiviert.



Die Demo-Projekte dienen als Einstieg zum Testen von unterschiedlichen Anwendungen. Folgende Projekte werden angeboten:

- DemoSwingTests
- DemoWebRapTests
- DemoWebserviceRestTests
- DemoWebserviceSoapTests
- DemoWebTests
- DemoWebVaadinTests
- TestEditorTests



Hinweis auf das Administrator-Handbuch

Im Administrator-Handbuch ist beschrieben, welche Schritte notwendig sind um die **Browser-Unterstützung** im *Test-Editor* einzustellen, dies ist wichtig für alle Demo-Projekte, die eine Web-Anwendung fernsteuern.

DemoSwingTests

Dieses Projekt zeigt, wie **Swing Anwendungen** (Java) automatisiert getestet werden können. Als Beispiel dient eine Anwendung zur Verwaltung von Geburtstagen. Die "Geburtstagslisten-Verwaltung" ist bereits Teil des Demos und muss nicht extra installiert werden, folgender Screenshot zeigt die Beispiel-Anwendung:

Name	Vorname	Geburtsdatum	Geschlecht
Müller	Paul	13.1.1970	männlich
Meier	Gabi	20.8.1978	weiblich

Die "Geburtstagslisten-Verwaltung" ermöglicht die Eingabe von Name, Vorname, Geburtsdatum und Geschlecht über verschiedene **Widgets** (Eingabefelder, Auswahllisten, Radio-Buttons, Checkboxes, Buttons...). Diese Widgets werden in den automatisierten **Testfällen** entsprechend ferngesteuert.

Die **DemoSwingTests** beinhaltet bereits folgende Elemente:

- **TestSzenarien:**
 - **ApplikationStartSzenario** startet die Anwendung "Geburtstagslisten-Verwaltung"
 - **ApplikationStopSzenario** stoppt die Anwendung "Geburtstagslisten-Verwaltung"
 - **NeuanlageSzenario** beinhaltet alle notwendigen Testschritte um einen neuen Eintrag der Tabelle hinzuzufügen, die Personendaten werden als Parameter übergeben
- **GeburtstagsVerwaltungsSuite:**
 - **AnlegenUndPruefenTest** füllt die Felder aus, um eine neue Person anzulegen. Sowohl vor als auch nach dem Speichern in der Tabelle wird überprüft ob die Daten korrekt eingetragen wurden
 - **MassenAnlageTest** legt mehrere Personen als Masseneingabe an und speichert diese in der Tabelle

DemoWebRapTests

Die Demo für **Eclipse-RAP-Anwendungen** testet gegen die Demo-Seite der Remote Application Platform (<http://rap.eclipsesource.com/demo/release/rapdemo/examples>). Die Demo-Seite beinhaltet eine Vielzahl der Basis-Widgets in den unterschiedlichsten Ausprägungen. Daher eignet sich die Seite sehr gut, um die Steuerung von RAP-Anwendungen zu demonstrieren.

The screenshot displays the RAP Demo web application interface. At the top, there is a blue header with the 'RAP' logo and the text 'REMOTE APPLICATION PLATFORM Demo'. Below the header is a navigation bar with links: 'Basic Widgets', 'Trees & Tables', 'Key Features', 'Layouts', and 'Custom Widgets'. The main content area is titled 'Input Widgets' and is divided into two columns. The left column, 'Simple Input Widgets', contains several form fields: 'First Name', 'Last Name', 'Passphrase' (masked with dots), 'Age' (with a spinner), 'Country' (a dropdown menu), 'Class' (a dropdown menu), and 'Date' (a date picker). Below these fields is a checkbox labeled 'Enabled'. The right column, 'Input validation with visual feedback', shows examples of validation: 'Digits Only' with a text field containing '4711 abcd' and an orange error bar, and 'Mandatory' with a text field and a yellow warning icon. Below these are 'Multiline Texts' with a text area containing placeholder text, and another text area with a scrollbar. At the bottom right of the page, a footer indicates 'RAP version: 2.2.0 (Build 201311142247)'.

Die Tests sind in mehrere **Suiten** untergliedert:

- **ButtonWidgets** - getestet wird neben dem Standard-Button auch Checkboxes und Radiobuttons
- **InputWidgets** - beinhaltet neben dem normalen Eingabefeld auch Tests zu DropDown-Listen und Datumseingabe
- **InputValidationWidgets** - testet Eingabefelder für die es eine Validierung gibt, wie beispielsweise Pflichtfelder
- **TextWidgets** - getestet werden Texteingabefelder mit und ohne Textumbruch
- **DialogWidgets** - es werden unterschiedliche Arten von Dialogen getestet
- **NavigationWidgets** - testet die Navigation zwischen den einzelnen Demo-Seiten

DemoWebserviceRestTests

Diese Demo ruft **REST Webservices** von Google auf, folgende Suiten stehen zur Verfügung:

- **GoogleApiDirectionSuite** überprüft die Verarbeitung von Eingabewerten der Google Maps API für Streckenverläufe
- **GoogleApiTimeZoneSuite** überprüft die Verarbeitung von Eingabewerten der Google Maps API für Zeitangaben

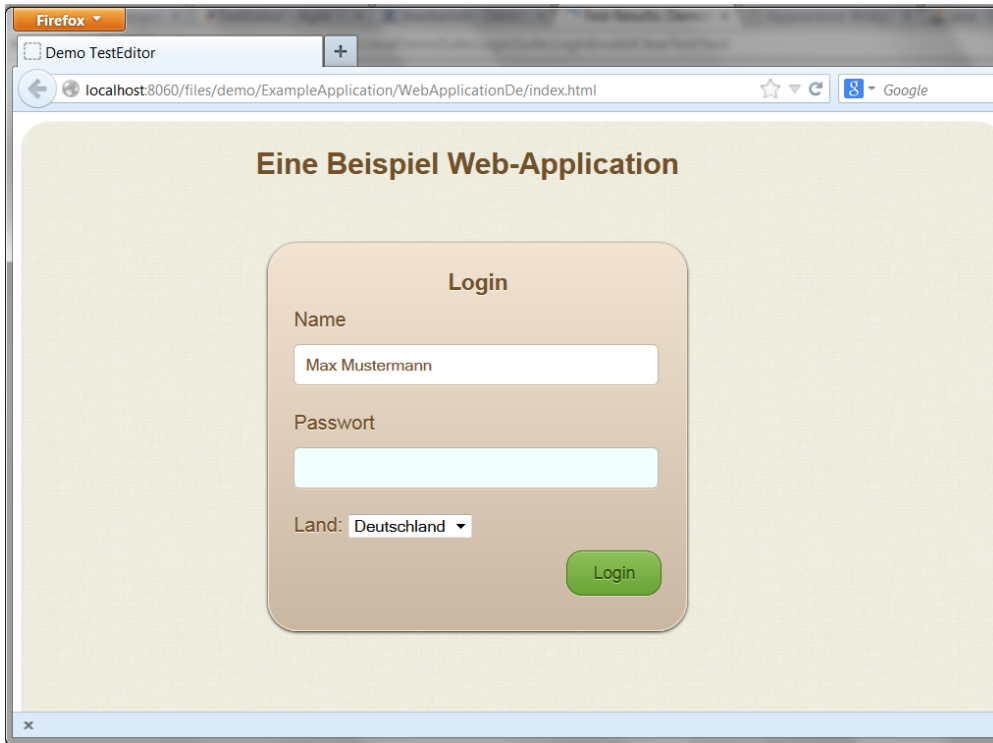
DemoWebserviceSoapTests

Diese Demo ruft einen **SOAP Webservice** auf und überprüft den Response, folgende Suite steht zur Verfügung:

- **SunRiseSoapSuite** fragt den Sonnenaufgang für eine bestimmte Zeitzone zu einem bestimmten Datum an.

DemoWebTests

Die Demo zeigt wie **Web-Anwendungen** getestet werden. Basis ist standardmäßig der Firefox-Browser, der aber zentral für alle Tests in dem ApplikationStartSzenario umgestellt werden kann. Es stehen zwei Suiten zur Verfügung: Eine Suite testet gegen eine **lokale Anwendung** (vgl. Screenshot), die zweite steuert eine **Google-Suche** fern.

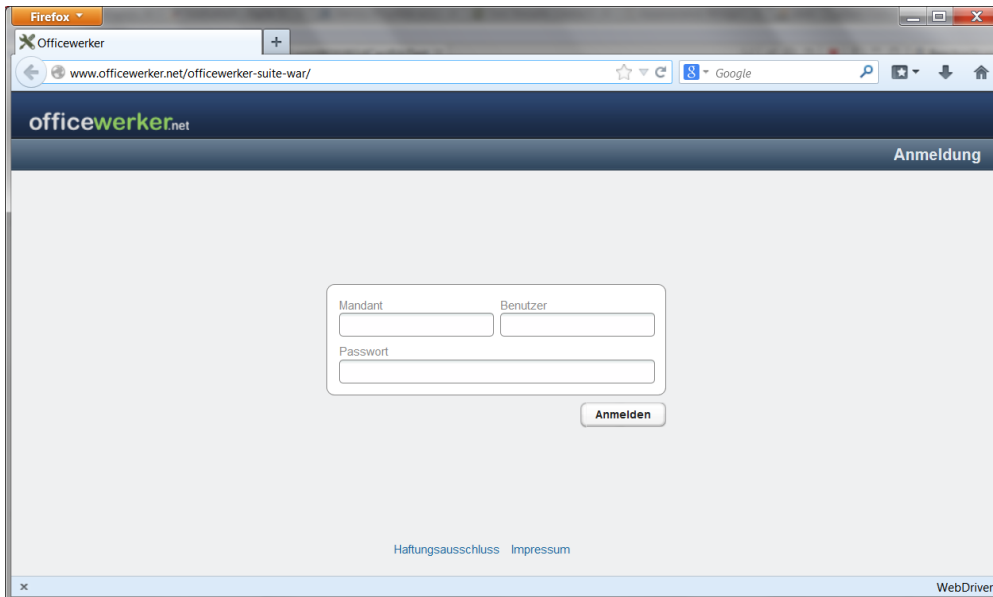


Es stehen folgende Suiten und Testfälle zur Verfügung:

- **GoogleSucheSuite** zeigt wie eine Google-Suche ferngesteuert werden kann (Internet-Verbindung vorausgesetzt)
 - **SucheAkquinetTest** sucht über Google nach dem Wort "akquinet" und überprüft das "www.akquinet.de" gefunden und "www.signal-iduna.de" nicht gefunden wurde
 - **SucheSignalIdunaTest** sucht über Google nach dem Wort "Signal Iduna" und überprüft das "www.signal-iduna.de" gefunden und "www.akquinet.de" nicht gefunden wurde
- **LocalDemoSuite** enthält Testfälle für eine lokale Webanwendung. Die hierfür aufgerufene Beispielanwendung wird mit der Demo ausgeliefert und muss nicht extra installiert werden
 - **LoginValidMassenTestSuite** enthält Testfälle zum Login vieler Anwender
 - **LoginValidMassenTest** testet das Login mit verschiedenen Testdaten hintereinander weg.
 - **LoginSuite** enthält Testfälle zum Login
 - **LoginInvalidClearTest** testet ob nach dem Versuch eines Logins mit einem ungültigen Benutzernamen eine entsprechende Meldung ausgegeben wird und der Benutzername gelöscht wird
 - **LoginInvalidTest** testet ob nach dem Versuch eines Logins mit einem ungültigen Benutzernamen eine entsprechende Meldung ausgegeben wird
 - **LoginValidTest** testet ob ein gültiger Benutzer sich anmelden kann. Die Meldung zum erfolgreichen Login vom System wird überprüft

DemoWebVaadinTests

Diese Demo steuert eine über das Internet zugängliche **Demo Vaadin-Anwendung** fern. Basis ist standardmäßig der Firefox-Browser, der aber zentral für alle Tests in dem ApplikationStartSzenario umgestellt werden kann.



Es stehen folgende Suites und Testfälle zur Verfügung:

- **OfficeWerkerSuite** beinhaltet Testfälle für die Webanwendung "OfficeWerker". Die notwendigen Logindaten sind Teil der Testfälle.
 - **AufwandSuite** enthält Tests für den Bereich "Aufwand".
 - **NeuerAufwandTest** legt einen Aufwand an und überprüft ob der Aufwand korrekt angelegt wurde. Der neu angelegte Aufwand wird dann wieder gelöscht.
 - **SucheAufwandTest** sucht nach einem Aufwand und überprüft das Suchergebnis.
 - **EingangsrechnungenSuite** enthält Tests für den Bereich "Eingangsrechnungen".
 - **SucheRechnungTest** ruft die Eingangsrechnungsübersicht auf und überprüft das Suchergebnis.
 - **LoginSuite** enthält positive und negative Testfälle zum Login-Test.
 - **LoginBenutzerInvalidTest** testet das Login mit einem unzulässigen Benutzer und überprüft ob der Benutzer als unzulässig erkannt wurde
 - **LoginMandantInvalidTest** testet das Login mit einem unzulässigen Mandanten und überprüft ob der Mandant als unzulässig erkannt wurde.
 - **LoginPasswortInvalidTest** testet das Login mit einem unzulässigen Passwort und überprüft ob das Passwort als unzulässig erkannt wurde.
 - **LoginValidTest** testet das Login mit zulässigen Daten und überprüft ob das Login erfolgreich war.
 - **LoginWithWildCardIdsTest** schreibt in die Felder der Login-Seite Daten, löscht diese dann wieder und meldet sich als Admin an und wieder ab.

TestEditorTests

Die TestEditorTests beinhalten die Testfälle um die Anwendung *Test-Editor* selbst zu testen. Um diese Tests auszuführen, ist derzeit ein tieferes Entwickler-Know-How des *Test-Editors* notwendig, was im Administrator-Handbuch weiter beschrieben wird.

- **AcceptanceForConDelivery** beinhaltet den Akzeptanztest für die Auslieferung von Projekten.
- **PerformanceAnalyse** beinhaltet Testfälle, deren Performanz analysiert werden.
- **ProjektSuite** beinhaltet alle Testfälle zum Anlegen und Editieren von Projekten.
 - **ProjektKonfigurationSuite** beinhaltet Testfälle, die das Bearbeiten von Projekten überprüfen.
 - **ProjektTeilenSuite** beinhaltet einen Testfall, zum Testen des Projektteilens.
- **SzenarienSuite** beinhaltet alle Testfälle zum Anlegen und Editieren von Szenarien.
- **SzenariensuiteSuite** beinhaltet Testfälle, die das Anlegen und Editieren von Szenarien testen.
- **TestfallSuite** beinhaltet Testfälle, die das Anlegen und Editieren von Testfällen testen.
 - **TestfallEditierenSuite** beinhaltet Testfälle, die das Verhalten von Testfällen überprüfen.
- **TestsuiteSuite** beinhaltet Testfälle, die das Verhalten von TestSuiten überprüfen.

Außerdem stellt TestEditorTests folgende Testszenarien zur Verfügung:

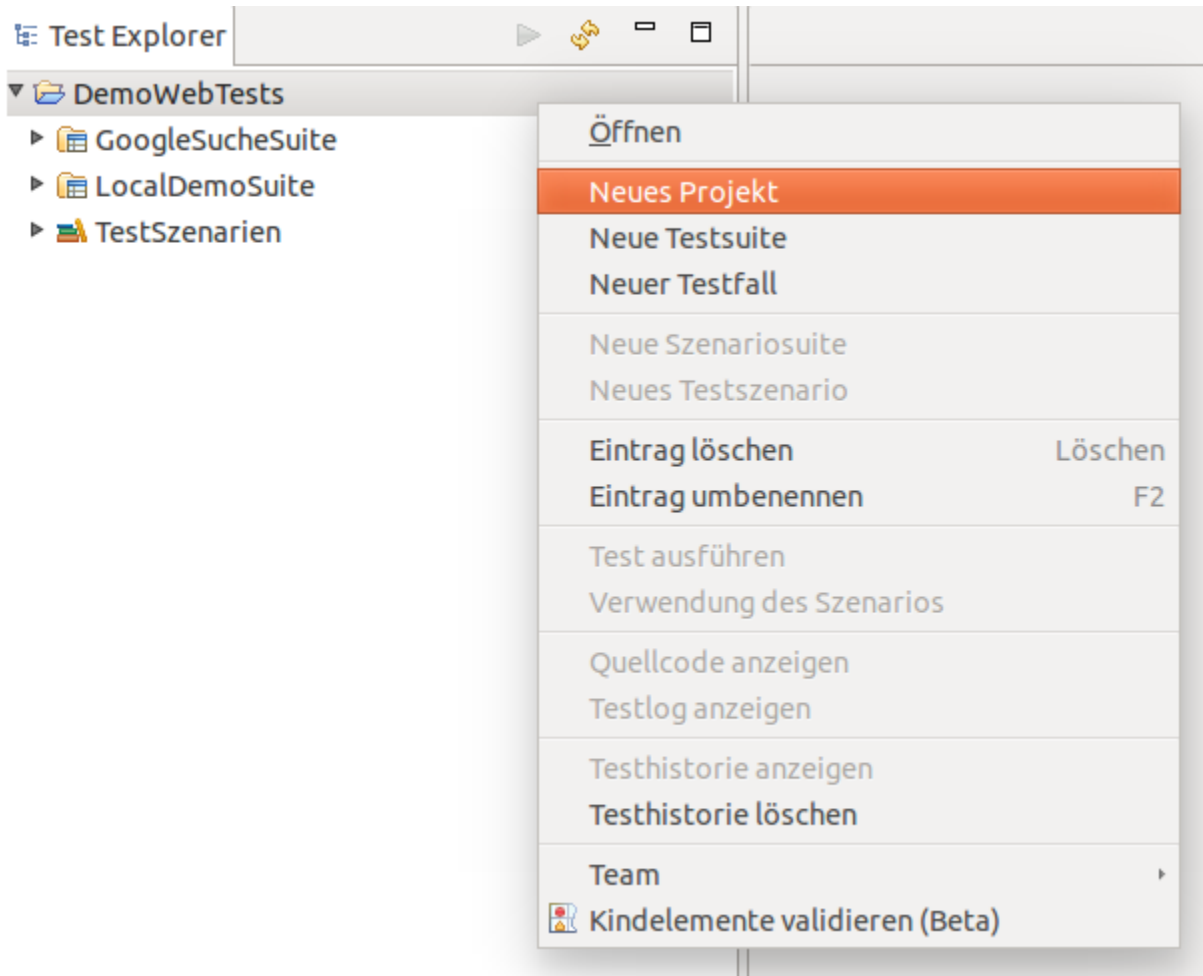
- **TestExplorerSzenarien** beinhaltet Szenarien zum Löschen, Umbenennen und Prüfen des Quellcodes.
- **TestfallSzenarien** beinhaltet Szenarien für Basistestfälle.
- **WizardSzenarien** beinhaltet Szenarien für Wizardtestfälle.
- **ApplikationStartSzenario** startet die Anwendung.
- **ApplikationStopSzenario** beendet die Anwendung.

Projekte anlegen und konfigurieren

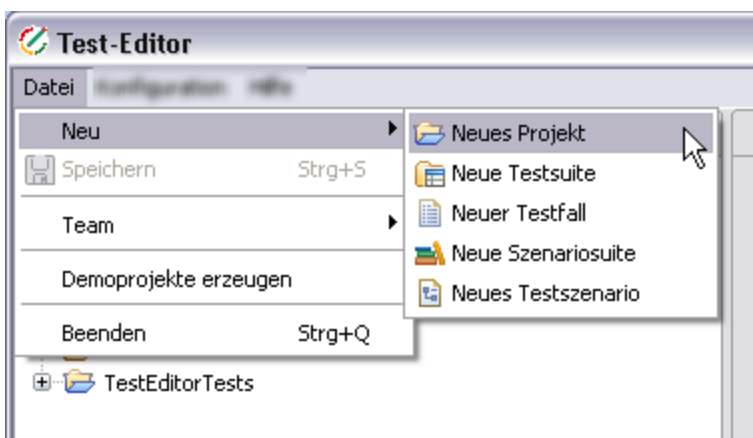
Ein **Projekt** bezieht sich in der Regel auf eine zu testende Applikation und beinhaltet alle notwendigen **Testfälle**. Die zu testende Applikation wird häufig auch als **AUT** (Application Under Test) beschrieben.

Projekt anlegen

Ein neues Projekt kann über das **Kontextmenü (Test-Explorer-> DemoWebTests unter Windows rechte Maustaste)** im Test-Explorer angelegt werden:

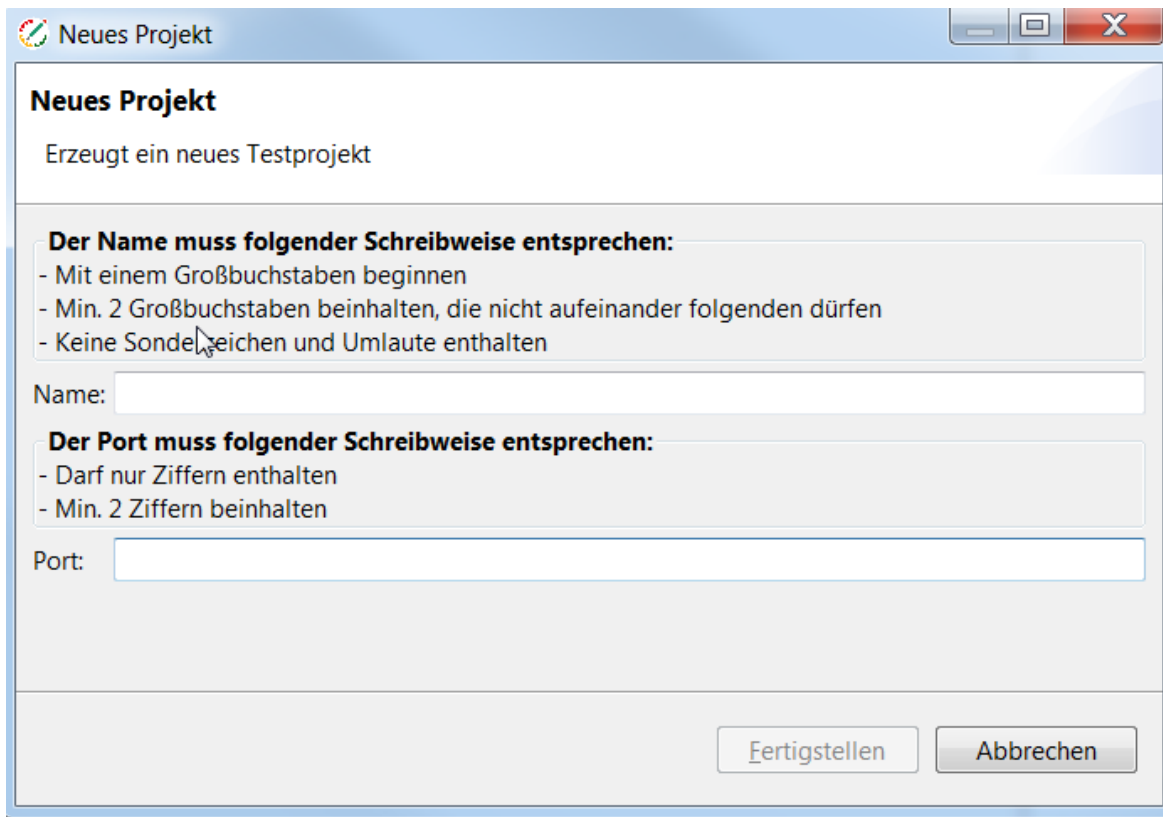


Alternativ steht die Funktion über **Menüeintrag Datei -> Neu -> Neues Projekt** zur Verfügung:



Es öffnet sich der **Dialog Neues Projekt**:

- In das **Feld Name** wird ein eindeutiger Projektname festgelegt. Wichtig zu beachten sind die Namenskonventionen, die im Dialog angezeigt werden.
- In das **Feld Port** ist ein, noch von keiner Applikation verwendeter Port, für den lokalen **FitNesse** Server einzutragen.



Neues Projekt

Erzeugt ein neues Testprojekt

Der Name muss folgender Schreibweise entsprechen:

- Mit einem Großbuchstaben beginnen
- Min. 2 Großbuchstaben beinhalten, die nicht aufeinander folgenden dürfen
- Keine Sonderzeichen und Umlaute enthalten

Name:

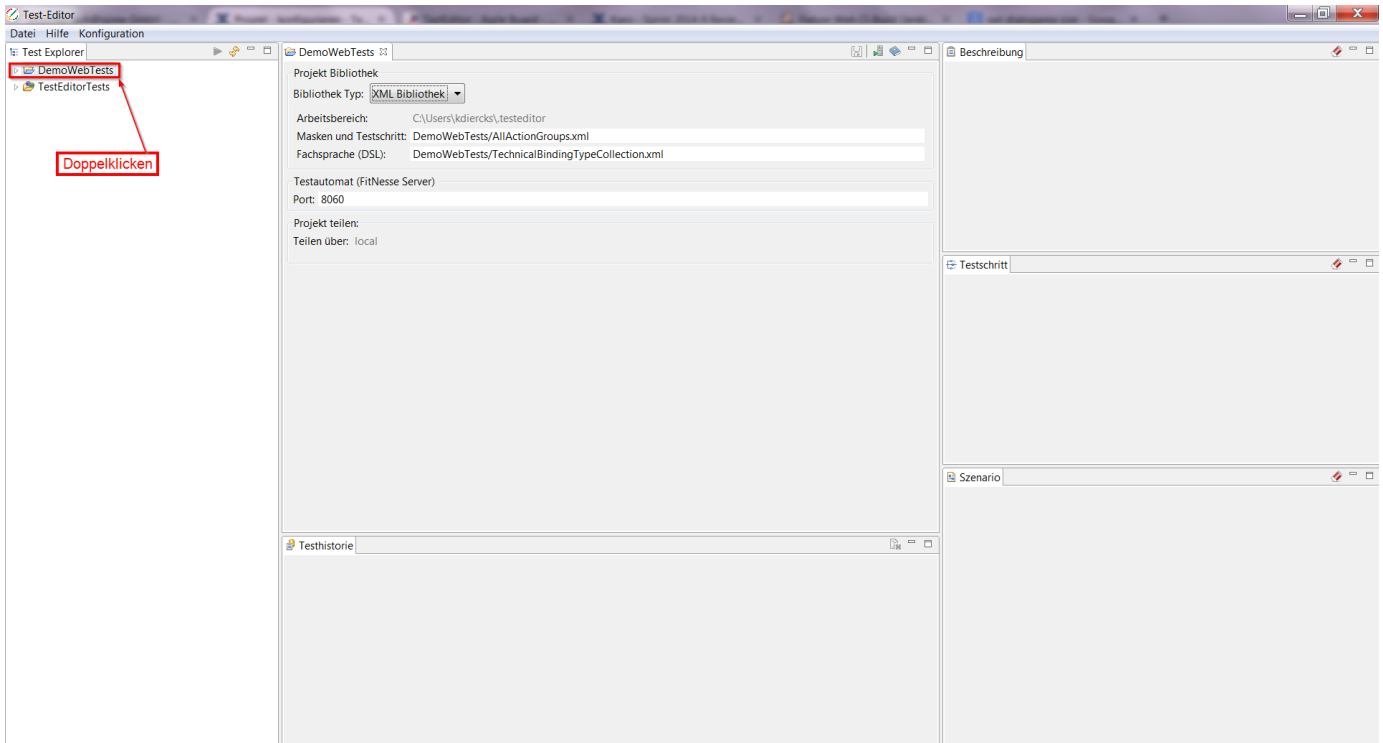
Der Port muss folgender Schreibweise entsprechen:

- Darf nur Ziffern enthalten
- Min. 2 Ziffern beinhalten

Port:

Projekt konfigurieren

Über einen **Doppelklick** auf dem **Projekt** im Test-Explorer öffnet sich die Projektkonfiguration:



Auf der Konfigurationsseite können folgende Einstellungen vorgenommen werden:

Projekt Bibliothek

- **Bibliothek Typ:** In den meisten Fällen wird die **XML** Library verwendet, d.h. die Metadaten in Form von XML definiert; der *Test-Editor* unterstützt allerdings auch eigene Plugins, um diese Daten z.B. durch eine projektspezifische Implementierung zu laden (z.B. aus einer Datenbank)
- **Masken und Testschritte:** Pfad zur XML Datei, welche die **Masken** und möglichen **Testschritte** beschreibt
- **Fachsprache (DSL):** Pfad zur XML Datei, welche die für die Tests verwendete domänenspezifische Sprache beschreibt



Hinweis auf das Administrator-Handbuch

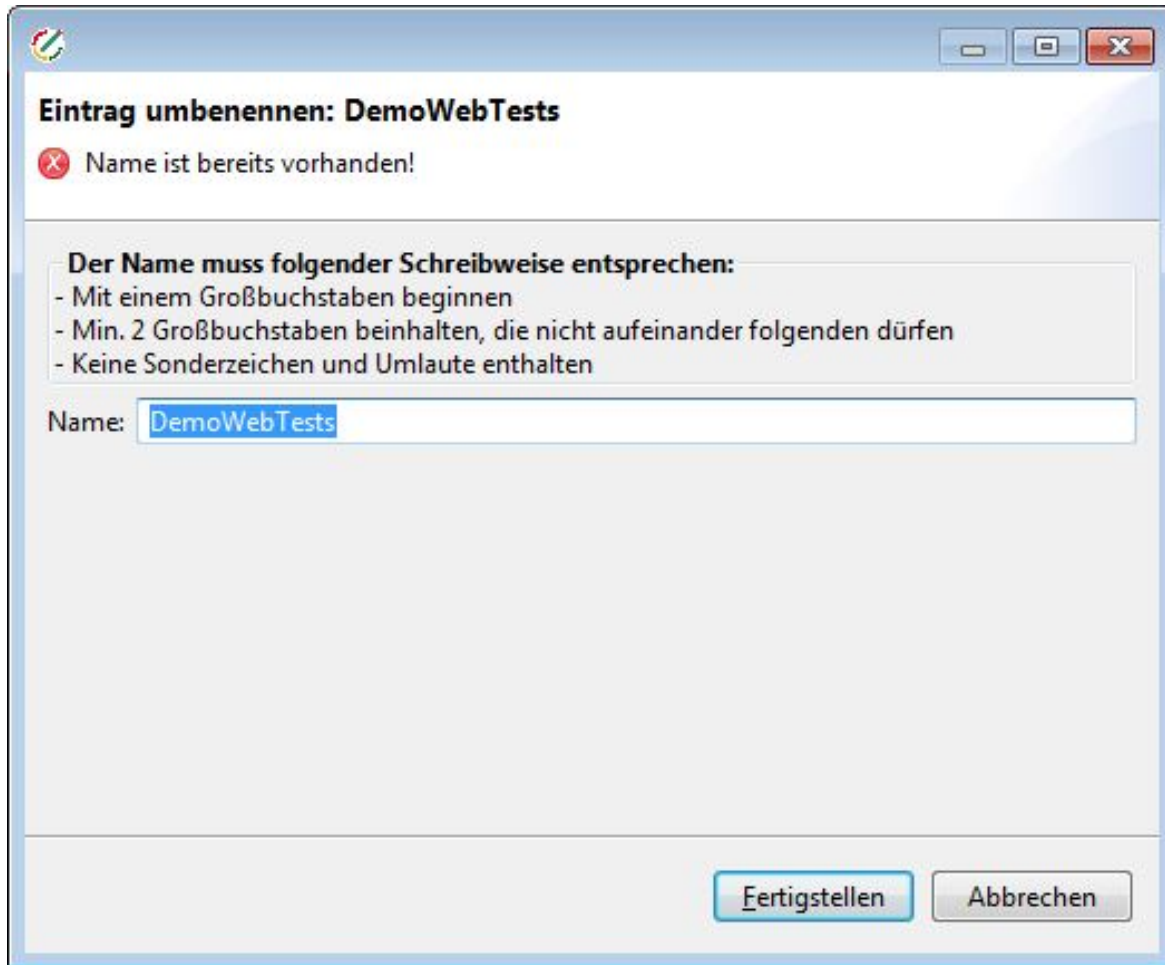
Im Administrator-Handbuch ist beschrieben, wie die **XML-Dateien zur Konfiguration eines Projektes** erzeugt und editiert werden können. In diesen Dateien werden die möglichen Testschritte einer Anwendung definiert, die im *Test-Editor* zur Auswahl bei der Erzeugung von Testschritten stehen.

Testautomat

- **Port:** Der lokale Port des FitNesse-Servers (falls dieser beim Start des Test-Editors belegt sein sollte, bitte ändern)

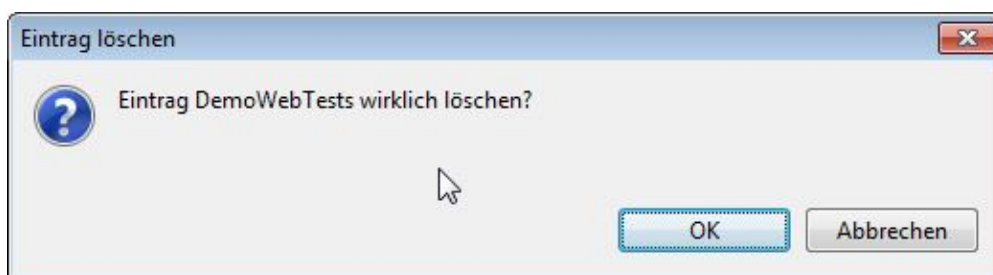
Projekt umbenennen

Ein Projekt kann über das **Kontextmenü (Test-Explorer) des Projektes** umbenannt werden, es öffnet sich folgender Dialog:



Projekt löschen

Ein Projekt kann über das **Kontextmenü (Test-Explorer) des Projektes** gelöscht werden, es erscheint eine **Sicherheitsabfrage**, ob das ausgewählte Element wirklich gelöscht werden soll. Es werden generell alle Kind-Objekte ([Testsuiten](#), [Testfälle](#) etc.) ebenfalls gelöscht. Ggf. werden die entsprechenden Elemente im Editor-Bereich automatisch geschlossen.

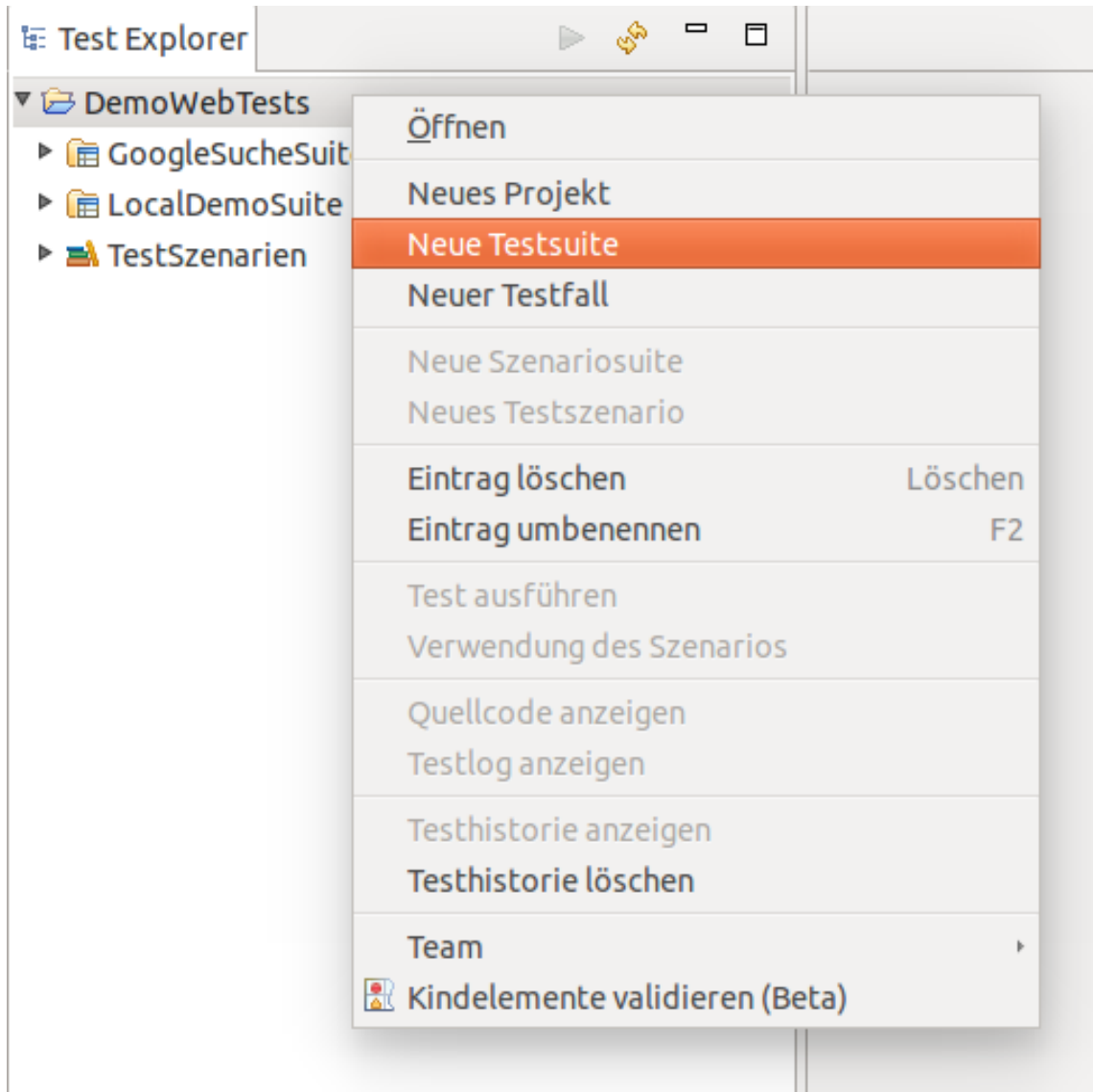


Mit Testsuiten gruppieren

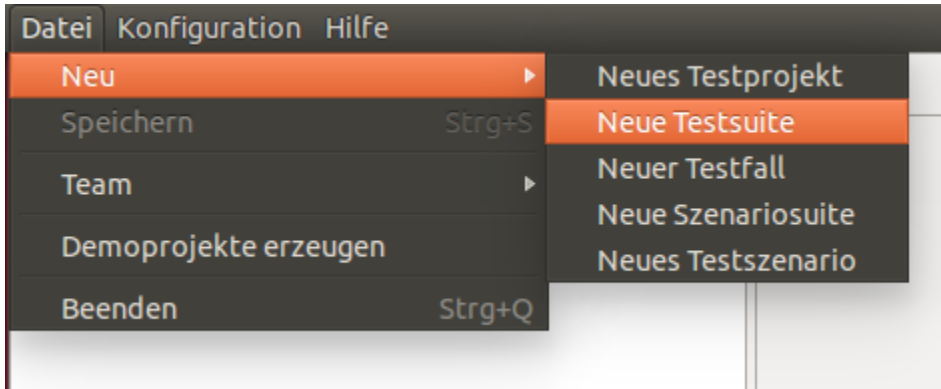
Testsuiten werden im Kontext eines Projektes erzeugt und dienen als Gruppierung von mehreren **Testfällen** und Unter-Testsuiten. Suiten sind vergleichbar mit Ordnern. Neben der Möglichkeit Testfälle als Kind-Elemente zu Erzeugen, gibt es auch die Möglichkeit bestehende Testfälle in die Testsuite zu referenzieren.

Testsuite anlegen

Eine neue Testsuite kann über das **Kontextmenü (Test-Explorer)** auf einem beliebigen **Projekt oder Testsuite** angelegt werden:

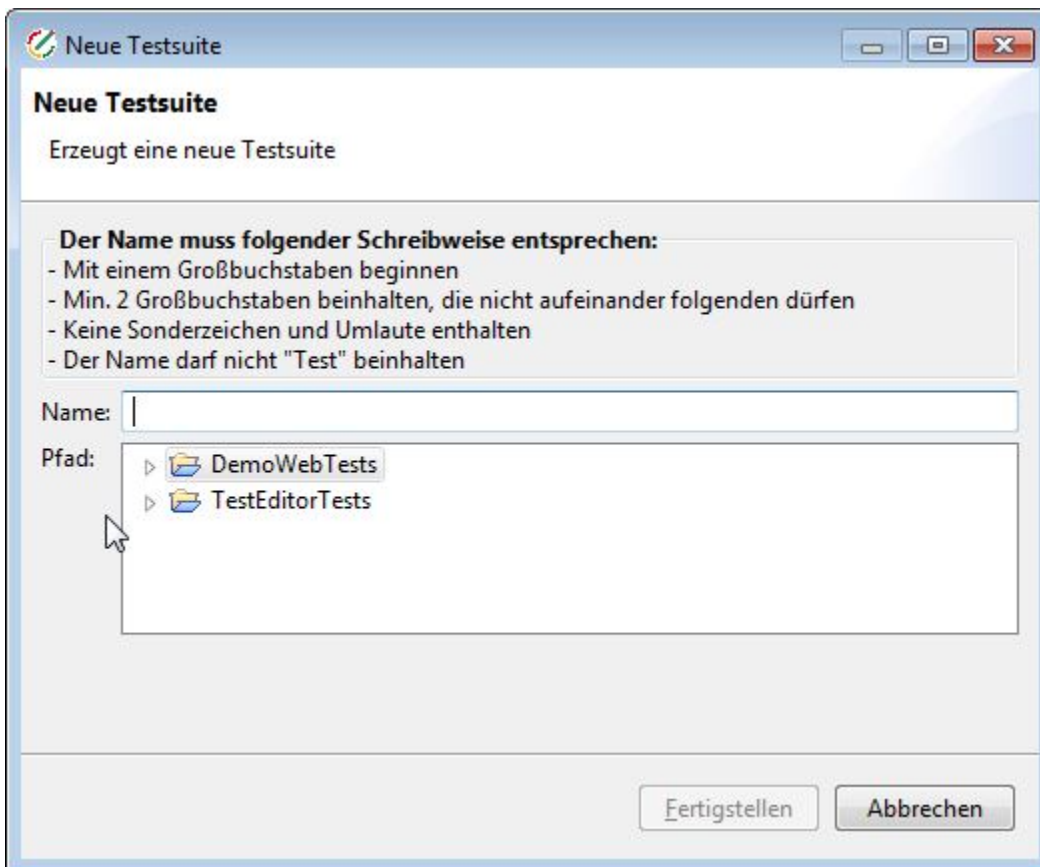


Alternativ steht die Funktion über **Menüeintrag Datei -> Neu -> Neue Testsuite** zur Verfügung:



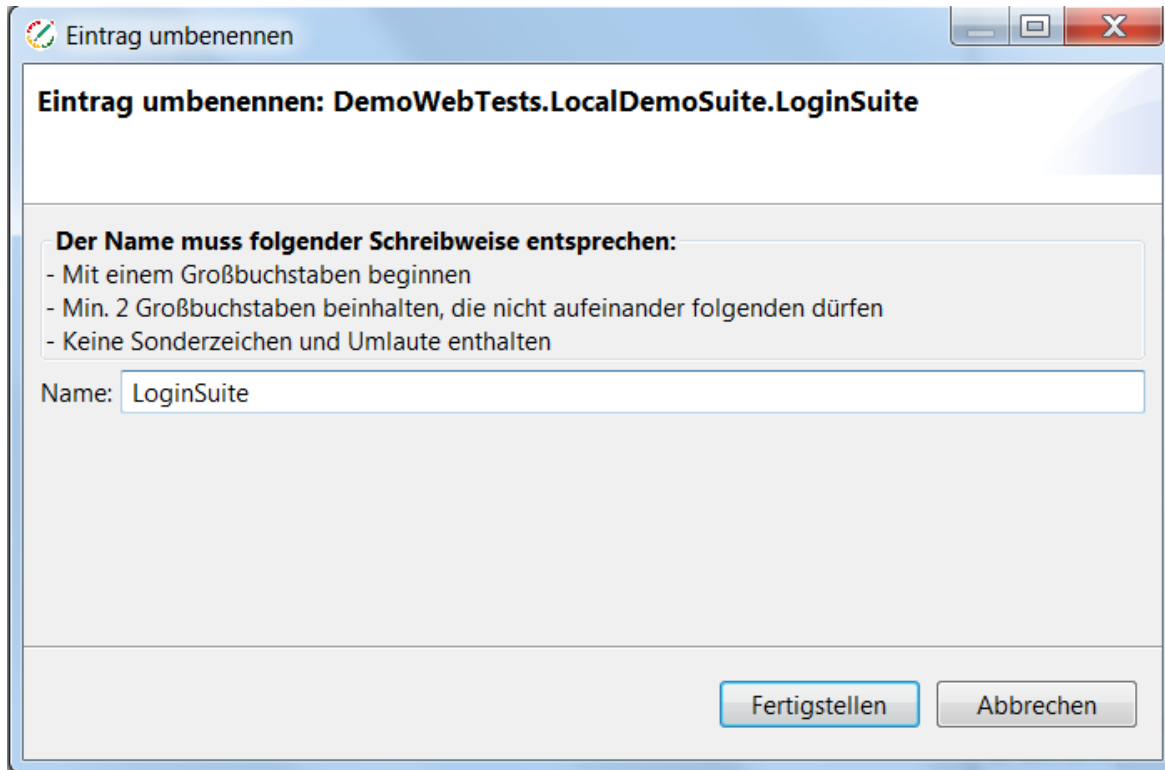
Es öffnet sich der **Dialog Neue Testsuite**:

- In das **Feld Name** wird der innerhalb des Vater-Elements (Projekt oder Testsuite) eindeutige Name für die Suite festgelegt. Wichtig zu beachten sind die Namenskonventionen, die im Dialog angezeigt werden
- In dem **Bereich Pfad** kann gewählt werden, wo in der Hierarchie die Suite erzeugt werden soll



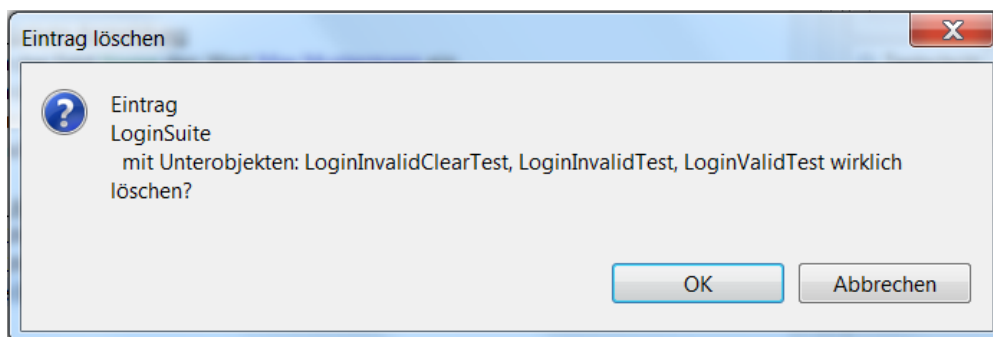
Testsuite umbenennen

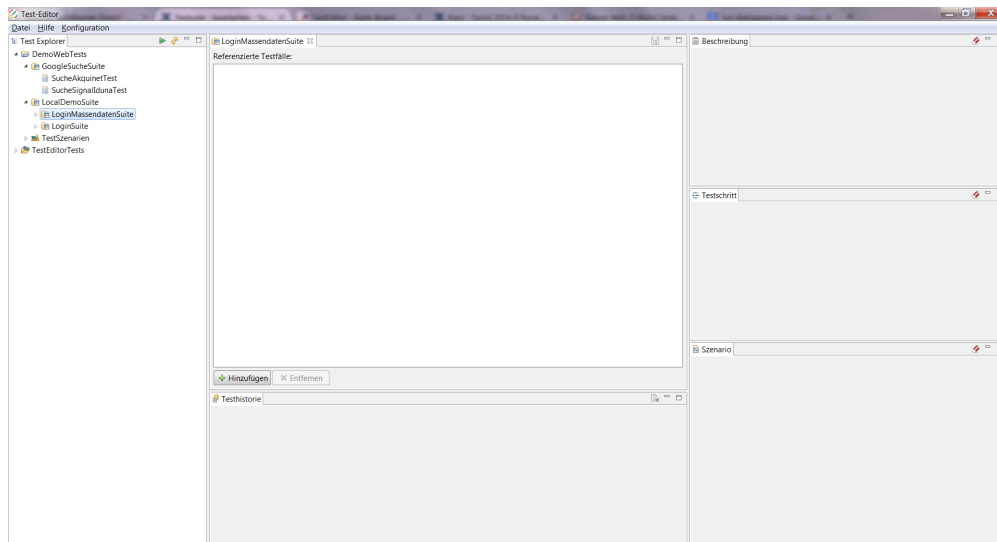
Eine Testsuite kann über das **Kontextmenü (Test-Explorer) der Suite** umbenannt werden, es öffnet sich folgender Dialog:



Testsuite löschen

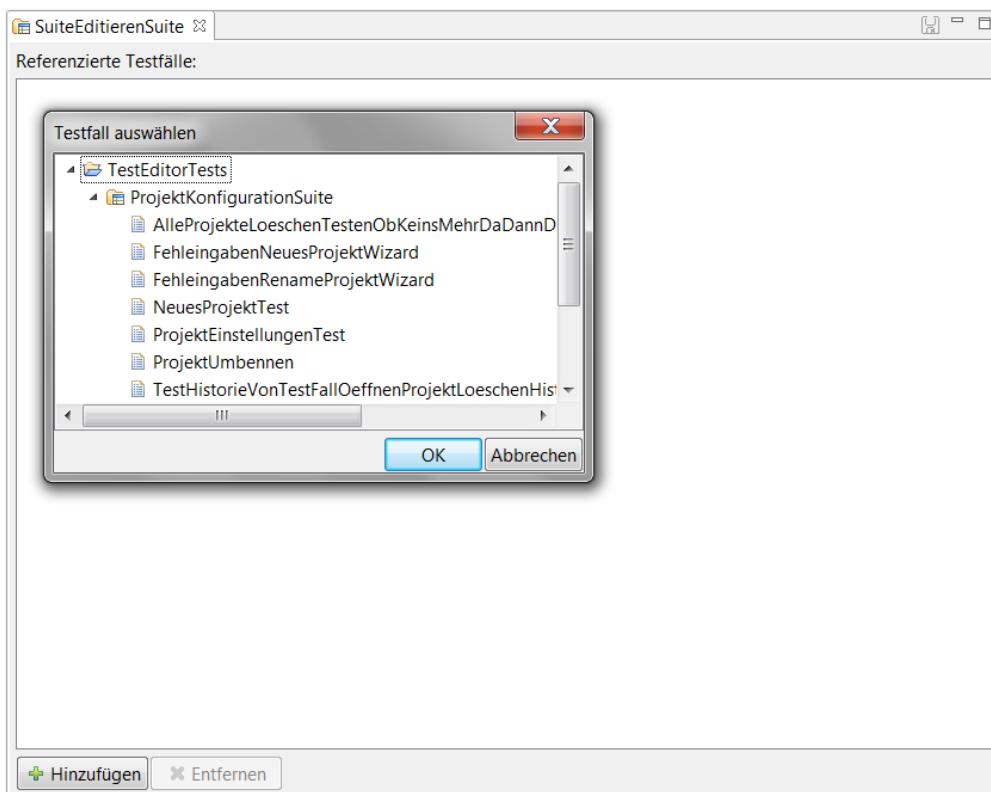
Eine Testsuite kann über das **Kontextmenü (Test-Explorer) der Suite** gelöscht werden, es erscheint eine **Sicherheitsabfrage**, ob das ausgewählte Element wirklich gelöscht werden soll. Es werden generell alle Kind-Elemente (Untersuiten, Testfälle etc.) ebenfalls gelöscht. Ggf. werden die entsprechenden Elemente im Editor-Bereich automatisch geschlossen.





Eine Testsuite kann nicht nur Testfälle wie einen Ordner aufnehmen und strukturieren, sondern sie kann auch Testfälle referenzieren. Dies kann genutzt werden um Testsuiten für bestimmte Zwecke anzulegen, die sich dann quer durch verschiedene Rubriken bestücken lassen. Eine Testsuite kann im Test-Explorer zum bearbeiten geöffnet werden:

Mittels des Hinzufügen-Buttons kann ein bestehender Testfall aus einem Dialog hinzugefügt werden:



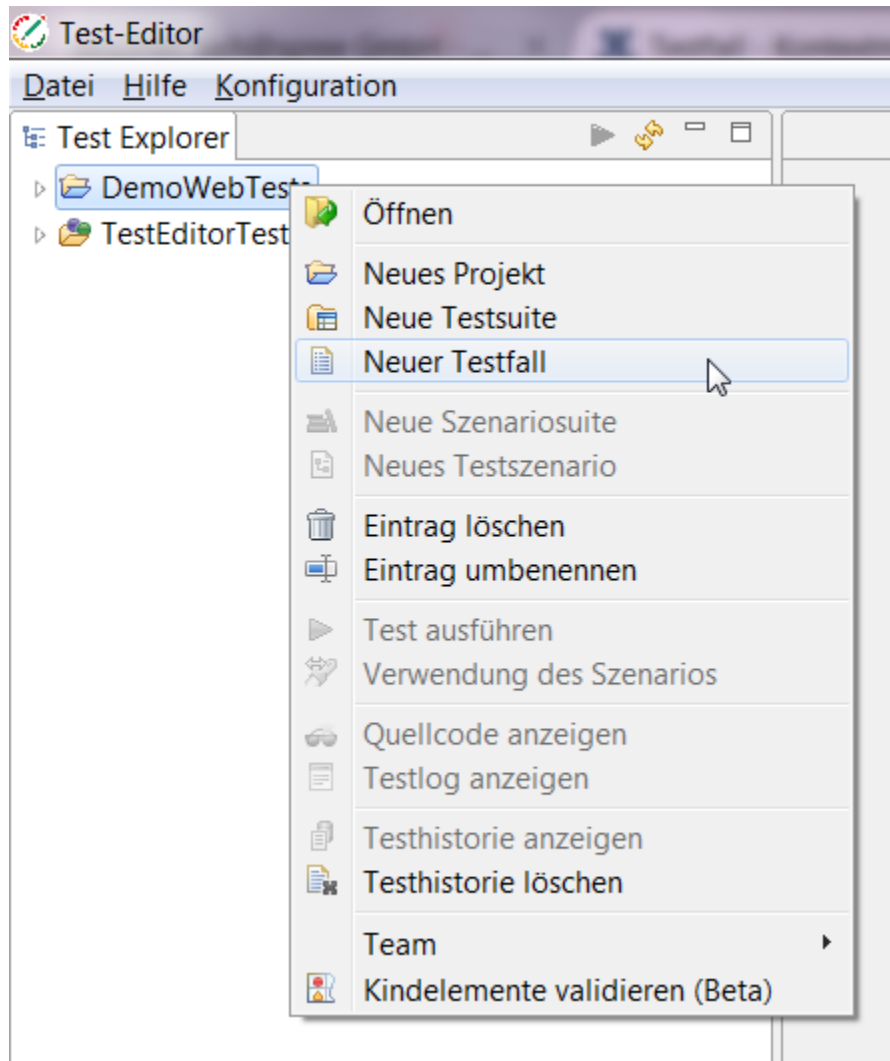
Über den X-Button können selektierte Testfälle wieder gelöscht werden.

Testfälle anlegen und bearbeiten

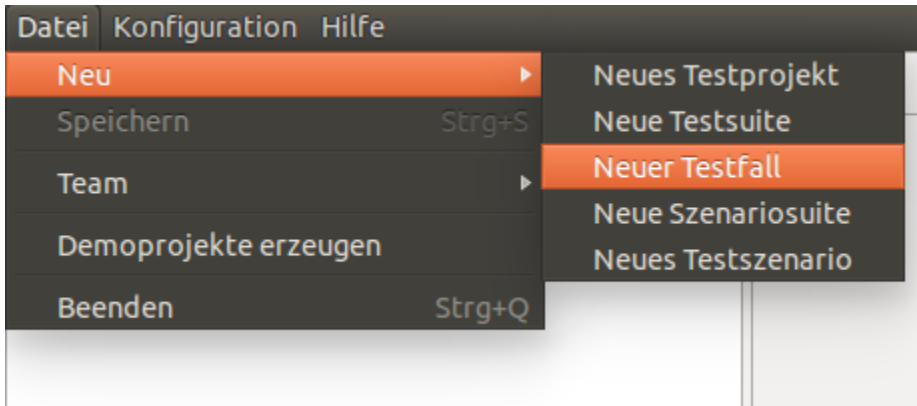
Ein **Testfall** bildet die kleinste abgeschlossene Einheit für einen **Test**. Ein Testfall bezieht sich immer auf eine konkrete Arbeitsabfolge in der zu testenden Applikation (**AUT**). Ein definierter Ausgangszustand (Testdaten, Testsystem, etc.) ist hierfür unabdingbar. Testfälle werden immer im Kontext eines Projektes erzeugt und bilden die kleinste Einheit in der Hierarchie des Projektes ab.

Testfall anlegen

Ein neuer Testfall kann über das **Kontextmenü (Test-Explorer)** auf einem beliebigen **Projekt** oder **Testsuite** angelegt werden:

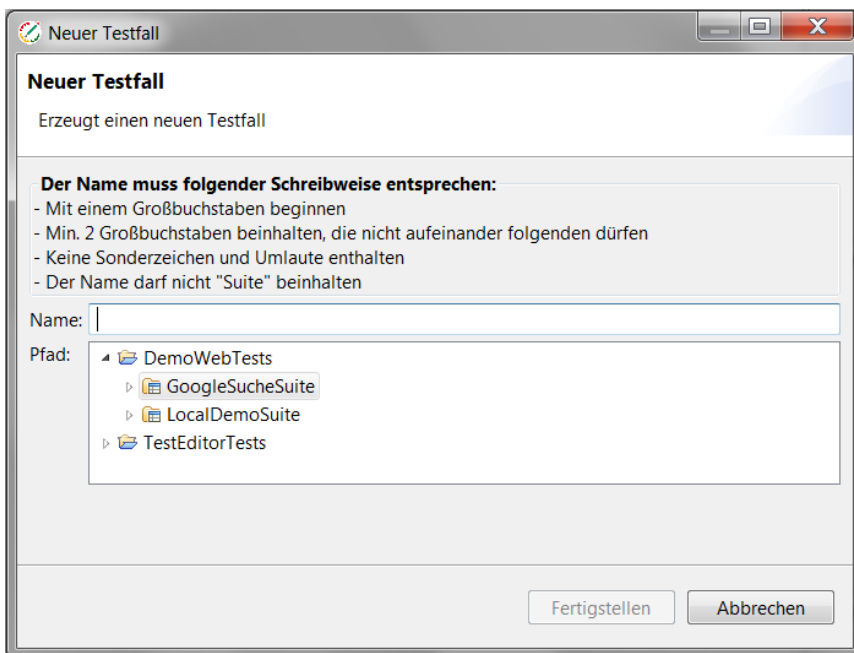


Alternativ steht die Funktion über den **Menüeintrag Datei -> Neu -> Neuer Testfall** aus dem Hauptfenster zur Verfügung:

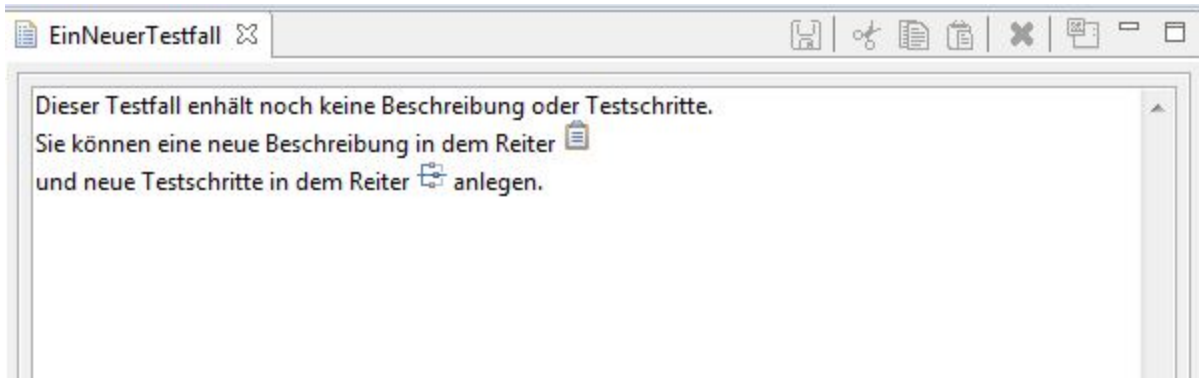


Es öffnet sich der **Dialog Neuer Testfall**:

- In das Feld **Name** wird der innerhalb des Vater-Elements (Projekt oder Test-Suite) eindeutige Name für den Testfall festgelegt. Wichtig zu beachten sind die Namenskonventionen, die im Dialog angezeigt werden
- In der **Pfad-Angabe** kann gewählt werden, wo in der Hierarchie der Testfall erzeugt werden soll



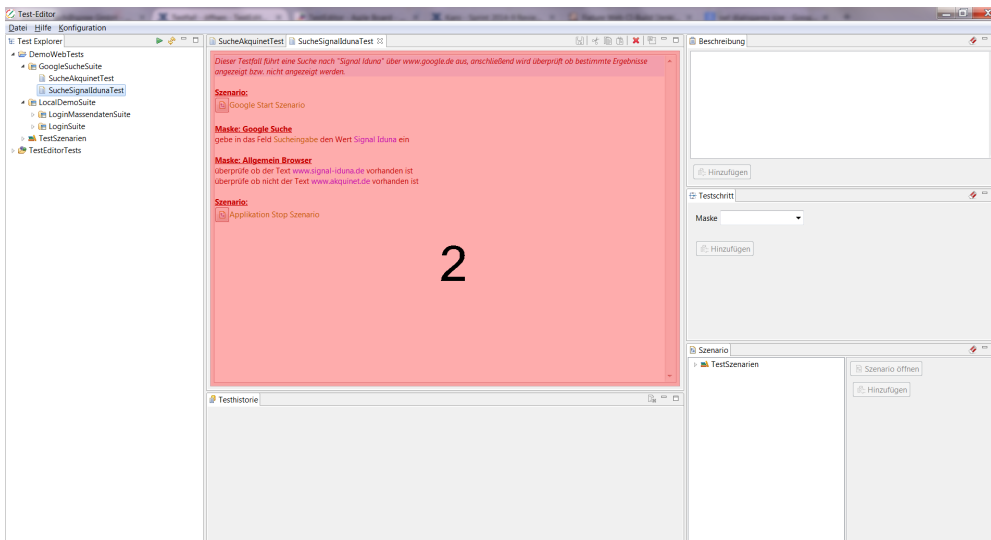
Nach der Anlage wird der Testfall in der Hierarchie im Test-Explorer angelegt und im Editor-Bereich geöffnet. Da der Testfall noch keinen Inhalt enthält, entspricht das Aussehen der folgenden Abbildung.



Sobald einzelne **Testschritte**, Beschreibungen oder Verweise auf **Szenarien** im Testfall erfasst wurden, werden die einzelnen Schritte im **Editor-Bereich** angezeigt.

Testfall öffnen

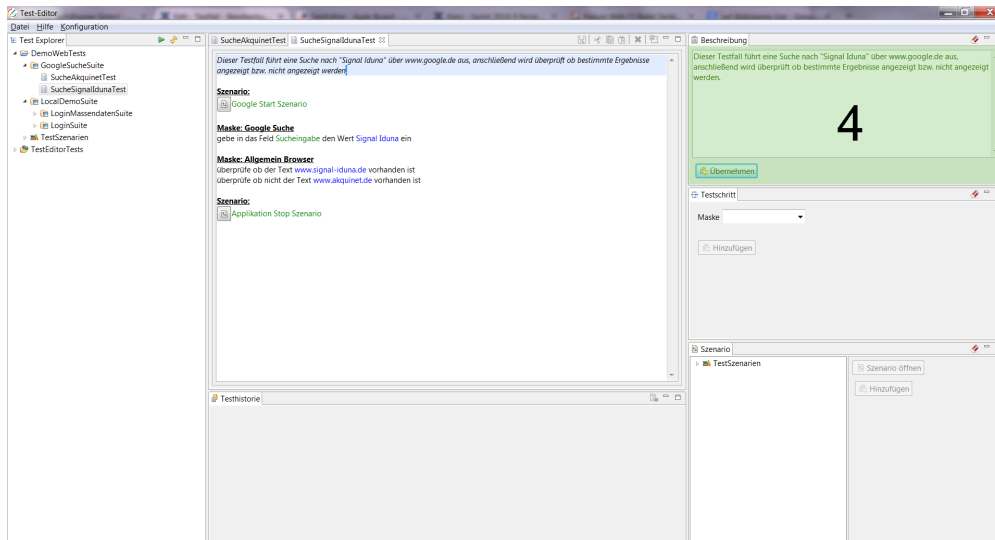
Über einen **Doppelklick** auf einem Testfall im **Test-Explorer** wird der Testfall geladen. Der Inhalt wird im Editor-Bereich (in Bereich 2 dargestellt) angezeigt, hier für den Testfall *SucheAkquinetTest*:



Beschreibungen zum Testfall hinzufügen

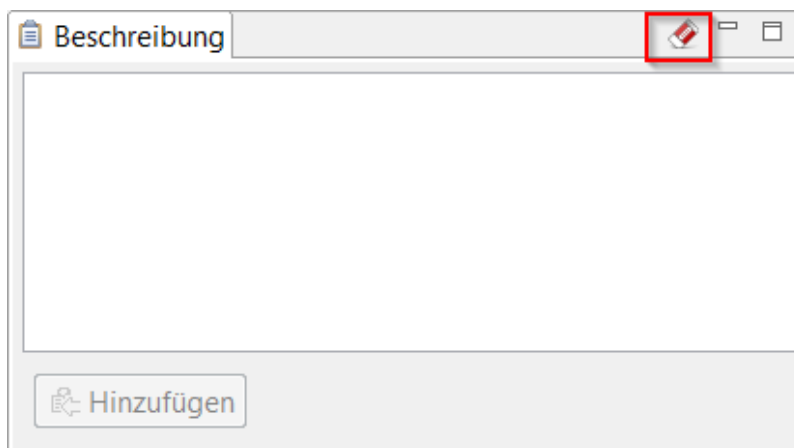
Zu Beginn eines Testfalls bietet es sich an, die Intension des Testfalls zu beschreiben. Auch innerhalb eines Testfalls sind zusätzliche Beschreibungen hilfreich um den Ablauf zu verdeutlichen. Beschreibungen haben keine Auswirkung auf die eigentliche Testausführung.

Im **Eingabefeld Beschreibung** (in Bereich 4 dargestellt), wird die neue Beschreibung erfasst und eine neue Zeile im Editor-Bereich an der Position des Cursors mit Hilfe des **Hinzufügen Buttons** erzeugt. Wurde zuvor eine Zeile ausgewählt erscheint statt dem Hinzufügen der **Übernehmen Button**, die bestehende Zeile wird also überschrieben.



Beschreibungsinhalte im Eingabefeld löschen

Die Inhalte im Eingabe können durch den rot markierten **Radiergummi-Button** entfernt werden um eine neue Beschreibung zu erfassen.



Alternativ können Beschreibungen zu einem Testfall auch über einen Popup-Dialog über die **Funktionstaste F7** erfasst werden:

Beschreibung

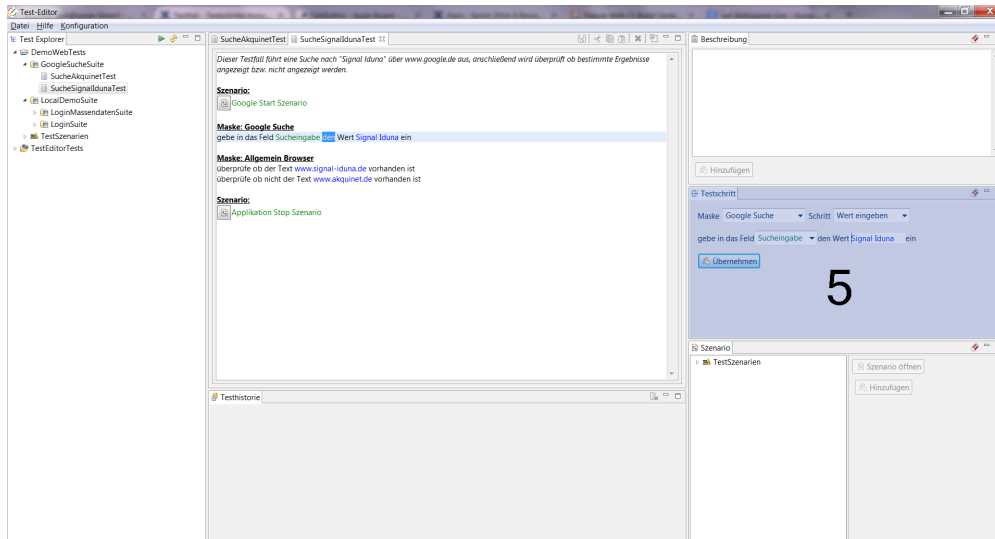
Hinzufügen

Hinzufügen und schließen

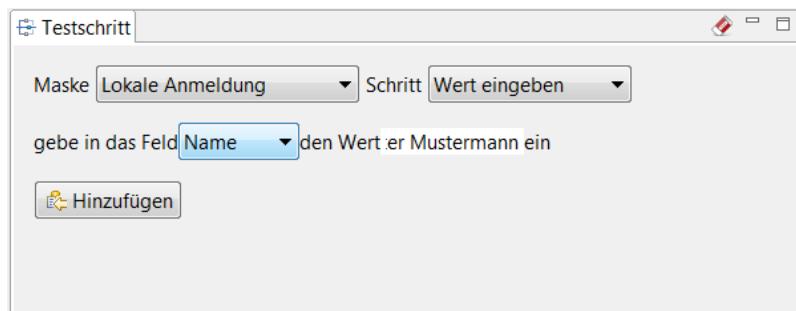
Testschritte zum Testfall hinzufügen

Testschritte definieren einzelne Abläufe, die in der zu testenden Applikation (AUT) ausgeführt werden sollen. Dies kann z.B. eine Eingabe in einem Feld sein, oder das Klicken eines Buttons.

In dem Bereich **Testschritte** (in Bereich 5 dargestellt) ist es möglich, einzelne Schritte für den Testfall auszuwählen bzw. anzupassen. Die Anweisungen werden in einer natürlichen **Fachsprache** erfasst und die Auswahlmöglichkeiten sind projektabhängig. Die möglichen Testschritte sind in **Masken** gegliedert. D. h. es können alle Eingaben und Prüfungen, die sich auf einer Seite des Testobjektes (z.B. eine Seite einer Web-Anwendung) befinden, gruppiert werden. Der Vorteil liegt darin, dass nur die Testschritte und Felder angezeigt werden, die auch auf der Maske existieren.



Beispiel: Die Maske *Lokale Anmeldung* bietet die Schritte *Button betätigen* und *Wert eingeben* an. Konkret kann der Benutzer z.B. einen Login wie folgt definieren: Aus dem Auswahlfeld **Maske** die Auswahl *Lokale Anmeldung* wählen und aus dem Auswahlfeld **Schritt** die Auswahl *Wert eingeben* wählen. Nach der Auswahl öffnet sich eine neue Zeile, in welcher die Eingaben für den Testschritt spezifiziert werden. Im folgenden Beispiel wurde für das Feld **Name** der Wert *Peter Mustermann* ausgewählt.



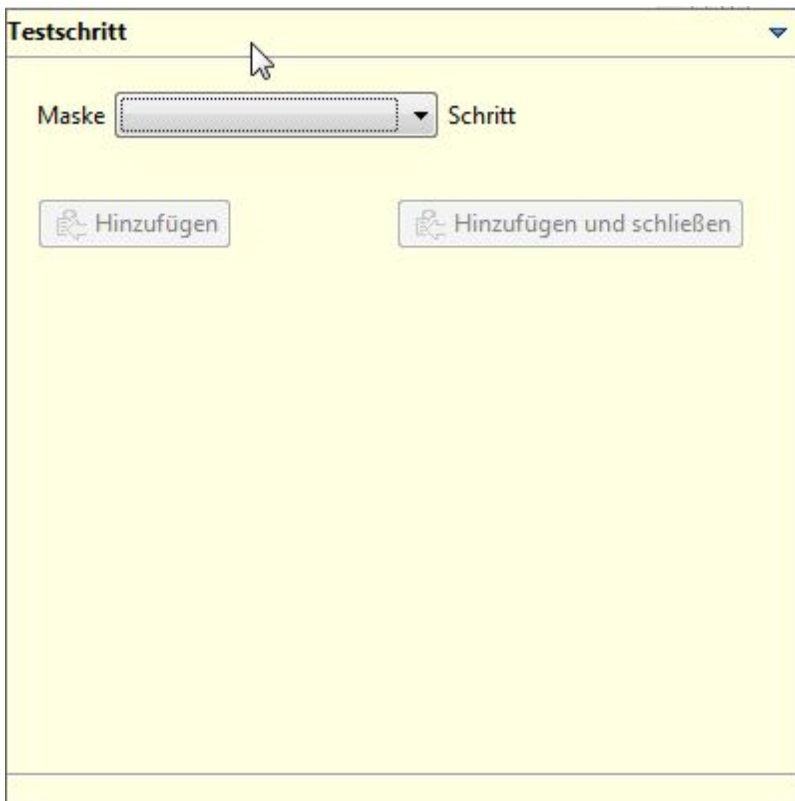
Mit dem Button **Hinzufügen** (bzw. **Übernehmen** bei Änderungen) werden die Eingaben in den Editor-Bereich übernommen.

Testschrittinhalte im Eingabefeld löschen

Die Inhalte im Eingabefeld können durch den rot markierten **Radiergummi-Button** entfernt werden, so dass ein neuer Testschritt erfasst werden kann.

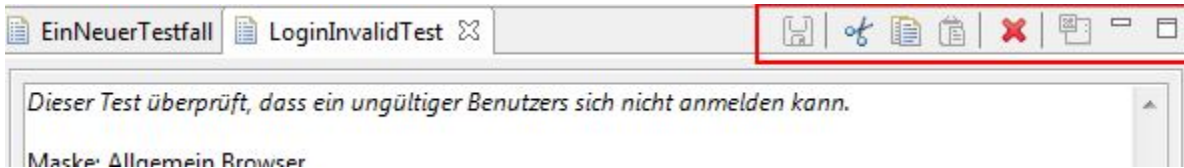


Alternativ können Testschritte zu einem Testfall auch über einen Popup-Dialog über die **Funktionstaste F8** erfasst werden:

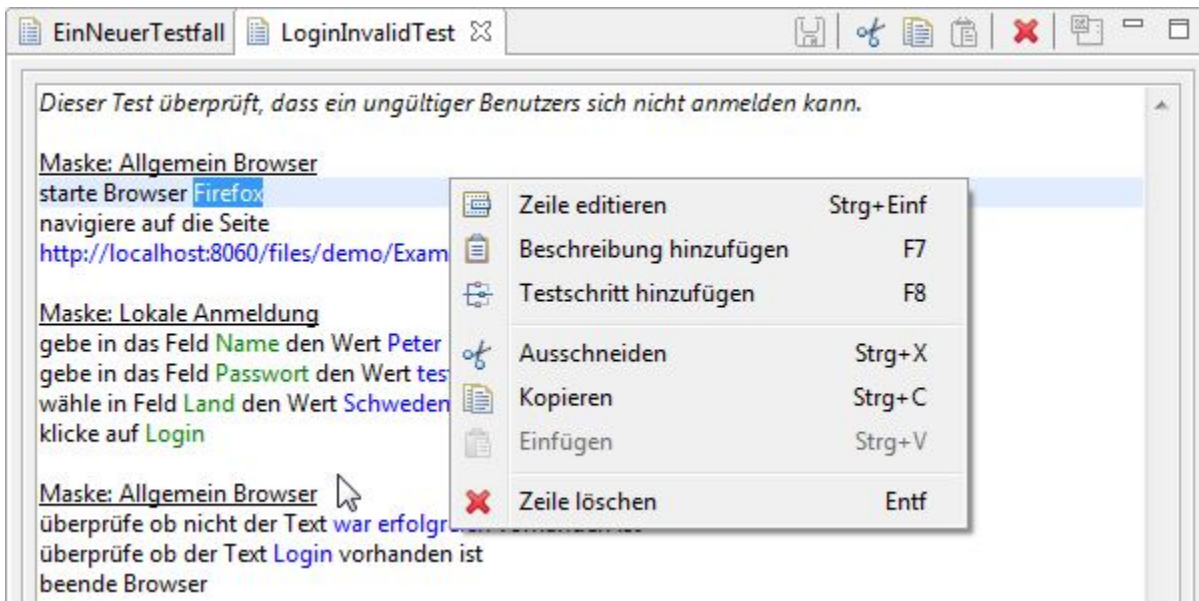


Testfall bearbeiten

In dem **Editor-Bereich** können die Operationen **kopieren**, **ausschneiden**, **löschen** und **einfügen einzelner Zeilen** am Testfall durchgeführt werden. Diese Operationen können über die **Menüleiste im Editor-Bereich** aufgerufen werden:



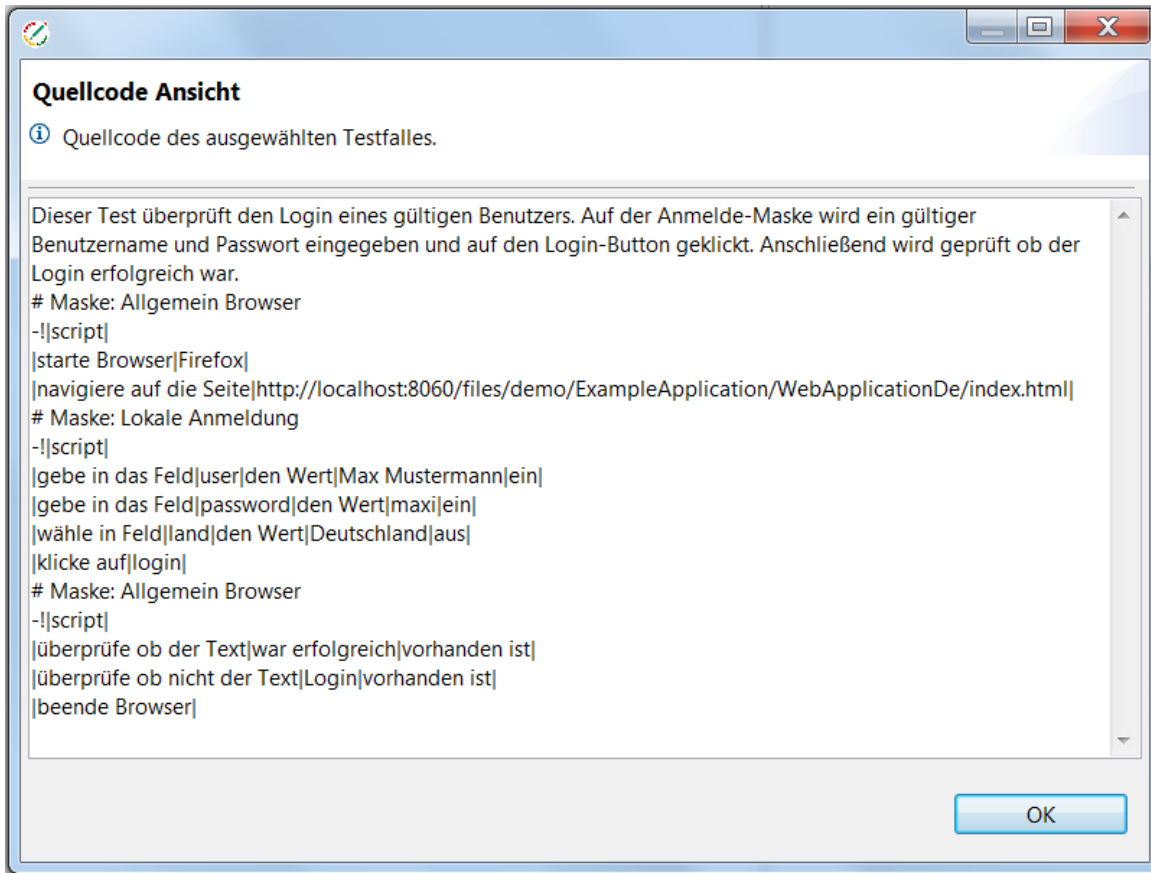
Alternativ lassen sich die Funktionen über das **Kontextmenü** aufrufen:



Hinweis: Wenn eine Zeile markiert ist, so wird diese beim Hinzufügen einer Beschreibung, Testschritts oder **Szenario** überschrieben. Wenn nur der Cursor in dem Editor-Bereich im Testfall steht, so wird an dieser Position das neue Element eingefügt. Beschreibungen oder Testschritte können innerhalb eines Testfalls ausgeschnitten oder kopiert und anschließend wieder eingefügt werden. Es ist auch möglich die kopierten Beschreibungen oder Testschritte in einem anderen Testfall des selben Projekts einzufügen. Eine Kopie über die Projektgrenze hinweg ist nicht möglich.

Quellcode eines Testfalls anzeigen

Alle Elemente im Test-Explorer (Testfälle, Suiten, Szenarios usw.) werden im Hintergrund auf einem FitNesse Server gespeichert. Der Server erwartet ein bestimmtes Format, wie diese Elemente abgelegt werden. Im Test-Explorer besteht die Möglichkeit sich diesen Quellcode anzeigen zu lassen, er dient in der Regel zur Fehlersuche und ist eher ein Feature für Entwickler.



Testfall umbenennen und löschen

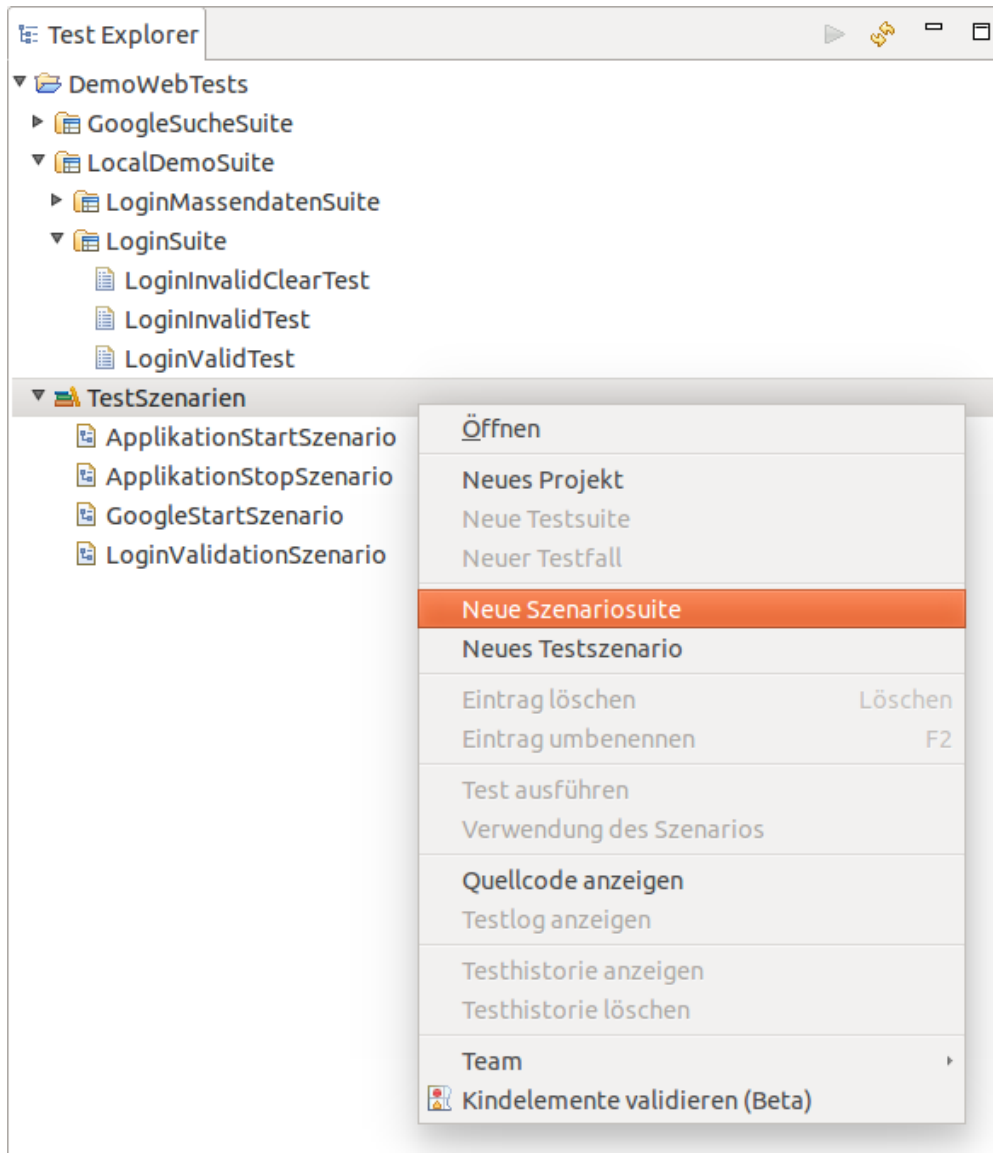
Analog zum **Umbenennen** und **Löschen** von Testsuiten können Testfälle über das Kontextmenü im Test-Explorer umbenannt bzw. gelöscht werden.

Mit Szenariensuiten arbeiten

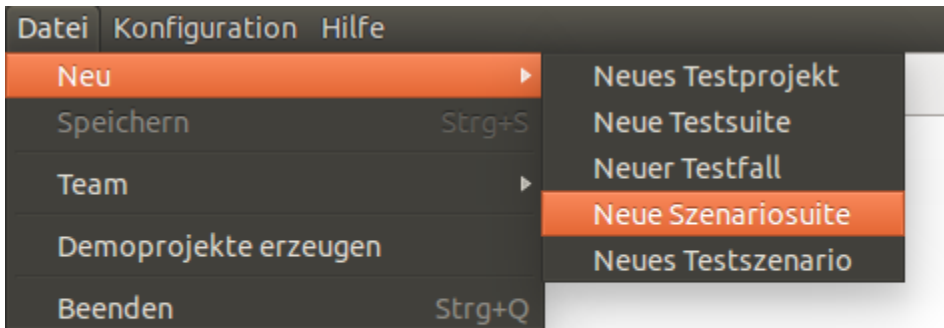
Szenariensuiten werden im Kontext eines Projektes erzeugt und dienen als Gruppierung von mehreren **Szenarien** und Unter-Szenariensuiten. Suites sind vergleichbar mit Ordnern. Neben der Möglichkeit Szenarien als Kind-Elemente zu erzeugen, gibt es auch die Möglichkeit bestehende Szenarien in die Szenariensuite zu referenzieren.

Szenariensuite anlegen

Eine neue Szenariensuite kann über das **Kontextmenü (Test-Explorer)** auf einem beliebigen **Projekt oder Szenariensuite** angelegt werden:

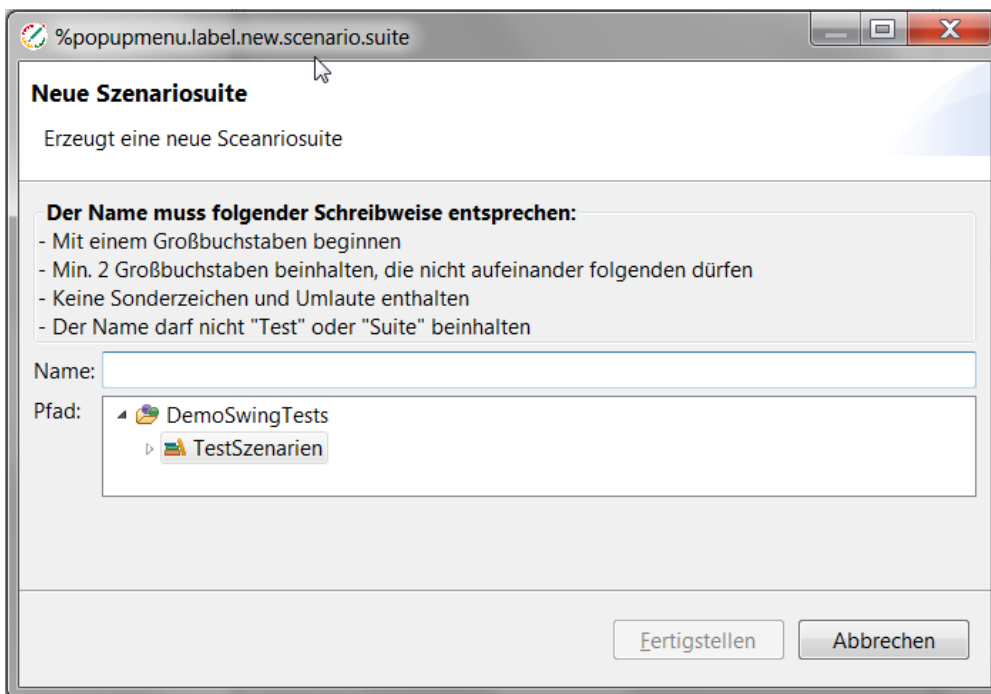


Alternativ steht die Funktion über **Menüeintrag Datei -> Neu -> Neue Testsuite** zur Verfügung:



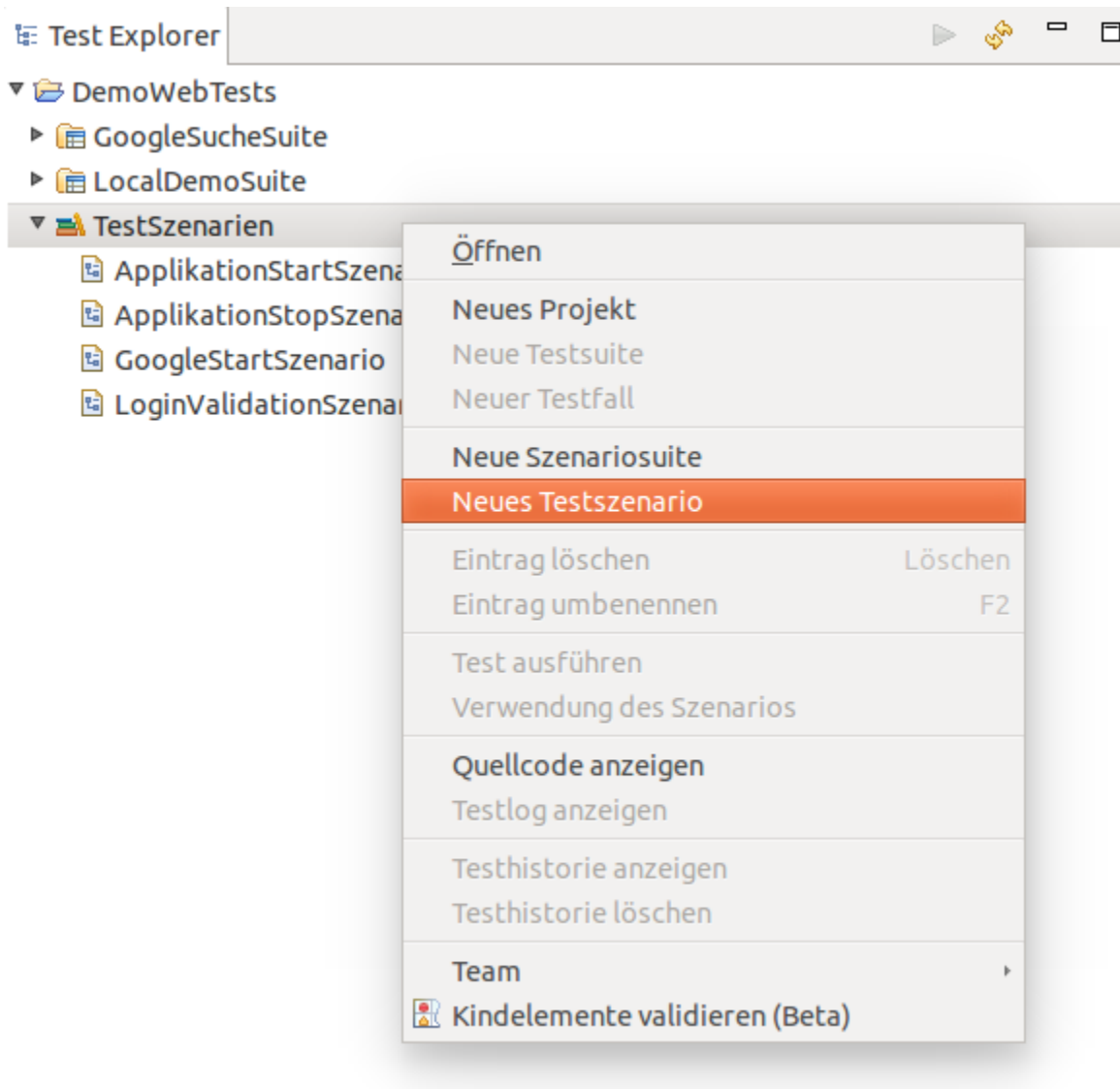
Es öffnet sich der **Dialog Neue Szenariosuite**:

- In das **Feld Name** wird der innerhalb des Vater-Elements (Projekt oder Szenariosuite) der eindeutige Name für die Suite festgelegt. Wichtig zu beachten sind die Namenskonventionen, die im Dialog angezeigt werden
- In dem **Bereich Pfad** kann gewählt werden, wo in der Hierarchie die **Suite** erzeugt werden soll

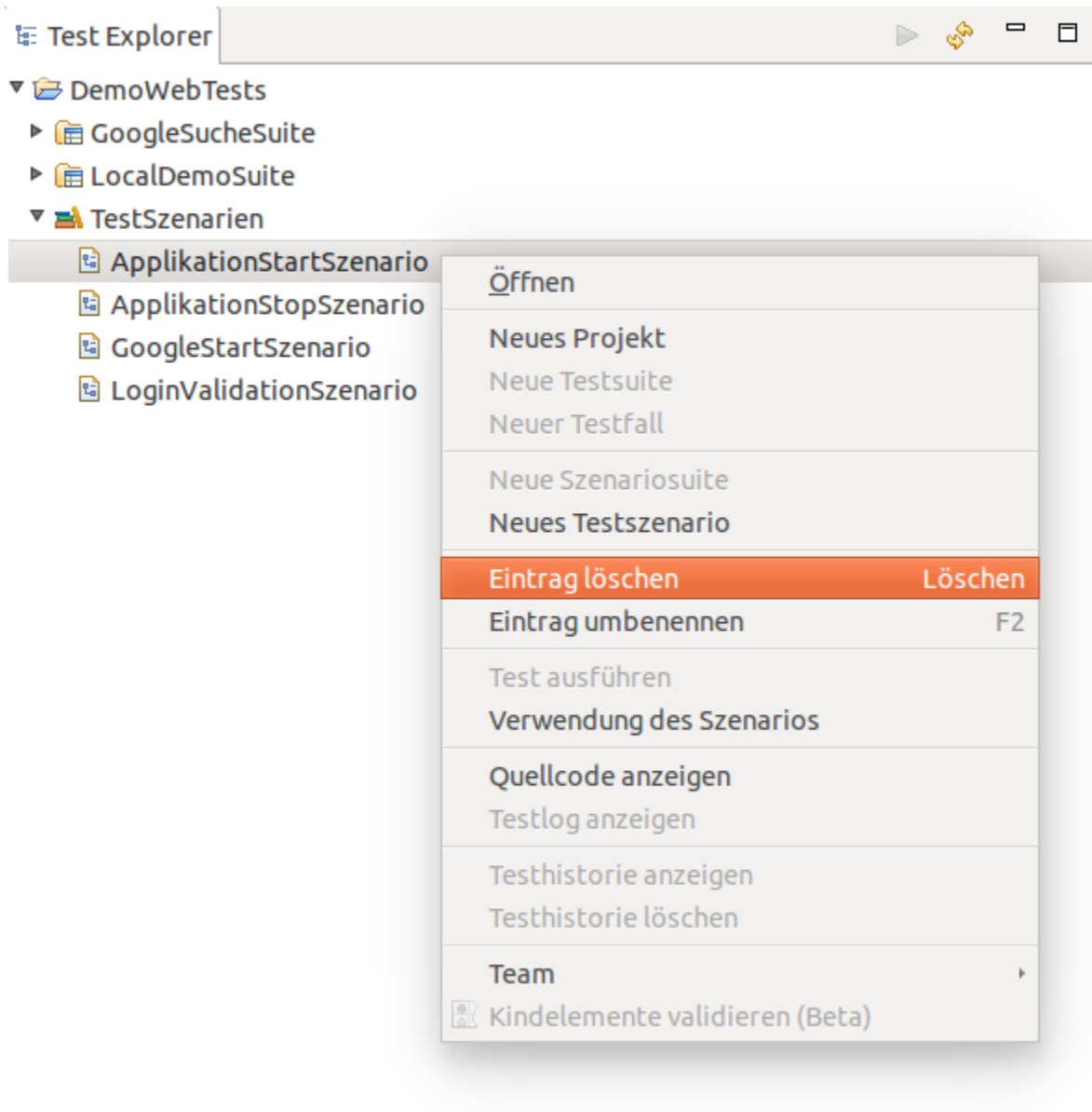


Szenariosuite bearbeiten

Mittels des Kontext-Menüs können einer Szenariosuite neue Szenarien oder weitere Szenariosuiten hinzugefügt werden.



Über das Kontext-Menü kann ein Testscenario auch wieder aus der Szenariosuite gelöscht werden.



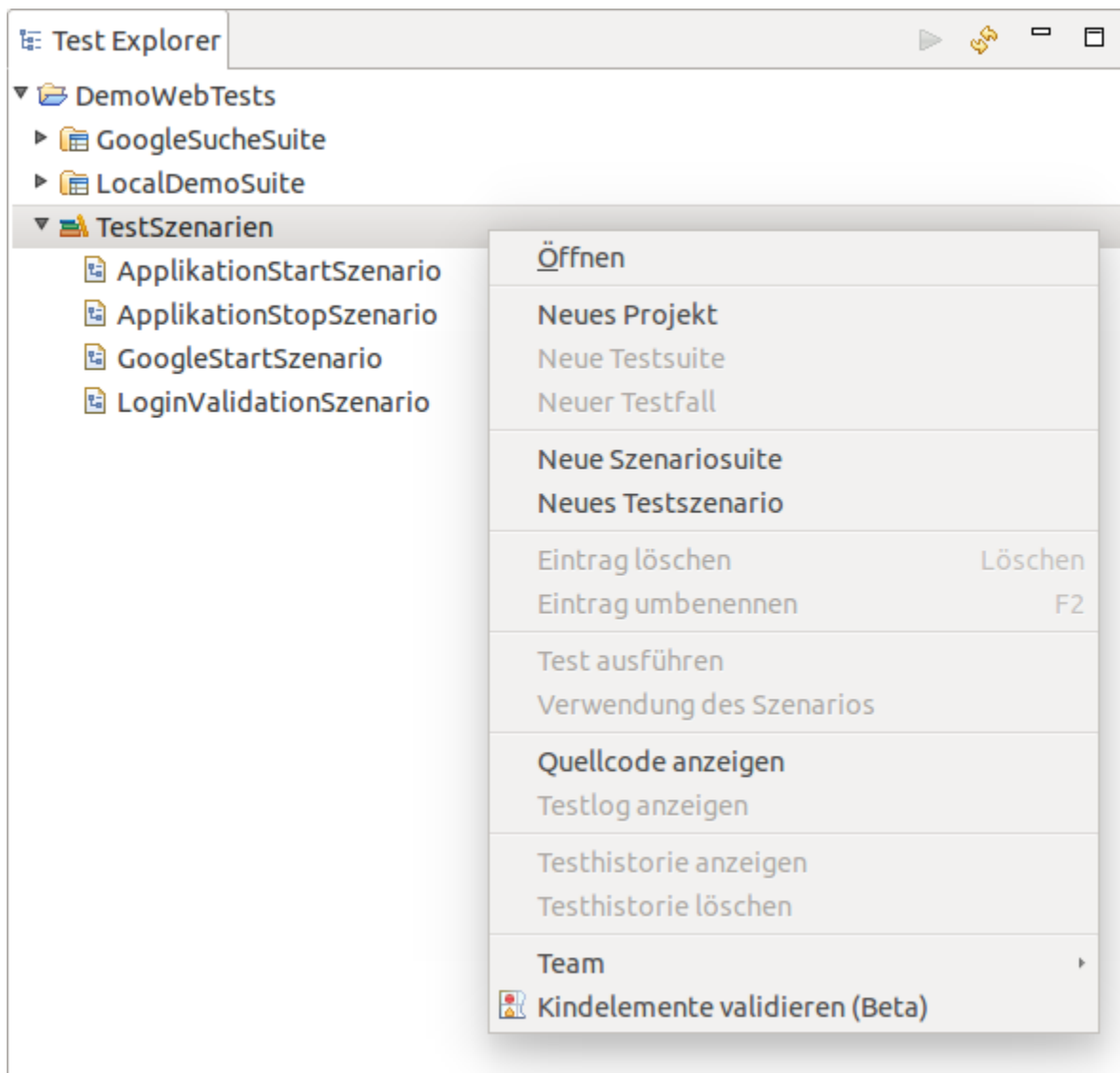
Hinweis: Wenn man von einem Szenario ein anderes Szenario einliest, kann die angezeigte Wertetabelle nur eine Zeile aufnehmen. Die Tabelle kann nicht erweitert werden.

Szenarien in Testfällen verwenden

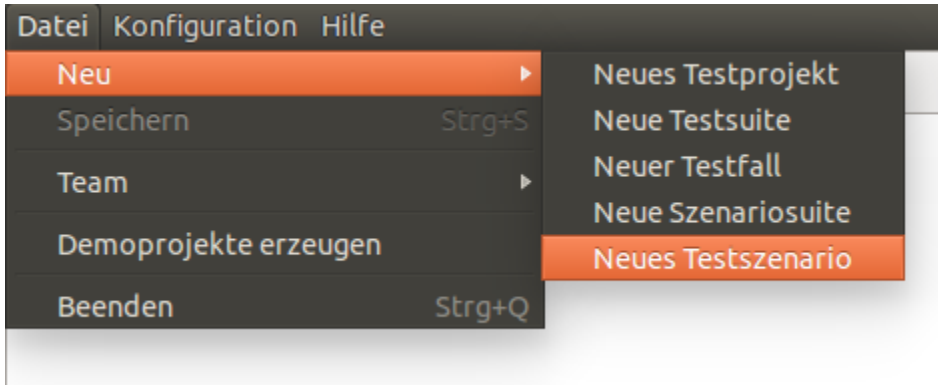
Szenarien dienen als Schablone um wiederkehrende **Testschritte** zentral und nur einmal zu definieren. Ein Szenario kann dann in **Testfällen** wieder verwendet werden.

Szenarien anlegen

Ein neues Szenario kann über das **Kontextmenü (Test-Explorer)** auf **TestSzenario** angelegt werden:

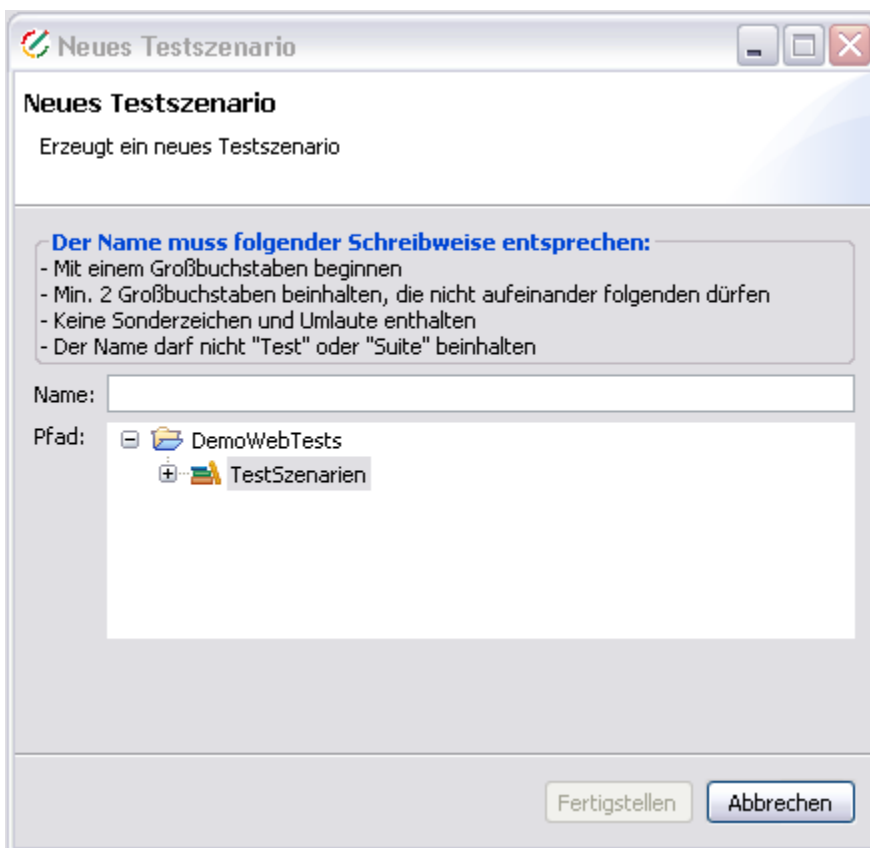


Alternativ steht die Funktion über den Menüeintrag **Datei -> Neu -> Neues Testszenario** zur Verfügung:



In beiden Fällen öffnet sich der **Dialog Neues Testszenario**:

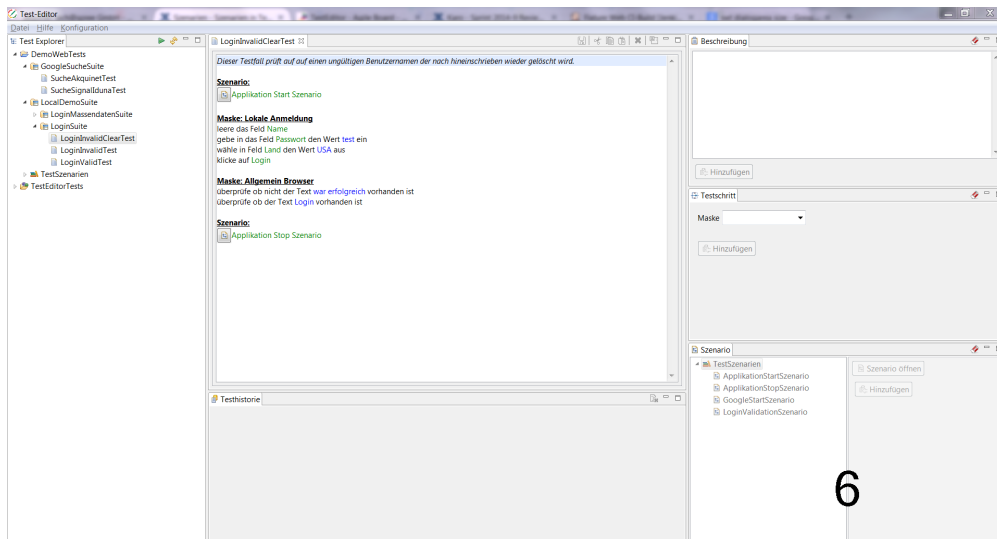
- In das **Feld Name** wird der eindeutige Name für das Szenario festgelegt. Wichtig zu beachten sind die Namenskonventionen, die im Dialog angezeigt werden
- In der **Pfad-Angabe** kann nur das jeweilige Projekt ausgewählt werden und **Szenariosuite** ausgewählt werden



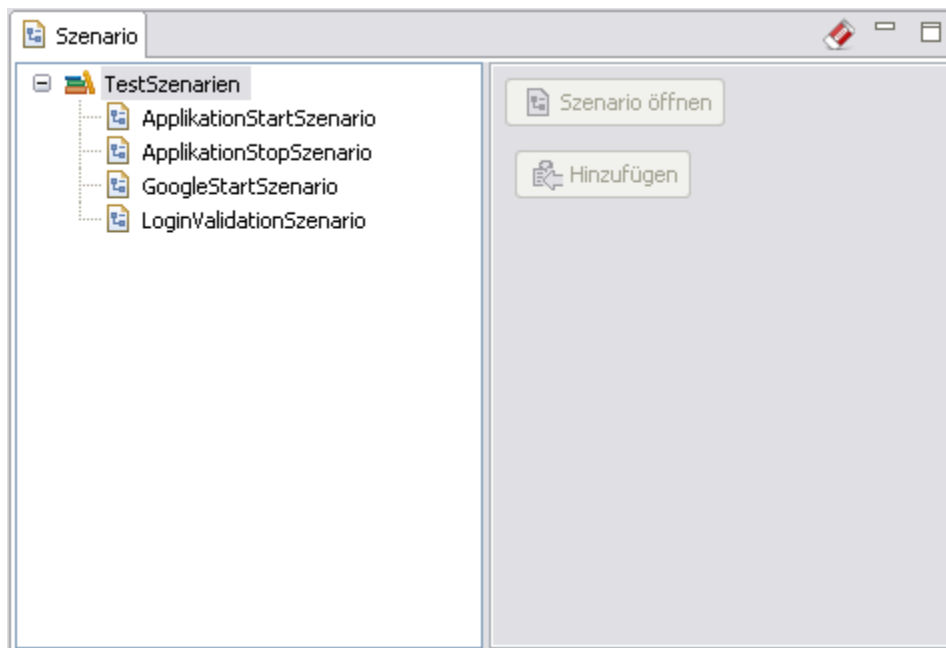
Das Szenario wird vergleichbar zum Testfall erfasst (einzelne Testschritte können definiert werden, etc.). Zusätzlich können Platzhalter (Parameter) bei der Erfassung von Testschritten mit dem Schlüssel @ (z.B. @name) definiert werden, dann erwartet das Szenario, dass ein konkreter Wert bei der Verwendung des Szenarios übergeben wird (siehe weiter unten). Der Platzhalter wird dann bei der Testausführung durch den Wert ersetzt.

Szenarien in Testfällen verwenden

Szenarien können in Testfällen oder wieder in anderen Szenarien verwendet werden. Die Verwendung ist identisch und die Beschreibung hier geht von der Verwendung in einem Testfall aus. Ein Szenario wird im Bereich Szenario (Bereich 6) ausgewählt und so dem Testfall hinzugefügt:



Beispiel: Das Szenario *ApplikationStartSzenario* wird ausgewählt und über den **Hinzufügen Button** zum aktuellen Szenario hinzugefügt. Der Button **Szenario Öffnen** dient als Schnellnavigation und öffnet das ausgewählte Szenario.

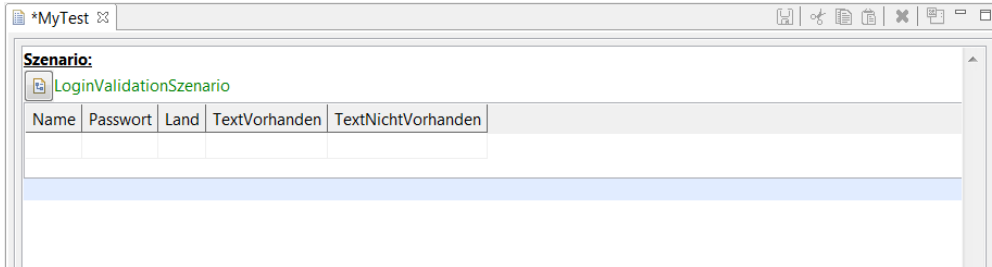


Im Testfall erscheint jetzt eine Referenz und ein Link auf das hinzugefügte Szenario. Wie das Szenario dargestellt wird, hängt davon ab ob das Szenario Parameter erwartet oder nicht.

Beispiel: Referenz ohne Parameter



Beispiel: Referenz mit Parametern (=> leere Tabelle)



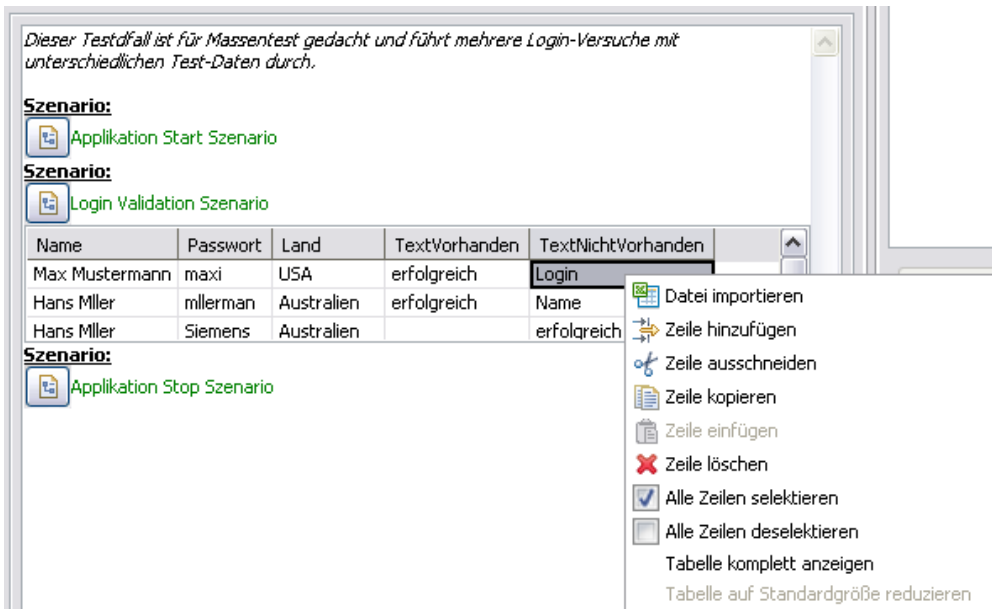
Wenn das Szenario mit Parametern aufgerufen werden muss, so wird in den Testfall zusätzlich eine Tabelle eingefügt. Die Tabelle enthält als Spalten die Parameter des Szenarios.

Szenario Tabelle mit Daten füllen


Die Tabelle kann einerseits **direkt im Editor-Bereich** editiert werden, andererseits bietet der *Test-Editor* die Möglichkeit die Daten **via CVS bzw. Excel** zu importieren.


Daten im Editor editieren

Es können ganze Zeilen in der Tabelle über das Kontextmenü editiert werden. Für das **Kontextmenu in der Zeile** muss die gesamte Zeile markiert werden. Neben den Editierfunktionen kann auch über das Kontextmenü die gesamte Tabelle angezeigt werden. Die Tabelle wird dann vollständig angezeigt und kann einfacher bearbeitet werden.




Dieser Testfall ist für Massentest gedacht und führt mehrere Login-Versuche mit unterschiedlichen Test-Daten durch.

Szenario:
 Applikation Start Szenario

Szenario:
 Login Validation Szenario

Name	Passwort	Land	TextVorhanden	TextNichtVorhanden
Max Mustermann	maxi	USA	erfolgreich	Login
Hans Mller	mllerman	Australien	erfolgreich	Name
Hans Mller	Siemens	Australien		erfolgreich
Moxen	chicago	USA		Login
Hilde	swiz	Schweden	erfolgreich	Deutschland
Jan	admin	Deutschland		erfolgreich

Szenario:
 Applikation Stop Szenario

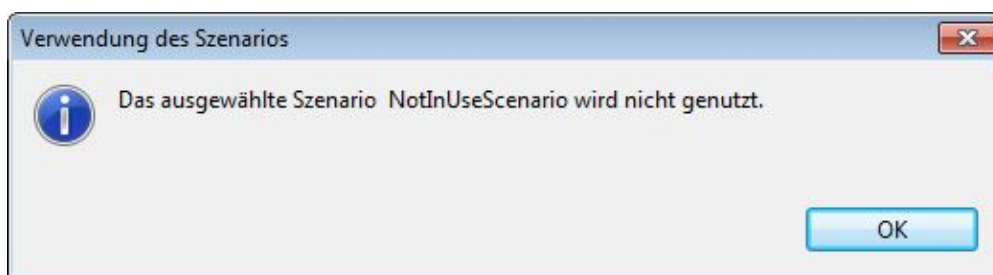
Datenimport via CSV oder Excel

Daten in die Tabelle können aus Excel oder aus einer CSV-Datei importiert werden. Dazu muss die erste Zeile in der Tabelle markiert werden und über das **Kontextmenu in der Tabelle** oder den **Button Excel-Import im Editor-Toolbar** der Import gestartet werden. Alle Spalten, die in der Tabelle des Szenarios vorhanden sind, müssen auch in der CSV- oder Excel vorhanden sein. Im Importprozess werden die Spalten automatisch in der Reihenfolge im Szenario importiert.

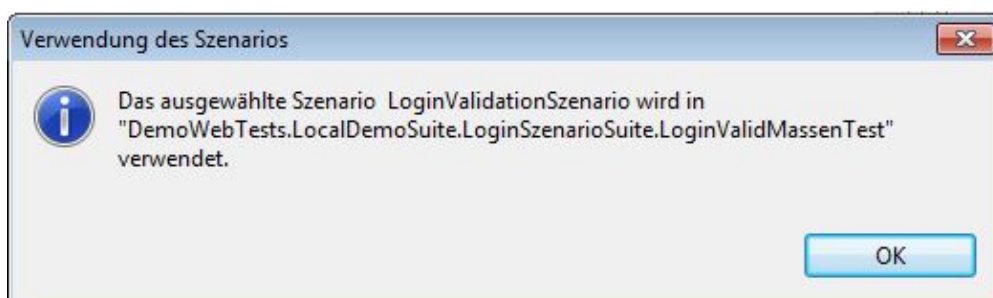
Szenarien Verwendung anzeigen

Über das **Kontextmenü (Test-Explorer) auf einem beliebigen Testszenario -> Verwendung des Szenarios** kann angezeigt werden, welche Testfälle das Szenario referenzieren.

Das folgende Beispiel zeigt, dass das Szenario in keinem Testfall verwendet wird:



Während dieses Szenario in mehreren Testfällen verwendet wird:



Wird ein Szenario verwendet, so kann es weder gelöscht noch umbenannt werden (die Kontextmenü Einträge sind deaktiviert).

Besonderheiten bei der Verwendung von Szenarien in Szenarien


Bei der Verwendung von Szenarien in Szenarien ist folgendes zu beachten:

1. Wenn das verwendete Szenario Aufrufparameter erwartet, so müssen diese an der Stelle des Aufruf angegeben werden.

Scenario Parameter:

Dieses Szenario startet die lokale Demo-Web-Applikation. Die Applikation ist bereits Bestandteil dieses Demo-Projektes, so dass keine Internet-Verbindung für die Ausführung notwendig ist. Der zu verwendene Browser kann hier bei Bedarf umgestellt werden.

Szenario:

 Seite In Browser Oeffnen Szenario	
Page	
http://localhost:8060/files/demo/ExampleApplication/WebApplicationDe/index.html	

2. Es kann nur ein Set von Parametern angegeben werden. Daher hat die Tabelle in der Darstellung auch nur eine Zeile und kann nicht erweitert werden. Ein Datenimport kann vorgenommen werden.


Tests ausführen


Ein **Test** ist eine konkrete Ausführung eines **Testfalles**. Es können auch **Testsuiten** als Test gestartet werden, was dazu führt das alle Testfälle der Suite als Test ausgeführt werden.

Test starten


Ein Test kann für einen Testfall oder Testsuite über das **Kontextmenü im Test-Explorer** ausgeführt werden. **Alternativ** steht der **grüne Ausführungs-Button** im Test-Explorer zur Verfügung:







Dieser Testfall ist für Massentest gedacht und führt mehrere Login-Versuche mit unterschiedlichen Test-Daten durch.

Szenario:
 Applikation Start Szenario

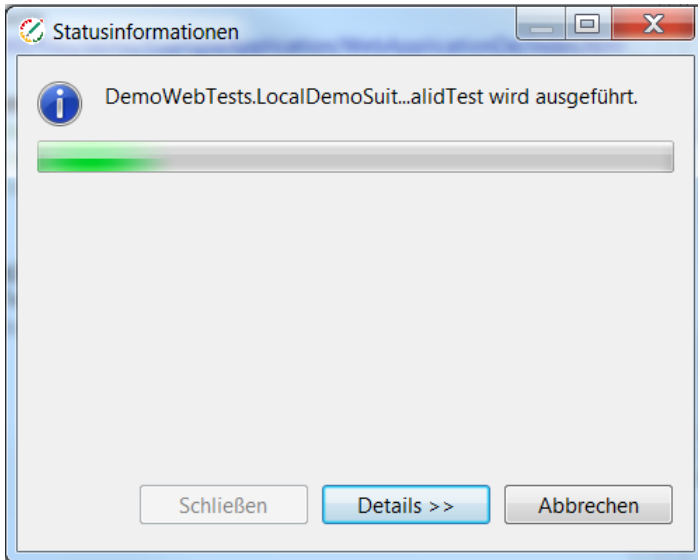
Szenario:
 Login Validation Szenario

Name	Passwort	Land	TextVorhanden	TextNichtVorhanden
Max Mustermann	maxi	USA	erfolgreich	Login
Hans Mller	mllerman	Australien	erfolgreich	Name
Hans Mller	Siemens	Australien		erfolgreich

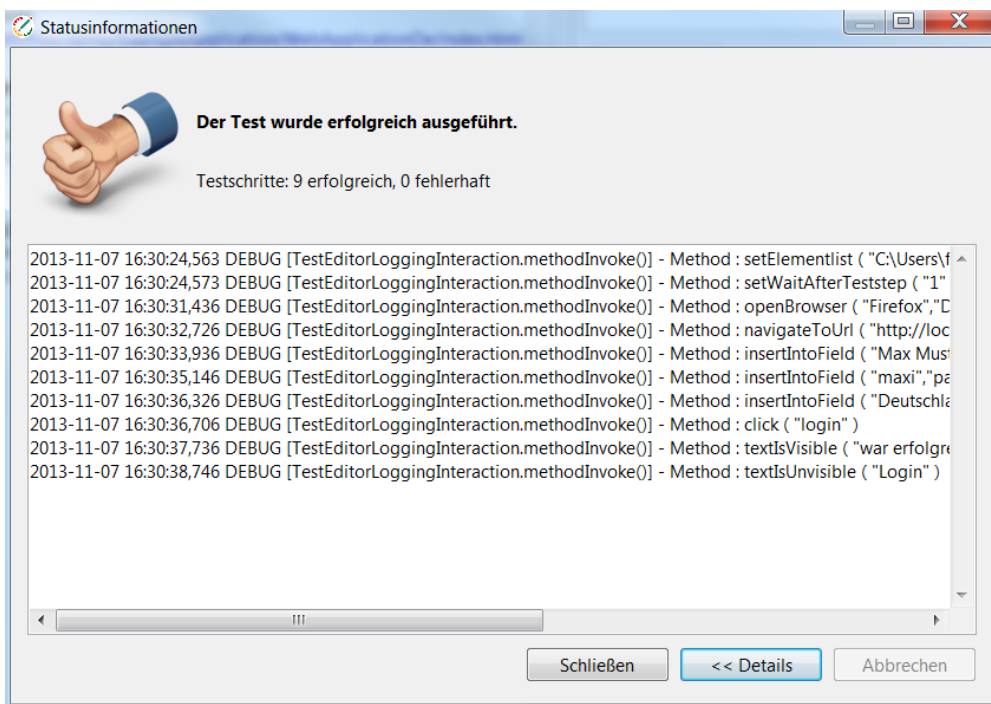
Szenario:
 Applikation Stop Szenario

-  Datei importieren
-  Zeile hinzufügen
-  Zeile ausschneiden
-  Zeile kopieren
-  Zeile einfügen
-  Zeile löschen
- ☒ Alle Zeilen selektieren
- ☐ Alle Zeilen deselektieren
- Tabelle komplett anzeigen
- Tabelle auf Standardgröße reduzieren

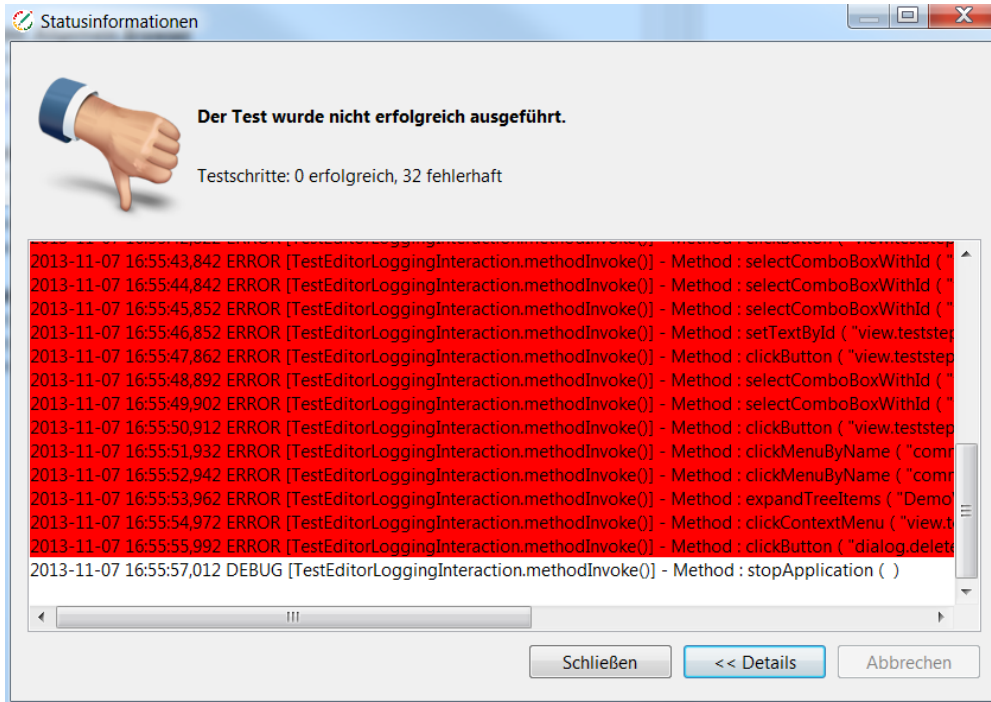
Es öffnet sich ein neuer Dialog, der den **Status zur Testausführung** zeigt:



Nach erfolgreich ausgeführtem Test wird ein Dialog mit den Testergebnissen angezeigt:

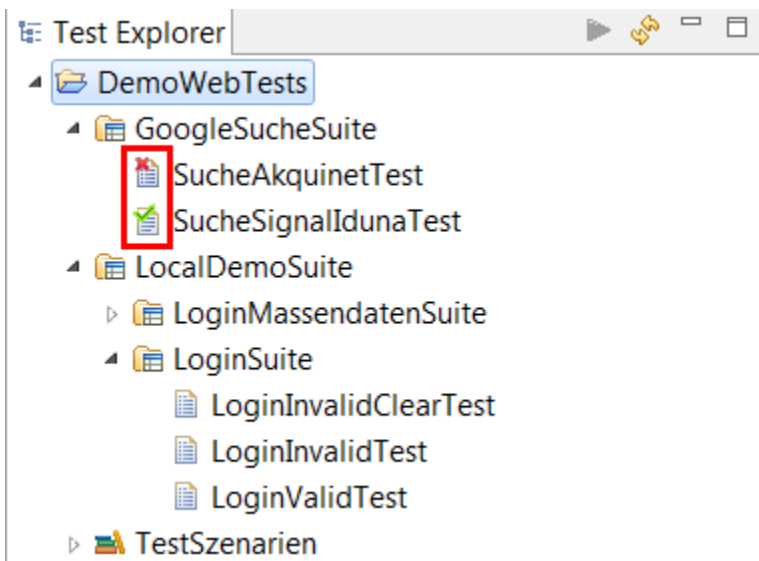


Wird der Test nicht erfolgreich ausgeführt, wird ein Dialog vergleichbar mit Folgendem angezeigt (die Zeilen mit dem Fehler sind rot markiert):



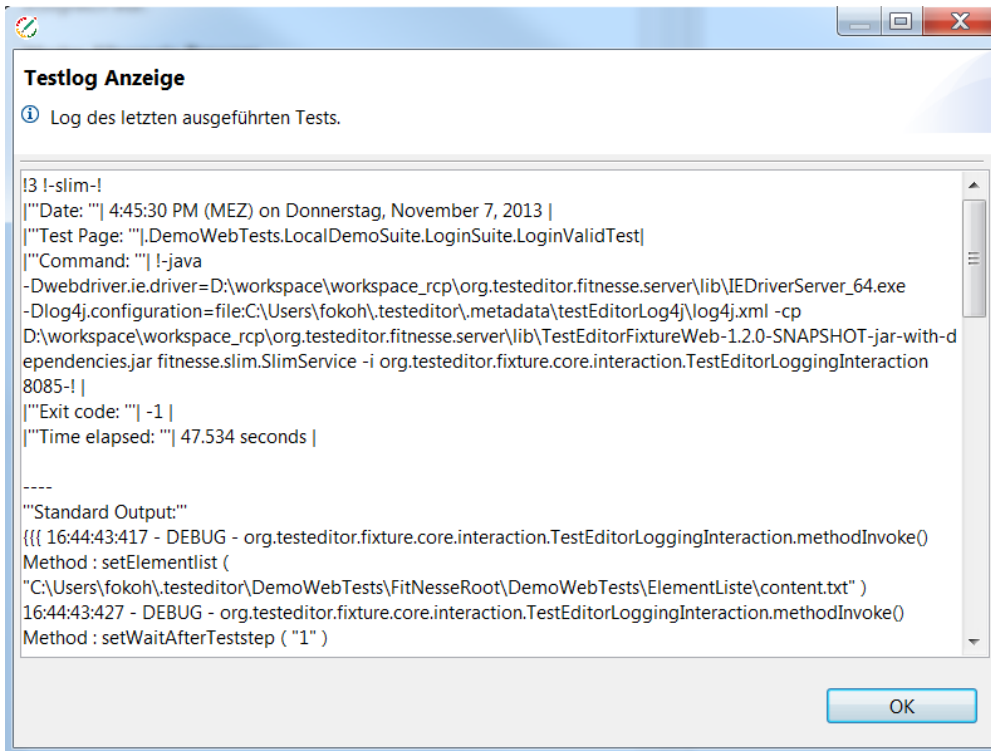
Ergebnisse der Tests erkennen

Auch nach der Testausführung ist im **Test-Explorer** erkennbar, ob ein Testfall zuletzt erfolgreich oder nicht erfolgreich getestet wurde. Der Testfall ist im positiven Fall mit einem **grünen Haken** markiert, im negativen Fall mit einem **roten Kreuz**.



Testlog anzeigen

Wurde ein Test ausgeführt, kann das **Testlog** über das **Kontextmenü im Test-Explorer** angezeigt werden:

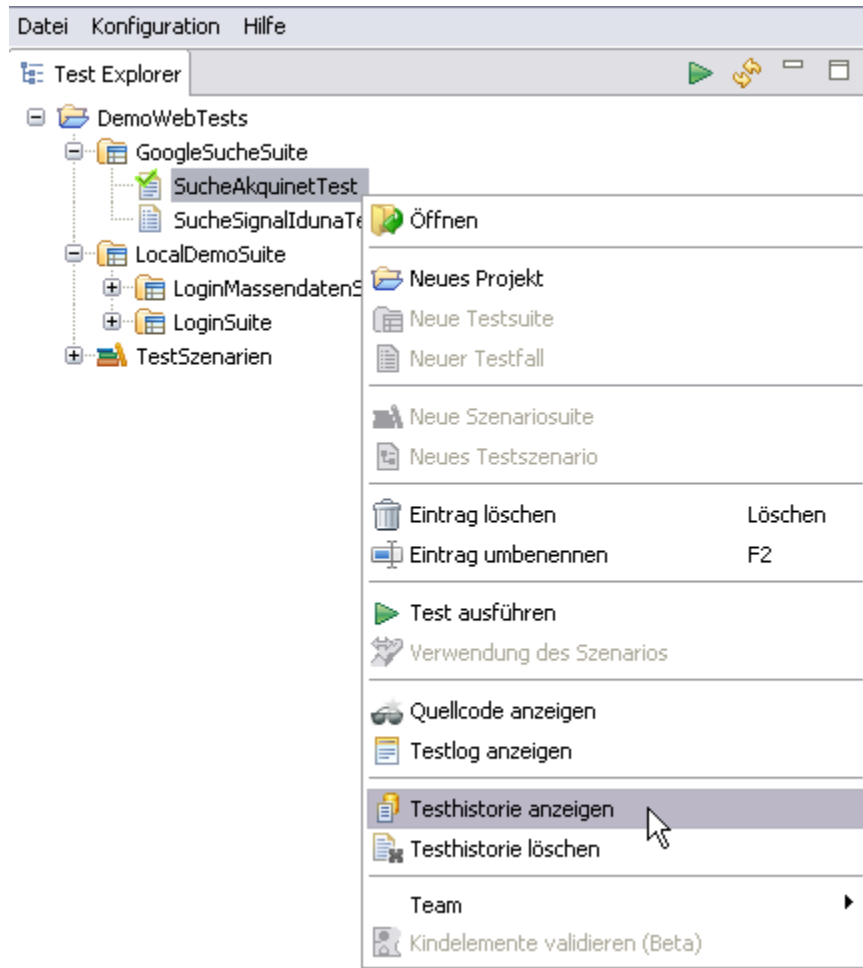


Testhistorie anzeigen

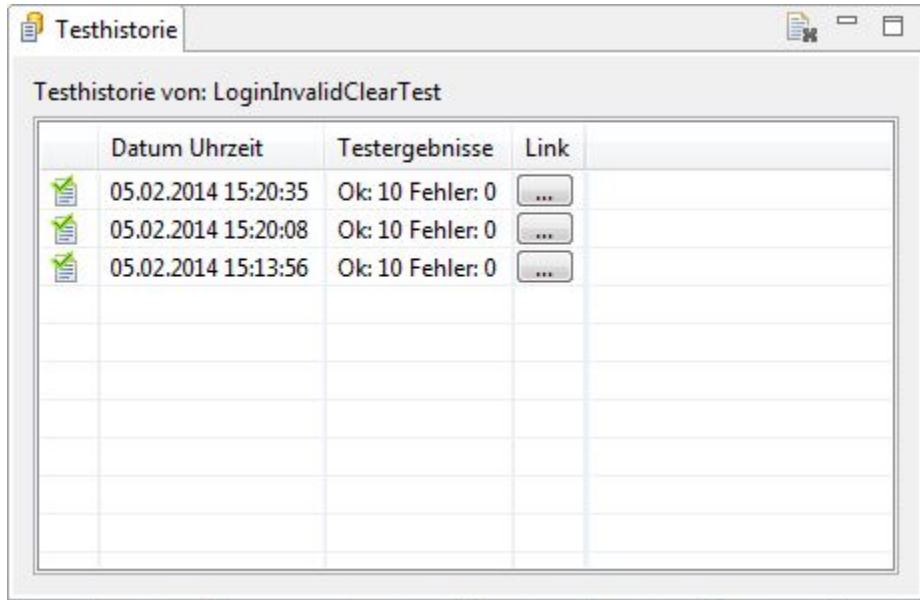
Wenn ein **Test** für einen **Testfall** oder eine **Suite** ausgeführt wurde, kann dazu die Historie angezeigt werden.

Öffnen der Testhistorie

Die Testhistorie zu einem Test, eines Testfalls oder einer Suite kann über das **Kontextmenü im Test-Explorer** geöffnet werden.



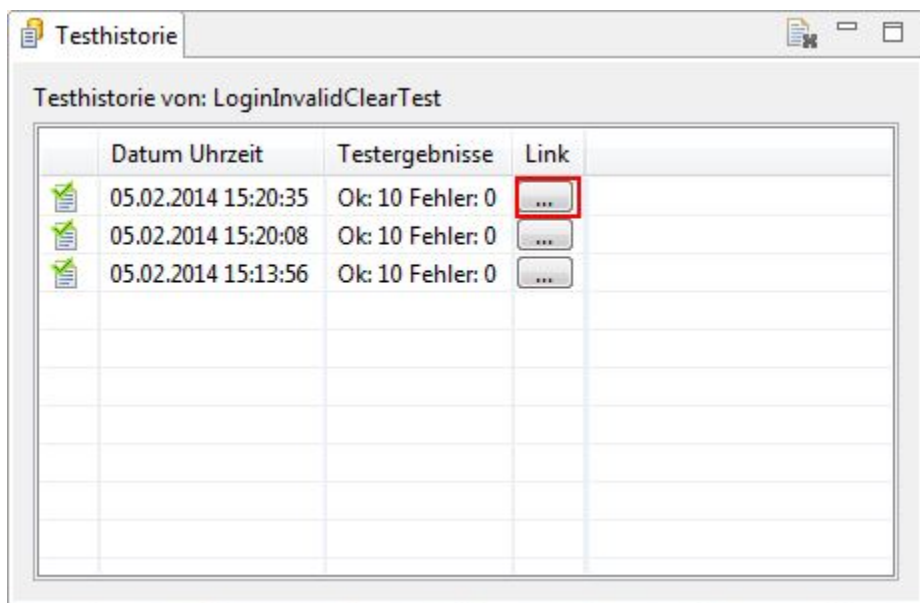
Anschließend wird in dem Bereich Testhistorie die Historie der Testläufe des ausgewählten Objekts angezeigt.



The screenshot shows a window titled 'Testhistorie' with a sub-header 'Testhistorie von: LoginInvalidClearTest'. It contains a table with four columns: 'Datum Uhrzeit', 'Testergebnisse', and 'Link'. The first column also contains status icons (green checkmarks). The table lists three test runs, all with a status of 'Ok' and 10 errors.

	Datum Uhrzeit	Testergebnisse	Link
	05.02.2014 15:20:35	Ok: 10 Fehler: 0	...
	05.02.2014 15:20:08	Ok: 10 Fehler: 0	...
	05.02.2014 15:13:56	Ok: 10 Fehler: 0	...

Die erste Spalte enthält den Status des Testlaufs, die zweite das Ausführungsdatum, die dritte das Testergebnis als Summe der erfolgreichen und nicht erfolgreichen Schritte. In der vierten Spalte ist ein Link zu der Detailansicht der Testhistorien, die über den Fitness-Server erreicht wird, enthalten.



This screenshot is identical to the one above, but with a red rectangular box highlighting the first 'Link' cell, which contains an ellipsis (...).

	Datum Uhrzeit	Testergebnisse	Link
	05.02.2014 15:20:35	Ok: 10 Fehler: 0	...
	05.02.2014 15:20:08	Ok: 10 Fehler: 0	...
	05.02.2014 15:13:56	Ok: 10 Fehler: 0	...



DemoWebTests > LocalDemoSuite > LoginSuite
LoginInvalidClearTest

View

Wed Feb 05 15:20:35 CET 2014 | FitNesse Version: v20121220

LoginInvalidClearTest	10 Right	0 Wrong	0 Ignores	0 Exceptions	20191 ms
------------------------------	----------	---------	-----------	--------------	----------

► [Precompiled Libraries](#)

[Expand All](#) | [Collapse All](#)

Dieser Test prüft auf einen ungültigen Benutzernamen der nach hineinschreiben wieder gelöscht wird.

▼ [Included page: <DemoWebTests.TestKomponenten.BrowserStartSzenario \(edit\)](#)

[Expand All](#) | [Collapse All](#)

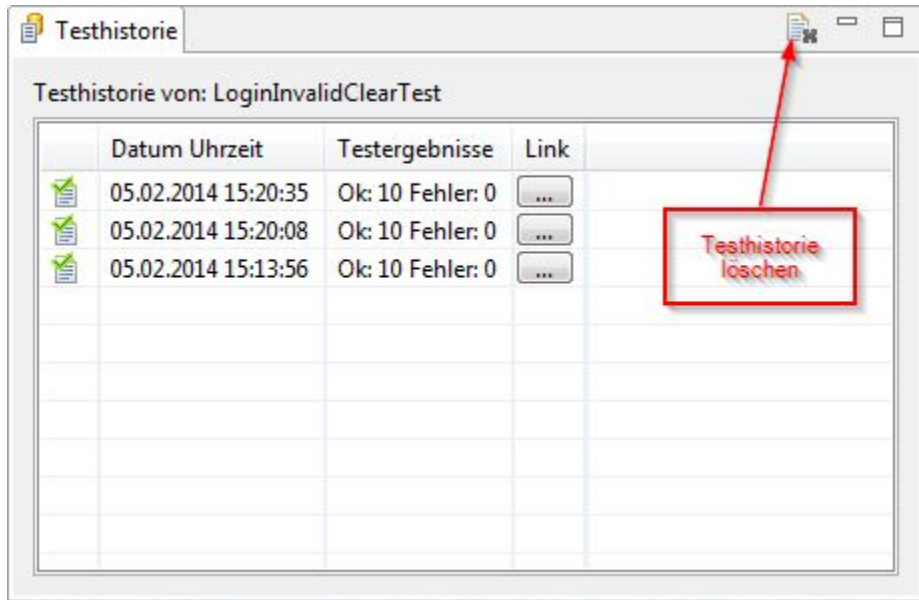
scenario	BrowserStartSzenario _
note	Maske: Allgemein Browser
starte Browser	Firefox

script

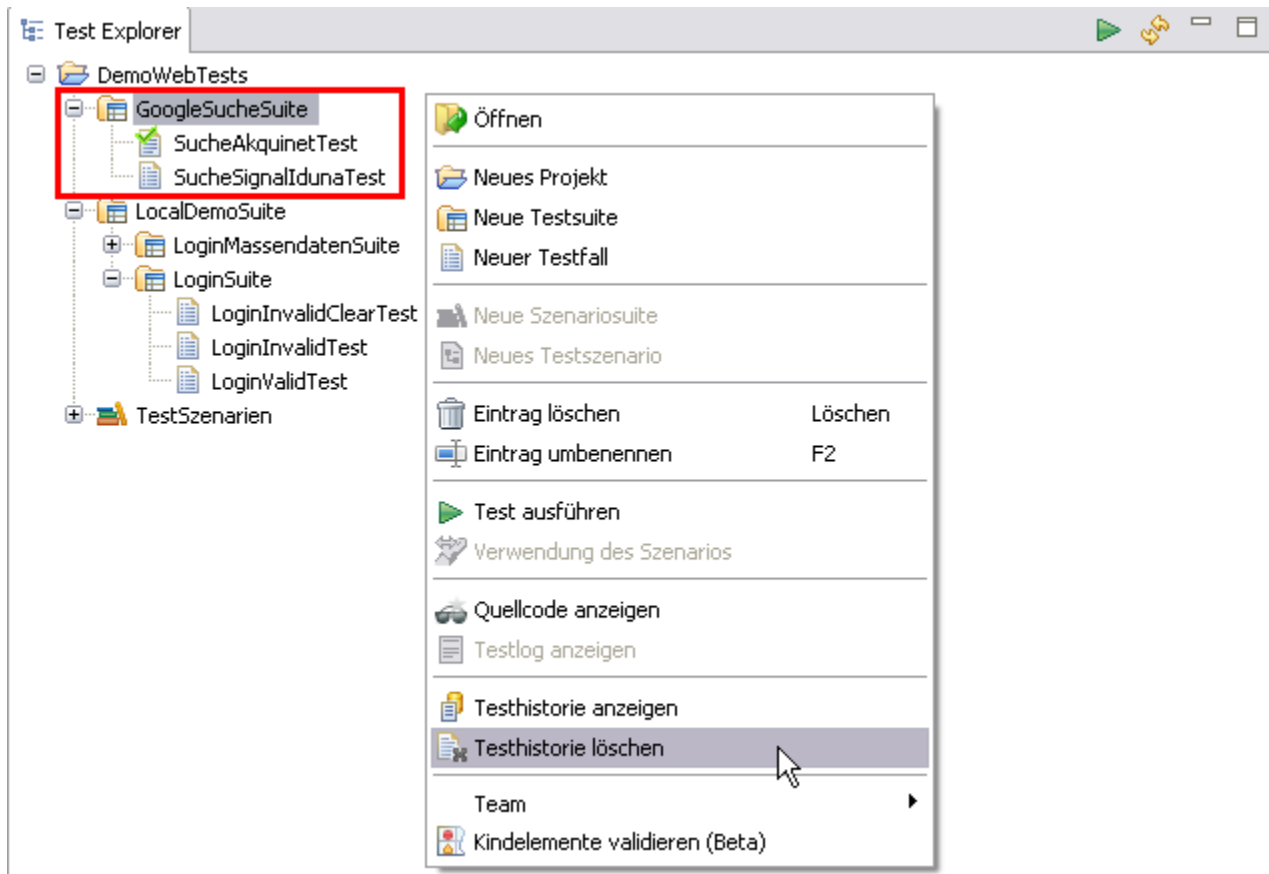
► Browser Start Szenario				
► navigiere auf die Seite	http://localhost:8060/files/demo/ExampleApplication/WebApplicationDe/index.html			
► gebe in das Feld	user	den Wert	Max Mustermann	ein
► leere das Feld	user			
► gebe in das Feld	password	den Wert	test	ein
► wähle in Feld	land	den Wert	USA	aus
► klicke auf	login			
► überprüfe ob nicht der Text	war erfolgreich		vorhanden ist	
► überprüfe ob der Text	Login		vorhanden ist	
► beende Browser				

Löschen der Testhistorie

die angezeigte Testhistorie kann direkt mit dem Löschenbutton gelöscht werden



Weiterhin kann die Testhistorie über das Menü in dem Explorerbaum gelöscht werden. Dann wird die Testhistorie aller selektierten Teststrukturen und von deren Kindern (untergeordneten Objekten) gelöscht.



In diesem Beispiel werden dann die Testhistorien von "SucheAkquinetTest" und von "SucheSignalIdunaTest" und deren Kindern gelöscht.

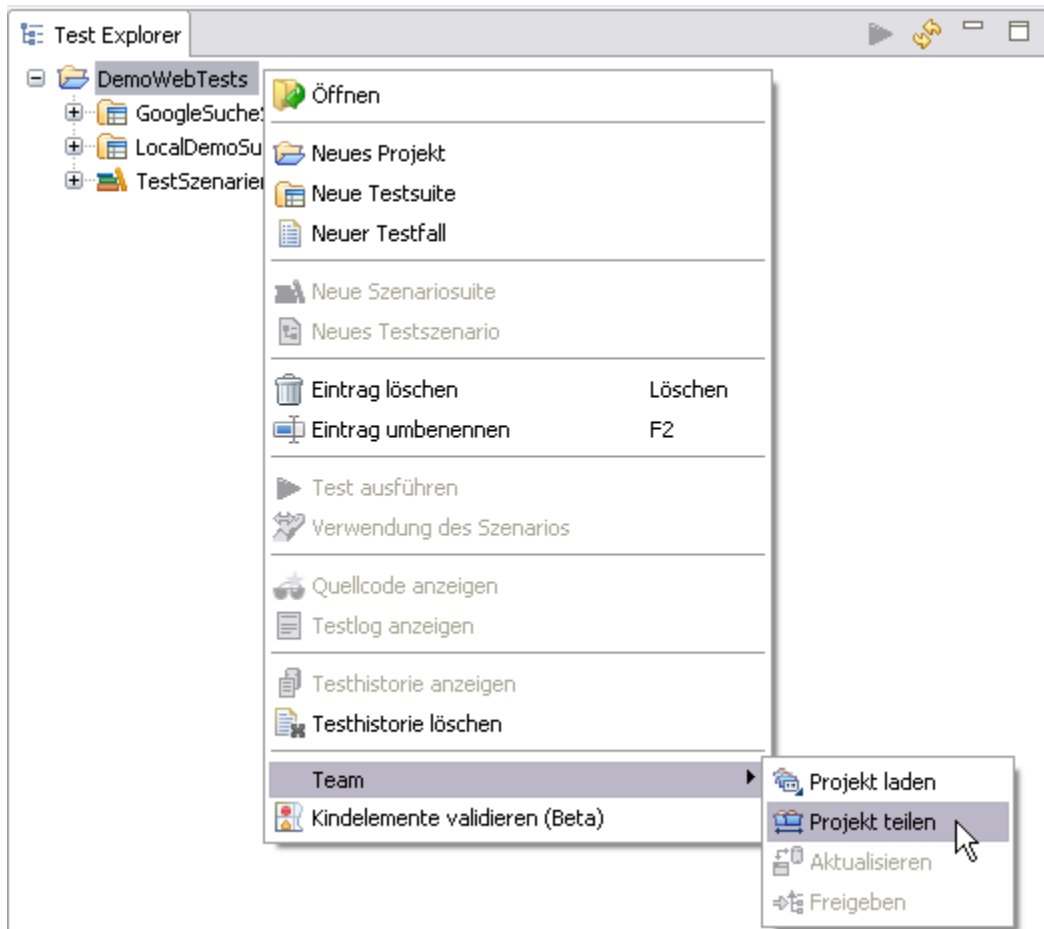
Projekte im Team teilen

Im *Test-Editor* ist die Benutzung von SVN als Versionsverwaltung integriert. Der *Test-Editor* unterstützt folgende Operationen:

- Testprojekt anderen Teammitgliedern zur Verfügung stellen
- Testprojekte aus der Versionsverwaltung laden
- Änderungen an Testobjekte dem Team zur Verfügung stellen
- Testobjekte aus der Versionsverwaltung aktualisieren

Wenn diese Möglichkeit in einem Projekt angeboten werden soll, so muss ein entsprechendes SVN-System eingerichtet werden (das SVN-System bzw. die benötigten Daten sollten vom Administrator zur Verfügung gestellt werden).

Projekte anderen Teammitgliedern zur Verfügung stellen



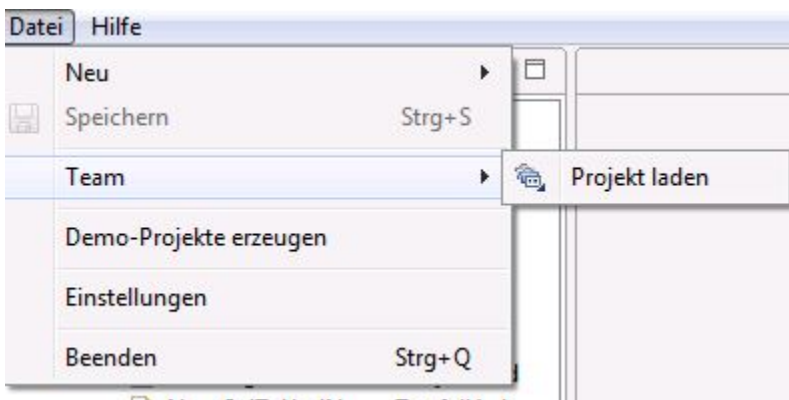
Über den Menüpunkt an einem Projekt im Explorerbaum *Team* -> *Projekt teilen* wird folgender Dialog geöffnet:

The dialog box is titled 'Projekt teilen' and contains the instruction 'Ein Projekt über eine Versionskontrolle im Team bereitstellen.' Below this is a 'Hinweis' section stating 'Alle Felder der Konfiguration müssen gefüllt werden.' The main area has a 'Teilen über:' label followed by a dropdown menu set to 'Subversion'. Below the dropdown are three input fields labeled 'URL:', 'Benutzername:', and 'Passwort:'. At the bottom right are two buttons: 'Fertigstellen' and 'Abbrechen'.

Nach Eingabe der Werte für die Felder SVN-Adresse, Benutzername und Passwort wird das Projekt in SVN hochgeladen (SVN checkin).

Testprojekte aus dem SVN laden

Über den Menüpunkt *Team* -> *Projekte laden* wird folgender Dialog geöffnet:



Projekt laden

Lädt ein Projekt aus einem Quellverwaltungssystem.

Hinweis
Alle Felder der Konfiguration müssen gefüllt werden.

Projektname:

Teilen über: Subversion ▼

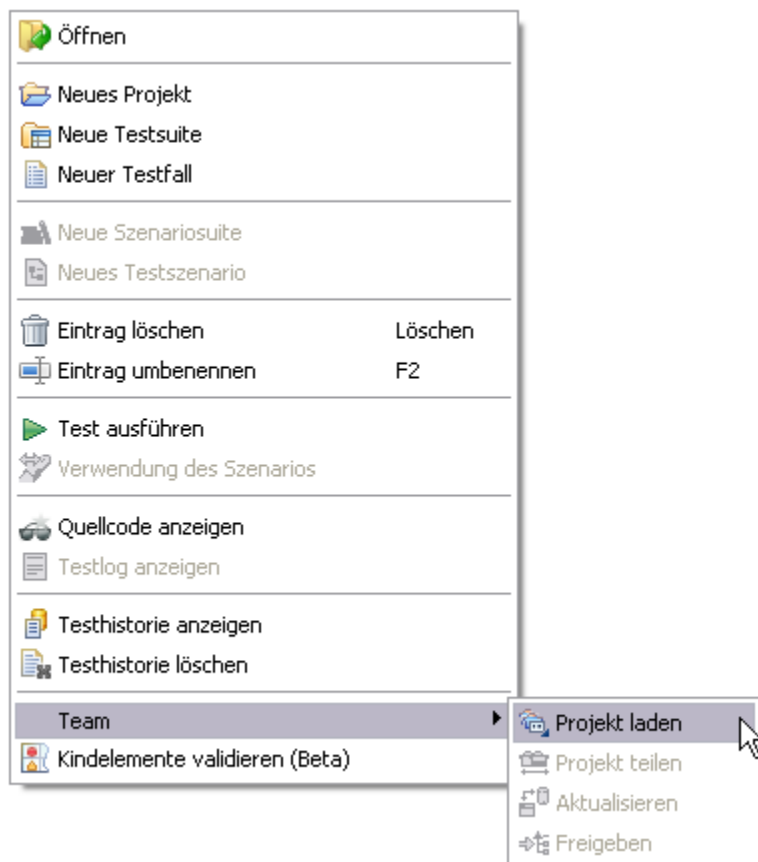
URL:

Benutzername:

Passwort:

Fertigstellen Abbrechen

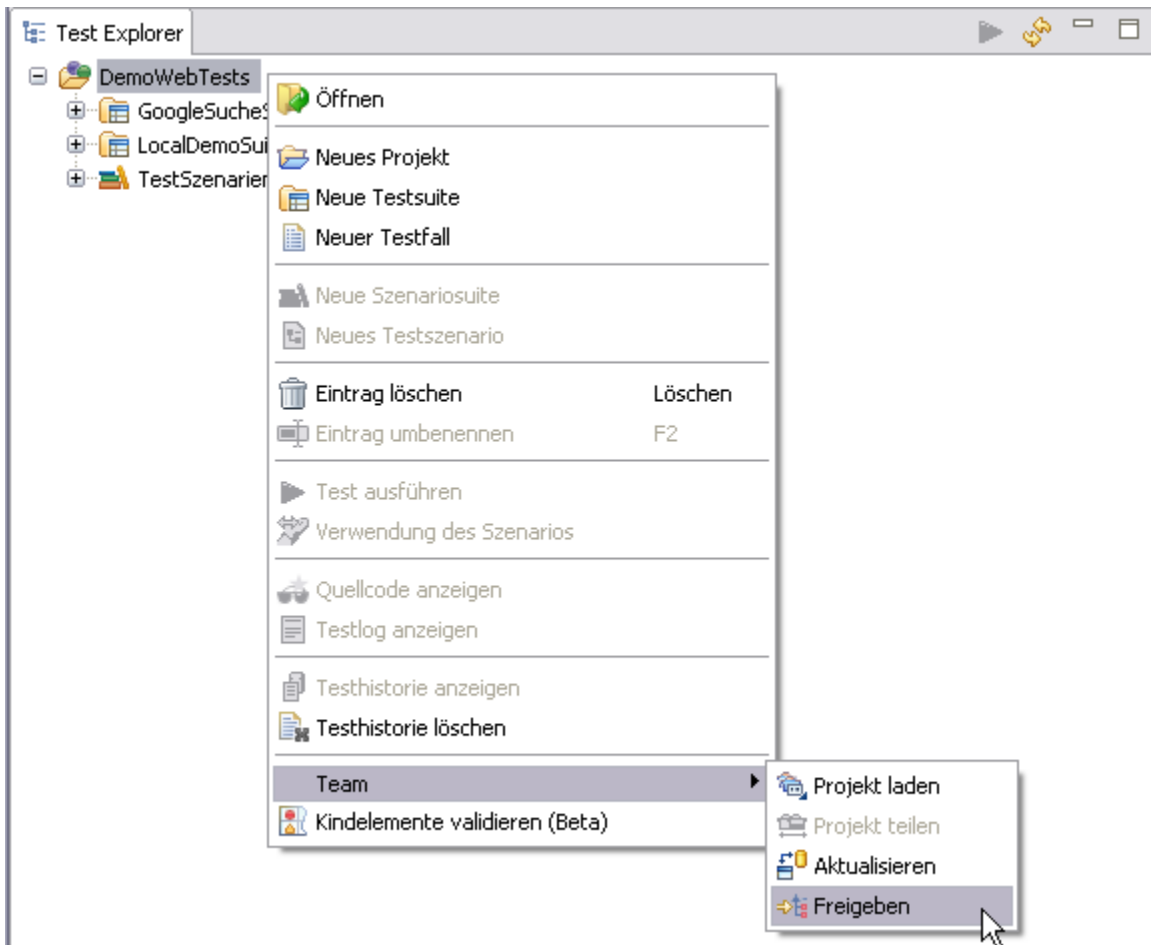
Über das Kontextmenü im Test-Explorerbereich kann man ebenfalls über den Menüpunkt *Team -> Projekte* laden ein neues Projekt laden.



Nach Eingabe der Werte für die Felder Projektname, SVN-Adresse, Benutzername und Passwort kann das Projekt aus dem SVN geladen werden.

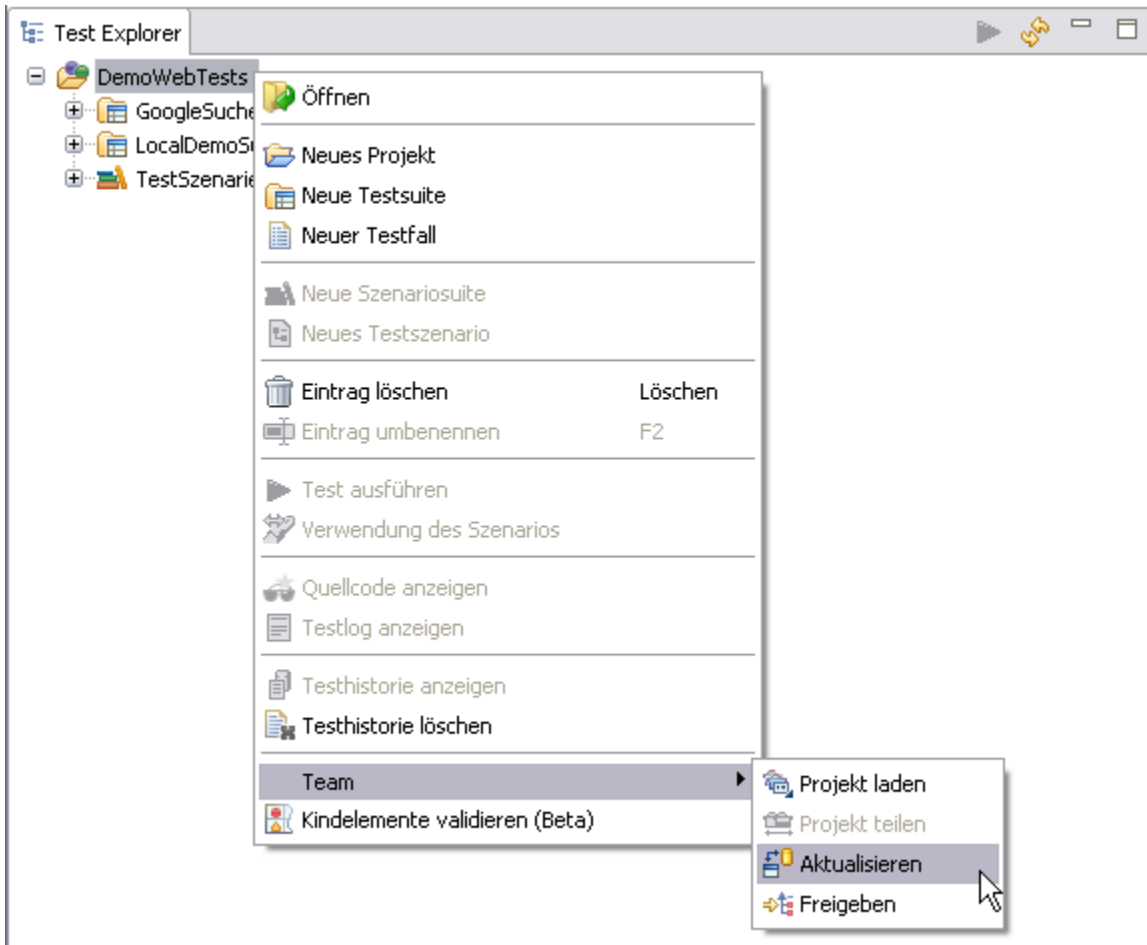
Änderungen an "Testobjekte" dem Team zur Verfügung stellen

Wenn Änderungen an einem Testobjekt vorgenommen wurden, so kann über den Menüpunkt **Team ->Freigegeben** im Test-Explorer, auf dem Testobjekt oder einem darüber liegenden Testobjekt freigegeben werden.



Testobjekte aktualisieren

Die übrigen Teammitglieder können anschließend über den Menüpunkt Team -> *Aktualisieren* im Test-Explorer, auf dem Testobjekt oder einem darüber liegenden Testobjekt die Änderungen aktualisieren.



Glossar

Akzeptanztest

Mit Hilfe des Akzeptanztests oder auch Abnahmetests kann der Anwender überprüfen ob die gewünschten Funktionalitäten einer Software vorhanden sind. Dabei werden die Funktionalitäten anhand der Projektvorgaben überprüft. Der Anwender nutzt die Software und muss entsprechend die technischen Details der Umsetzung nicht kennen. Oft werden Alpha- oder Beta-Tests als Akzeptanztest genutzt. Der Test-Editor unterstützt den Akzeptanztest, indem die Testfälle bereits zur Konzeptionierungsphase erfasst werden können. Dadurch können sinkt die Wahrscheinlichkeit das Sonderfälle im späteren Testing vergessen werden.

AUT

Application Under Test (auch SUT, System Under Test) meint die zu testende Software.

Element-Liste

Die Element-Liste ist in der Datei **ElementList.conf** abgebildet und besteht aus Key-Value Paaren. Sie muss ebenfalls vom Projekt gepflegt werden. Der Key ist identisch mit dem "Locator" in der ActionGroup.xml und der Value ist jeweils der technische Schlüssel um ein Element auf dem AUT zu adressieren (also z.B. eine HTML ID oder ein X-Path Ausdruck).

Fachsprache (DSL)

Eine Fachsprache, häufig auch als Domain Specific Language (DSL) bezeichnet, ist nichts anderes als eine Definition wie Testschritte sich generell zusammen setzen. So ist beispielsweise definiert ob ein Testschritt zur Erfassung eines Textes in einem Input-Feld "gebe in das Feld XYZ den Wert ABC ein" heißt, oder ob z.B. eine englische oder eine andere Formulierung gewählt werden soll. Diese Informationen werden in der Datei **TechnicalBindingTypeCollection.xml** hinterlegt. Änderungen sind in der Regel nur dann vorzunehmen, wenn ein Projekt eine eigene Fachsprache entwickelt.

FitNesse

FitNesse ist ein automatisiertes Test-Framework, mit dem Testfälle und Testergebnisse in einer Wiki-Struktur darstellt werden. Der Test-Editor setzt auf diese Wiki-Struktur auf und gibt ihr eine Nutzeroberfläche, die auch technisch weniger versierten Nutzern ohne Vorkenntnisse nutzen können.

Fixture

Fixtures verbinden die Wiki-Struktur von FitNesse mit der Application Under Test (AUT). Im Falle des Test-Editors handelt es sich dabei um Java-Klassen.

Masken und Testschritte

Die nächste Konfigurationsdatei muss von einem Projekt entsprechend erweitert werden, sie beschreibt welche Testschritte auf den jeweiligen Masken der AUT überhaupt möglich sind. Wichtig ist zu unterscheiden, dass hier keine konkreten Testfälle gespeichert werden, sondern nur die Meta-Informationen zu einem Projekt, also welche Aktionen überhaupt möglich sind (z.B. Klicke auf **Login**, Gebe in **Name** einen **Wert** ein usw.). Diese Informationen sind in der **ActionGroup.xml** gespeichert. Sie verweist dabei auf die Fachsprache der TechnicalBinding.xml.

Port

Ihre Aktivitäten im Internet werden durch sogenannte „Services“ organisiert, von denen jeder seine Kennnummer hat, die als „**Portnummer**“ (kurz: Port) bezeichnet wird.

RAP

Remote Application Platform ist ein Eclipse-Plugin, zur Entwicklung von Web-2.0-Anwendungen.

REST

Representational State Transfer ist ein Programmierparadigma für Webanwendungen.

SOAP

Simple Object Access Protocol mit dessen Hilfe Daten zwischen Systemen ausgetauscht werden können. Dazu gehören auch sogenannte Remote Procedure Calls, also der Aufruf einer Prozedur durch ein anderes System.

Szenario

Szenarien sind eine Gruppe von Testschritten die zu einem oder mehreren Testfällen hinzugefügt werden können. Diese Testschritte sind so angeordnet das sie logisch sinnvoll sind. So wird z. B. das Login für mehrere Testfälle benötigt. Indem man das Login als Szenario vorbereitet muss für jeden Testfall nur noch das Szenario hinzugefügt und mit Login-Daten bestückt werden. Diese Definition entspricht auf der Szenariodefinition von FitNesse.

Szenariosuite

Szenariosuiten können angelegt werden um Szenarien thematisch zu gruppieren.

Test

Ein Test oder Testlauf die Ausführung eines Testfalls oder eine Testsuite

Testfall

Ein Testfall ist ein logische Abfolge von Testschritten, die in der Ausführung eine Funktionalität des Systems prüfen.

Jeder Testfall beschreibt:

- was (Funktion, Genauigkeit, usw.) zu prüfen ist,
- welche Ausgangssituation hierfür erforderlich ist,
- welche Eingaben (Daten, Signale, Zeitbedingungen, usw.) notwendig sind und
- welche Ergebnisse (Ausgaben, Reaktionen) zu erwarten sind.

Testlog

Das Testlog stellt die Meldungen des Systems dar, die während der entsprechende Test durchgelaufen ist, aufgezeichnet wurden.

Testsuite

Um Testfälle logisch gruppieren zu können, können diese in einer entsprechenden Testsuite abgelegt werden.

Test-Treiber

Ein Test-Treiber ist eine Software, die eine Schnittstelle erzeugt, um Systeme zu steuern. Die Treiber müssen dabei abhängig von der jeweiligen Software gewählt werden.

Art der Anwendung	Treiber
Swing	FEST
Web	Selenium
SWT und RCP	SWT Bot
Datenbankzugriffe	DB Slim

Widget

Widget ist ein Überbegriff für die Bedienelemente einer Oberfläche. Bedienelemente können Eingabefelder, Buttons, usw. sein.

XML

Die [Extensible Markup Language](#) ist eine Sprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien.

Akzeptanztest

Mit Hilfe des Akzeptanztests oder auch Abnahmetests kann der Anwender überprüfen ob die gewünschten Funktionalitäten einer Software vorhanden sind. Dabei werden die Funktionalitäten anhand der Projektvorgaben überprüft. Der Anwender nutzt die Software und muss entsprechend die technischen Details der Umsetzung nicht kennen. Oft werden Alpha- oder Beta-Tests als Akzeptanztest genutzt. Der Test-Editor unterstützt den Akzeptanztest, indem die Testfälle bereits zur Konzeptionierungsphase erfasst werden können. Dadurch können sinkt die Wahrscheinlichkeit das Sonderfälle im späteren Testing vergessen werden.

AUT

Application Under Test (auch SUT, System Under Test) meint die zu testende Software.

Element-Liste

Die Element-Liste ist in der Datei **ElementList.conf** abgebildet und besteht aus Key-Value Paaren. Sie muss ebenfalls vom Projekt gepflegt werden. Der Key ist identisch mit dem "Locator" in der ActionGroup.xml und der Value ist jeweils der technische Schlüssel um ein Element auf dem AUT zu adressieren (also z.B. eine HTML ID oder ein X-Path Ausdruck).

Fachsprache (DSL)

Eine Fachsprache, häufig auch als Domain Specific Language (DSL) bezeichnet, ist nichts anderes als eine Definition wie Testschritte sich generell zusammen setzen. So ist beispielsweise definiert ob ein Testschritt zur Erfassung eines Textes in einem Input-Feld "gebe in das Feld XYZ den Wert ABC ein" heißt, oder ob z.B. eine englische oder eine andere Formulierung gewählt werden soll. Diese Informationen werden in der Datei **TechnicalBindingTypeCollection.xml** hinterlegt. Änderungen sind in der Regel nur dann vorzunehmen, wenn ein Projekt eine eigene Fachsprache entwickelt.

FitNesse

[FitNesse](#) ist ein automatisiertes Test-Framework, mit dem Testfälle und Testergebnisse in einer Wiki-Struktur dargestellt werden. Der Test-Editor setzt auf diese Wiki-Struktur auf und gibt ihr eine Benutzeroberfläche, die auch technisch weniger versierten Nutzern ohne Vorkenntnisse nutzen können.

Fixture

Fixtures verbinden die Wiki-Struktur von FitNesse mit der Application Under Test (AUT). Im Falle des Test-Editors handelt es sich dabei um Java-Klassen.

Masken und Testschritte

Die nächste Konfigurationsdatei muss von einem Projekt entsprechend erweitert werden, sie beschreibt welche Testschritte auf den jeweiligen Masken der AUT überhaupt möglich sind. Wichtig ist zu unterscheiden, dass hier keine konkreten Testfälle gespeichert werden, sondern nur die Meta-Informationen zu einem Projekt, also welche Aktionen überhaupt möglich sind (z.B. Klicke auf **Login**, Gebe in **Name** einen **Wert** ein usw.). Diese Informationen sind in der **ActionGroup.xml** gespeichert. Sie verweist dabei auf die Fachsprache der TechnicalBinding.xml.

Port

Ihre Aktivitäten im Internet werden durch sogenannte „Services“ organisiert, von denen jeder seine Kennnummer hat, die als „**Portnummer**“ (kurz: **Port**) bezeichnet wird.

RAP

Remote Application Platform ist ein Eclipse-Plugin, zur Entwicklung von Web-2.0-Anwendungen.

REST

Representational State Transfer ist ein Programmierparadigma für Webanwendungen.

SOAP

Simple Object Access Protocol mit dessen Hilfe Daten zwischen Systemen ausgetauscht werden können. Dazu gehören auch sogenannte Remote Procedure Calls, also der Aufruf einer Prozedur durch ein anderes System.

Szenario

Szenarien sind eine Gruppe von Testschritten die zu einem oder mehreren Testfällen hinzugefügt werden können. Diese Testschritte sind so angeordnet das sie logisch sinnvoll sind. So wird z. B. das Login für mehrere Testfälle benötigt. Indem man das Login als Szenario vorbereitet muss für jeden Testfall nur noch das Szenario hinzugefügt und mit Login-Daten bestückt werden. Diese Definition entspricht auf der Szenariodefinition von FitNesse.

Szenariosuite

Szenariosuiten können angelegt werden um Szenarien thematisch zu gruppieren.

Test

Ein Test oder Testlauf die Ausführung eines Testfalls oder eine Testsuite

Testfall

Ein Testfall ist eine logische Abfolge von Testschritten, die in der Ausführung eine Funktionalität des Systems prüfen.

Jeder Testfall beschreibt:

- was (Funktion, Genauigkeit, usw.) zu prüfen ist,
- welche Ausgangssituation hierfür erforderlich ist,
- welche Eingaben (Daten, Signale, Zeitbedingungen, usw.) notwendig sind und
- welche Ergebnisse (Ausgaben, Reaktionen) zu erwarten sind.

Testlog

Das Testlog stellt die Meldungen des Systems dar, die während der entsprechende Test durchgelaufen ist, aufgezeichnet wurden.

Testsuite

Um Testfälle logisch gruppieren zu können, können diese in einer entsprechenden Testsuite abgelegt werden.

Test-Treiber

Ein Test-Treiber ist eine Software, die eine Schnittstelle erzeugt, um Systeme zu steuern. Die Treiber müssen dabei abhängig von der jeweiligen Software gewählt werden.

Art der Anwendung	Treiber
Swing	FEST
Web	Selenium
SWT und RCP	SWT Bot
Datenbankzugriffe	DB Slim

Widget

Widget ist ein Überbegriff für die Bedienelemente einer Oberfläche. Bedienelemente können Eingabefelder, Buttons, usw. sein.

XML

Die [Extensible Markup Language](#) ist eine Sprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien.

Release Notes

Die folgenden Release-Notes dienen der Übersicht, welche Features in welchem *Test-Editor* Release umgesetzt wurden:

Release 1.6.1 (Oktober 2014)

Fehlerbehebung	<ul style="list-style-type: none">• Copy/Paste einer Tabelle innerhalb eines Szenarios (TE-1475)• Scenariotabellen in Szenarien werden fehlerhaft abgespeichert (TE-1477)
-----------------------	--

Release 1.6.0 (Oktober 2014)

UI-Scanner	<ul style="list-style-type: none">• Webseiten können gescannt werden, um Vorlagen für die AllActionGroups und die TechnicalBindingTypeCollections zu erzeugen
Test-Suite	<ul style="list-style-type: none">• Wenn einer Suite neue referenzierte Testfälle hinzugefügt werden, wird beim wiederholten Öffnen des Dialogs die alte Auswahl wiederhergestellt• Wenn man die Liste der referenzierten Testfälle öffnet, wird die Anzahl der referenzierten Testfälle angezeigt
SVN Kommentare	<ul style="list-style-type: none">• Beim Teilen von Projekten und freigeben von Änderungen kann jetzt ein Kommentar eingegeben werden
Fehlerbehebung	<ul style="list-style-type: none">• Fehler und Fehler-Handling im SVN-Bereich wurden verbessert• Bearbeiten der Testkonfiguration bereinigt• Exception-Handling wurde verbessert

Release 1.5.4 (Oktober 2014)

Fehlerbehebung	<ul style="list-style-type: none">• Tabellen (Excel, CVN) Import mit leeren Zellen funktioniert (TE-1412)• Einsatz von Umgebungsvariablen in Szenarien möglich (TE-1425)• Umbenennen eines nicht geteilten Projektes (TE-1416)• SzenarioView wird korrekt aktualisiert (TE-1409)• Status in der Testhistorie stimmt mit dem Status des Test-Explorers überein (TE-1400)• Geöffnete Testobjekte werden nach Aktualisierung des Baumes neu geladen (TE-1407, TE-1427)• Beim Fehlen von XSD-Schemata Dateien wird eine sprechende Fehlermeldung ausgegeben (TE-1311)• Leere Testsuite ist nun ausführbar (TE-1379)• Darstellung von Parametern im Test-Explorer abhängig von ihrem Inhalt (TE-1138)
-----------------------	--

Release 1.5.3 (Juli 2014)

Testbestand prüfen	<ul style="list-style-type: none">• Testfallbeschreibung kann automatisch auf Konsistenz bezüglich der genutzten Bibliothek (TechnicalBindingTypeCollection.xml und AllActionGroups.xml) geprüft werden. Inkonsistenzen werden im Test-Editor als fehlerhaft dargestellt.
Perspektive speichern	<ul style="list-style-type: none">• Die Anordnung der Bereiche (Test-Explorer, Beschreibung, Testschritt, usw.) des Test-Editors werden beim Beenden gespeichert. Diese können jederzeit auf den Auslieferungszustand des Test-Editors zurückgesetzt werden.
Performance	<ul style="list-style-type: none">• Performanceverbesserung (Öffnen von größeren Testfällen sowie Öffnen von Suites mit einer großen Anzahl an Testfällen ist schneller)
Fehlerbehebung	<ul style="list-style-type: none">• SVN- Handling Fehlermeldungen verbessert• Wiki Parseranpassungen (Parameter werden besser unterstützt und Fehlerhandling wurde verbessert)

Release 1.5.2 (Juni 2014)

Verbesserte Bedienung der Tabellen	<ul style="list-style-type: none">• Die Tabellen für die Systemkonfiguration und die Massentests können einfacher bedient werden
Fortschrittsanzeige bei Teamarbeit	<ul style="list-style-type: none">• Beim Freigeben/Aktualisieren von Testfällen/Szenarien/Suites wird der Fortschritt angezeigt
Performance	<ul style="list-style-type: none">• Performanceverbesserung

Fehlerbehebung	<ul style="list-style-type: none"> • Es wurden diverse Fehler behoben
-----------------------	--

Release 1.5.1 (April 2014)

Test abbrechen	<ul style="list-style-type: none"> • Das Abbrechen eines laufenden Tests kann jetzt schneller durchgeführt werden, ohne langes Warten
Fortschrittsanzeige bei Teamarbeit	<ul style="list-style-type: none"> • Beim Freigeben/Aktualisieren von Testfällen/Szenarien/Suites wird der Fortschritt angezeigt
Performance	<ul style="list-style-type: none"> • Performanceverbesserung
Fehlerbehebung	<ul style="list-style-type: none"> • Es wurden diverse Fehler behoben

Release 1.5 (April 2014)

TestSuite verwalten	<ul style="list-style-type: none"> • TestSuiten können Verweise auf Testfälle verwalten und so als Container für die Testausführung dienen.
Projekte verwalten	<ul style="list-style-type: none"> • Erweiterung der Unterstützung bei der Synchronisierung von Projektdateien über SVN.
Testscenarien verwalten	<ul style="list-style-type: none"> • Umbenennung von TestKomponenten zu TestSzenarien • Die TestSzenarien sind jetzt keine Suite mehr und daher nicht mehr als Test ausführbar • Es sind Untergruppierungen für die Szenarien unterhalb der TestSzenarien möglich.

Release 1.4 (Februar 2014)

Test-Editor Einstellungen	<ul style="list-style-type: none"> ▪ Globale Variablen (z.B. Browser-Pfad, Fixture-Pfad) können zentral gesetzt werden (absolute Pfade in einzelnen Konfigurationsseiten entfallen damit) ▪ Projekt Konfiguration für die Demo-Projekte vereinheitlicht
Projekte verwalten	<ul style="list-style-type: none"> ▪ Projekte können über SVN geteilt werden ▪ Projekte können aus dem SVN geladen werden ▪ Testobjekte eines Projekt (SVN) können aktualisiert/freigegeben werden
Testfälle editieren	<ul style="list-style-type: none"> ▪ Auswahlboxen können jetzt editiert werden, wodurch ein direkter Filter der möglichen Ergebnisse gesetzt wird ▪ In Szenarien können auch für Auswahllisten Platzhalter mit @ gesetzt werden ▪ Bei Wechsel zwischen mehreren Testfällen bleibt die Auswahl (z.B. Testschritt) erhalten
Testhistorie löschen	<ul style="list-style-type: none"> ▪ Testhistorie kann gelöscht werden

Release 1.3 (Dezember 2013)

Testfälle editieren	<ul style="list-style-type: none"> ▪ Performanceoptimierung beim Öffnen und Bearbeiten von Testfällen
Testhistorie anzeigen	<ul style="list-style-type: none"> ▪ Anzeigen einer lokalen Historie der Test-Ausführung

Folgende **Bugs** wurden behoben:

- Speichern vor Testausführung kann korrekt abgebrochen werden
- Beim Einfügen von Testschritten bleibt der Scrollbereich bestehen
- Copy und Paste Funktion in Eingabefeldern sichergestellt

Release 1.2 (November 2013)

Projekte verwalten	<ul style="list-style-type: none"> ▪ Projekte können über entsprechende Wizards angelegt, umbenannt und gelöscht werden ▪ Optimierung der Fehlermeldungen in allen Wizards (Anlage, Bearbeiten, Löschen)
Testfälle editieren	<ul style="list-style-type: none"> ▪ Massentestdaten (Tabelle mit Testdaten) werden optimiert dargestellt

Test ausführen	<ul style="list-style-type: none"> ■ eine Suite (auch mit verschachtelten Sub-Suiten) kann als Test ausgeführt werden ■ Optimierung des Live-Logs bei der Test-Ausführung
-----------------------	---

Release 1.1 (Oktober 2013)

Szenarien editieren	<ul style="list-style-type: none"> ■ Es können Szenarien erstellt werden, die in Testfällen benutzt werden können ■ Szenarien können 0-n Parameter erwarten, Parameter werden über @... bei der Eingabe in Textfeldern festgelegt
Testfälle editieren	<ul style="list-style-type: none"> ■ Ein Testfall kann ein bzw. mehrere Szenarien verwenden ■ Erwartet das Szenario Parameter werden die Daten tabellarisch erfasst oder via Excel- bzw. CSV-Datei importiert ■ Dadurch dass die Tabelle n-Zeilen beinhaltet kann, können verschiedene Testdurchläufe definiert werden (= Massentest)
Test-Bibliothek verwalten	<ul style="list-style-type: none"> ■ Pro Projekt kann eingestellt werden, ob die standardisierte XML Bibliothek des Test-Editors oder eine eigene projektspezifische Impl. verwendet wird ■ Die projektspezifische Impl. wird als eigenes Bundle im Zusammenhang mit dem Test-Editor verwendet (Plugin-Mechanismus) ■ Die technische Locator ID (z.B. als Key zur Elementliste) kann direkt in der XML Konfiguration gepflegt werden, wodurch die Konfiguration vereinfacht wurde
Test ausführen	<ul style="list-style-type: none"> ■ Während der Testausführung wird ein "Live-Log" angezeigt, der einzelne Testschritte dokumentiert

Release 1.0 (Juni 2013)

Test-Editor starten	<ul style="list-style-type: none"> ■ Der Test-Editor ist über eine ausführbare Datei startbar (Windows und Linux) ■ Anzeige eines Splash-Screens mit Logo ■ Starten der FitNesse Server: Pro Projekt wird ein FitNesse Server lokal gestartet ■ Beenden der FitNesse Server: Beim Schließen des Test-Editors werden diese gestoppt
Projekte verwalten	<ul style="list-style-type: none"> ■ Unterstützung mehrerer Projekte im Homeverzeichnis des Test-Editors ■ Je Projekt gibt es eine Projekt Konfigurationsdatei (z.B. mit FitNesse Port etc.) ■ Zentrale Projektkonfigurationen können über die UI eingestellt werden
Test-Explorer bedienen	<ul style="list-style-type: none"> ■ Testfälle können geöffnet werden (im Hauptbereich wird der Testfall angezeigt) ■ Testfälle und Suiten können angelegt und umbenannt werden ■ Testfälle und Suiten können einzeln oder auch mehrere gleichzeitig gelöscht werden ■ Ein Test kann zu einem Testfall gestartet werden ■ Quellcode zu einer Suite oder Testfall kann angezeigt werden
Testfälle editieren	<ul style="list-style-type: none"> ■ Beschreibungen können erzeugt, geändert und gelöscht werden ■ Testschritte können erzeugt, geändert und gelöscht werden (Auswahl über Maske und Schritt) ■ Bestehende Testschritte werden validiert und eine Fehlermeldung angezeigt, wenn sie ungültig sind ■ Einzelne oder mehrere Zeilen können kopiert, ausgeschnitten und in einem anderen Testfall eingefügt werden ■ markierten Zeilen können in einem Testfall verschoben werden
Test-Bibliothek verwalten	<ul style="list-style-type: none"> ■ Projekte können eine eigene DSL (Fachsprache) verwenden (XML-Konfiguration) ■ Die möglichen Testschritte zu einzelnen Masken (die sog. Test-Bibliothek) kann über XML je Projekt konfiguriert werden
Test ausführen	<ul style="list-style-type: none"> ■ Tests können ausgeführt werden und es wird angezeigt, ob der Test erfolgreich/nicht erfolgreich war ■ Testfalls erhalten ein grünes bzw. rotes Icon, nachdem ein Test ausgeführt wurde (bis zur ersten Ausführung ist es grau)