

ex|xcellent  
solutions



Benjamin Schmid | eXXcellent solutions gmbh

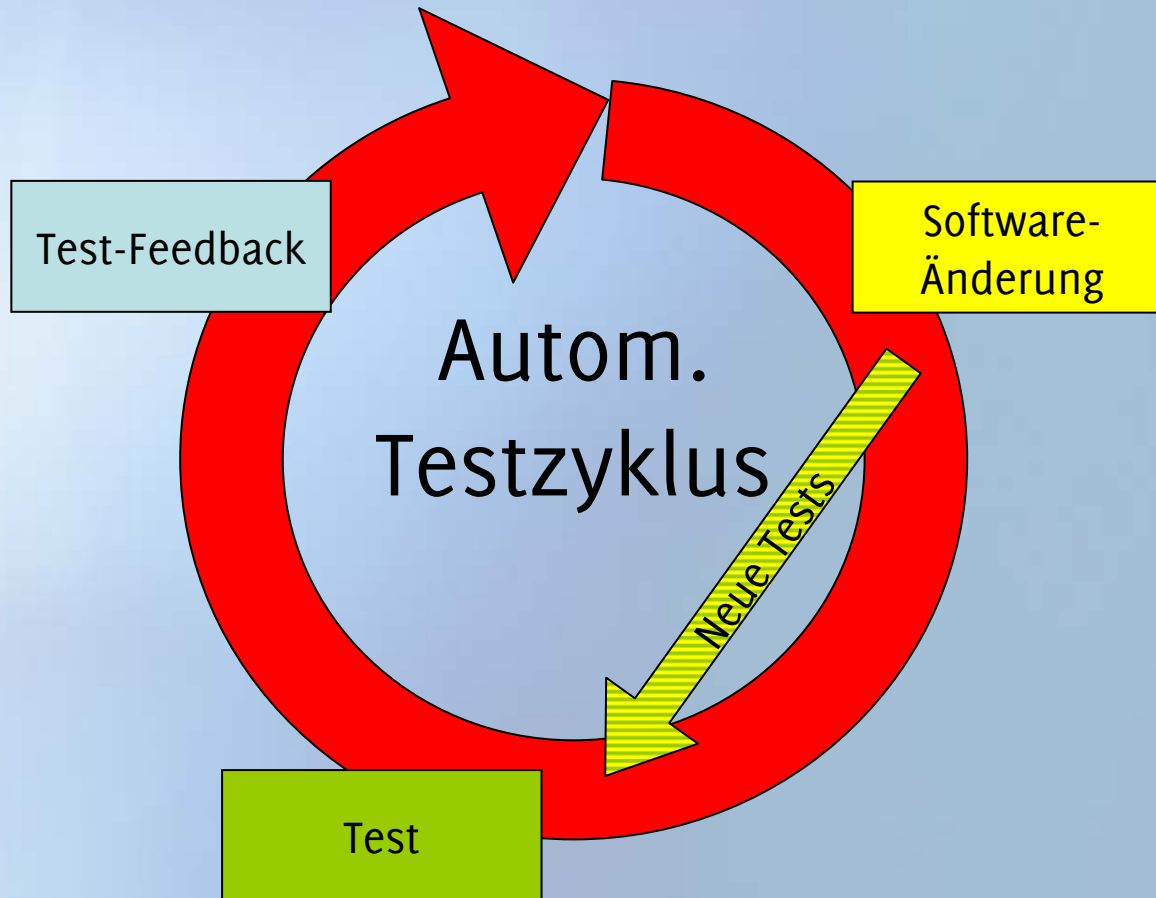
## **Automatisierte GUI-Tests mit Selenium und TestNG in der Praxis**

# Inhalt

---

- » Vision
- » Vorstellung der Technologien
  - Selenium
  - TestNG
- » Die Herausforderungen der Praxis
- » Plugging everything together: Live-Demo
- » Fazit und Ausblick

# Vision



---

# Die Technologien

## Selenium

# Was ist Selenium

- » ... ein Testwerkzeug für **Web-Applikationen**
- » ... steuert einen **Browser direkt**, wie ein normaler Benutzer
- » ... ist **Freie Software / Open Source**
- » ... läuft auf:



IE



Firefox



Konqueror



Safari



Camino



Opera

Seamonkey

## Bestandteile von Selenium

- » Core Steuerungs-Engine für den Browser (JavaScript)
- » IDE Scripting-Entwicklungswerkzeug (FireFox)
- » RC Remote Control (API für Java, C#, Python, ...)
- » weitere Selenium Grid, ...

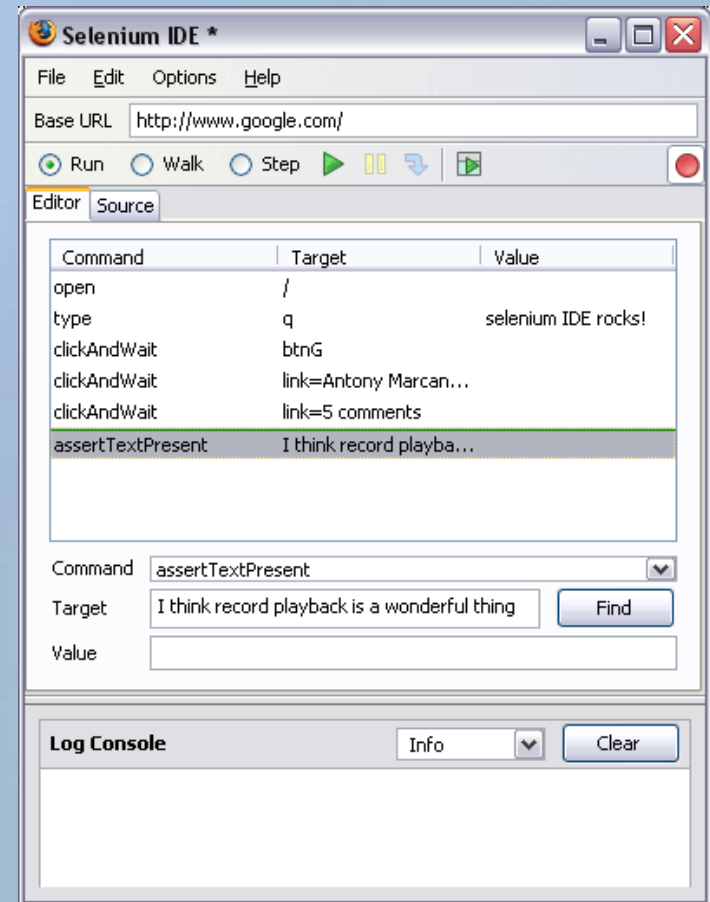
# Wie funktioniert's? – Selenium Core/IDE

## Selenium Core

- » 100% DHTML/JavaScript-Engine
- » „Bedient“ Browser, z.B.:  
open (*url*), click (*locator*),  
waitForCondition (...), keyUp (...),  
mouseDownAt (...), ...

## Selenium IDE

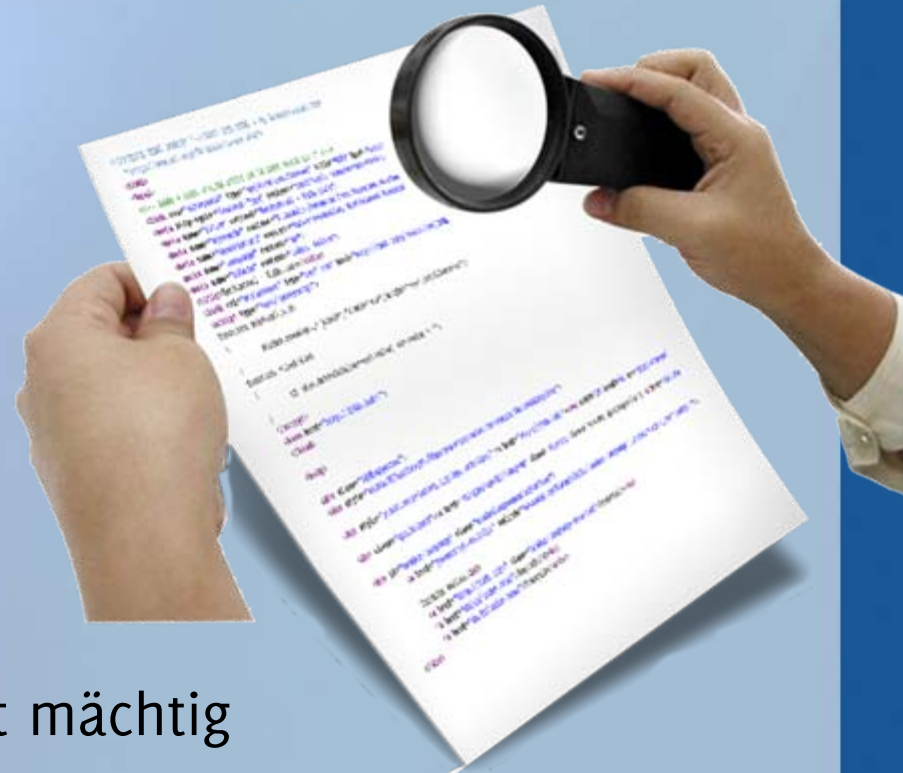
- » FireFox Plugin
- » Scripting von Selenium Core
- » Autom. Record & Playback  
(nur bei stabilen IDs/Namen!)



# Wie funktioniert's? – Selenium Locators

Wie können Elemente auf der HTML-Seite adressiert werden

- » `id=id`
- » `name=name`
- » `link=textPattern`
- » `css=cssSelectorSyntax`
- » `dom=javascriptExpression`
- » `xpath=xpathExpression`

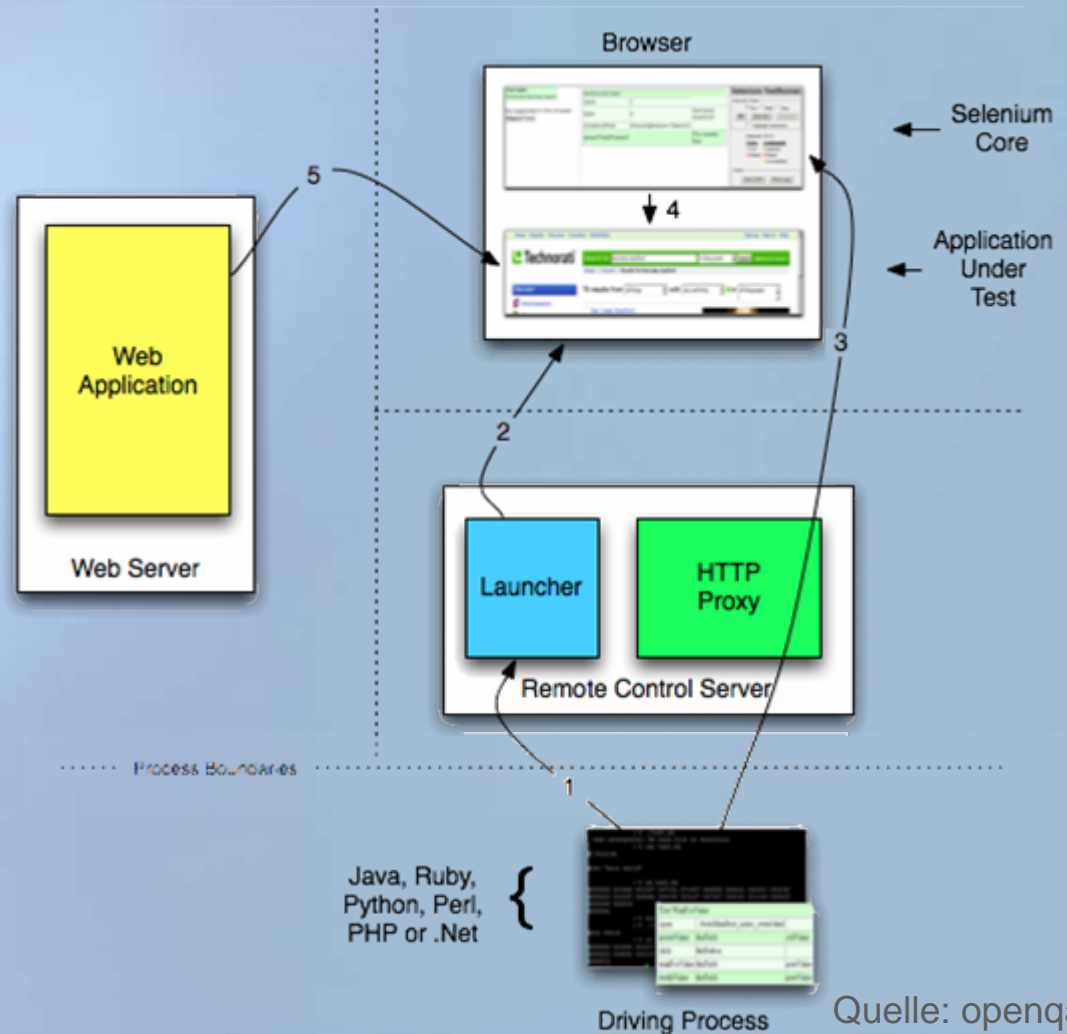


➔ Vor allem die XPath-Notation ist mächtig

Extension Selenium UI Elements: `ui=issuePages::article(index=2)`

# Wie funktioniert's? – Selenium RC

- » RC Server konfiguriert, startet & stoppt den gew. Browser
- » Injiziert via Proxy die Selenium-Core-Engine
- » Dient als Vermittler für die Anfragen und Befehle der RC-Clients



Quelle: openqa.org

# Selenium – „Standard Selenium“-Entwicklung

## Vom Framework nahe gelegter Entwicklungszyklus:

1. Testpfad interaktiv mit Selenium IDE entwickeln
2. Entwurf als Java-RC/JUnit Code exportieren
3. Test anpassen, Prüfungen einfügen und einbinden
4. Tests starten (RC Server, Zielapplikation und Test)

# Wie funktioniert's? – IDE/Java-RC Skript

Beispiel: Mit Selenium IDE generiertes Java RC-Skript

```
public class YahooTest extends SeleneseTestCase {  
  
    public void setUp() throws Exception {  
        setUp("http://www.yahoo.de/", "*chrome");  
    }  
  
    public void testNew() throws Exception {  
        selenium.open("/");  
        selenium.type("p" "selenium test");  
        selenium.click("searchsubmit");  
        selenium.waitForPageToLoad("30000");  
        selenium.click("//div[@id='web']/ol/li[1]/div[2]/div[1]/h3/a/b");  
        selenium.waitForPageToLoad("20000");  
    }  
}
```

HTML name/id Selektor  
<input name="p">..</input>

IDE-generierter  
XPath Selektor

# Wo sind die Grenzen?

- » Grenzen durch **Sicherheitsmodelle** der Browser
  - Host Testserver  $\neq$  Host App.-Server (same-origin policy/Anti-XSS)
  - HTTPS (Zertifikate/Injection)
  - **Lösung:** Subprojekt Web Driver (Native Engine)
- » **Cross-Browser** Funktionalität nicht ganz gratis
- » Bedingte Eignung für **Last-Tests**
- » Sehr positiv: Kontinuierlich steigende Aufmerksamkeit & Projektaktivität!  
(Heavy User: Google)



# Selenium - Fazit (1/2)

- ✓ Browser „fernsteuerung“ → echter Blackbox Test
- ✓ gut gewappnet für Web 2.0, Ajax, Webframeworks:
  - dynamische Inhalte, IDs, Layouts → *Selektoren (XPath)*
  - Client-Logiken (Ajax, JS-Frameworks) → *End-to-End*
  - Proprietäre HTTP-Kommunikation → *End-to-End*
- ✓ Java-Bridge → **Gewohnte Sprachmittel/Umgebung**
- ✓ Ideal für **Funktions-/Abnahme-Tests**  
(auch Cross-Browser-Tests)

# Selenium - Fazit (2/2)



## Record & Playback mit Selenium IDE

- ✓ Komfortabler Start für einfache/kleine „Web 1.0“-Applikationen/Tests
- ↻ IDE Standardvorgehen: Schnell & Einfach
- ⚠ Keine Out-of-the Box Lösung für
  - dynamische IDs (Echo2, wingS, ...) und
  - langfristig stabile Tests (wechselnde HTML Layouts)

It's free software!

- Ideales Konzept/Technologie
- Potential was Verwendung anbelangt

---

# Die Technologien

## TestNG

# Warum kein JUnit?



- » Erstes, populäres Testframework (1998, Gamma & Beck)
- » Quasi de facto Standard
- » Funktionalität rein auf Unit-Tests ausgelegt
- » „Opfer“ des eigenen Erfolgs:
  - Regelmäßiger Einsatz außerhalb echter Unit-Tests
  - Lange Zeit kaum Updates (3.x)
  - Offene Wünsche für funktionale Tests

# Was ist TestNG?



- » Ein Testing-Framework für alle Testebenen, d.h.
  - Unit-Tests
  - Funktionale Tests
  - Integrations-Tests
  - etc.

» Test-Gliederung in: *Suite* -> *Test* -> *Class* -> *Method*  
+ parallel: *Groups*

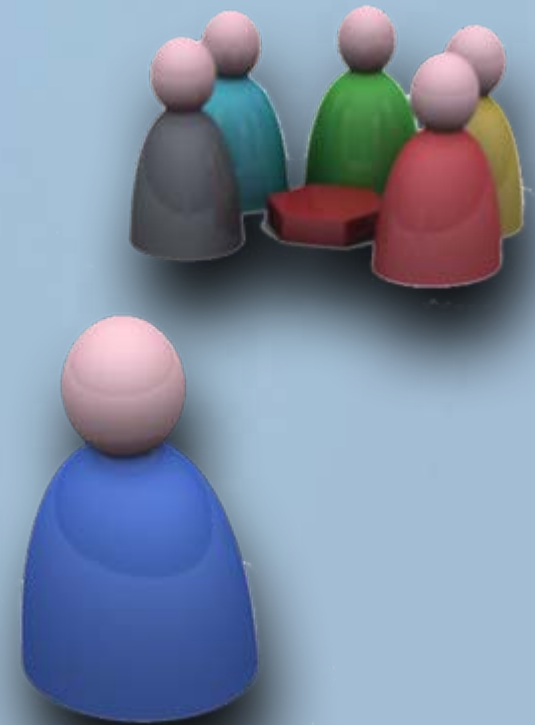
» JUnit3-Kompatibilitäts-Modus

» Plugins für Eclipse  und IntelliJ 

# TestNG – Strukturierung von Tests

## » Gruppierung mit *groups*

- Ein- und Ausschließen von Tests & Konfigurationen
- Mehrfache Gruppenzugehörigkeiten
- Auswahl durch reguläre Ausdrücke: *gui.\** -> *gui.basics*, *gui.longrunner*,...



## » Abhängigkeiten/Reihenfolge mit *dependsOnGroups/Methods*

JUnit: 1 PASSED and 9 FAILURES

TestNG: 1 PASSED, 1 FAILED and 8 SKIPPED

# TestNG - Konfiguration & Datenquellen



» Freie Konfigurationsmethoden je Ebene:

*@Before/After[Suite - Method]*

*@Before/AfterGroup*

» **Parameter:** Testausführung mit konfiguriertem Wert

» **DataProvider:** Test-Regression mit versch. Werten

*@Test(dataProvider= "user-agents")*

```
public void verifyUserAgent(String s, int code) {
```

```
    assert getReturnCodeFor(s) == code;
```

```
}
```

# Weitere interessante Features in TestNG

## » Parallelität und **Multithreading**

*@Test(invocationCount=1000, threadPoolSize=10)*

## » **Erfolgsquoten**

*@Test(invocationCount=100, successPercentage=95)*

## » Maximale Ausführungszeiten / Timeouts

*@Test(timeOut=10000)*

## » Automatischer generierte Testsuite der zuletzt fehlgeschlagenen Tests *testng-failed.xml*

## » Eigene Factories zur Test-Instanziierung

# TestNG – All-in-one Beispiel

```
public class TestNGSample {

    @BeforeTest(groups = {"gui.basics", "gui.advanced"})
    public void prepare() {
        Server.deployApp("demoapp");
    }

    /** Einfacher Login innerhalb 5s. */
    @Test ( groups = "gui.basics", timeout = 5000)
    public void checkLogin() {
        assert GUITest.login("admin", "admin123") == true;
    }

    /** Massen-Login mit 3 Clients parallel und logins aus Provider, falls
        der einfache Login erfolgreich war. */
    @Test ( groups = "gui.advanced", dataProvider = "login-provider",
        dependsOnMethods = "checkLogin",
        invocationCount = 100, threadPoolSize = 3)
    public void checkMultipleLogins(String user, String pass, boolean valid) {
        assert GUITest.login(user, pass) == valid;
    }
}
```

# TestNG - Fazit

---

» TestNG deckt deutlich mehr als nur Unit-Tests ab

» Für unseren Fall besonders attraktive Vorteile:

- ✓ **Gruppen** („GUI“, „Business“, „QuickTests“)
- ✓ **DataProvider** (Test-Regression mit versch. Daten)
- ✓ **Multi-Threading** (→ Selenium Grid)
- ✓ **Plugin-API** (Eigene Test-Reports, ...)

⚠ Grenzen: DataProvider + Dependencies → Factory

→ **Für unsere Anforderungen gegenüber JUnit deutlich überlegen**

---

# Die Herausforderungen der Praxis

# Challenge #1: Effiziente GUI-Test Erstellung

» Aufgezeichnete bzw. manuell codierte

Selektoren problematisch: `["//div[@id='web']/ol/li[1]/div[2]/div[1]/h3/a/b");`

- XPath & HTML-Wissen zum Verständnis erforderlich
- Ohne stabile & sprechende HTML IDs sehr komplex
- Nicht selbsterklärend; Blick in HTML-Seite notwendig
- Anfällig gegenüber Änderung am Layout/HTML-Code

Wunsch:

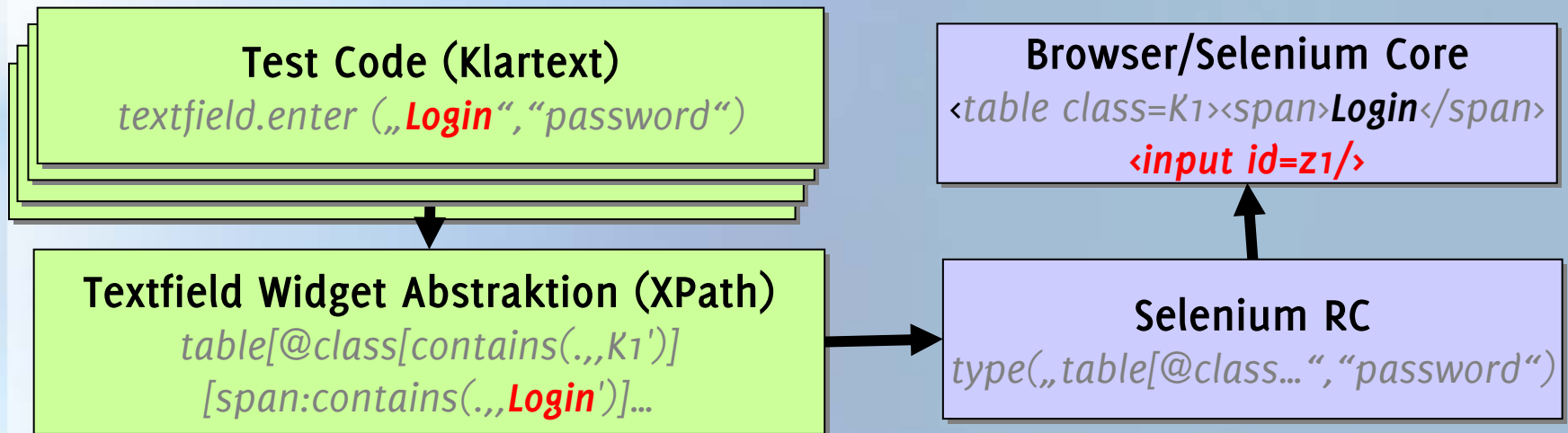
» Einfache und selbsterklärende API für Jedermann

» Einmal geschriebene Test sollen **robust** gegenüber internen Änderungen sein

# Lösung #1: Abstraktion der HTML Seite/Widgets

Sammlung semantischer UI-Elemente  
als Fassade über Selenium (Menü, Button, Textfeld)

- Elemente „wissen“ um ihre **HTML-Representation**  
(Projekt & Framework-Spezifisch)
- Transformieren ihr Adresse in passende XPath-Ausdrücke



# DSL – Ein Beispiel

```
@Test(enabled = true)
public void stateErfasst() throws Exception {
    final String id;

    openApplication();
    getTextField().enterViaLabel("Benutzerkennung", "BurgerM");
    getTextField().enterViaLabel("Passwort", "BurgerM");
    getButton().clickTextButton("Anmelden");
    waitForNewPage();

    getMenu().openMenu("Abteilungen", "Kundenbetreuung", "Eingangskorb");

    // Unter-Use Case: Auftrag erfassen
    id = auftragswriter.erfassen(PCDatenServiceauftrag.IN_AUFTRAG_KOMPLETT);

    TestLog.info("# Auftrag '" + id + "' über Globale Suche suchen und öffnen");
    getMenu().openMenu("Globale Suche");
    getTextField().enterViaLabel("Serviceauftrag-ID", id);
    getButton().clickTextButton("Serviceaufträge suchen");
    waitForNewPage();

    getTable().clickUniqueLink(id);
    waitForNewPage();
}
```

# Herausforderung #2: Die Nachvollziehbarkeit

- » Unbeaufsichtigte Ausführung automatischer GUI-Tests
- » Vielschichtige Testabdeckung
  - Viele Fehlerpotentiale
  - regelmäßige Fehleranalyse notwendig

## Wunsch nach Transparenz:

Einfaches nachvollziehen & verstehen von:

- » was passiert?
  - » was schlägt fehl?
- } und in welchem Kontext?

# Lösung #2

## Transparenz schaffen:

- » Lesbarer Test & Ablaufprotokoll durch semantische API und Logmessages
- » Anreicherung des Ablaufprotokolls um Screenshots
- » **Archivierung:** Ablaufprotokoll, Applikationslog, etc.

## Umsetzung:

- Screenshots via Logging
- Modifizierter TestNG TestReporter



# Herausforderung #3: Die grüne Wiese

- » Tests in datenlosen Systemen wenig gewinnbringend
- » Sinnvolle Quellen von Testdaten:
  - künstlich erzeugte Datenszenarien:  
Aufwändig zu konstruieren; definierte Konstellation
  - produktionsnahe Echt-Daten:  
realitätsnahe Mengengerüste; Stichproben geeignet
- » Es gilt bei Bedarf die umfangreiche Datenszenarios schnell wieder herstellen zu können, welche die Grundlage für den jeweiligen Test bildeten

# Lösung #3: Snapshots und RDBMS Refactoring

## Snapshot-Mechanismen von Datenbanken

- MS SQL: „Database Snapshots“  
(ab SQL Server Enterprise/Developer)
- Oracle: „Oracle Flashback“  
(ab Oracle 10gR2)

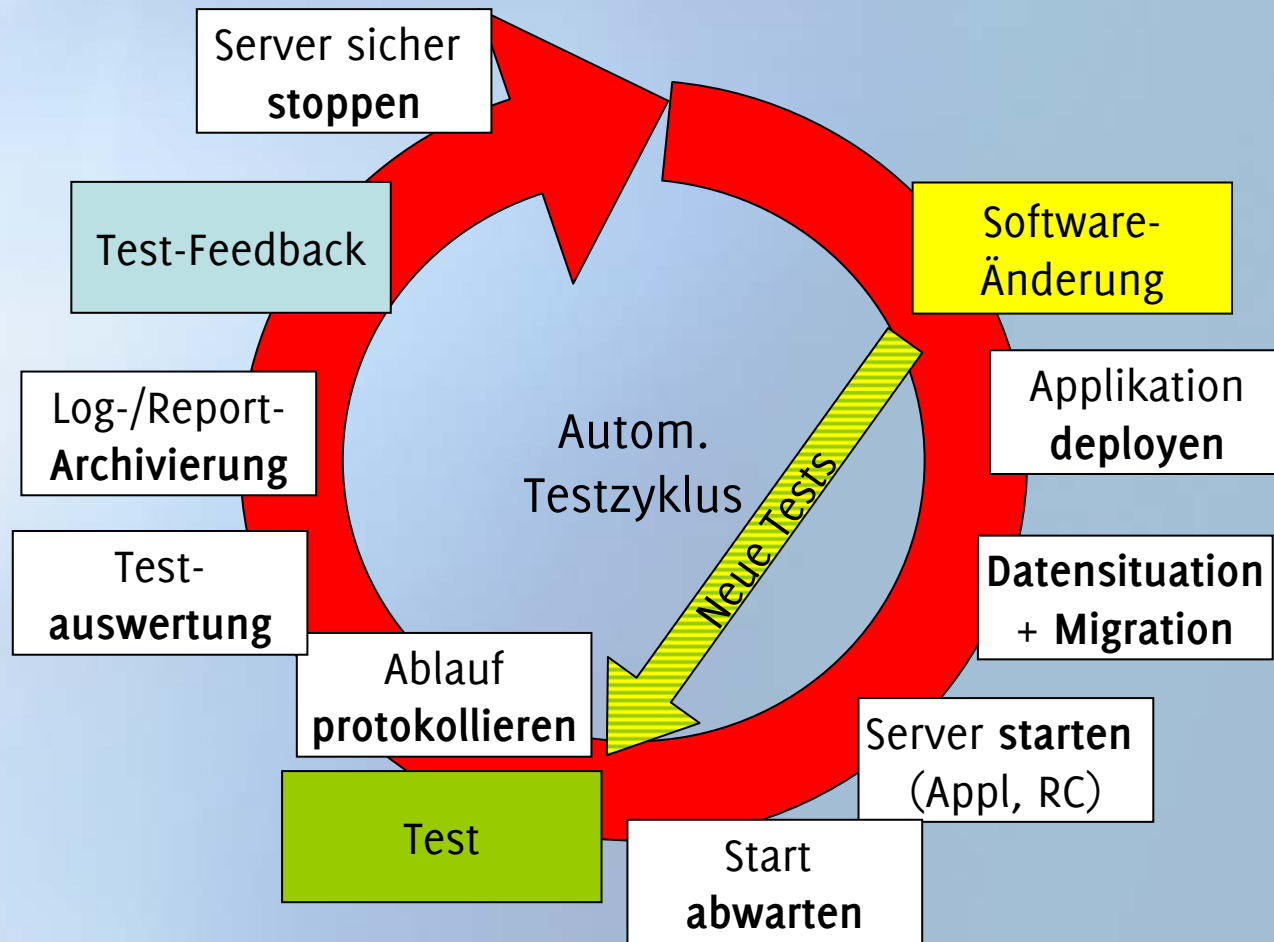
→ MS SQL: Snapshot Rollback von 20GB in ~1 Min!

## Automatische und Verlustfreie Migration

- Automatische Migration der archivierten Datenszenarios an aktuelles RDBMS Schema

→ LiquBase

# Herausforderung #4: Pluggin' all together



---

# Live Demo

---

# Fazit und Ausblick

# Erfahrungen aus der Praxis

---

## Erfahrungen im Projekt:

- » Gute Projekt-Akzeptanz
- » Einfach: Selbst Projektleiter entwirft UI-Tests [im Zug]
- » Hoher Reuse-Anteil dank Abstraktion des Konzepts

## Technische Pitfalls:

- » Autom. Integrationsprozess (Tomcat, etc.)
- » MS IE (XPath im deadly slow, delikat)
- » Selenium Feinheiten: (fehlende *waitfor()*, ...)

# Lohnen sich automatisierte GUI-Tests?

---

» Generell: GUI-Tests sind

- fragil, aufwendig & langsam
- Dafür aber auch **umfassend** und **realistisch**

» UI-Tests ersetzen keine Unit-/Abnahme-/Tests!

- Unit-Tests sind wertvolle White-Box Tests und sollten nicht vernachlässigt werden!
- Schnelle & einfache Basis für weitere Tests

» UI-Tests sinnvoller Teil einer Gesamt-Teststrategie

# Lohnen sich GUI-Tests also?

---

## » Ja, auf jeden Fall!

Die Frage ist in welchem Umfang!

## » Basis-Checks: Eigentlich Immer!

(Deployen, Starten, Login, wichtigste GUI-Pfade)

→ Kontinuierliche Regression des Gesamtsystems

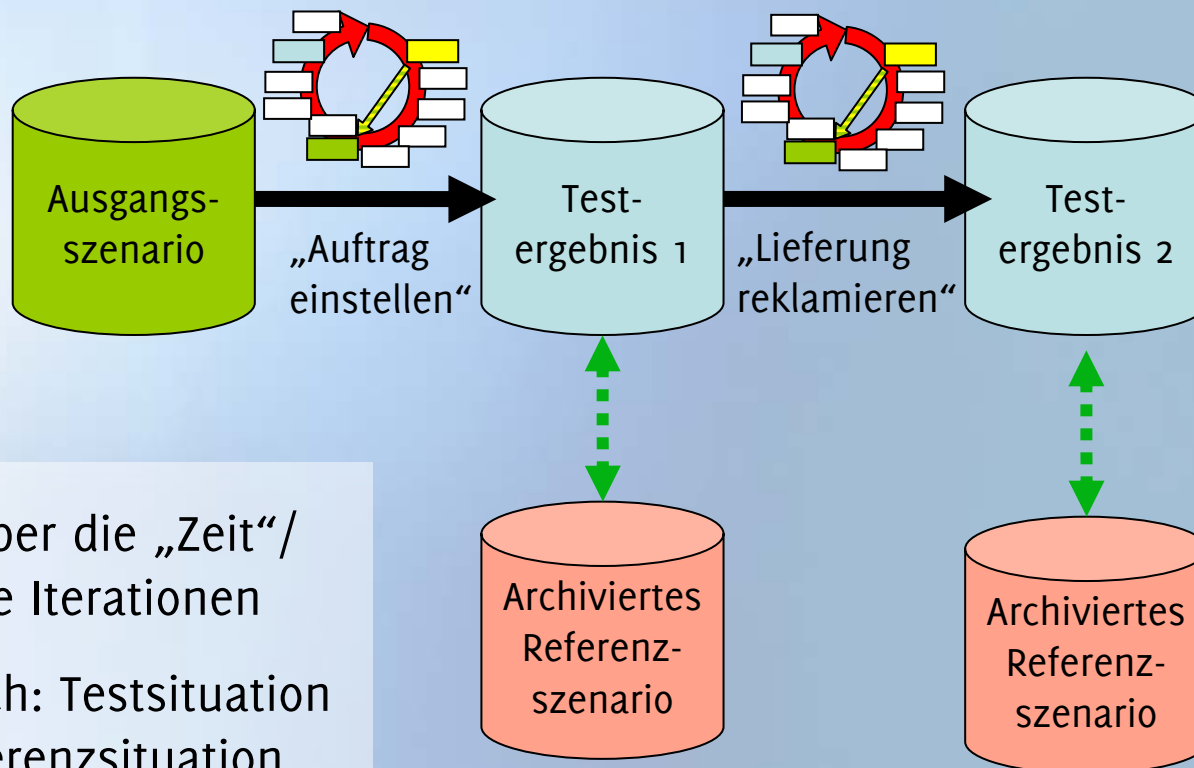
→ Bereits mit einfachen Mitteln gut realisierbar!

## » Ausführlichere Funktionschecks: Vor allem bei

- Projekte mit langer Laufzeit / Produktcharakter
- Kritischen Use Cases
- Kurzen Release-Zyklen

# Ausblick – Tests des Gesamtprozesses

z.B. am Beispiel einer Abrechnungssoftware



- Tests über die „Zeit“/ mehrere Iterationen
- Vergleich: Testsituation vs. Referenzsituation

# Vielen Dank für Ihre Aufmerksamkeit!

**Besuchen Sie uns!**

ex|Xcellent  
solutions

**Direkt am Eingang der Ausstellung**

*<http://www.excellent.de/>*

## Referenzen & weitere Infos

- » Web Driver: [code.google.com/p/webdriver/](http://code.google.com/p/webdriver/)
- » Selenium UI Elements: [ttwhy.org/home/blog/category/selenium/](http://ttwhy.org/home/blog/category/selenium/)
- » XPather: [addons.mozilla.org/en-US/firefox/addon/1192](http://addons.mozilla.org/en-US/firefox/addon/1192)
- » Selenium User Meeting:  
[raibledesigns.com/rd/entry/last\\_night\\_s\\_selenium\\_users](http://raibledesigns.com/rd/entry/last_night_s_selenium_users)
- » Selenium Homepage: [www.openqa.org](http://www.openqa.org)
- » TestNG Homepage: [www.testng.org](http://www.testng.org)
- » LiquiBase: [www.liquibase.org](http://www.liquibase.org)

