

# Continuous Delivery ohne Kopfschmerzen

Martin Reinhardt (Holisticon AG)

 @mreinhardt



# Agenda

- Einführung / Build Pipeline
- Deployment
- Toggle Integration
- Testautomatisierung
- Ausblick
- Links

# Einführung

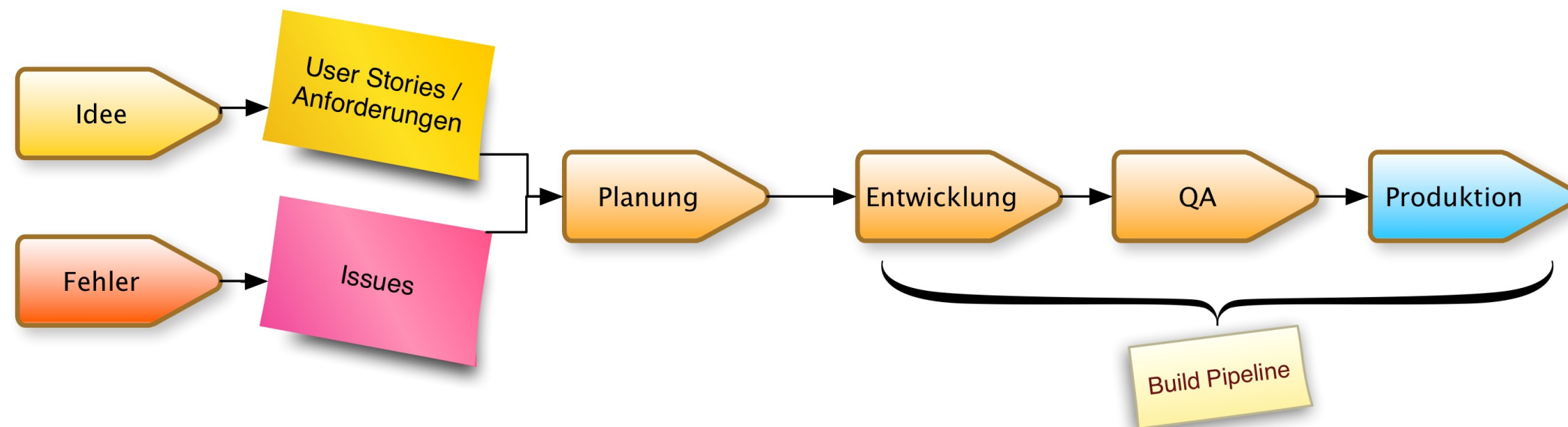
- Continuous Delivery ist die logische Fortsetzung von Continuous Integration

# Einführung

- Continuous Delivery ist die logische Fortsetzung von Continuous Integration
- Die Idee dahinter
  - ☐ Software mit agilen Methoden kann nicht komplett (manuell) getestet werden
  - ☐ Alle 2 Wochen gesamten Funktionsumfang abtesten ist utopisch
  - ☐ Testumgebungen stehen meist nicht ausreichend zur Verfügung

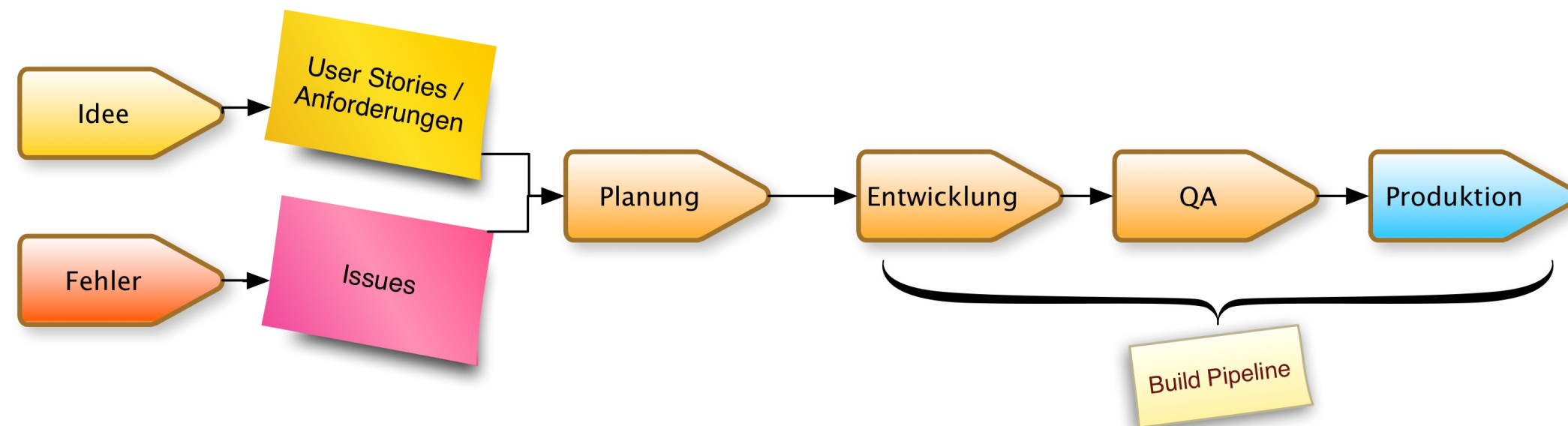
# Einführung

- Continuous Delivery ist die logische Fortsetzung von Continuous Integration
- Die Idee dahinter
  - Software mit agilen Methoden kann nicht komplett (manuell) getestet werden
  - Alle 2 Wochen gesamten Funktionsumfang abtesten ist utopisch
  - Testumgebungen stehen meist nicht ausreichend zur Verfügung



# Einführung

- Continuous Delivery ist die logische Fortsetzung von Continuous Integration
- Die Idee dahinter
  - Software mit agilen Methoden kann nicht komplett (manuell) getestet werden
  - Alle 2 Wochen gesamten Funktionsumfang abtesten ist utopisch
  - Testumgebungen stehen meist nicht ausreichend zur Verfügung



# Warum Build Pipelines

# Warum Build Pipelines

- Agile Softwareentwicklung arbeitet kleinteilig
  - ☐ Software oft und zuverlässig in Produktion
  - ☐ Nutzung der IDE != Automatisierung
  - ☐ Wesentlich ist dabei die Build Pipeline



# Warum Build Pipelines

## ■ Agile Softwareentwicklung arbeitet kleinteilig

- ☐ Software oft und zuverlässig in Produktion
- ☐ Nutzung der IDE != Automatisierung
- ☐ Wesentlich ist dabei die Build Pipeline

## ■ Wie?

- ☐ Geschwindigkeit
- ☐ Automatisierung

# Warum Build Pipelines

## ■ Agile Softwareentwicklung arbeitet kleinteilig

- ☐ Software oft und zuverlässig in Produktion
- ☐ Nutzung der IDE != Automatisierung
- ☐ Wesentlich ist dabei die Build Pipeline

## ■ Wie?

- ☐ Geschwindigkeit
- ☐ Automatisierung



# Build Pipeline

# Build Pipeline

- wesentliches Ziel ist schnelles Feedback, also Geschwindigkeit
  - Einzelne Schritt schnell abarbeiten (5 - 10 Minuten)
  - Möglichst früh Fehler finden (**Unit-Tests**)

# Build Pipeline

- wesentliches Ziel ist schnelles Feedback, also Geschwindigkeit
  - ☐ Einzelne Schritt schnell abarbeiten (5 - 10 Minuten)
  - ☐ Möglichst früh Fehler finden (**Unit-Tests**)
- unabhängig vom (Build-)Tool
  - ☐ Build muss reproduzierbar sein

# Build Pipeline

- wesentliches Ziel ist schnelles Feedback, also Geschwindigkeit
  - ☐ Einzelne Schritt schnell abarbeiten (5 - 10 Minuten)
  - ☐ Möglichst früh Fehler finden (**Unit-Tests**)
- unabhängig vom (Build-)Tool
  - ☐ Build muss reproduzierbar sein
- CI-Tools
  - ☐ Jenkins, Go, Travis CI, XCode Bots ...
  - ☐ TestTools

# Build Pipeline

- wesentliches Ziel ist schnelles Feedback, also Geschwindigkeit
  - ☐ Einzelne Schritt schnell abarbeiten (5 - 10 Minuten)
  - ☐ Möglichst früh Fehler finden (**Unit-Tests**)
- unabhängig vom (Build-)Tool
  - ☐ Build muss reproduzierbar sein
- CI-Tools
  - ☐ Jenkins, Go, Travis CI, XCode Bots ...
  - ☐ TestTools
- Quellcode muss Continuous Delivery gerecht werden

# Continuous Delivery

- CI Server stellt dabei auch die finale Software auf dem Zielsystem bereit
  - ☐ "Infrastructure as code" (Puppet, Chef, Docker)
  - ☐ Deployment Pipeline



# Continuous Delivery

- CI Server stellt dabei auch die finale Software auf dem Zielsystem bereit
  - ☐ "Infrastructure as code" (Puppet, Chef, Docker)
  - ☐ Deployment Pipeline
- Jeder gängige CI Server bietet Pipeline Bausteine an
  - ☐ somit CD Server
  - ☐ logische 1. Schritt ist Visualisierung
  - ☐ Übersicht aller Pipelines eines CD Servers
  - ☐ Zeitliche Aktivitäten

# Build Pipeline sinnvoll nutzen

## ■ 3 Grundprinzipien

- ☐ Zeit, Schritte schnell abschließen (3 - 5 min)
- ☐ Automatisierung
- ☐ Reproduzierbarkeit

## ■ Tests müssen schnell gefixt werden und stabil sein

## ■ Tests gruppieren & parallelisieren

## ■ Kritische Tests immer in Pipeline laufen lassen

## ■ weitere Teststrecken aufbauen

- ☐ Code Quality
- ☐ Migrationstests
- ☐ Nightly Builds
- ☐ Weitere Pipelines

# Jenkins Build Pipeline Plugin

- Feature von Jenkins 2
- Deployment Pipeline Erzeugung in Jenkins benötigt viele Plugins
- Build Pipeline Ansicht bietet detaillierten Überblick über einzelne Schritte der Pipeline

## Pipeline ContinuousDelivery\_Demo

Full project name: Public/ContinuousDelivery\_Demo



[Recent Changes](#)

### Stage View



# Jenkins 2 Build DSL

```
node {
  env.JAVA_HOME = tool 'jdk-8-oracle'
  def mvnHome = tool 'Maven 3.3.1'
  env.PATH="${env.JAVA_HOME}/bin:${mvnHome}/bin:${env.PATH}"

  stage 'Checkout'
  git url: 'https://github.com/holisticon/continuous-delivery-demo.git'

  stage 'Build'
  sh "${mvnHome}/bin/mvn clean package"

  stage 'Unit-Tests'
  sh "${mvnHome}/bin/mvn test"

  stage 'Integration-Tests'
  wrap([$class: 'Xvfb']) {
    sh "export DOCKER_HOST=tcp://localhost:4243 &&\n${mvnHome}/bin/mvn -Ddocker_clean_manifest"
```

# Deployment

- Wichtiger Punkt bei Planung
- Versionierung meist zu grob (1.0.3-SNAPSHOT)

```
mvn package -Dversion=20140325092556-d4dafcd-2341
```

- Einfaches Groovy Script (EnvInject-Plugin in Jenkins)
- kann mit Jenkins 2 direkt im Build-Skript hinterlegt werden

```
def date = new Date()
def datestring = date.format("yyyyMMddHHmmss")
def version = datestring + "-" + SHA + "-" + BUILD_NUMBER;
return [VERSION: version]
```

# Deployment

- Wichtiger Punkt bei Planung
- Versionierung meist zu grob (1.0.3-SNAPSHOT)

```
mvn package -Dversion=20140325092556-d4dafcd-2341
```

- Einfaches Groovy Script (EnvInject-Plugin in Jenkins)
- kann mit Jenkins 2 direkt im Build-Skript hinterlegt werden

```
def date = new Date()
def datestring = date.format("yyyyMMddHHmmss")
def version = datestring + "-" + SHA + "-" + BUILD_NUMBER;
return [VERSION: version]
```

- Verschiedene Arten neue Version bereitzustellen
  - ☐ Big Bang
  - ☐ Feature-Toggle (nur eine Version)
  - ☐ simultane Versionen (Blue/Green im Cluster)
  - ☐ Beta-Release für Nutzergruppen

# Deployment

- Wichtiger Punkt bei Planung
- Versionierung meist zu grob (1.0.3-SNAPSHOT)

```
mvn package -Dversion=20140325092556-d4dafcd-2341
```

- Einfaches Groovy Script (EnvInject-Plugin in Jenkins)
- kann mit Jenkins 2 direkt im Build-Skript hinterlegt werden

```
def date = new Date()
def datestring = date.format("yyyyMMddHHmmss")
def version = datestring + "-" + SHA + "-" + BUILD_NUMBER;
return [VERSION: version]
```

- Verschiedene Arten neue Version bereitzustellen
  - ☐ Big Bang
  - ☐ Feature-Toggle (nur eine Version)
  - ☐ simultane Versionen (Blue/Green im Cluster)
  - ☐ Beta-Release für Nutzergruppen
- Gerade letzten beiden Versionen erfordern vom Backend mehrere Versionen in DB etc. zu unterstützen

# Toggle-Integration

- praxis-bewährtes Konzept - FeatureToggle Pattern



# Toggle-Integration

- praxis-bewährtes Konzept - FeatureToggle Pattern
- Features nicht nur nach ihrem Fertigstellungsgrad nutzbar, sondern abhängig von
  - ☐ Benutzer
  - ☐ Umgebung
  - ☐ ...

# Toggle-Integration

- praxis-bewährtes Konzept - **FeatureToggle Pattern**
- Features nicht nur nach ihrem Fertigstellungsgrad nutzbar, sondern abhängig von
  - ☐ Benutzer
  - ☐ Umgebung
  - ☐ ...
- **Togglz**
  - ☐ Bibliothek für Java
  - ☐ Integration in Spring, JEE, CDI, JSF
  - ☐ Flexible **Persistierung** (Datei, Datenbank, ...)
  - ☐ Admin-Konsole zu Verwaltung

## Togglz (1)

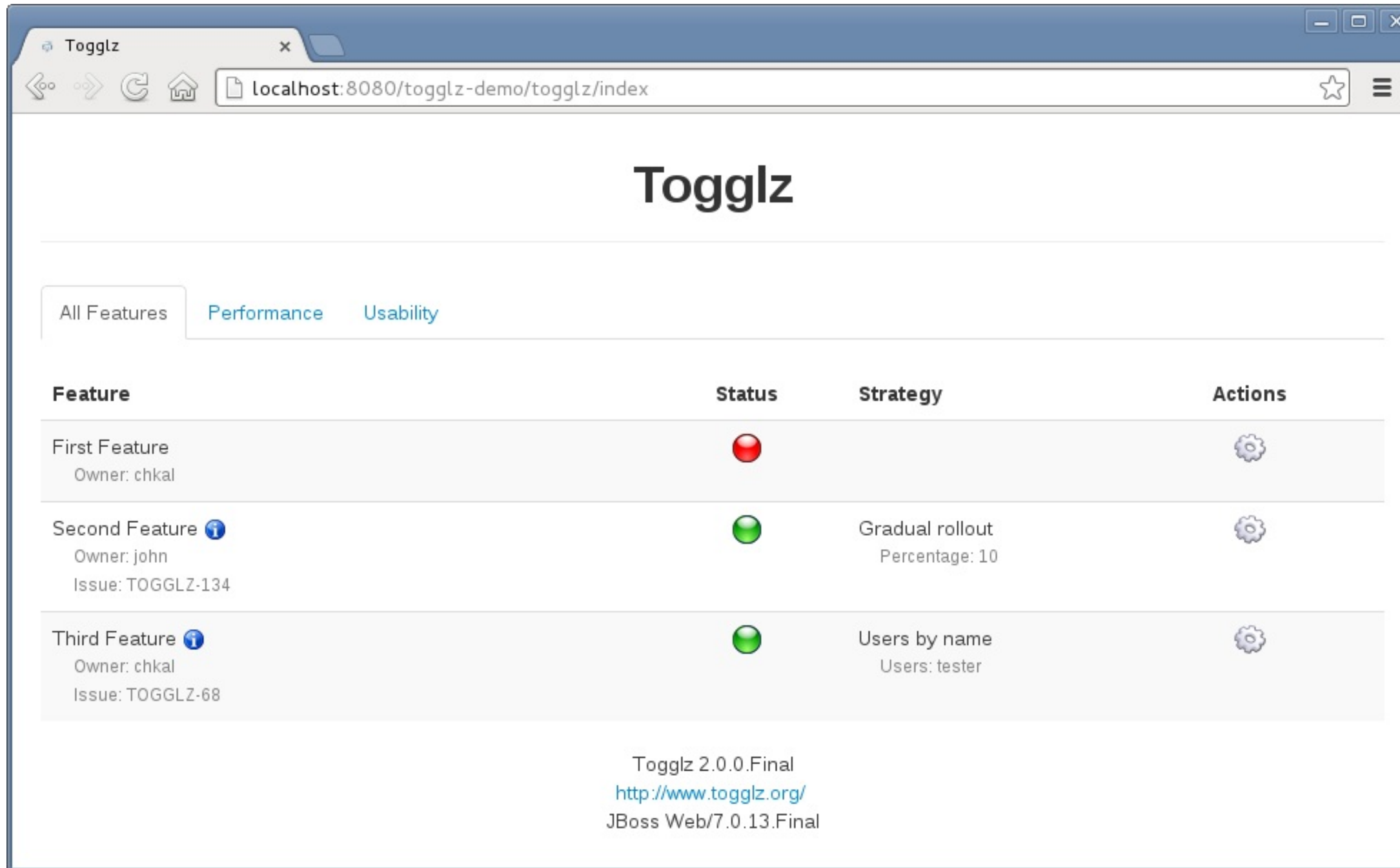
- Verschiedene Module integrierbar

```
<dependency><!-- Togglz core module (mandatory) -->
  <groupId>org.togglz</groupId>
  <artifactId>togglz-core</artifactId>
</dependency>
<dependency><!-- Togglz for Servlet environments (mandatory for webapps) -->
  <groupId>org.togglz</groupId>
  <artifactId>togglz-servlet</artifactId>
</dependency>
<dependency><!-- CDI integration (optional) -->
  <groupId>org.togglz</groupId>
  <artifactId>togglz-cdi</artifactId>
</dependency>
```









- Admin-Konsole kann ebenso optional integriert werden

# Togglz (2)

## ■ Admin-Konsole



The screenshot shows the Togglz Admin Console web application running in a browser. The browser's address bar shows the URL `localhost:8080/togglz-demo/togglz/index`. The application has a header with the "Togglz" logo and a navigation bar with tabs for "All Features", "Performance", and "Usability". Below the navigation bar is a table with four columns: "Feature", "Status", "Strategy", and "Actions". The table lists three features: "First Feature" (Status: red circle), "Second Feature" (Status: green circle, Strategy: Gradual rollout, Percentage: 10), and "Third Feature" (Status: green circle, Strategy: Users by name, Users: tester). Each feature row has a gear icon in the "Actions" column. At the bottom of the page, the version "Togglz 2.0.0.Final" and the URL <http://www.togglz.org/> are displayed, along with the JBoss Web version "JBoss Web/7.0.13.Final".

Feature	Status	Strategy	Actions
First Feature Owner: chkal			
Second Feature  Owner: john Issue: TOGGLZ-134		Gradual rollout Percentage: 10	
Third Feature  Owner: chkal Issue: TOGGLZ-68		Users by name Users: tester	

Togglz 2.0.0.Final  
<http://www.togglz.org/>  
JBoss Web/7.0.13.Final

## Togglz (3)

### ■ Features als Enum-Klassen

```
public enum AppFeatures implements Feature {  
    @Label("Portal")  
    PORTAL,  
  
    @EnabledByDefault  
    @Label("Enable project support in CV")  
    FEATURE_CV_PROJECT_SUPPORT;  
  
    public boolean isActive() {  
        return FeatureContext.getFeatureManager().isActive(this);  
    }  
}
```

### ■ Direkte Nutzung in Code

```
if (AppFeatures.PORTAL.isActive()) {  
    ...  
}
```

## Togglz (4)

### ■ Nutzung in Tests

```
public class AuthFilterTest {
    @Rule
    public TogglzRule togglzRule = TogglzRule.allEnabled(AppFeatures.class);

    @Test
    public void testNoRedirectToLoginWithUser() throws Exception{
        final StringBuffer requestURL = new StringBuffer("abc/xyz");
        final User user = new User.Builder().withUsername("test")...

        final UserSession userSession = mock(UserSession.class);
        final HttpSession httpSession = mock(HttpSession.class);
        final FilterChain chain = mock(FilterChain.class);
        request = mock(HttpServletRequest.class);
        response = new HttpServletResponse();
        when(request.getRequestURL()).thenReturn(requestURL);
        when(request.getSession()).thenReturn(httpSession);
        when(userSession.getUser()).thenReturn(user);
        // all features are active by default
    }
}
```

# Testautomatisierung

- Anschluss der Unit-Tests erzeugten Artefakte weiter testen
  - ☐ Ablage in Repository
  - ☐ Deployment in Test-Umgebung
- Testumgebungen problemebehaftet
  - ☐ Dedizierte Slaves (Labels)
  - ☐ Docker Images
- Docker besonders hilfreich (lokales Testen, Ressourcennutzung)
  - ☐ [Maven-Plugin](#) verfügbar
- Feedback ist ein Muss
  - ☐ TestTools müssen aussagekräftige Reports liefern
  - ☐ reproduzierbar & nachvollziehbar
  - ☐ gerade bei Oberflächen-Tests

# Responsive Testing mit Galen



# Responsive Testing mit Galen

- Unterschiedliche Darstellungsformen für Bilder, wie Produktabbildungen, je nach verwendetem Endgerät

# Responsive Testing mit Galen

- Unterschiedliche Darstellungsformen für Bilder, wie Produktabbildungen, je nach verwendetem Endgerät
- Mobile und Desktop-Ansichten liefern gleiche User Experience im Responsive Design wie getrennt gepflegte Varianten

# Responsive Testing mit Galen

- Unterschiedliche Darstellungsformen für Bilder, wie Produktabbildungen, je nach verwendetem Endgerät
- Mobile und Desktop-Ansichten liefern gleiche User Experience im Responsive Design wie getrennt gepflegte Varianten
- Responsive Webdesign erhöht Testaufwand für verschiedene Auflösungen/Geräteklassen

# Responsive Testing mit Galen

- Unterschiedliche Darstellungsformen für Bilder, wie Produktabbildungen, je nach verwendetem Endgerät
- Mobile und Desktop-Ansichten liefern gleiche User Experience im Responsive Design wie getrennt gepflegte Varianten
- Responsive Webdesign erhöht Testaufwand für verschiedene Auflösungen/Geräteklassen
- Agile Entwicklung wiederholt gerade Testschritte in mehreren Sprints

# Über Galen



*"Galen is a OSS tool for testing html layout in a real browser"*

# Über Galen



*"Galen is a OSS tool for testing html layout in a real browser"*

## ■ Grundprinzipien

- ☐ Test beschreiben im BDD-Stil das gewünschte Verhalten
- ☐ "menschlich lesbar"
- ☐ Verhalten wie ein Nutzer

# Über Galen



*"Galen is a OSS tool for testing html layout in a real browser"*

## ■ Grundprinzipien

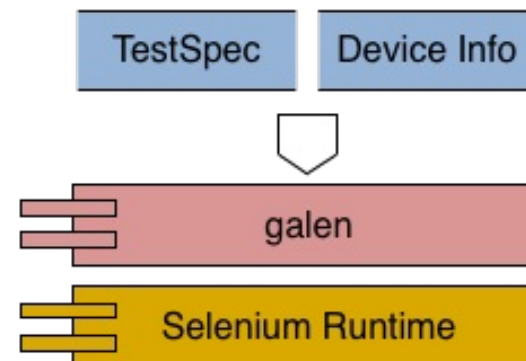
- ☐ Test beschreiben im BDD-Stil das gewünschte Verhalten
- ☐ "menschlich lesbar"
- ☐ Verhalten wie ein Nutzer

## ■ Features

- ☐ DSL für die Beschreibung der Specs
- ☐ Eigene Testsyntax mit Data-Providern (optional)
- ☐ Selenium für die Ausführung
- ☐ Ausführliches Reporting

# Architektur

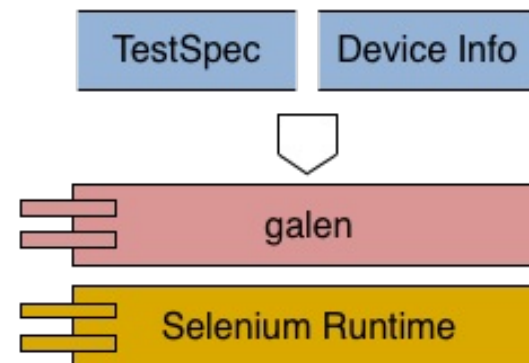
- Test Spezifikationen und Device Infos werden interpretiert
- Tests werden mit Selenium ausgeführt





# Architektur

- Test Spezifikationen und Device Infos werden interpretiert
- Tests werden mit Selenium ausgeführt



- Schnittstellen zu Java und JavaScript

# Analyse design



- Galen wurde aufgrund von Responsive Webdesign entwickelt
- Geräte stehen stellvertretend für Browser und Bildschirmgröße
- Tags sorgen für Zuordnung
- Zur Ausführung wird Browsergröße angepasst und Test ausgeführt

# Galen Spec Language

```
@objects
  content      #content
  heading      .tile_head_line_icon
  Titel
= Content should not exceed screen width =
  content:
    width 90 to 100% of viewport/width
= Content should not exceed component width =
  @on mobile
    content.heading:
      inside parent ~0px left
  @on desktop
    content.heading:
      inside parent ~5px left
  Spec-Regel
```

- Elemente werden zunächst bekannt gemacht
- Geräte(klassen) werden über Tags festgelegt
- Elemente erhalten Regeln, die geprüft werden

# Beispiel

```
@objects
  content      #content
    heading    .tile_head_line_icon

= Content should not exceed screen width =
  content:
    width 90 to 100% of viewport/width

= Content should not exceed component width =
  @on mobile
    content.heading:
      inside parent ~0px left
  @on desktop
    content.heading:
      inside parent ~5px left
```

- Ohne Tag gilt es für alle Varianten

## Wiederverwendung

Prinzipiell mehrere Arten

- Importieren von Specs
- Auslagerung in Komponenten
- Rules

## Import von allgemeinen Specs

```
@objects
  tile-image-* .tile_inline_first
  tile-text-*  .tile_inline_second

@import shared/commonLayout.gspec

= Content should not exceed screen width =
  tile-image-*:
    aligned horizontally all tile-image-1
```

# Komponenten

- Komponenten verhalten sich wie Widgets, z.B. Menüeinträge oder Listeneinträge
- Pseudo-Element 'parent' für Referenzierung

```
@objects
  tile-image-*    .tile_inline_first
  tile-text-*     .tile_inline_second

= Content should not exceed screen width =
  tile-image-*:
    aligned horizontally all tile-image-1

= Info section =
  tile-text-*:
    component shared/tileTextComponent.gspec
```

# Custom Rules

- werden als Extra-Datei abgelegt

```
# Check elements horizontal alignment top e.g.  
@rule %{objectPattern} are aligned horizontally top next to each other  
  @if ${count(objectPattern)} > 1  
    @forEach [${objectPattern}] as item, next as nextItem  
      ${item}:  
        aligned horizontally top ${nextItem}
```

- Über Pattern-Matching nutzbar und mit Pipe in Spec nutzbar

```
@objects  
  login_box    #login  
  label-*      .label  
  username     #username  
  password     #password  
  submit       button  
  
@groups  
  login_box_elements login.label-*,login.username,login.password,login.submit  
  
= Login box looks good =  
  @on desktop, tablet  
    | &login_box_element are aligned horizontally top next to each other
```



# Aufbau von Tests

- Galen erlaubt Parametrisierung:

```
@@ table devices
| device | size |
| mobile | 500x700 |
| tablet | 900x600 |
| desktop | 1300x700 |

@@ parameterized using devices
homepage on ${device}
  ${websiteUrl} ${size}
  check specs/homepage.spec --include "${device}"
```

Ausführung über Kommandozeile

```
galen test my.test -DwebsiteUrl='https://getbootstrap.com' --htmlreport reports/
```

# Testausführung

Einzelne Tests können auch direkt ausgeführt werden

```
$ galen check home-page.spec  
  --url "http://example.com/home"  
  --size 1024x800  
  --include desktop  
  --htmlreport report.html
```

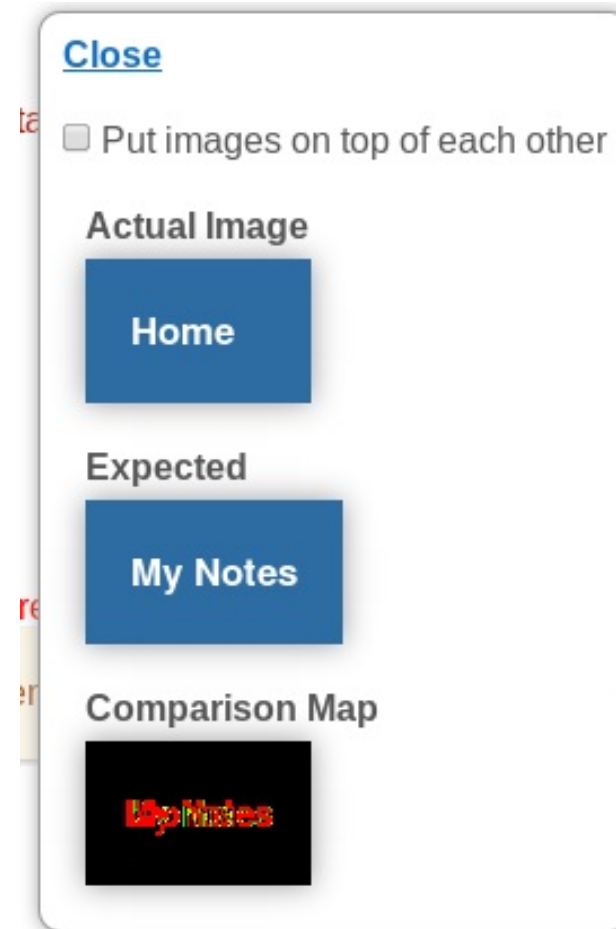
# Bildvergleich

- Pagedump kann Screenshots der aktuellen Seite erzeugen

```
galen dump "specs/homepage.spec"  
  --url "http://galenframework.com"  
  --size "1024x768"  
  --export "dumps/homepage-dump"  
  --max-width "200"  
  --max-height "200"
```

# Vergleich in Spec

```
header-logo  
  image: file imgs/header-logo.png, error 4%
```



# Report

**ERROR** sample.layout.JavascriptLayoutTest->shouldShowCorrectBaseLayout on TestDevice [name=fullhd]

Overall layout

navigation shown on desktop

Content should fit to screen size

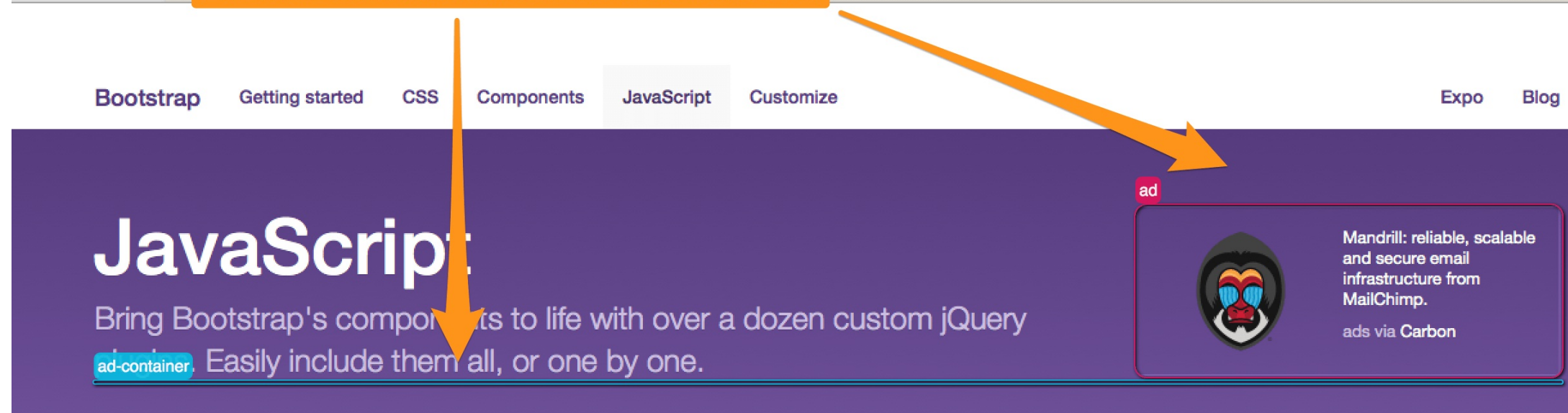
Code snippets should be vertical aligned

Ad should be centered

ad

 centered horizontally on: ad-container

"ad" is not centered horizontally on "ad-container". Offset is 810px



- Report hilft durch direkte Anzeige der Elemente bei Fehlersuche

# Beispiel-Report

## Galen Test Report

<div>TestsGroups</div>									
Test	Passed	Failed	Warnings	Total	Groups	Started	Duration		
<a href="#">Home page on Chrome browser</a>	2	2	0	4		18-08-2015 17:45:13	8s	<div></div>	
<a href="#">Home page on Firefox browser</a>	2	2	0	4		18-08-2015 17:45:01	12s	<div></div>	
<a href="#">Home page on IE 11 browser</a>	2	2	0	4		18-08-2015 17:45:22	16s	<div></div>	
<a href="#">Home page on Opera browser</a>	2	2	0	4		18-08-2015 17:44:49	11s	<div></div>	
<a href="#">Home page on Safari on Mac browser</a>	2	2	0	4		18-08-2015 17:44:23	25s	<div></div>	

# Pagedump

## All objects:

content

header

header-container

js-code-snippet-1

js-code-snippet-10

js-code-snippet-11

js-code-snippet-12

js-code-snippet-13

js-code-snippet-14

js-code-snippet-15

<http://getbootstrap.com/javascript/> - Galen page

viewport

navbar

navbar-item-1

navbar-item-2

navbar-item-3

navbar-item-4

navbar-item-5

navbar-item-6

header

header-container

Neuer Tab

```
galen dump "specs/javascriptPageLayout.gspec" \  
  --url "http://getbootstrap.com/javascript/" \  
  --size "1024x768" --export "dumps/javascript" \  
  --max-width "200" max-height "200"
```

js-code-snippet-1

# Demo

```
# basic.test
@@ table browsers
  | browserName | browser |
  | Chrome      | chrome  |
  | Firefox     | firefox |
@@ table devices
  | deviceName | tags      | size      |
  | Mobile     | mobile    | 320x600   |
  | Desktop    | desktop   | 1024x800  |
@@ parameterized using browsers
@@ parameterized using devices
@@ groups home, page
homepage on ${deviceName} in ${browserName} browser
  selenium ${browser} ${websiteUrl} ${size}
  check specs/homePageLayout.spec --include "${device}"
# run
galen test basic.test -DwebsiteUrl="getbootstrap.com" \
  html-reporter reports
```



# Saucelabs

■ Testing in Cloud möglich

```
@@ set
  sauceKey      aff16b42-9c23-4cb6-adf7-38da9e02193a
  sauceUser     hypery2k
  gridLogin     ${sauceUser}:${sauceKey}
  gridUrl       http://${gridLogin}@ondemand.saucelabs.com:80/wd/hub
  website       http://testapp.galenframework.com

@@ table browsers
  | browserName | gridArgs |
  | Safari on Mac | --browser "safari" --version 6 --dc.platform "OS X 10.8" |
  | Opera | --browser "opera" --version 12 --dc.platform "Linux" |
  | Firefox | --browser "firefox" --version 34 --dc.platform "Linux" |
  | Chrome | --browser "chrome" --version 39 --dc.platform "Linux" |
  | IE 11 | --browser "internet explorer" --version 11 --dc.platform "Windows 8.1" |

@@ parameterized using browsers
Home page on ${browserName} browser
  selenium grid ${gridUrl} --page ${website} ${gridArgs}
  check homepage.spec
```

# Interaktion

- **GalenPages API** hilft bei Interaktionen
- Einfache PageObject-Pattern Bibliothek

```
var login = arg.login, password = arg.password; // test data
this.LoginPage = $page("Login page", {
  email: "input.email", // css locator
  password: "xpath: //input[@class='password']", // xpath locator
  submitButton: "id: submit", // id locator

  load: function () {
    this.open("http://example.com/login");
    return this.waitForIt();
  },
  loginAs: function (userName, password) {
    this.email.typeText(userName);
    this.password.typeText(password);
    this.submitButton.click();
  }
}); // now you can use it like this
new LoginPage(driver).load().loginAs(login, password);
```

# Kombination von GalenPages und Specs

```
@@ table browsers
  | browserName | browser |
  | Chrome      | chrome  |
  | Firefox     | firefox |
@@ table devices
  | deviceName | tags      | size      |
  | Mobile     | mobile    | 320x600   |
  | Desktop    | desktop   | 1024x800  |

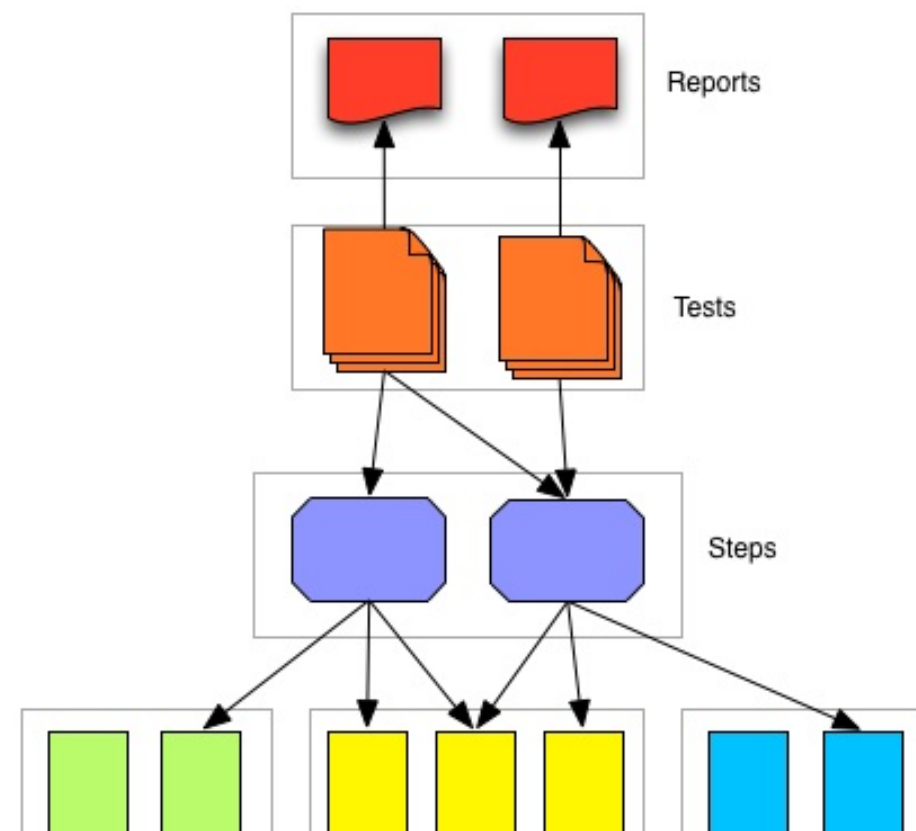
@@ parameterized using browsers
@@ parameterized using devices
@@ groups login, page
login on ${deviceName} in ${browserName} browser
  selenium ${browser} ${websiteUrl} ${size}
  check specs/loginPageLayout.gspec --include "${device}"
  run login.js '{login: "user", password: "password"}'
  check specs/eventPageLayout.gspec --include "${device}"
```

# Oberflächen Testing mit Stil - Serenity

- Tool für ATDD (acceptance test driven development)
- Macht Oberflächen-Tests lesbarer, wartbarer und wiederwendbar
- Nutzt Selenium als technische Basis
- Kann als Java und JBehave-Variante genutzt werden
- Nutzt PagePattern
  - ☐ Page als Model der Webseite mit allen Elemente.
  - ☐ Steps als Gruppierung der möglichen Interaktionsmöglichkeiten, z.B. Klicken
  - ☐ Test ist eine Kollektion von Tests die Benutzereingaben reflektieren und das Ergebnis verifizieren
- Stellt Reporting für Tracking & Fehlersuche zur Verfügung
- Verbindung mit Ticket-System
- Anforderungen auf Tests mappen

# Serenity (2) - Page-Object Pattern

- Testautomatisierungspattern für effiziente und lesbare Tests
- Jede UI Seite wird auf eine Page-Klasse gemappt
- Interaktion wird über Steps-Klassen umgesetzt
- Tests nutzen Steps um Anwendungsverhalten zu simulieren



# Serenity (3) - Beispiel

- Anforderung: Als Nutzer möchte zum Ergebnis meiner Suche navigieren
- Testfall:
  - ☐ Starte Google-Suche
  - ☐ Tippe Suchbegriff ein
  - ☐ Suche Ergebnis in Trefferliste
  - ☐ Navigiere zu Ergebnis
  - ☐ Validiere Ergebnis

# Serenity (4) - Seitenstruktur

```
public class GoogleSearchPage extends PageObject {  
    private WebElement searchInput;    //add your WebElement to the Page  
  
    public GoogleSearchPage (WebDriver driver){  
        super(driver); //add constructor due to PageObject  
    }  
}
```

# Serenity (4) - Seitenstruktur

```
public class GoogleSearchPage extends PageObject {  
    private WebElement searchInput;    //add your WebElement to the Page  
  
    public GoogleSearchPage (WebDriver driver){  
        super(driver); //add constructor due to PageObject  
    }  
}
```

```
public class GoogleSearchPage extends PageObject {  
    // add your WebElement to the Page  
    @FindBy()  
    private  
  
    // add c  
    public G  
    super(  
    }  
}
```

- className : String – FindBy
- css : String – FindBy
- how : How – FindBy
- id : String – FindBy
- jquery : String – FindBy
- linkText : String – FindBy
- name : String – FindBy
- ngModel : String – FindBy
- partialLinkText : String – FindBy
- sclocator : String – FindBy
- tagName : String – FindBy

Press '^Space' to show Template Proposals



# Serenity (5) - Seitenstruktur

- Suchergebnisse werden in Page-Klasse extrahiert

```
public class GoogleResultsPage extends PageObject{
    ...
    @FindBy(id="search")
    private WebElement searchResults;

    public void findResult(String resultsTerm){
        element(searchResults).waitUntilVisible();
        waitFor(ExpectedConditions.presenceOfAllElementsLocatedBy(
            By.cssSelector("div#search li.g")));
        List<WebElement> resultList = searchResults.findElements(
            By.cssSelector("li.g"));

        theFor:
        for(WebElement elementNow:resultList)
            if(elementNow.getText().contains(resultsTerm)){
                element(elementNow).waitUntilPresent();
                elementNow.findElement(By.cssSelector("h3.r a")).click();
                break theFor;
            }
    }
}
```

# Serenity (6) - Interaktion über Steps

- Jegliche Benutzerinteraktion wird in Steps abgebildet
- Steps werden im Report aufgezeigt
- Parameter werden im Report angezeigt
- Methodennamen werden CamelCase gesplittet

```
public class GoogleSteps extends ScenarioSteps{  
    ...  
  
    @Step  
    public void inputSearchTerm(String search){  
        googleSearchPage().inputTerm(search);  
    }  
  
    @Step  
    public void clickOnSearch(){  
        googleSearchPage().clickOnSearch();  
    }  
  
    @StepGroup  
    public void performSearch(String search){  
        inputSearchTerm(search);  
        clickOnSearch();  
    }  
}
```

# Serenity (7) - Tests

- Anforderungen werden im Test abgeglichen
- BDD-like:

```
@RunWith(SerenityRunner.class)
public class GoogleSearchTest {
    @Managed(uniqueSession = true)
    public WebDriver webdriver;

    @ManagedPages(defaultUrl = "https://www.google.com")
    public Pages pages;

    @Steps
    public GoogleSteps googleSteps;

    @Test
    public void googleSearchTest(){
        googleSteps.performSearch("evozon");
        googleSteps.findSearchResult("on Twitter");
        googleSteps.verifyUrl("twitter.com/evozon");
    }
}
```

- CSV-Dateien für Testdaten nutzbar

# Serenity (8) - Report Beispiel

The screenshot displays a Serenity report on the left and the corresponding Java code on the right. The report is titled 'Steps' and lists four test steps. The first three steps are successful, indicated by green checkmarks: 'Perform search: 42', 'Input search term: 42', and 'Click on search'. The fourth step, 'Find search result: Stupidedia', is also successful. The final step, 'Verify url: http://www.wikipedia.org/42', is failed, indicated by a red 'X' icon and a red background. Green arrows point from the first three steps to their respective code blocks, and a red arrow points from the failed step to its code block.

Step	Code
Perform search: 42	<pre>61 @StepGroup 62 public void performSearch(String search) { 63     inputSearchTerm(search); 64     clickOnSearch(); 65 }</pre>
Input search term: 42	<pre>67 @Step 68 public void inputSearchTerm(String search) { 69     googleSearchPage().inputTerm(search); 70 }</pre>
Click on search	<pre>72 @Step 73 public void clickOnSearch() { 74     googleSearchPage().clickOnSearch(); 75 }</pre>
Find search result: Stupidedia	<pre>77 @Step 78 public void findSearchResult(String search) { 79     googleResultsPage().findResult(search); 80 }</pre>
Verify url: http://www.wikipedia.org/42	<pre>82 @Step 83 public void verifyUrl(String url) { 84     assertThat("Url pattern does not match! ", 85</pre>

# Serenity (9) - Demo

[Back to serenity\\_gradle\\_sample](#) [index](#)

[Zip](#)



Home

☒ Overall Test Results

[Requirements](#)

*i* Report generated 20-05-2016 15:04

## Test Results: All Tests

9 test scenarios (11 tests in all, including 3 rows of test data)

3 passed , 2 pending , 6 failed , 0 errors, 0 compromised, 0 ignored, 0 skipped [CSV]

Test Count

Weighted Tests

Total number of tests that pass, fail, or are pending.

Show/Hide Pie Chart

### Test Result Summary

Test Type	Total	Pass	Fail	Pending	Ignored
Automated	11	3 (27%)	6 (55%)	2 (18%)	0 (0%)
Manual	0	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Total	11	3 (27%)	6 (55%)	2 (18%)	0 (0%)
Total Duration	1 minutes 50 seconds				

### Related Tags

% Passed

Test  
count

Features

Search

22.2%



9

Stories

Google Login Test

0%



1

# Fazit & Ausblick

- Feedback ist ein Muss

# Fazit & Ausblick

- Feedback ist ein Muss

- ☐ Teststufen kontinuierlich Feedback über aktuellen Qualitätsstand

# Fazit & Ausblick

## ■ Feedback ist ein Muss

- ☐ Teststufen kontinuierlich Feedback über aktuellen Qualitätsstand
- ☐ Tools müssen das unterstützen



# Fazit & Ausblick

- Feedback ist ein Muss
  - ☐ Teststufen kontinuierlich Feedback über aktuellen Qualitätsstand
  - ☐ Tools müssen das unterstützen
- Im Bereich von CD viele Tools im Bereich Testing & Automatisierung

# Fazit & Ausblick

- Feedback ist ein Muss
  - ☐ Teststufen kontinuierlich Feedback über aktuellen Qualitätsstand
  - ☐ Tools müssen das unterstützen
- Im Bereich von CD viele Tools im Bereich Testing & Automatisierung
- Gefahr von "Over-Engineering" der Schritte in Pipeline

# Fazit & Ausblick

- Feedback ist ein Muss
  - ☐ Teststufen kontinuierlich Feedback über aktuellen Qualitätsstand
  - ☐ Tools müssen das unterstützen
- Im Bereich von CD viele Tools im Bereich Testing & Automatisierung
- Gefahr von "Over-Engineering" der Schritte in Pipeline
- Bewegung im Tool-Bereich (Thoughtworks Go, Docker ...)

# Fazit & Ausblick

- Feedback ist ein Muss
  - ☐ Teststufen kontinuierlich Feedback über aktuellen Qualitätsstand
  - ☐ Tools müssen das unterstützen
- Im Bereich von CD viele Tools im Bereich Testing & Automatisierung
- Gefahr von "Over-Engineering" der Schritte in Pipeline
- Bewegung im Tool-Bereich (Thoughtworks Go, Docker ...)
- Continuous Integration zwingend für Continuous Delivery

# Links

- FeatureToggle Pattern
- Beispiel Anwendung
- GitHub-Projekt Serenity
- Galen Beispiele
- Rock CI mit Jenkins 2 und Docker
- Beispiel Serenity
- Jenkins
  - ☐ Docker image for workflow demo
  - ☐ Workflow-Beispiel Job

*There is no one-size-fits-all solution to the complex problem of implementing a deployment pipeline.” Continuous Delivery, J. Humble, D. Farley*

# About me

■ Martin Reinhardt (Holisticon AG)



■ [github.com/hyper2k](https://github.com/hyper2k)

■ [twitter.com/mreinhardt](https://twitter.com/mreinhardt)