Integration in den Build

Einbinden in Maven, Gradle und Co.

Integration in den Erstellungs-prozess

Sämtlich vorgestellte Test-Frameworks und Test-Ebenen lassen sich in den Erstellungsprozess integrieren.

Tests müssen mit jedem Build automatisiert ausgeführt werden. (CI)

Maven

Integration in Maven

In Maven erfolgt die Integration der Tests über Plugins.

- UnitTests: Surefire
- Integration/ComponentTests: Failsafe

UnitTests in Maven

- Laufen in der Testphase ab ($ mvn test)
- werden über Surefire-Plugin ausgeführt
- http://maven.apache.org/surefire/maven-surefire-plugin

Surefire-Plugin-Bespiel

```
<plugins>
    ...
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.16</version>
        <configuration>
            <argLine>-Dfile.encoding=UTF-8</argLine>
             <includes>
                 <include>**/*Test.*</include>
             </includes>
        </configuration>
    </plugin>
    ...
</plugins>
```

Integration / ComponentTests in Maven

- Laufen in der Integrationstestphase ab ($ mvn integration-test oder $ mvn verify)
- werden über Failsafe-Plugin ausgeführt
- http://maven.apache.org/surefire/maven-failsafe-plugin

Failsafe-Plugin

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-failsafe-plugin</artifactId>
    <version>2.14</version>
    <configuration>
        <argLine>-Dfile.encoding=UTF-8</argLine>
        <includes><include>**/*CT.*</include></includes>
    </configuration>
    <executions>
        <execution>
            <goals>
                <goal>verify</goal>
                <goal>integration-test</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

SmokeTests und SystemIntegrationTests in Maven

Wie realisiere ich die Spitze der Pyramide?

- Dafür gibt es Maven weder eine Phase noch einen Implementierungsvorschlag.
- Maven sieht lediglich 2-Testphasen vor

Lösung 1: Verpacken in JAR

- Test können mit eigener Endung versehen (*IT.*) und in ein ausführbares JAR verpackt werden.
- gegebenfalls eigens Maven-Projekt
- JAR wird dann in Repo-Server hochgeladen
- CI-Server lädt sich JAR aus Repo-Server
- CI-Server führt das JAR und damit die Tests gegen ein Test-System aus

Lösung 2: Eigene POM oder Profil mit Surefire-Plugin

- Tests können mit eigener Endung versehen werden (*IT.*)
- Zusätzlich pom.xml mit anderen Namen in Projekt (z.B. it.pom.xml)
- it.pom.xml verwendet Surefire-Konfig mit *IT.*
- CI-Server checkt Projekt aus
- CI-Server führt Maven parametrisiert aus ($ mvn -f it.pom.xml test)

Gradle

UnitTest mit Gradle

- In Gradle bringt das Java und/oder Groovy Plugin diese Funktionalität mit.
- Kein weiteres Plugin nötig.
- $ gradle test

Integration in Gradle

- In Gradle per se nicht vorgesehen muss über eigene Tasks realisiert werden.
- siehe Beispiel

Integration-Testbeispiel Gradle

```
test {
    maxParallelForks = 5
    forkEvery = 50
    include '**/Test*.*'
}

task integrationTest(type: Test, dependsOn: "test") << {
   include '**/IntegrationTest*.*'
}
```

SmokeTests und SystemIntegrationTests mit Gradle

Ähnlich wie Maven durch Erzeugen eines ausführbaren JARs

oder gleiche Lösung wie bei den Integration-Tests. Eigenen Task definieren:

```
...
task componentTest(type: Test) << {
   include '**/ComponentTest*.*'
}
...
```

Tools und Frameworks

weiter Tools und Frameworks

- UI-Tests: Webdriver / Sellenium
  - http://docs.seleniumhq.org/projects/webdriver

- REST und SOAP-Tests: SoapUI, HttpBuilder
  - http://www.soapui.org (kommerziell)
  - http://groovy.codehaus.org/modules/http-builder

Grails

Testphasen in Grails

Grails sieht per se 3 Testphasen vor

- Unit
- Integration
- Functional

Implementiert aber nur Unit und Integration

Functional kann z. B. Geb und Spock nachgerüstet werden.

Befehle

$ grails test-app

$ grails test-app unit:

$ grails test-app integration:

$ grails test-app functional:

...

SmokeTests und SystemIntegrationTests mit Grails

Sind in Grails nicht vorgesehen. Folgende Lösungen bieten sich an:

1. Eigene Grails-App die diese Tests enthält, ähnlich wie bei Maven, und das vom CI-Server gegen die Test-Umgebung ausgeführt wird.
2. Über eigenes Plugin eine 4. Testphase hinzufügen die nach dem Checkout des Projektes vom CI-Server ausgeführt werden kann z. B. `$grails test-app systemtests:`