HSLU Hochschule Luzern

Verteilte Systeme und Komponenten

Professionelle Entwicklung mit Java

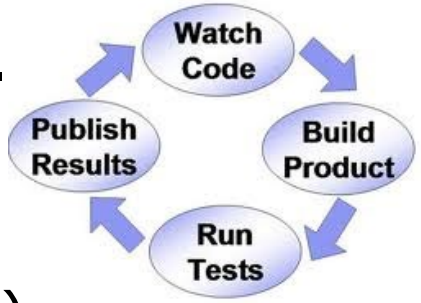
nach den Prinzipen der Continuous Integration (CI)

Roland Gisler

FH Zentralschweiz

Entwicklungsprozess: Ziele im Modul VSK

- Entwicklung eines Java-Projektes aus mehreren Komponenten (bzw. Modulen), im Team, in einer vorbereiteten Projektstruktur.
  - Projekte sind im VCS für Sie bereits erstellt!
- Verwendung eines Versionskontrollsystems (VCS)
  - **git** und **GitLab**, auf Enterprise Lab zur Verfügung gestellt.
- Einsatz einer zeitgemässen, integrierten Entwicklungsumgebung
  - NetBeans, Eclipse, IntelliJ, … – Sie haben die Wahl!
- Continuous Integration (CI)
  - Automatisierter Buildprozess mit Apache Maven.
  - Automatisierte Unit-Tests mit JUnit/AssertJ.
  - Zentrale Codeverwaltung mit Git und GitLab.
  - Zentraler Buildserver mit Jenkins (und GitLab CI).



Software Stack

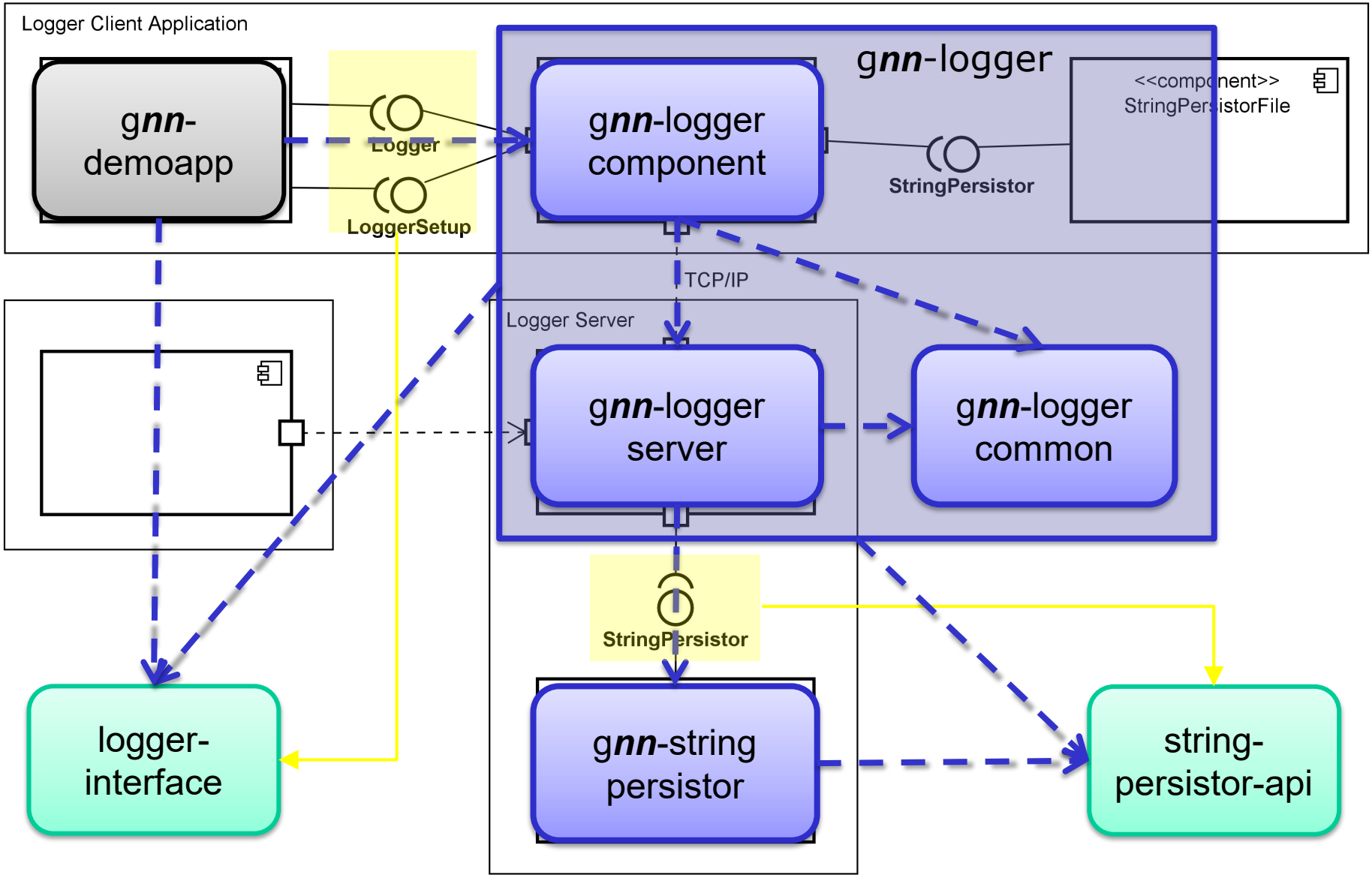
- Oracle Java JDK (inkl. JRE), Version **17.0.4.1 LTS**
  - Wichtig: Wir verwenden alle diesen **einheitlichen** Stand!
  - Nur so ist die Austauschbarkeit zwischen den Teams gegeben.
  - Ältere Java-Versionen werden nicht mehr unterstützt!
- Apache Maven 3.8.6
  - In IDE's enthalten, aber freistehende Installation empfohlen!
  - Konfiguration (**`settings.xml`** im **`$HOME/.m2`**) wichtig!
- git 2.37.3
  - Empfehlung für zusätzlichen Client (IDE-unabhängig): SmartGit von http://www.syntevo.com/smartgit/
- Anleitung für SW-Migration- und Installation: siehe ILIAS!
- SW-Archiv auf SWITCHdrive: http://bit.ly/2OH3Uhh

Java 17 LTS

- Im September 2021 wurde wie geplant der zweite LTS (nach neuem Releaseplan) wenige Tage vor Semesterstart veröffentlicht.
- Dieser LTS löst offiziell die bisherige LTS-Version **11**(.0.13) ab.
- Oracle hat versprochen, nun im Rhythmus von zwei (oder drei) Jahren (2018, 2021, 2023,…) jeweils wieder LTS Versionen (11, 17, 21, …) zu veröffentlichen, welche für den produktiven Einsatz vorgesehen sind.

➔ Darum nutzen wir im HS22 alle einheitlich die Java Version **17.0.4**.1 **LTS** – was besonders im Hinblick auf die Austauchbarkeit von Modulen und Komponenten als Binaries wichtig ist.

Projektstruktur – Logger System (mit DemoApp)



Projekte, Komponenten, Libraries, Applikationen…

- **g*nn*-demoapp** und **g*nn*-loggerserver** sind Applikationen.
  - enthalten typisch eine **main()**-Methode.
- **g*nn*-loggercomponent** und **g*nn*-stringpersistor** sind Komponenten, welche je ein Interface implementieren.
- **loggerinterface** und **stringpersistor-api** sind Schnittstellen.
  - noch spezifischer: API – Application Programmers Interface
- **g*nn*-loggercommon** ist eine reine Library zur Modularisierung.

- Alles sind Projekte, wobei **g*nn*-logger** drei Submodule (**component**, **server** und **logger**) hierarchisch aggregiert.
  - Feature von Apache Maven, erlaubt es u.a. die Module logisch als **eine** einzige Releaseeinheit zu behandeln ➔ Input folgt.

VSK-Projekte auf HSLU-GitLab SCM

- Pro Gruppe vier git-Repositories, mit vorbereiteten Projekten
  - Bekannte Struktur und Build mit Apache Maven.
- Basierend auf dem von OOP und AD bekannten Java-Template.
- Gruppe für Moduldurchführung, Rechte pro Benutzer auf Gruppe.
  - **`VSK-22HS01/g`*`nn`* – **VSK**, **H**erbst**S**emester 20**22**, Kurs **01**
- Struktur/Namensgebung der Projekte (*nn*=Gruppennummer):
  - **`g`*`nn``-demoapp`** – Demo-Applikation für Integration und Test.
  - **`g`*`nn``-stringpersistor`** – Komponente für Dateispeicherung
  - **`g`*`nn``-logger`** – Logger Server, Multimodulprojekt, enthält:
    - **`loggercomponent`** – Logger-Komponente.
    - **`loggerserver`** – Logger Serveranwendung.
    - **`loggercommon`** – Gemeinsame Library (optionale Nutzung).
  - **`g`*`nn``-documentation`** – Für Dokumentation (kein Build/CI).

Schedule der EP-Inputs

- Teil 1 (SW02):
  - Versionskontrollsysteme mit git
  - Buildautomatisation mit Apache Maven
- Teil 2 (SW03):
  - Dependency Management mit Apache Maven
  - Buildserver Technologien mit GitLab CI und Jenkins
- Teil 3 (SW07):
  - Prinzipien der Continuous Integration (CI)

Demo

- Projekte auf GitLab:
  https://gitlab.enterpriselab.ch/vsk-22hs01

Wichtige Hinweise / Empfehlungen

- **git**-Client(s) bitte mit **Klarnamen** (Vorname Name) und korrekter EMail-Adresse konfigurieren, **bitte keine Synonyme** verwenden.
  - siehe Konfiguration in: **`~/.gitconfig`** (im User-**`$HOME`**).
- Ziel: Im VCS befindet sich immer ein kompilierbarer, lauffähiger Stand aller Projekte.
  - Vor dem Commit mindestens kompilieren und testen!
  - Am einfachsten über **`mvn package`** oder **`mvn verify`**.
- Beim Commit schreiben wir **immer** einen aussagekräftigen Kommentar, wenn vorhanden **mit** einer **Issue-Number** (z.B. **`#23`**).
  - Commits werden dadurch in GitLab dem Issue zugeordnet, was die Nachvollziehbarkeit extrem erhöht!

Aufträge

- Klonen sämtlicher Projekte (einzeln, jedes Teammitglied).
  - Integration in die IDE, Verständnis der Funktionsweise.
- Ziel für Heute: Alle sind bereit, um am Logger-Projekt aktiv entwickeln zu können.
- Hilfsmittel für Installation (Java, Maven etc.):
  Anleitung «`OOP_JavaDevelopmentManual_jdk17.pdf`»
  - Erweiterte und aktualisierte Fassung, bekannt aus OOP & AD.
  - Zu finden im ILIAS unter:

HSLU Hochschule Luzern

Fragen?