séance 2 - 30/11/2020

Architectures N-Tiers

Web Ecosystem

- Web languages
- Libraries – Frameworks
- Patterns
- Legacy MVC stacks
- Contemporary stacks
- Data access
- Deployment solutions

Web languages

Description

Document description

1. Html (DOM)
2. Css
3. Svg
4. ~~Xul~~
5. Markdown

Etc…

Data description

1. Sql
2. Jpql
3. Xml
4. ASN.1
5. Json
6. MathML
7. MusicML

Etc…

Execution

Server-side

1. Java
2. Php
3. C,C++
4. Python
5. Ruby
6. Javascript

Etc…

Client-side

1. ~~Java (applets)~~
2. ~~Actionscript~~
3. Javascript

Libraries - Frameworks

- Web languages
- Libraries – Frameworks
- Patterns
- Legacy MVC stacks
- Contemporary stacks
- Data access
- Deployment solutions

Style (libraries)

1. Bootstrap
2. W3c
3. Bulma
4. Uikit
5. Semantic UI
6. Materialize (Google)
7. Pure (Yahoo)

Etc…

Application (libraries)

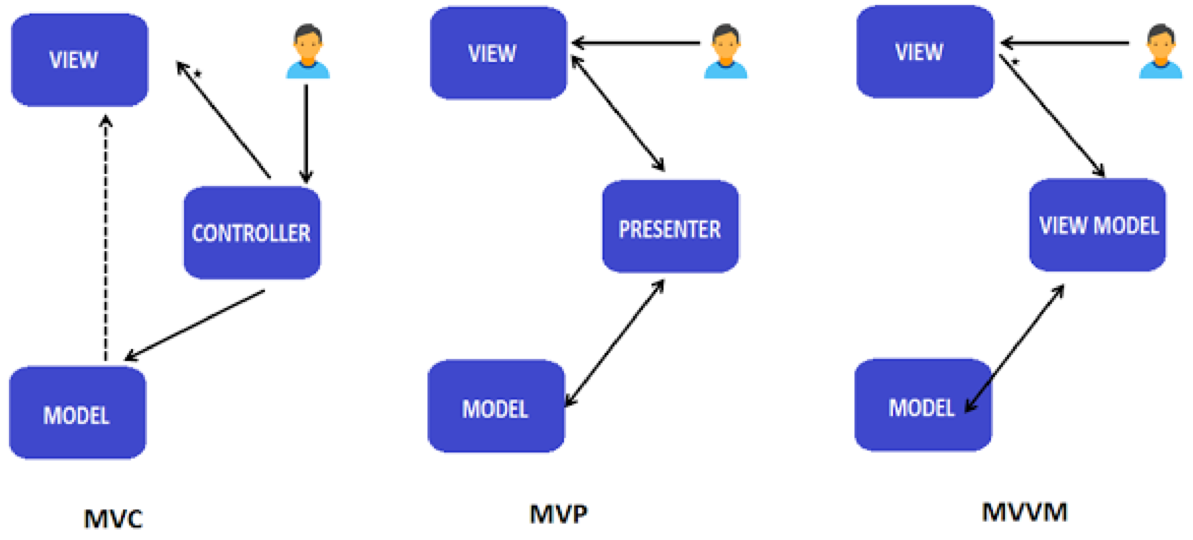
1. jQuery
2. Vue
3. Datatable.net
4. Etc…

Application (frameworks)

1. Angular
2. React
3. Vue
4. Django
5. Flask
6. Laravel
7. Ruby on Rails
8. Backbone

Patterns

- Web languages
- Libraries – Frameworks
- Patterns
- Legacy MVC stacks
- Contemporary stacks
- Data access
- Deployment solutions



https://medium.com/@ankit.sinhal/mvc-mvp-and-mvvm-design-pattern-6e169567bbad

https://www.dotnetdojo.com/mvvm/

Legacy MVC stacks

- Web languages
- Libraries – Frameworks
- Patterns
- Legacy MVC stacks
- Contemporary stacks
- Data access
- Deployment solutions

1. Struts
2. JEE JSP
3. JEE
4. JSF
5. Google GWT
6. Eclipse RAP
7. JavaFx
8. Flex
9. Tapestry
10. Vaadin

Etc …

Contemporary Stacks

Java

1. Spring (Java Backend) + Any Front
2. JEE (Java Backend) + Any Front

Other

1. Node.js - Express (Javascript)
2. Laravel (Php)
3. Symphony (Php)
4. Zend (Php)
5. Django (Python)

Data access

Aka micro-services

Structure

Native

1. SQL

Containers

1. EJB
2. JPA

Transport

Remote call

1. Java RMI
2. Corba
3. Unix ONC RPC
4. Microsoft DCOM

Protocols

1. REST
2. JaxRS
3. JaxB

Deployment solutions

- Web languages
- Libraries – Frameworks
- Patterns
- Legacy MVC stacks
- Contemporary stacks
- Data access
- Deployment solutions

Private Cloud (Self hosted)

1. Virtual Private Server
2. Dedicated Server

Public Cloud

1. Heroku
2. Amazon Aws
3. Microsoft Azure
4. Google cloud

Etc…

Saas

1. Salesforce
2. Microsoft Teams
3. Slack
4. Github
5. Compta
6. Etc…

Scripting languages

1. Php
2. Groovy
3. Javascript
4. Python
5. Perl
6. Ruby

Etc...

Javascript engines

Platforms

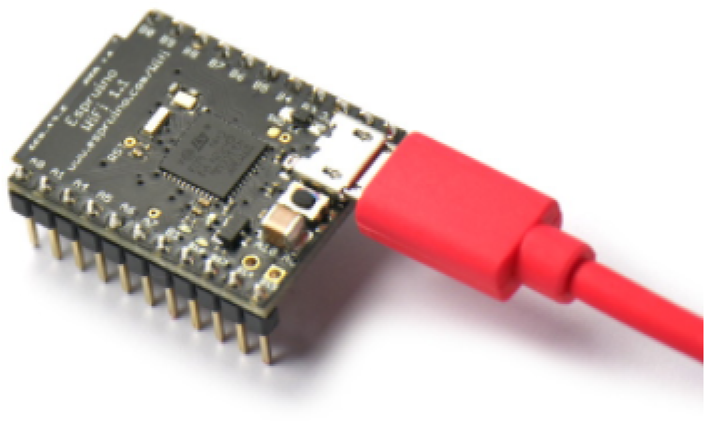
1. Browsers
2. Operating systems
3. Jvm
4. Bare-metal

(Javascript on MCU)

Espruino - Open-Source JavaScript Interpreter For STM32 Microcontrollers

Posted by Industry News on Mar 29th 2020

Share Tweet Share



The Espruino is an open-source JavaScript interpreter for microcontrollers. It is designed for devices with small amounts of RAM (as low as 8kB). It was created by Gordon Williams in 2012 as an effort to make microcontroller development genuinely multiplatform.

Though initially not open-source, the Espruino firmware was offered as a free download for STM32 microcontrollers. It was made open-source in 2013 after a successful Kickstarter campaign for a development board running the software. Since the original Espruino board, there have been many new official development boards.

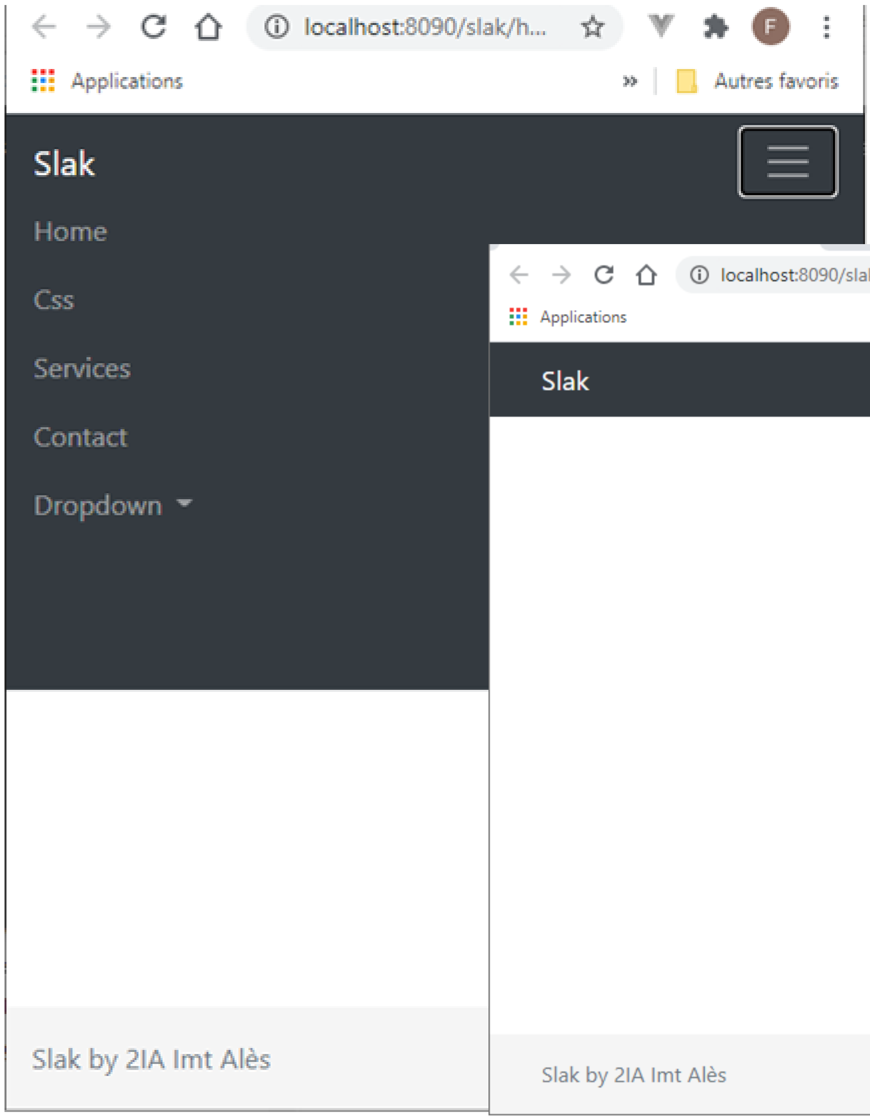
Base Application

Server: Wildfly 16
Maven archetype
Dependencies:

- jboss-jaxrs-api_2.0_spec
- hibernate-jpa-2.1-api
- jboss-ejb-api_3.2_spec
- hibernate-validator
- slf4j-api
- jboss-jsf-api_2.2_spec
- hibernate-jpamodelgen
- hibernate-validator-annotation-processor
- junit
- arquillian-junit-container
- arquillian-protocol-servlet
- httpclient
- httpclient
- javax.json-api
- javax.json
- javax.json.bind-api
- jackson-core
- jackson-dataformat-csv
- jboss-servlet-api_3.1_spec
- commons-lang3
- commons-fileupload
- javax.websocket-api
- mysql-connector-java

Responsive Framework

Bootstrap css



localhost:8090/slak/h...

Applications

Autres favoris

Slak

Home

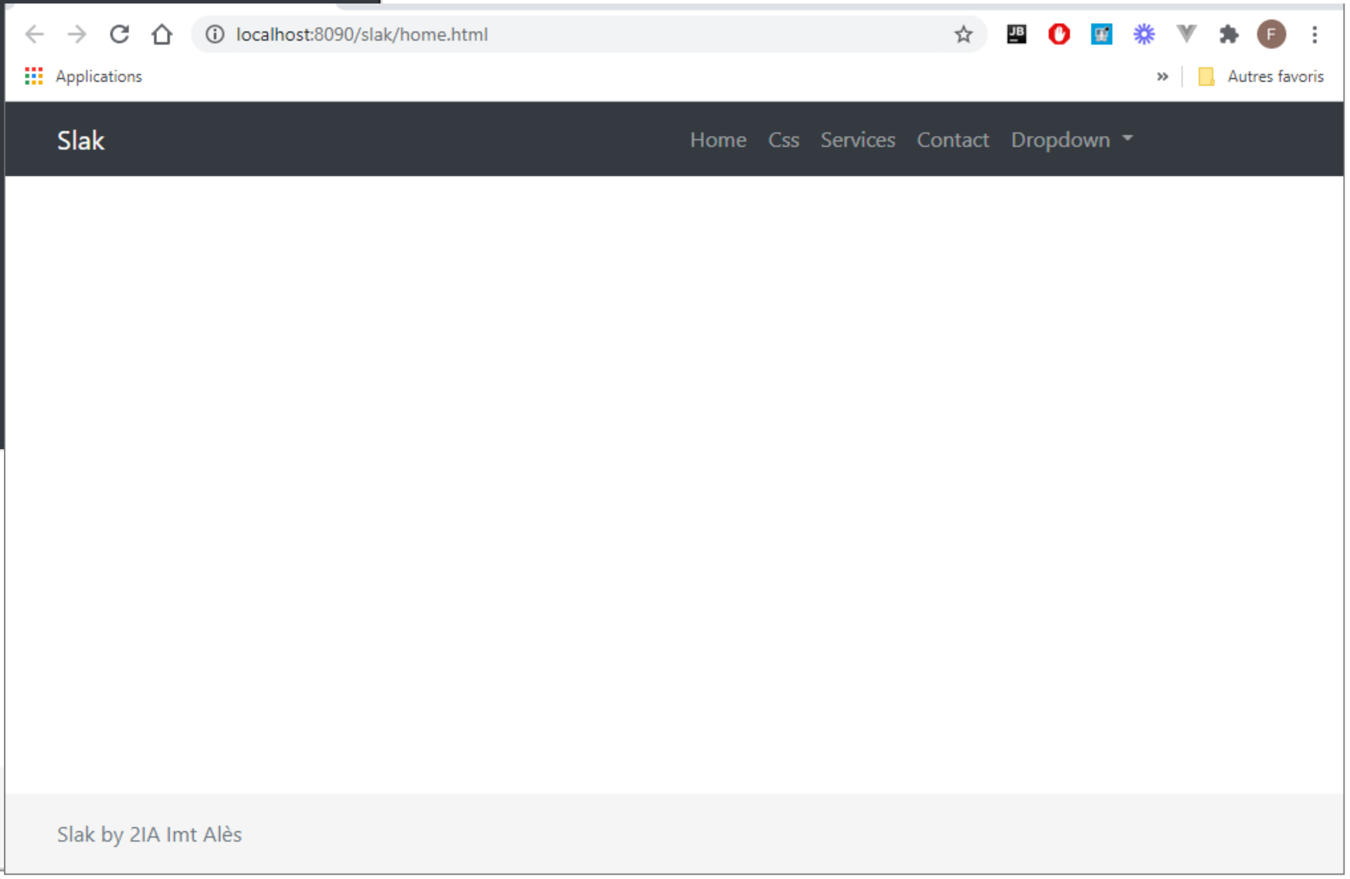
Css

Services

Contact

Dropdown

Slak by 2IA Imt Alès



localhost:8090/slak/home.html

Applications

Autres favoris

Slak

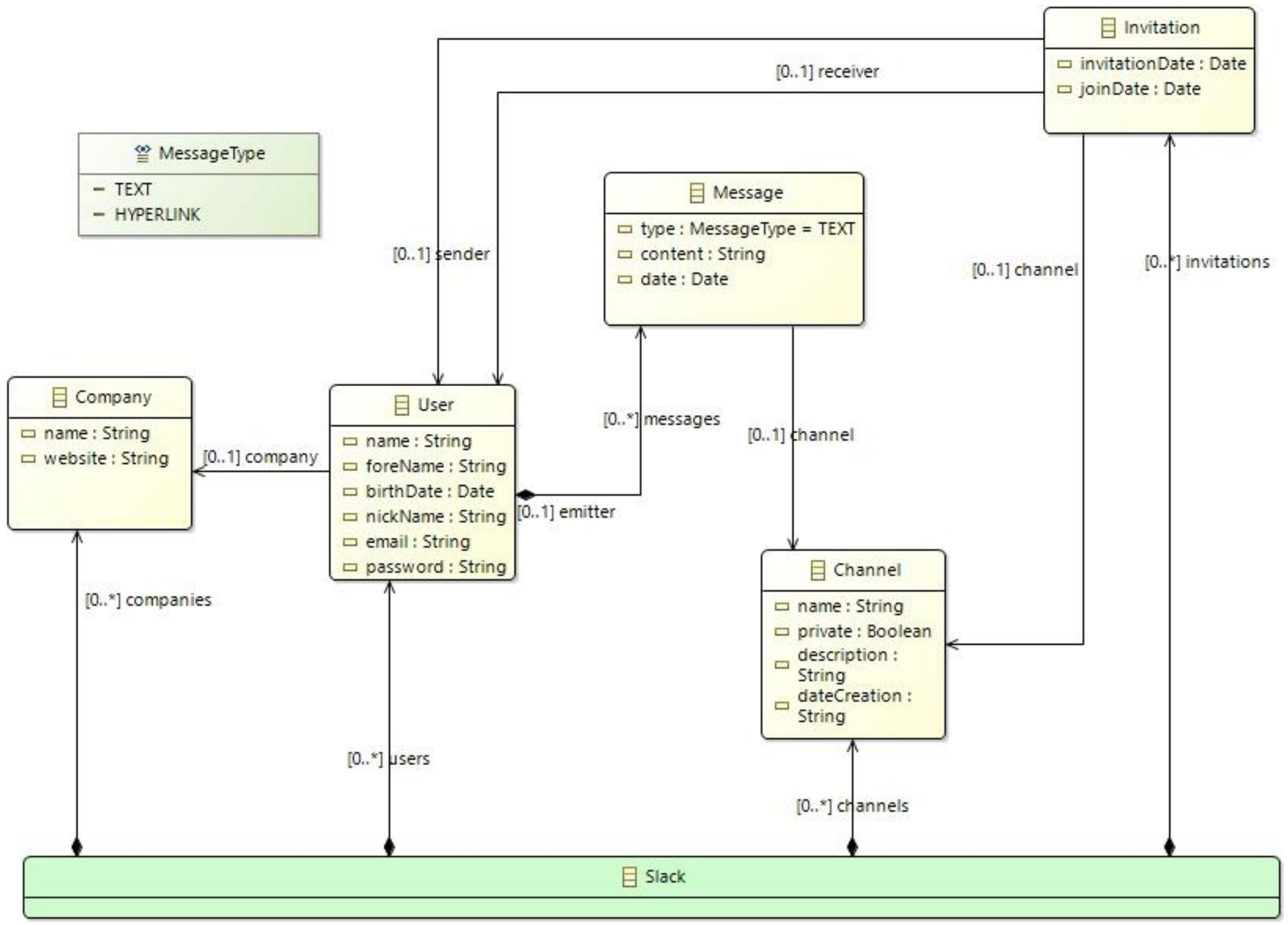
Home Css Services Contact Dropdown

Slak by 2IA Imt Alès

Css Exercises

Js Exercises

Domain model



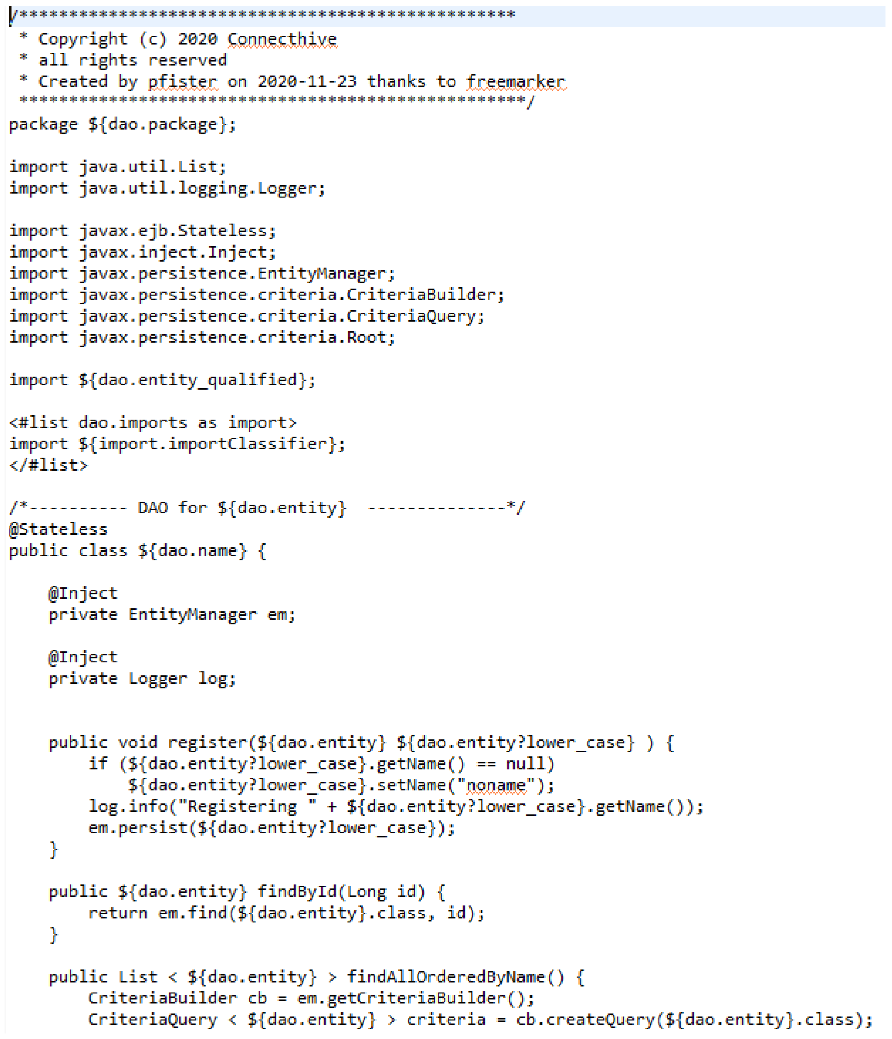
Sujet

L'application à développer est un clone de Slack (https://slack.com). Il s'agit d'un environnement de travail collaboratif dont la base est une messagerie de groupe. Les utilisateurs s'inscrivent à des canaux, ces canaux comportant éventuellement des espaces de travail. Les messages sont émis par l'utilisateur vers un canal et les utilisateurs partenaires de ces canaux reçoivent les messages.
Il est possible d'ajouter des plugins à Slack, ce qui transforme la messagerie en socle d'applications distribuées. Un exemple de plugin est le tableau blanc. On peut y ajouter la visio-conférence, etc...

Travail à faire:

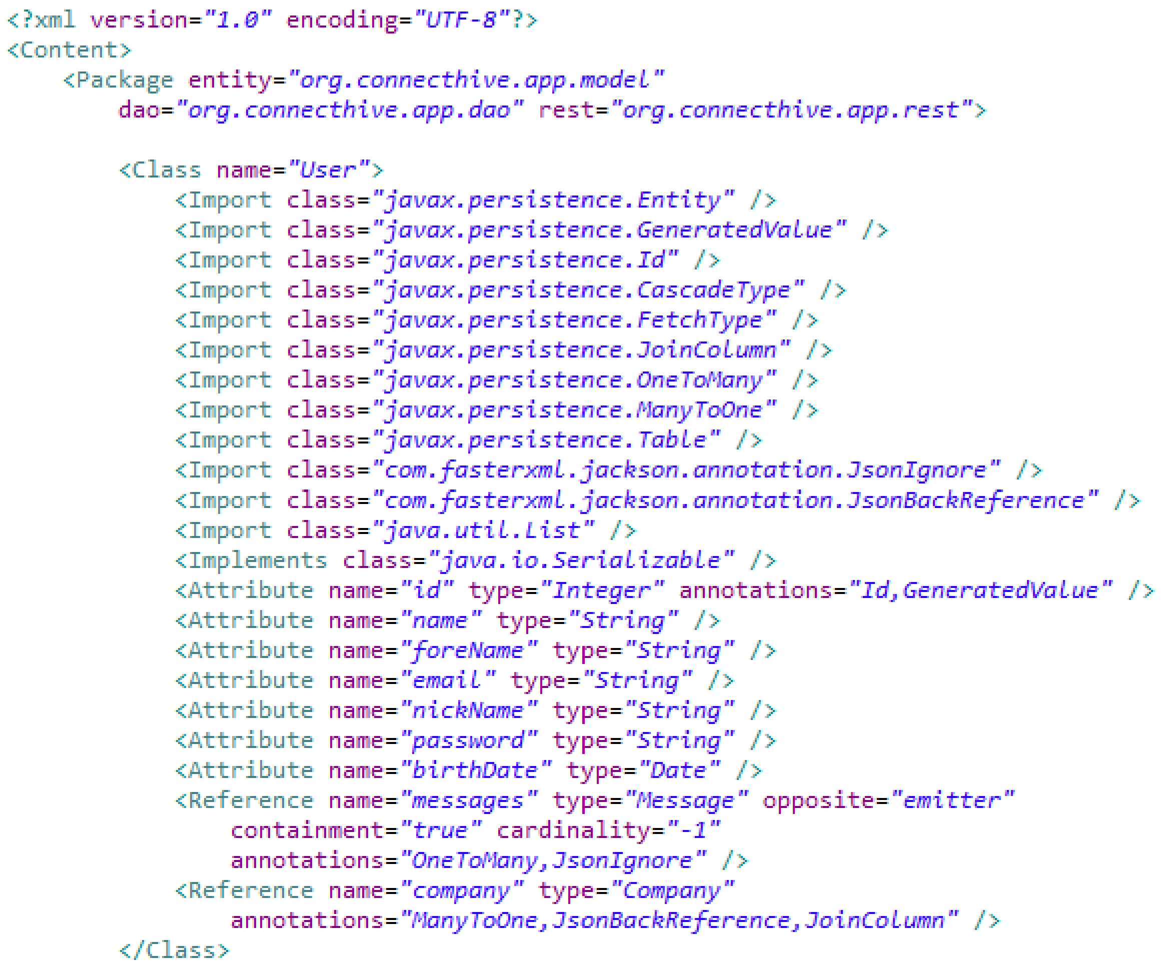
- Instancier un modèle métier basé sur EJB.
- Créer un jeu de données pour le test.
- Créer un serveur backend avec API REST (Jax-RS) pour les données et les règles de gestion.
- Créer un serveur web front-end (Html5 + vanilla javascript)
- Gérer les interactions en temps-réel en utilisant la technologie websocket.

Model transformation based on Freemarker



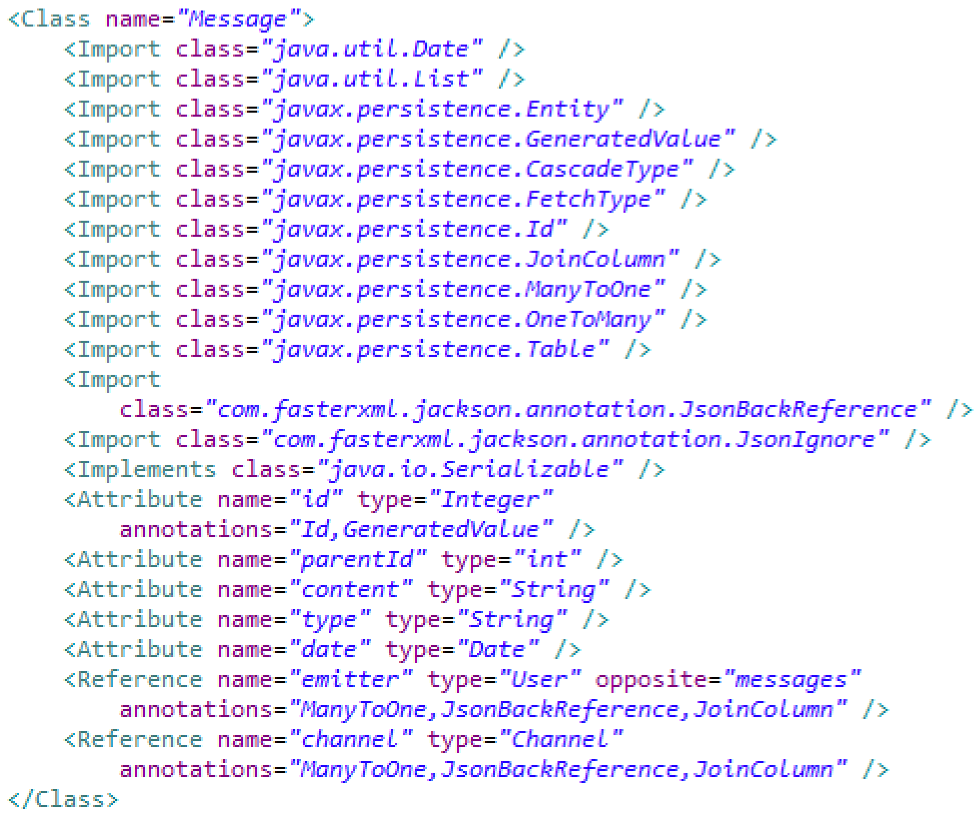
```
/****************************************************
 * Copyright (c) 2020 Connecthive
 * all rights reserved
 * Created by pfister on 2020-11-23 thanks to freemarker
 ****************************************************/
package ${dao.package};

import java.util.List;
import java.util.logging.Logger;

import javax.ejb.Stateless;
import javax.inject.Inject;
import javax.persistence.EntityManager;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;

import ${dao.entity_qualified};

<#list dao.imports as import>
import ${import.importClassifier};
</#list>

/*---------- DAO for ${dao.entity}  --------------*/
@Stateless
public class ${dao.name} {

    @Inject
    private EntityManager em;

    @Inject
    private Logger log;


    public void register(${dao.entity} ${dao.entity?lower_case} ) {
        if (${dao.entity?lower_case}.getName() == null)
            ${dao.entity?lower_case}.setName("noname");
        log.info("Registering " + ${dao.entity?lower_case}.getName());
        em.persist(${dao.entity?lower_case});
    }

    public ${dao.entity} findById(Long id) {
        return em.find(${dao.entity}.class, id);
    }

    public List < ${dao.entity} > findAllOrderedByName() {
        CriteriaBuilder cb = em.getCriteriaBuilder();
        CriteriaQuery < ${dao.entity} > criteria = cb.createQuery(${dao.entity}.class);
```

Transformation => xml



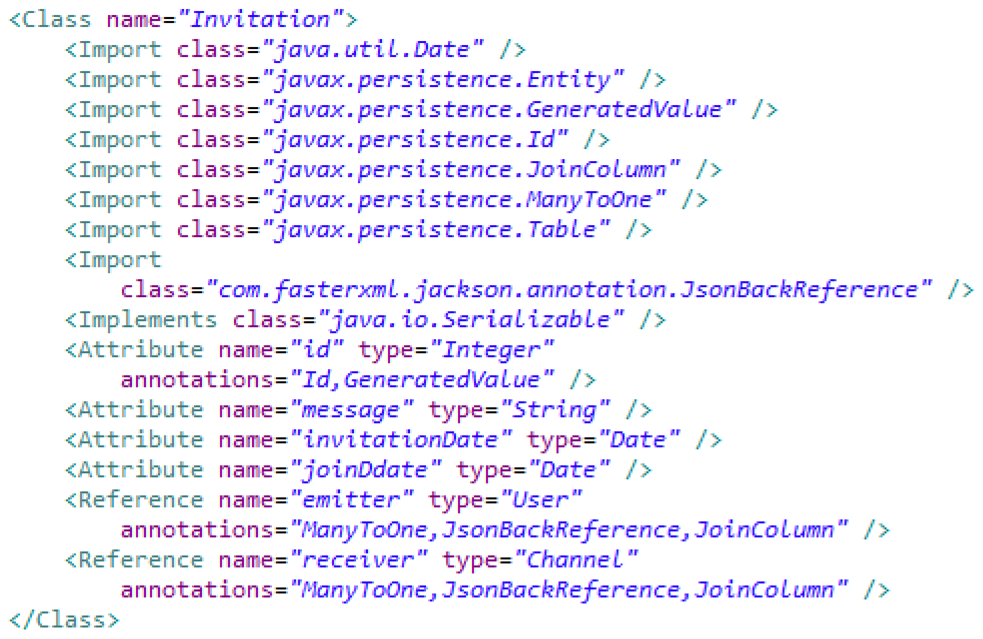
```xml
<?xml version="1.0" encoding="UTF-8"?>
<Content>
    <Package entity="org.connecthive.app.model"
        dao="org.connecthive.app.dao" rest="org.connecthive.app.rest">

        <Class name="User">
            <Import class="javax.persistence.Entity" />
            <Import class="javax.persistence.GeneratedValue" />
            <Import class="javax.persistence.Id" />
            <Import class="javax.persistence.CascadeType" />
            <Import class="javax.persistence.FetchType" />
            <Import class="javax.persistence.JoinColumn" />
            <Import class="javax.persistence.OneToMany" />
            <Import class="javax.persistence.ManyToOne" />
            <Import class="javax.persistence.Table" />
            <Import class="com.fasterxml.jackson.annotation.JsonIgnore" />
            <Import class="com.fasterxml.jackson.annotation.JsonBackReference" />
            <Import class="java.util.List" />
            <Implements class="java.io.Serializable" />
            <Attribute name="id" type="Integer" annotations="Id,GeneratedValue" />
            <Attribute name="name" type="String" />
            <Attribute name="foreName" type="String" />
            <Attribute name="email" type="String" />
            <Attribute name="nickName" type="String" />
            <Attribute name="password" type="String" />
            <Attribute name="birthDate" type="Date" />
            <Reference name="messages" type="Message" opposite="emitter"
                containment="true" cardinality="-1"
                annotations="OneToMany,JsonIgnore" />
            <Reference name="company" type="Company"
                annotations="ManyToOne,JsonBackReference,JoinColumn" />
        </Class>
```

Transformation => xml



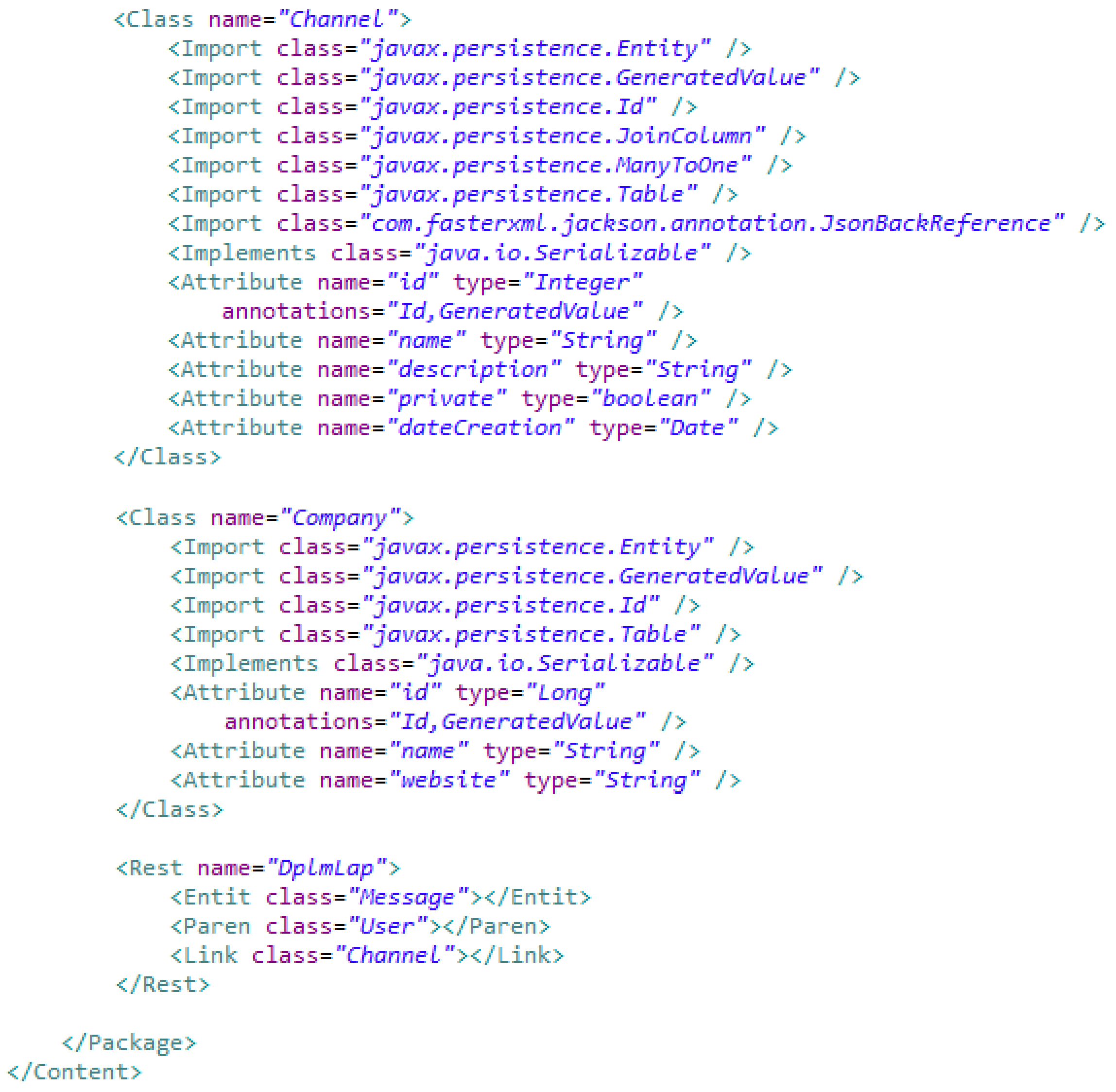
```xml
<Class name="Message">
    <Import class="java.util.Date" />
    <Import class="java.util.List" />
    <Import class="javax.persistence.Entity" />
    <Import class="javax.persistence.GeneratedValue" />
    <Import class="javax.persistence.CascadeType" />
    <Import class="javax.persistence.FetchType" />
    <Import class="javax.persistence.Id" />
    <Import class="javax.persistence.JoinColumn" />
    <Import class="javax.persistence.ManyToOne" />
    <Import class="javax.persistence.OneToMany" />
    <Import class="javax.persistence.Table" />
    <Import
        class="com.fasterxml.jackson.annotation.JsonBackReference" />
    <Import class="com.fasterxml.jackson.annotation.JsonIgnore" />
    <Implements class="java.io.Serializable" />
    <Attribute name="id" type="Integer"
        annotations="Id,GeneratedValue" />
    <Attribute name="parentId" type="int" />
    <Attribute name="content" type="String" />
    <Attribute name="type" type="String" />
    <Attribute name="date" type="Date" />
    <Reference name="emitter" type="User" opposite="messages"
        annotations="ManyToOne,JsonBackReference,JoinColumn" />
    <Reference name="channel" type="Channel"
        annotations="ManyToOne,JsonBackReference,JoinColumn" />
</Class>
```

Transformation => xml



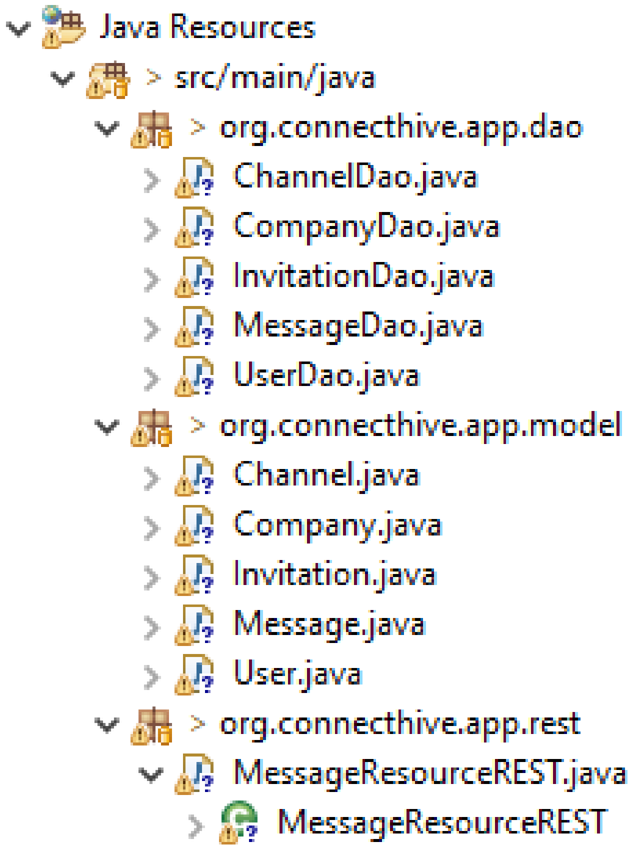
```xml
<Class name="Invitation">
    <Import class="java.util.Date" />
    <Import class="javax.persistence.Entity" />
    <Import class="javax.persistence.GeneratedValue" />
    <Import class="javax.persistence.Id" />
    <Import class="javax.persistence.JoinColumn" />
    <Import class="javax.persistence.ManyToOne" />
    <Import class="javax.persistence.Table" />
    <Import
        class="com.fasterxml.jackson.annotation.JsonBackReference" />
    <Implements class="java.io.Serializable" />
    <Attribute name="id" type="Integer"
        annotations="Id,GeneratedValue" />
    <Attribute name="message" type="String" />
    <Attribute name="invitationDate" type="Date" />
    <Attribute name="joinDdate" type="Date" />
    <Reference name="emitter" type="User"
        annotations="ManyToOne,JsonBackReference,JoinColumn" />
    <Reference name="receiver" type="Channel"
        annotations="ManyToOne,JsonBackReference,JoinColumn" />
</Class>
```

Transformation => xml



```xml
        <Class name="Channel">
            <Import class="javax.persistence.Entity" />
            <Import class="javax.persistence.GeneratedValue" />
            <Import class="javax.persistence.Id" />
            <Import class="javax.persistence.JoinColumn" />
            <Import class="javax.persistence.ManyToOne" />
            <Import class="javax.persistence.Table" />
            <Import class="com.fasterxml.jackson.annotation.JsonBackReference" />
            <Implements class="java.io.Serializable" />
            <Attribute name="id" type="Integer"
                annotations="Id,GeneratedValue" />
            <Attribute name="name" type="String" />
            <Attribute name="description" type="String" />
            <Attribute name="private" type="boolean" />
            <Attribute name="dateCreation" type="Date" />
        </Class>

        <Class name="Company">
            <Import class="javax.persistence.Entity" />
            <Import class="javax.persistence.GeneratedValue" />
            <Import class="javax.persistence.Id" />
            <Import class="javax.persistence.Table" />
            <Implements class="java.io.Serializable" />
            <Attribute name="id" type="Long"
                annotations="Id,GeneratedValue" />
            <Attribute name="name" type="String" />
            <Attribute name="website" type="String" />
        </Class>

        <Rest name="DplmLap">
            <Entit class="Message"></Entit>
            <Paren class="User"></Paren>
            <Link class="Channel"></Link>
        </Rest>

    </Package>
</Content>
```

Generation: Entity + Stateless + REST



```
Java Resources
  src/main/java
    org.connecthive.app.dao
      ChannelDao.java
      CompanyDao.java
      InvitationDao.java
      MessageDao.java
      UserDao.java
    org.connecthive.app.model
      Channel.java
      Company.java
      Invitation.java
      Message.java
      User.java
    org.connecthive.app.rest
      MessageResourceREST.java
        MessageResourceREST
```