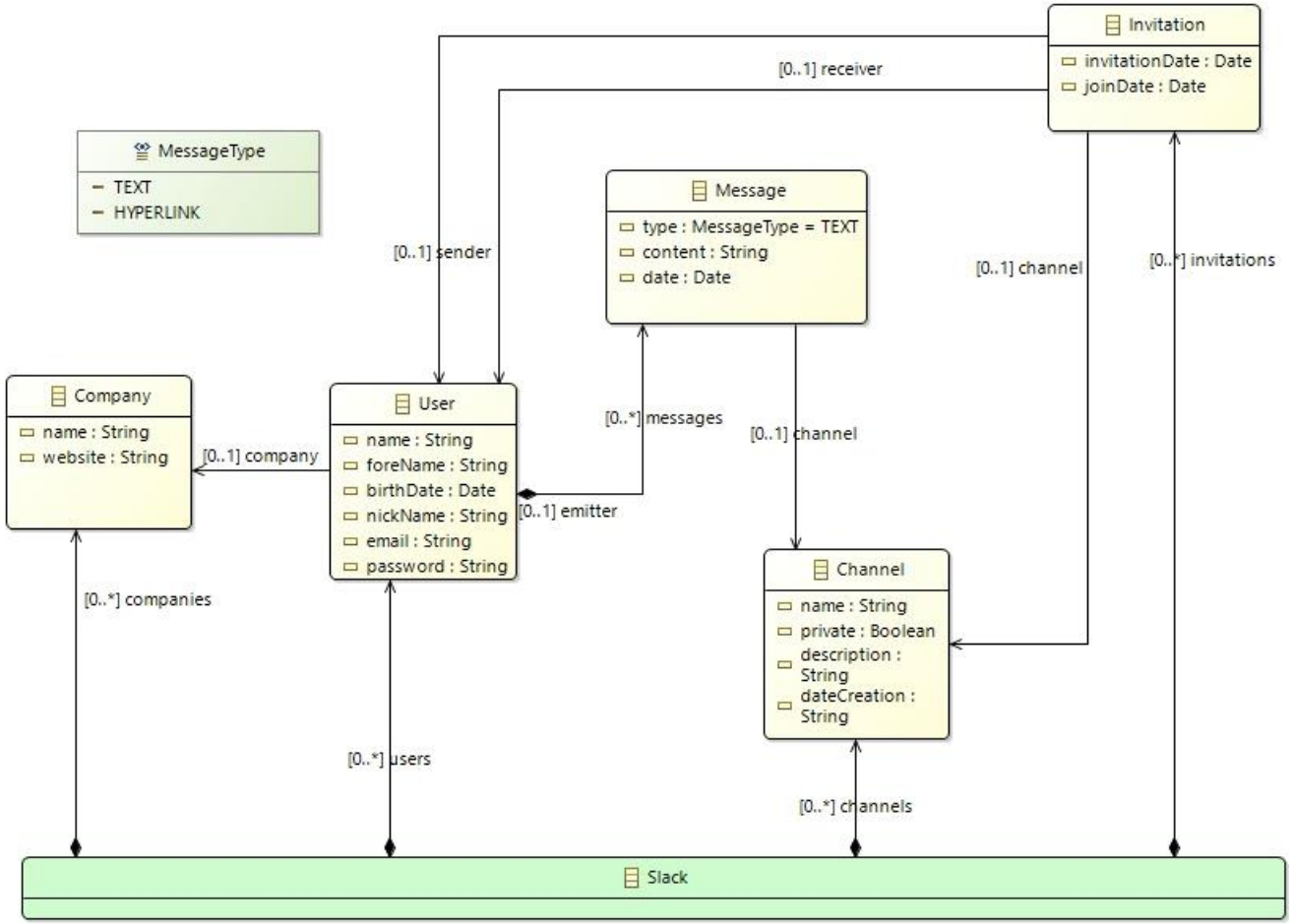
séance 3 – 02/12/2020

Architectures N-Tiers

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

```
<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

```
<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

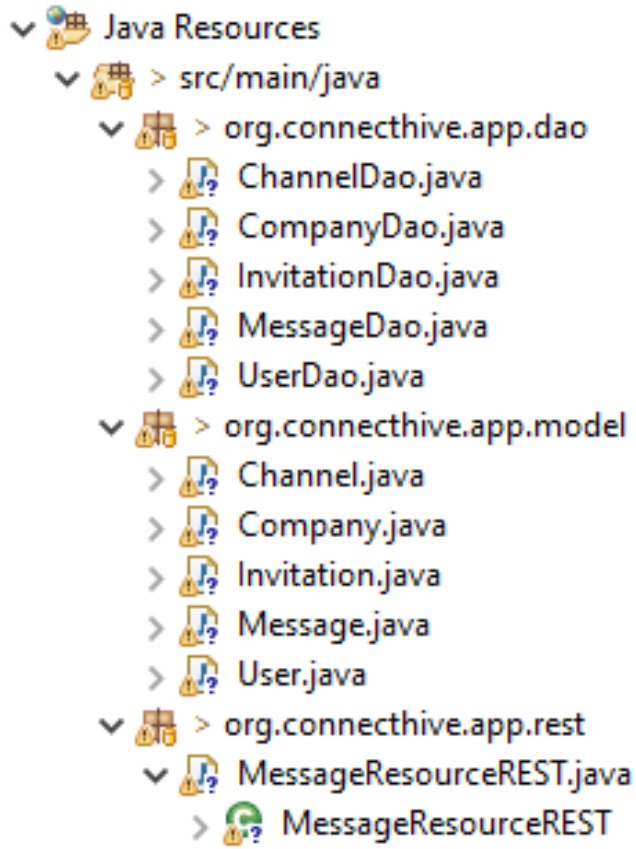
```
<Class name="Channel">
    <Import class="javax.persistence.Entity" />
    <Import class="javax.persistence.GeneratedValue" />
    <Import class="javax.persistence.Id" />
    <Import class="javax.persistence.JoinColumn" />
    <Import class="javax.persistence.ManyToOne" />
    <Import class="javax.persistence.Table" />
    <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



Installation et configuration wildfly

- Télécharger le serveur wildfly (16 par exemple)
- Onglet server: new server, indiquer le dossier d'installation du serveur.
- Faire la configuration CORS (voir document Configuration_wildfly.pdf)

Javascript Templating

Let's "templatize" the markup of one of the items by using placeholders for the data like:

```
<ul id="list">
  <!--Generated list items will go here-->
</ul>

<script id="template-list-item" type="text/template">
  <li>
    <a href="{{url}}">{{name}}</a
  </li>
</script>
```

Placeholder formats like `{{url}}` s search-and-replace, but they're arbitr whatever you want (i.e. `%url%`, `<%`

Templating with Objects

Now we need to parse the data and generate the HTML. Here's an example object:

```
0: {
  "id": 5,
  "name": "Basecamp",
  "city": "Chicago",
  "state": "IL",
  "url": "https://basecamp.com/"
},
1: {
  "id": 17,
  "name": "Google",
  "city": "Mountain View",
  "state": "CA",
  "url": "http://google.com/"
}
```

https://jonsuh.com/blog/javascript-templating-without-a-library/

JQuery serializeArray()

Output the result of form values serialized as arrays:

```
$("button").click(function(){
  var x = $("form").serializeArray();
  $.each(x, function(i, field){
    $("#results").append(field.name + ":" + field.value + " ");
  });
});
```

Definition and Usage

The serializeArray() method creates an array of objects (name and value) by serializing form values.

You can select one or more form elements (like input and/or text area), or the form element itself.

https://www.w3schools.com/jquery/ajax_serializearray.asp

https://apimirror.com/jquery/serializearray