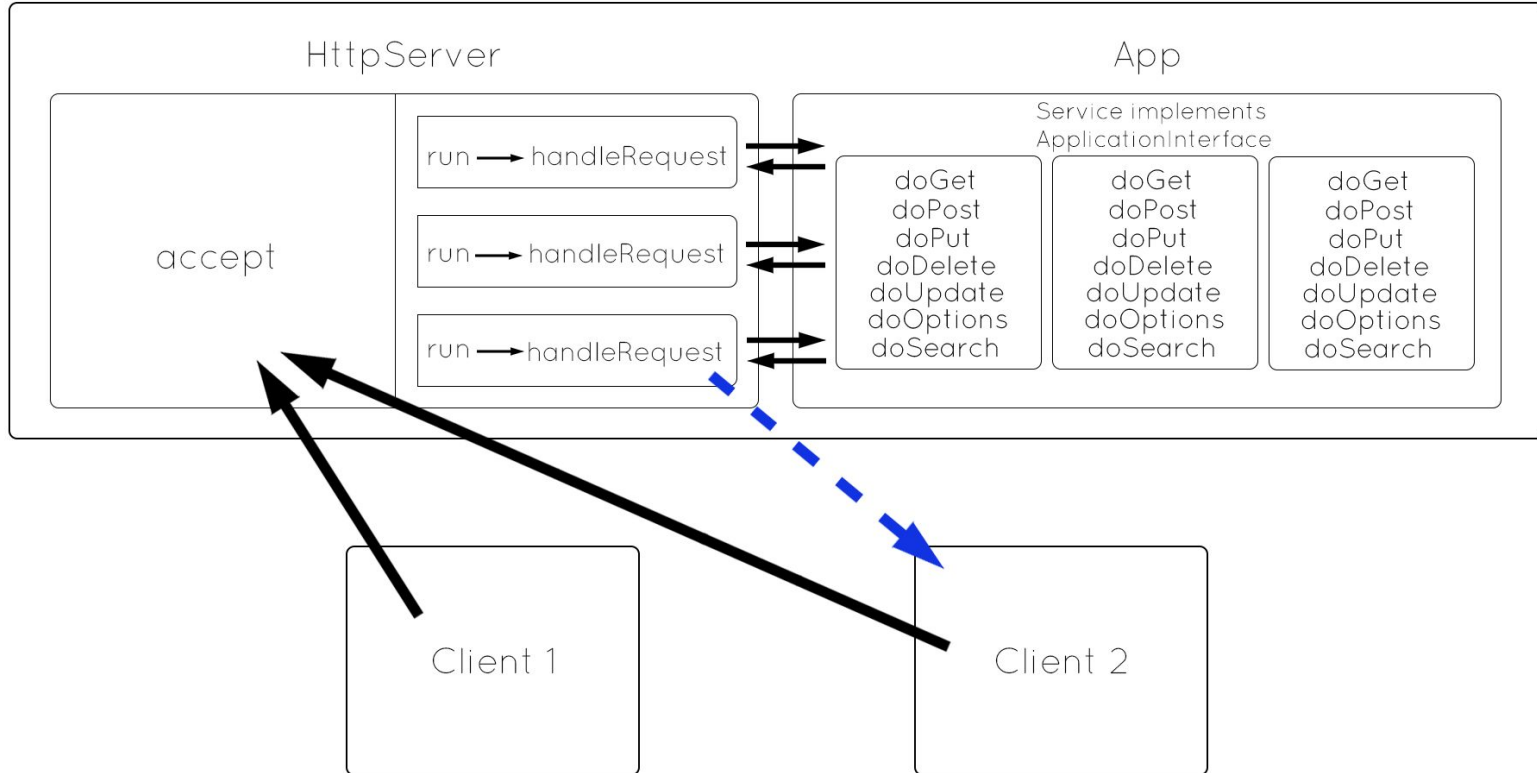


# { HT Framework }

Ladislav Halifa  
Patrick Tran

# 1. Serveur HTTP

## Serveur



## 2. Routage

- Basé sur un fichier XML

```
<routing>

  <mapping>
    <name>PointId</name>
    <class>apps.pointApp.PointId</class>
    <url-pattern>/Point/p/[0-9]+/(x|y)</url-pattern>
  </mapping>

  .....

</routing>
```

## 2. Routage

- Vérification des paramètres d'URL obligatoires

```
<mapping>  
  <name>PointId2</name>  
  <class>apps.pointApp.PointId2</class>  
  <url-pattern>/Point/p/[0-9]+\?x=\d+&y=\d+</url-pattern>  
</mapping>
```

## 2. Routage

- Vérification des paramètres d'URL facultatifs

```
<mapping>  
  <name>PointId2</name>  
  <class>apps.pointApp.PointId2</class>  
  <url-pattern>/Point/p/[0-9]+((\?x=\d+& y=\d+) | (\?x=\d+) | (\?y=\d+))?</url-pattern>  
</mapping>
```

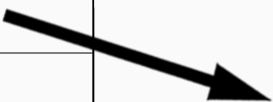
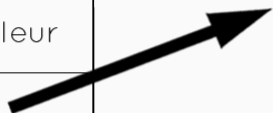
## 2. Routage

- Table de routage créée au lancement du serveur
- Pas de collisions
- Utilisation de l'API Java Réflexion
- Fichier web.xml commun à toutes les applications

### 3. Sessions

Cookie Table

clé	valeur
uniqueId	
uniqueId	



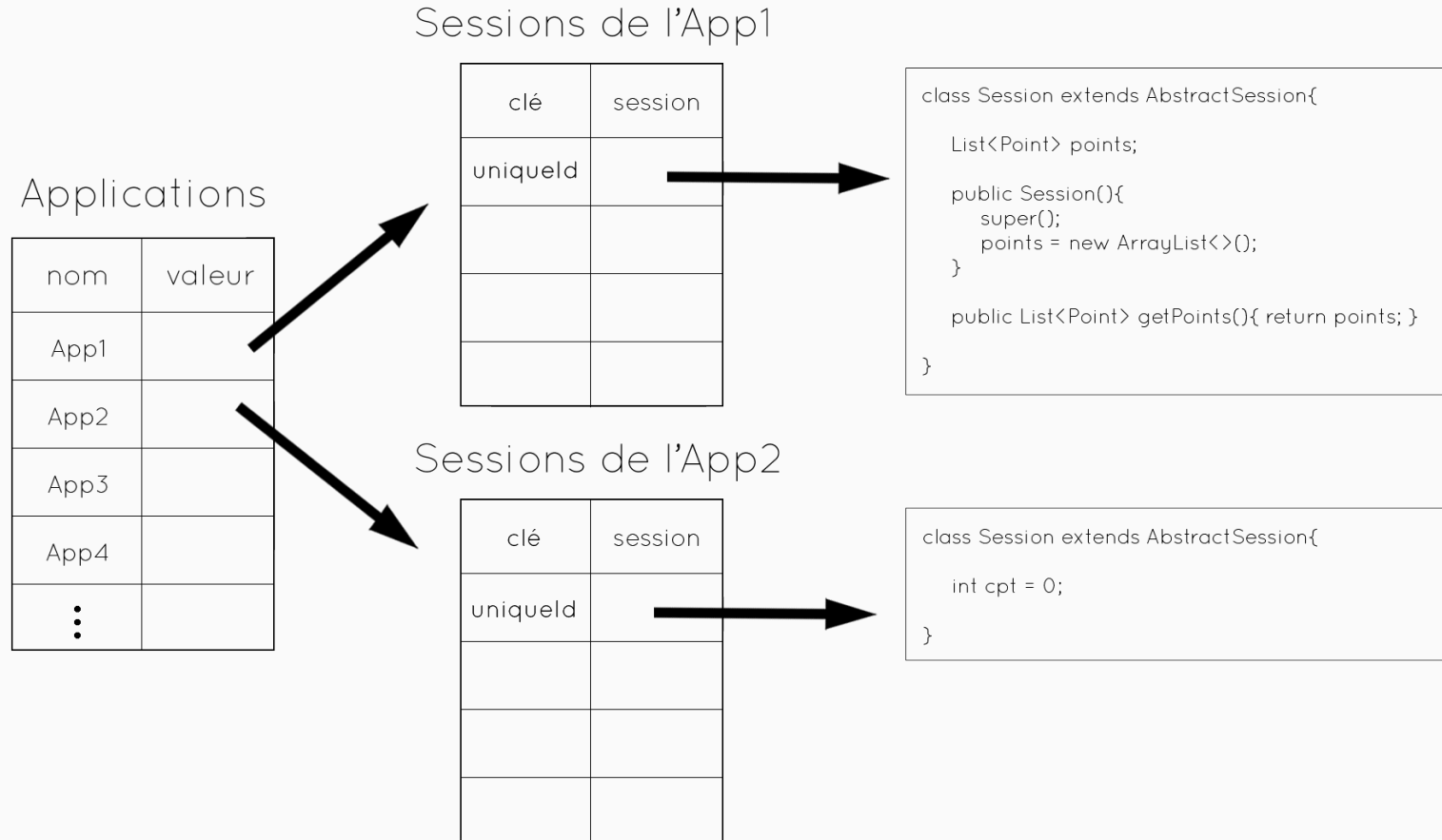
Cookies d'un utilisateur

userHash hash(IP+ userAgent)	clé valeur	clé valeur	...
------------------------------------	---------------	---------------	-----

Cookies d'un autre utilisateur

userHash hash(IP+ userAgent)	clé valeur	clé valeur	...
------------------------------------	---------------	---------------	-----

### 3. Sessions






## 4. Patrons

### Template

```
<table border="1px">
  <tr>
    <th>Date</th><td>%date%</td>
  </tr>
  <tr>
    <th>Adresse</th><td>%adresse%</td>
  </tr>
  <tr>
    <th>Username</th><td>%user.username%</td>
  </tr>
  <tr>
    <th>Prenom</th><td>%user.prenom%</td>
  </tr>
  <tr>
    <th>Nom</th><td>%user.nom%</td>
  </tr>
  <tr>
    <th>Age</th><td>%user.age%</td>
  </tr>
</table>
```

### Environnement

variables	valeurs
date	"21/03/17"
user	
adresse	"4 place Jussieu"



username	"toto"
prenom	"paul"
nom	"luap"
age	30

## 4. Patrons

→ Récupérer le template via son path

→ Matcher avec toutes les expressions avec le pattern `%((\w+)((\.)\w+))?)%`

→ Vérifier s'il s'agit d'un type simple ou d'un objet

↳ Type simple

→ Récupérer la valeur de la variable dans l'environnement

↳ Objet

→ Vérifier si l'attribut existe avec:

```
Class methodClass = Class.forName(env.get(varName).getClass().getName());  
Field[] classFields = methodClass.getDeclaredFields();
```

→ Invoquer le getter correspondant

```
getMethod = "get"+attrName.toUpperCase().charAt(0)+attrName.substring(1);  
Method method = methodClass.getMethod(getMethod);  
Object classInstance = env.get(varName);  
value = method.invoke(classInstance) + "";
```

→ Remplacer toutes les variables dans le template par la valeur correspondante

# 5. Démo

```

public class Login implements ApplicationInterface {

    @Override
    public ApplicationResponseInterface doPost(RequestInterface request, SessionInterface session) {
        ApplicationResponseInterface response = new ApplicationResponse();
        try {

            Session s;
            JSONObject jsonBody = new JSONObject(request.getBody());
            JSONObject jsonResponse = null;
            jsonResponse = UserServices.login(jsonBody.getString("login"), jsonBody.getString("password"));

            if (jsonResponse.has("id")) {
                int id = jsonResponse.getInt("id");
                if (session != null) {
                    s = (apps.todoList.setvlet.model.Session) session;
                    if (id != s.getId()) {
                        s.setId(id);
                        SessionManager.save(request, s);
                    }
                } else {
                    s = new Session();
                    s.setId(id);
                    SessionManager.save(request, s);
                }
            }

            response.setBody(jsonResponse);
            response.setContentType(Headers.APPLICATION_JSON);
        } catch (JSONException e) {
            response.setBody(ResponseBuilder.serverResponse(StatusCode.INTERNAL_SERVER_ERROR, request.getUrl().getPath()));
            response.setContentType(Headers.TEXT_HTML);
            e.printStackTrace();
        }
        return response;
    }
}

```