Nouvelles technologies de développement en JEE

- Stephane Epardaud
- Nicolas Leroux

Plan

- Introduction
- Conception
- Implémentation
  - Storage-tiers
  - Middle-tiers
  - Web-tiers
- Améliorations
- Conclusion

Plan

Introduction

LUNATECH
RESEARCH

Introduction

- Qui sommes-nous ?
- Java EE très populaire
- Beaucoup d'innovations
- Avec Java 5 (annotations), code très différent
- Présentations de certaines technos récentes

Plan

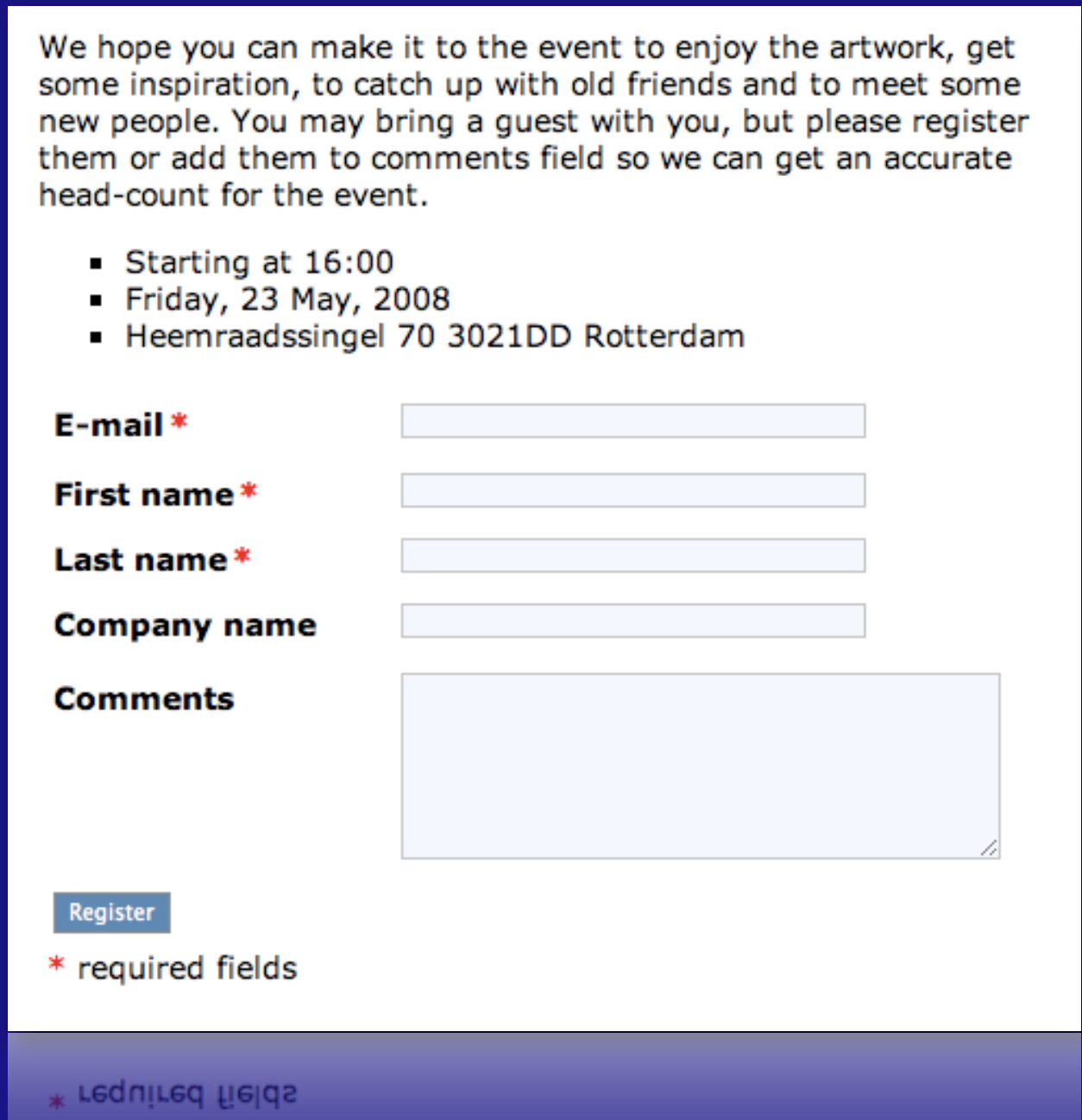
Conception

LUNATECH
RESEARCH

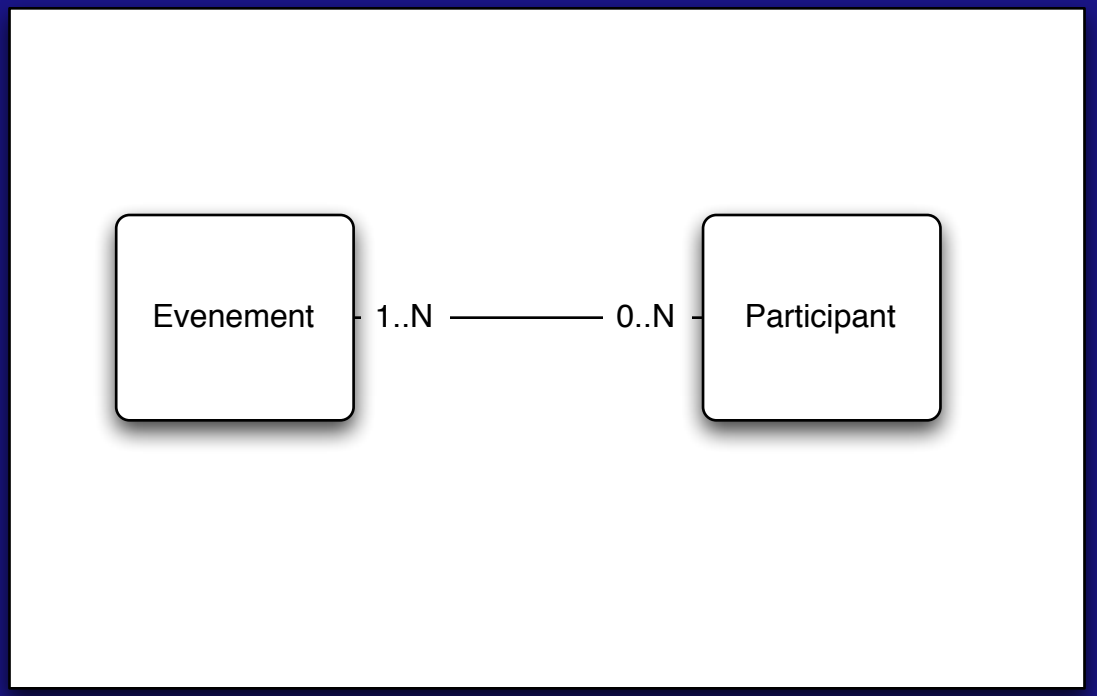
Le problème

- Créer une application web pour s'inscrire à des évenements
  - Combien de gens ?
  - Qui ?
  - Garder les contacts
  - Vue d'utilisateur
  - Vue d'administrateur

Prototype


We hope you can make it to the event to enjoy the artwork, get some inspiration, to catch up with old friends and to meet some new people. You may bring a guest with you, but please register them or add them to comments field so we can get an accurate head-count for the event.
Starting at 16:00
Friday, 23 May, 2008
Heemraadssingel 70 3021DD Rotterdam
E-mail *
First name *
Last name *
Company name
Comments
Register
* required fields

Le design: Modele



Le design: modèle

- Évenement
  - Titre
  - Date
  - Description
  - Identifiant
  - Participants

Le design: modèle

- Participant
  - Prénom
  - Nom
  - Companie
  - Adresse Mail
  - Évenements

Plan

Implémentation:

Le storage-tiers

LUNATECH
RESEARCH

Implementation: JPA

- Représentation de la base de donnée en Java
  - Entity Bean
- Table ⇔ Classe
- Ligne ⇔ Instance
- Définition du mapping par des annotations
- Hibernate, Toplink, etc...

Ex: Evenement.java

```
@Entity
public class Evenement implements Serializable {

 @GeneratedValue
 @Id
 private int id;

 private String titre;
 private Date date;
 private String description;
 private String identifiant;

 @ManyToMany
 private List<Participant> participants =
   new ArrayList<Participant>();

 // getters, setters...
}
```

Participant.java

```
@Entity
public class Participant implements Serializable {

 @GeneratedValue
 @Id
 private int id;

 private String prenom;
 private String nom;
 private String companie;
 private String addresseMail;

 @ManyToMany
 private List<Evenement> evenements =
   new ArrayList<Evenement>();

 // getters, setters...
}
```

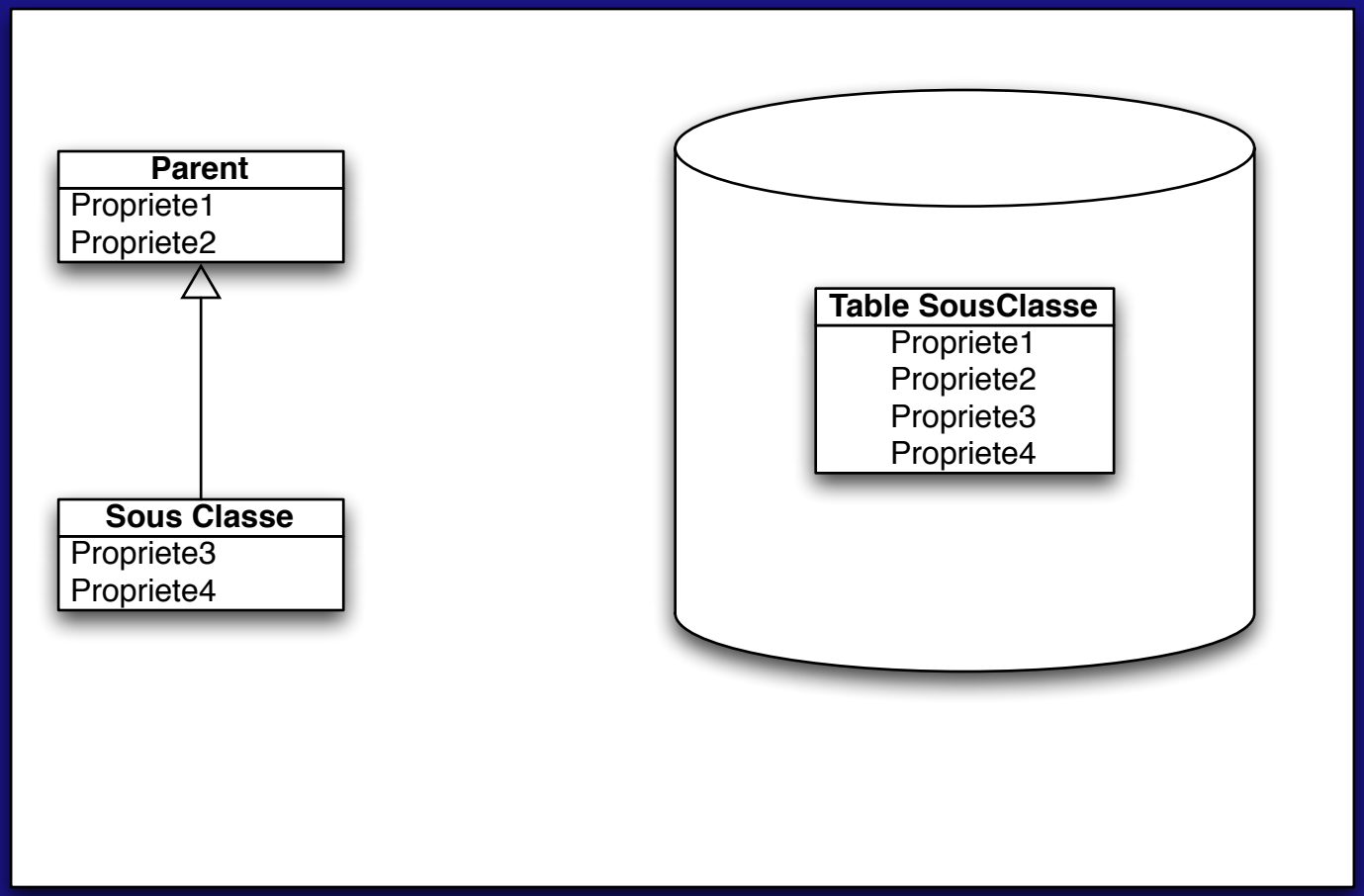
LUNATECH
RESEARCH

Autres annotations

- @Table
- @Column
- @OneToMany, @ManyToMany, @ManyToOne
  - @OrderBy
  - @Sort
- Héritage
  - @MappedSuperclass
  - @Inheritance, etc...

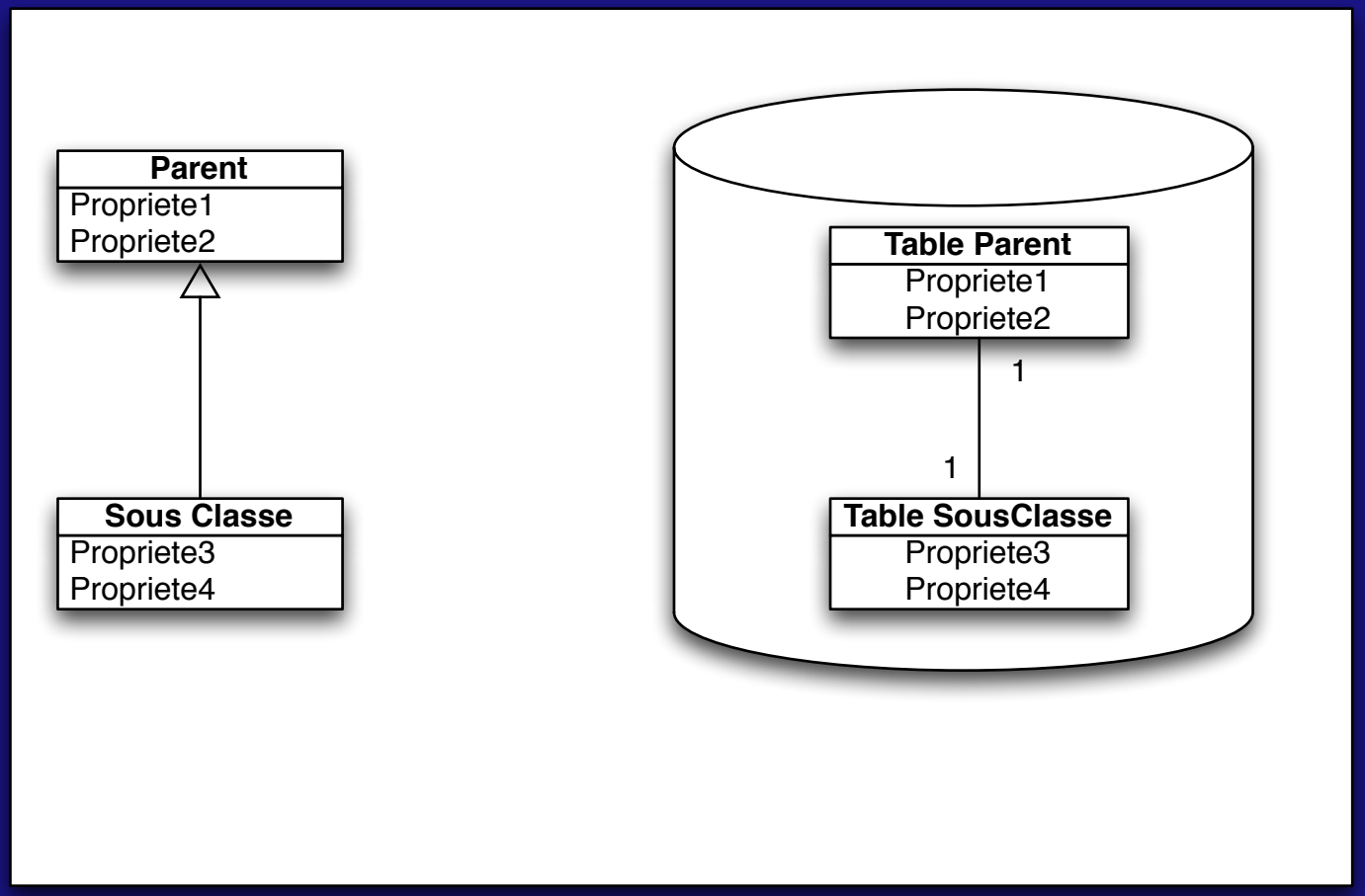
Types d'héritages 1

- @MappedSuperclass



Types d'héritages 2

- @Inheritance(strategy=JOINED)



Plan

Implémentation:
Le middle-tiers

Utilisation du modèle

- Framework: JBoss Seam
  - JPA (Hibernate), JAAS (Securite), Mail, PDF, etc...
  - Enterprise Java Beans 3 (EJB 3)
    - Middle-tier: web beans (JSR-299) et DAO (business logic)
  - Java Server Faces (JSF) / Facelets
    - Web tier
    - Templates

LUNATECH RESEARCH

Middle-tier

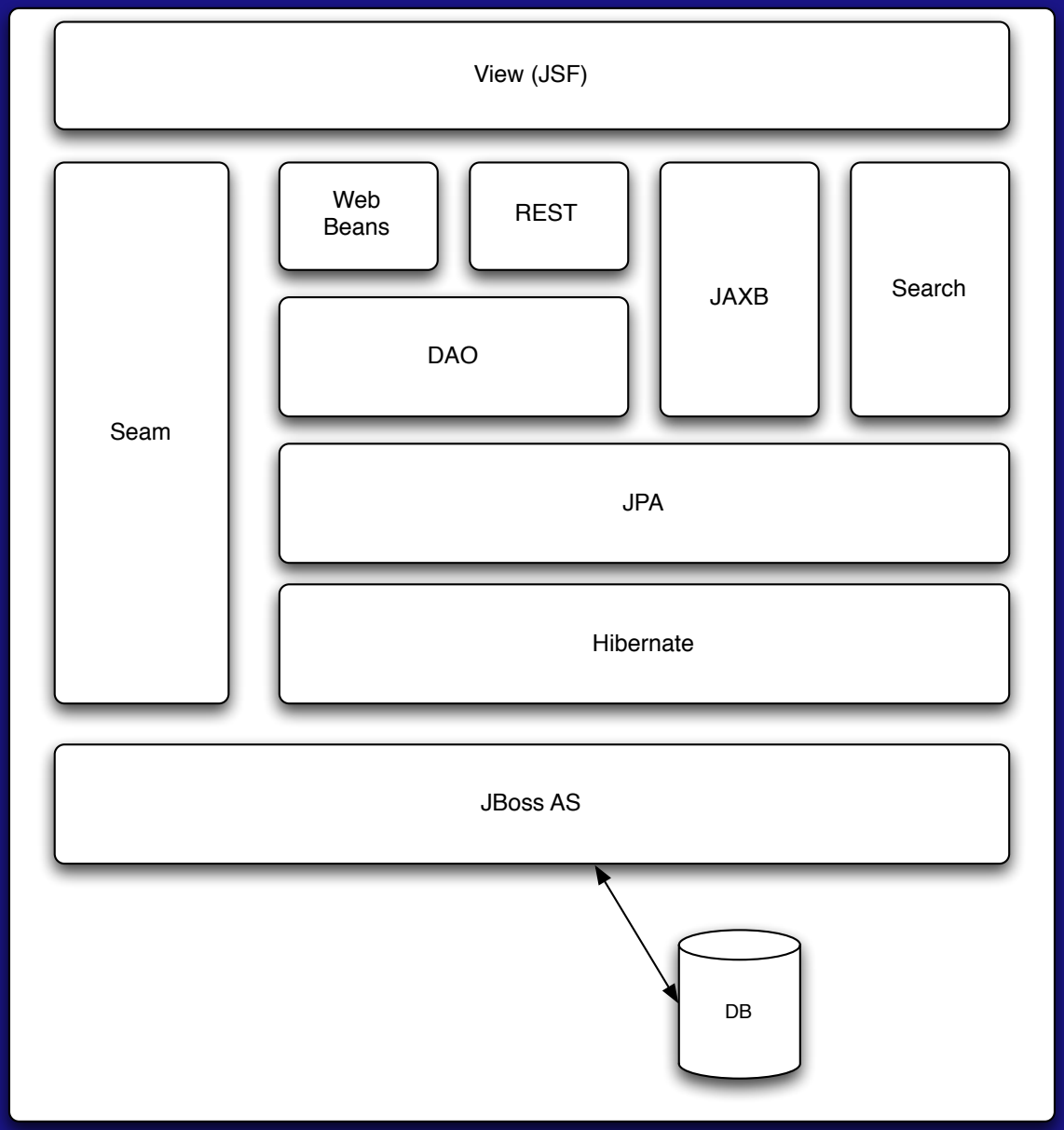
- Web Bean
  - Composant Seam
  - Actions du web-tier
  - Interface avec les DAOs
- Data Access Object
  - Composant EJB3 ou POJO
  - Interface avec JPA
  - Hibernate Query Language (HQL) - SQL façon objet

LUNATECH
RESEARCH

EJB3

- Interface @Local/@Remote
  - Ou Plain Old Java Object / POJO (EJB 3.1/JSR-318)
  - Lookup/Injection automatique
  - Proxy transparent
    - Répartition
  - Stateless
  - Stateful
  - MDB
  - *Entity* (maintenant JPA)

Middle-tiers



Ex: EvenementDAO

```
@Local
public interface EvenementDAO {

 public List<Evenement> findEvenements();
 public Evenement findEvenement(String id);
 public Evenement save(Evenement evt);
 public void delete(Evenement evt);

}
```

EvenementDAOBean

```
@Stateless
@Name("evenementDAO")
public class EvenementDAOBean implements EvenementDAO {

@In
private EntityManager entityManager;

public List<Evenement> findEvenements() {
    return (List<Evenement>)
      entityManager.createQuery("FROM Evenement ORDER BY date, titre")
       .getResultList();
}

public Evenement findEvenement(String identifiant) {
    return (Evenement)
      entityManager.createQuery("FROM Evenement where identifiant = :identifiant")
        .setParameter("identifiant", identifiant)
        .getSingleResult();
}
```

HQL

- FROM Evenement AS evt WHERE evt.participants.size > 0 ORDER BY evt.date, evt.titre
- HQL:
  - FROM Foo AS foo WHERE foo.bar.customer.address.city = 'Nice'
- SQL:
  - SELECT foo.id, foo.bar, ... FROM foo LEFT JOIN bar ON bar.id = foo.bar LEFT JOIN customer ON bar.customer = customer.id LEFT JOIN address ON address.id = customer.address WHERE address.city = 'Nice'

Les web beans

- Bijection
  - @In/@Out(scope = ScopeType.PAGE)
- Contextes
  - Event
  - Page
  - Conversation
  - Session
  - Application

LUNATECH RESEARCH

EvenementWeb

```
@Stateless
public interface EvenementWeb {

  public void initEvenementList();

  public void add();

  public void edit();

  public void save();

  public void cancel();

  public void remove();

  public void destroy();
}
```

EvenementWebBean 1

LUNATECH RESEARCH

```
@Stateful
@Name("evenementAction")
public class EvenementWebBean implements EvenementWeb {

  @DataModel
  private List<Evenement> evenementList;

  @DataModelSelection
  private Evenement selectedEvenement;

  @In(required = false)
  @Out(required = false)
  private Evenement editedEvenement;

  @In @Out
  private boolean isAddingEvenement;

  @EJB
  private EvenementDAO evenementDAO;

  // ...
```

EvenementWebBean 2

```
@Factory("evenementList")
public void initEvenementList(){
    evenementList = evenementDAO.findEvenements();
}
public void add(){
  editedEvenement = new Evenement();
  isAddingEvenement = true;
}

public void edit(){
  editedEvenement = selectedEvenement;
  isAddingEvenement = false;
}

public void remove(){
  evenementDAO.delete(selectedEvenement);
  evenementList = null;
}
// ...
```

EvenementWebBean 3

```
    // ...

 public void cancel(){
    isAddingEvenement = false;
    editedEvenement = null;
  }

  public void save() {
    evenementDAO.save(editedEvenement);
    editedEvenement = null;
    isAddingEvenement = false;
  }

  @Remove
  @Destroy
  public void destroy() { }

}
```

Plan

Implémentation:
Le web-tiers

LUNATECH
RESEARCH

Les vues

- Java Server Faces
  - XHTML + tags spéciaux
  - Facelets
- Templates
- Unified Expression Language (UEL)
  - Mapping des beans
  - #{evenement.titre}
  - #{evenementAction.save()}

Le template 1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/
html4/strict.dtd">

<html xmlns:f="http://java.sun.com/jsf/core"
      xmlns:ui="http://java.sun.com/jsf/facelets">
 <f:view contentType="text/html">
  <head>
   <title><ui:insert name="title">Template</ui:insert></title>
   <link rel="stylesheet" type="text/css" href="/stylesheet/main.css"/>
   <ui:insert name="head"/>
  </head>
  <body class="#{identity.loggedIn ? 'signedin':'signedout'}">
   <div id="masthead">
    <h1>Foo Bar</h1>
    <ui:fragment rendered="#{identity.loggedIn}">
     <a href="/evenements/">Evenements</a>
     <a href="/participants/">Participants</a>
    </ui:fragment>
   </div>
```

Le template 2

```
<!-- ... -->

<div class="content">
 <ui:insert name="content"/>
</div>

</body>
<f:view>
</html>
```

Vues: evenements 1

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
                xmlns:h="http://java.sun.com/jsf/html"
                xmlns:s="http://jboss.com/products/seam/taglib"
                xmlns:ui="http://java.sun.com/jsf/facelets"
                template="/theme/template.xhtml">
<ui:define name="title">Évenements</ui:define>
<ui:define name="content">
 <h:form>
  <table>
   <tr><th>Date</th><th>Titre</th></tr>
   <ui:repeat value="#{evenementList}" var="evenement">
    <tr>
     <td>
      <s:fragment rendered="#{editedEvenement == evenement}">
       <h:inputText value="#{editedEvenement.date}"/>
      </s:fragment>
      <s:fragment rendered="#{editedEvenement != evenement}">
       <h:ouputText value="#{evenement.date}">
         <f:convertDateTime pattern="dd MM yyyy"/>
       </h:outputText>
      </s:fragment>
     </td>
```

Vues: evenements 2

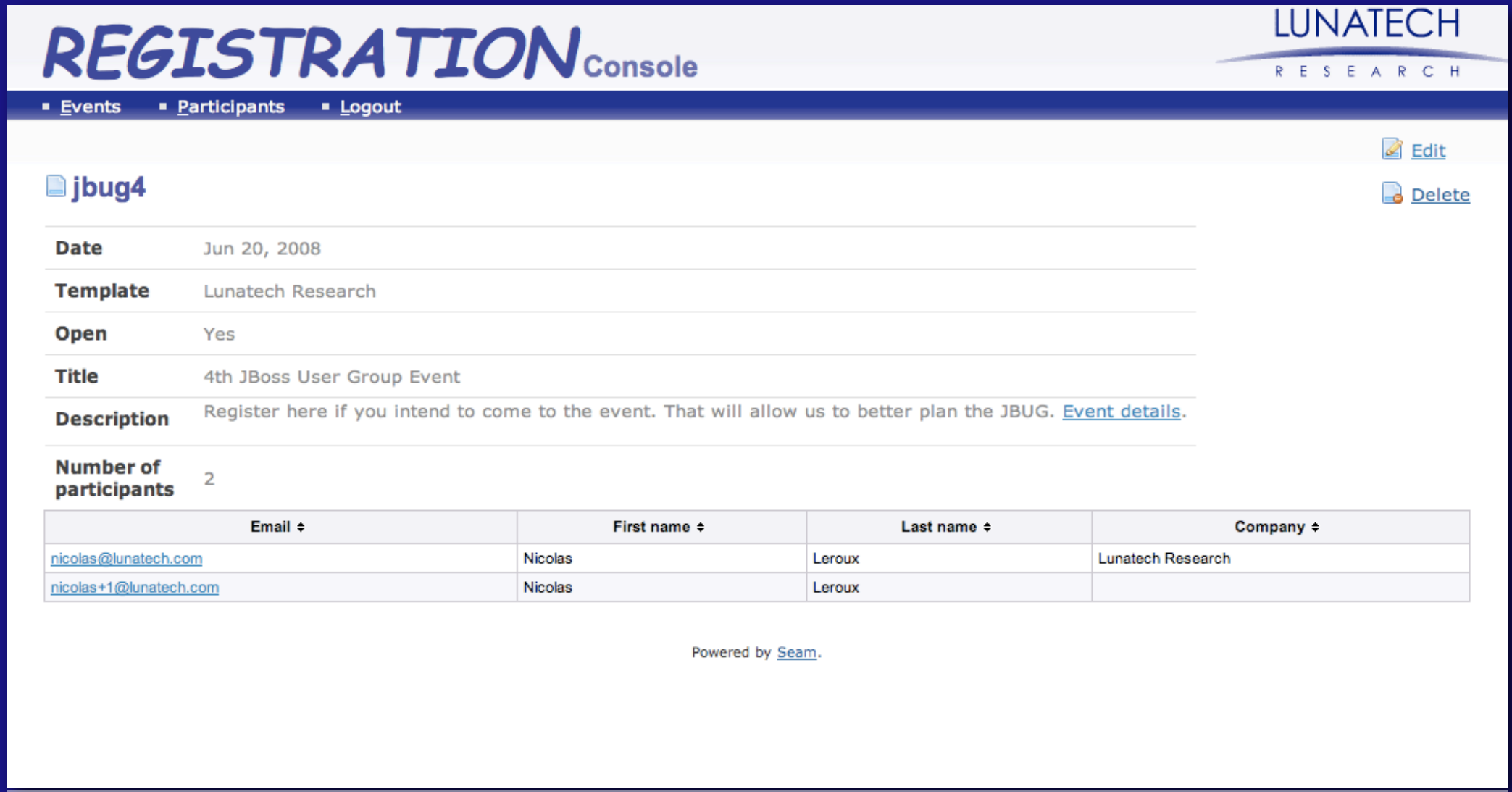
```
<!-- ... -->
<td>
 <s:fragment rendered="#{editedEvenement == evenement}">
  <h:inputText value="#{editedEvenement.titre}"/>
 </s:fragment>
 <s:fragment rendered="#{editedEvenement != evenement}">
  #{evenement.titre}
 </s:fragment>
</td>
<td>
 <s:fragment rendered="#{editedEvenement == evenement}">
  <h:commandButton action="#{evenementAction.save}"/>, or
  <h:commandLink action="#{evenementAction.cancel}"/>
 </s:fragment>
 <s:fragment rendered="#{editedEvenement != evenement}">
  <h:commandLink action="#{evenementAction.edit}"/>, or
  <h:commandLink action="#{evenementAction.remove}"/>
 </s:fragment>
</td>
</tr>
</ui:repeat>
<!-- ... -->
```

Vues: evenements 3

```
    <!-- ... -->

  <s:fragment rendered="#{isAddingEvenement}">
   <tr>
    <td>
     <h:inputText value="#{editedEvenement.date}"/>
    </td>
    <td>
     <h:inputText value="#{editedEvenement.titre}"/>
    </td>
    <td>
     <h:commandButton action="#{evenementAction.save}"/>, or
     <h:commandLink action="#{evenementAction.cancel}"/>
    </td>
   </tr>
  </s:fragment>
 </table>
</h:form>
</ui:define>
</ui:composition>
```

Résultat


REGISTRATION Console
LUNATECH RESEARCH
Events
Participants
Logout
Edit
Delete
jbug4
Date
Jun 20, 2008
Template
Lunatech Research
Open
Yes
Title
4th JBoss User Group Event
Description
Register here if you intend to come to the event. That will allow us to better plan the JBUG. Event details.
Number of participants
2
Email
First name
Last name
Company
nicolas@lunatech.com
Nicolas
Leroux
Lunatech Research
nicolas+1@lunatech.com
Nicolas
Leroux
Powered by Seam.

Plan

Ajout de fonctionalités:

Validation

Validation

- Problème de la validation
  - ~~Dans chaque tiers ?~~
- Hibernate Validation
  - JSR 303: Bean Validation
- Définition dans le modèle
  - Par propriété / pour tout le bean
  - Messages d'erreurs
  - Automatiquement utilisé par les vues
  - Extensible

Hibernate Validation

- @Email
- @NotEmpty
- Custom
  - Définition d'une annotation @Foo
  - Extension de Validator<Foo>

Participant.java

```
@Entity
public class Participant implements Serializable {
 // ...

 @NotEmpty(message = “{participant.invalid.prenom}”)
 private String prenom;

 @NotEmpty
 private String nom;

 @NotEmpty
 private String companie;

 @NotNull
 @EMail
 private String addresseMail;

 // ...
```

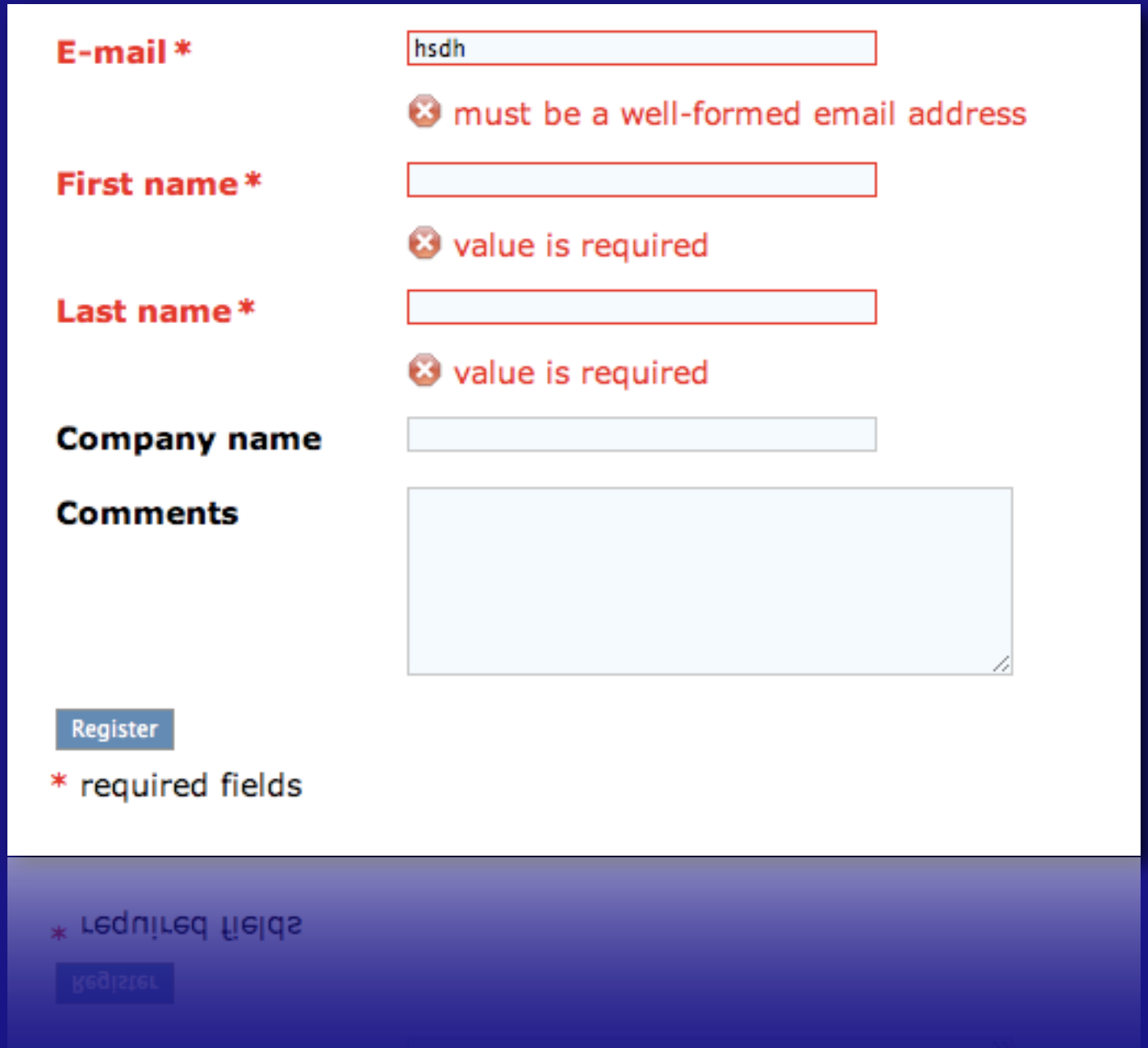
Vues: evenements

```
<!-- ... -->


<ui:define name=”content”>
 <h:form>
  <f:facet name="afterInvalidField">
   <span class="validation error"><s:message /></span>
  </f:facet>
  <table>

  <!-- ... -->
```

Resultat


E-mail *
hsdh
must be a well-formed email address
First name *
value is required
Last name *
value is required
Company name
Comments
Register
* required fields

Plan

Ajout de fonctionalités:

Recherche

Ajout de la recherche

- Tout le monde veut un mini-google
  - Full-text search
- Apache Lucene
  - Indexation de documents
  - Recherche par champs
    - age:2 titre:foo
- Hibernate Search
  - @Indexed sur l'entité
  - @Field sur les champs

Evenement.java

```
@Entity
@Indexed
public class Evenement implements Serializable {
 // ...

 @Field
 private String titre;

 @Field
 private Date date;

 @Field
 private String description;

 @Field(index = Index.UN_TOKENIZED)
 private String identifiant;
```

EvenementDAOBean

```
public class EvenementDAOBean implements EvenementDAO {

  @In
  private FullTextEntityManager entityManager;

  public List<Evenement> findEvenements(String recherche){
    BooleanQuery query = new BooleanQuery();
    for (final String term : searchField.split(" "))
      query.add(new TermQuery(term), BooleanClause.Occur.MUST);
    return (List<Evenement>)
      entityManager.createFullTextQuery(query, Evenement.class)
      .getResultList();
  }

  // ...
}
```

EvenementWebBean

```
@Stateful
@Name("evenementAction")
public class EvenementActionBean implements EvenementAction {

  @RequestParameter
  private String q; // à la google!

  @DataModel
  private List<Evenement> evenementList;

  @Factory("evenementList")
  public void initEvenementList(){
    if (q == null || q.length() == 0)
      evenementList = evenementDAO.findEvenements();
    else
      evenementList = evenementDAO.findEvenements(q);
  }
```

Le template

```
<!-- ... -->


<div id="masthead">
 <h1>Foo Bar</h1>
 <ui:fragment rendered="#{identity.loggedIn}">
  <a href="/evenements/">Evenements</a>
  <a href="/participants/">Participants</a>
  <h:form>
   <h:inputText name=”q”/>
   <h:commandButton action=”#{evenementAction.initEvenementList}”>
    Recherche
   </h:commandButton>
  </h:form>
 </ui:fragment>
</div>

<!-- ... -->
```

LUNATECH
RESEARCH

Plan

Ajout de fonctionalités:
Backups

Backup XML

- Sauver les données en XML
- Les restaurer aussi
- Java API for XML Binding (JAXB: JSR-222)
  - @XmlRootElement sur une entité
  - @XmlElement, @XmlAttribute ou @XmlValue sur une propriété
- Sérialisation / désérialisation automatique
  - Plus de parseurs !

Evenement.java

```
@Entity
@XmlRootElement
public class Evenement implements Serializable {

 @Id @GeneratedValue
 @XmlTransient
 private int id;

 private String titre;
 private Date date;
 private String description;
 private String identifiant;

 @XmlWrapperElement(name = “participants”)
 @XmlElement(name = “participant”)
 private List<Participant> participants =
   new ArrayList<Participant>();
 // ...
```

BackupWebBean

```
@Stateless
@Name("backupAction")
public class BackupWebBean implements BackupWeb {

  @In
  private FacesContext facesContext;

  @In
  private List<Evenement> evenementsList;

  public void backup() {
    HttpServletResponse response = (HttpServletResponse)
      facesContext.getExternalContext().getResponse();
    response.setContentType("text/xml");
    response.setCharacterEncoding("UTF-8");
    OutputStream outputStream = response.getOutputStream();
    JAXBContext jaxb = JAXBContext.newInstance(Evenement.class);
    Marshaller marshaller = jaxb.createMarshaller();
    marshaller.marshall(evenementsList, outputStream);
}
```

Le template

```
<!-- ... -->

<div id="masthead">
 <h1>Foo Bar</h1>
 <ui:fragment rendered="#{identity.loggedIn}">
  <a href="/evenements/">Evenements</a>
  <a href="/participants/">Participants</a>
  <h:form>
   <h:commandLink
     action="#{backupAction.backup}">Backup</h:commandLink>
   <h:inputText name="q"/>
   <h:commandButton action="#{evenementAction.initEvenementList}">
    Recherche
   </h:commandButton>
  </h:form>
 </ui:fragment>
</div>

<!-- ... -->
```

XML

```
<?xml encoding="UTF-8"?>
<evenements>
 <evenement>
  <titre>Présentation à l'IUT</titre>
  <date>15/05/2008</date>
  <description>Présentation de plein (trop?) de technos Java devant des étudiants attentifs (hum!)</
description>
  <identifiant>iut-05-2008</identifiant>
  <participants>
   <participant>
    <prénom>Nicolas</prénom>
    <nom>Leroux</nom>
    <companie>Lunatech Research</companie>
   </participant>
   ...
  </participants>
 </evenement>
 <evenement>
  <titre>JBug Juin 2008 Rotterdam</titre>
  <date>20/06/2008</date>
  <description>JBoss User Group Netherlands Juin 2008</description>
  ...
```

Plan

Ajout de fonctionalités:
RESTful web services

RESTful webservices

- Web services basés sur HTTP
  - Action: GET/PUT/POST/DELETE
  - Objet: URL
  - Résultat: réponse
  - Format du contenu: XML, JSON, ...
  - Pas d'état
- Java API for RESTful web services
  - JAX-RS: JSR-311

Exemples

- Obtenir des événements:
  - GET /rest/evenements
  - GET /rest/evenements?q=IUT
- Obtenir un événement:
  - GET /rest/evenement/iut-05-2008
- Ajouter un événement:
  - POST /rest/evenements

RESTWebBean

```
@Stateless
@Name(“RESTAction”)
@Path("/rest")
@ProduceMime({"application/xml", "text/plain"})
public class RestWebBean implements RestWeb {

  @EJB
  private EvenementDAO evenementDAO;

  @Path("/evenements")
  @GET
  public List<Evenement> get(@QueryParam("q") String recherche){
    if (recherche != null && recherche.length() > 0)
      return evenementDAO.findEvenements(recherche);
    return evenementDAO.findEvenements();
  }

  // ...
```

RESTActionBean 2

```
@Path("/evenement/{id}")
@GET
public Evenement getOne(@PathParam String id){
  if (id == null || !id.matches("^[-_A-Za-z0-9]+$"))
    throw new WebApplicationException("Invalid ID");
  Evenement evt = evenementDAO.findEvenement(id);
  if (evt != null)
    return evt;
  throw new WebApplicationException("Invalid ID");
}

@Path("/evenements")
@POST
public Response post(Evenement newEvenement, @Context UriInfo uriInfo){
  evenementDAO.save(newEvenement);
  URI uri = uriInfo.getBaseUriBuilder().path(getClass(), "getOne")
            .build(newEvenement.getId());
  return Response.created(uri).build();
 }
}
```

Exemples

GET /rest/evenements HTTP/1.1

Host: www.foo.bar

HTTP/1.1 200 OK
Content-Type: application/xml

<?xml encoding=”UTF-8”?>
...

POST /rest/evenements HTTP/1.1
Host: www.foo.bar
Content-Type: application/xml

<?xml encoding=”UTF-8”?>
<evenement>
 <titre>Pause café</titre>
 <date>**tous les jours**</date>
</evenement>

HTTP/1.1 400 Bad Request
Content-Type: text/plain

Invalid XML

Plan

DEMO

Plan

Conclusion

Conclusion

- Beaucoup de buzzwords
- Beaucoup d'annotations
- Moins de code (qu'avant)
- Beaucoup d'innovations
  - Nouveaux JSRs en préparation
- Nombreux autres frameworks
  - Technos similaires

Contactez-nous!

- www.lunatech-research.com
- employment@lunatech.com
- +31 10 750 2600