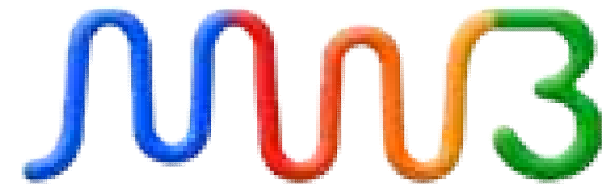


웹기반 AJAX개발을 위한 프레임워크 - 메타웍스3

유엔진 오픈소스 BPM 프로젝트
장진영
jyjang@uengine.org



Spring
Java Application Framework



Concurrent Web-based Applications :



You are the mother, How about your Object ?

Mother cares All the things?



Do that by himself ?



Introducing Metaworks3 –



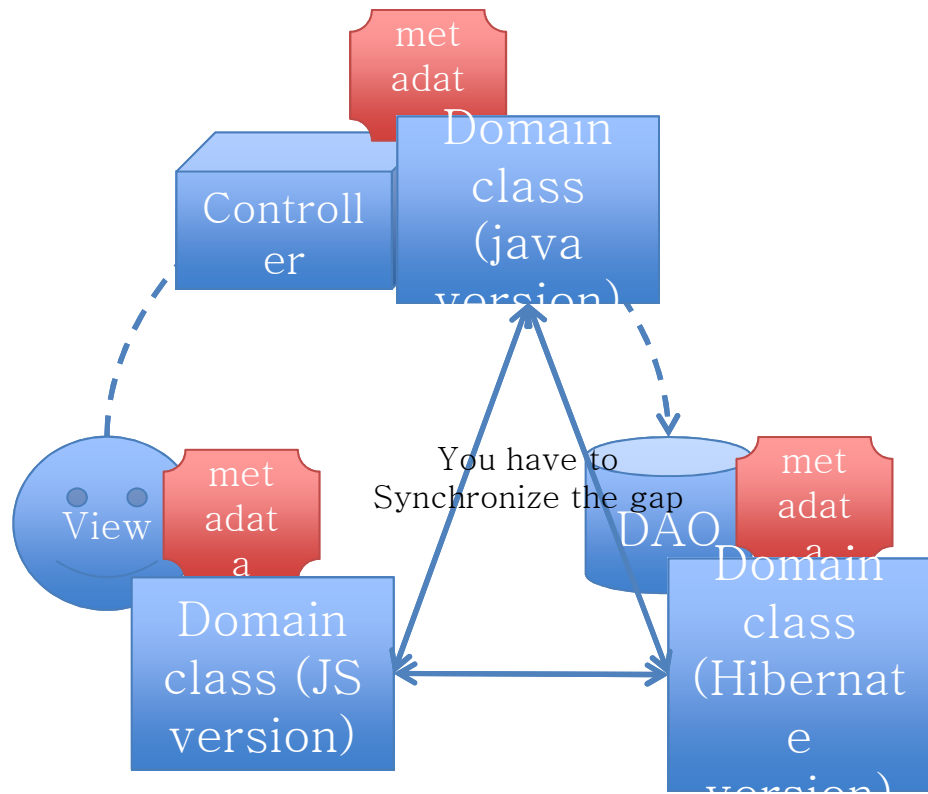
- Metaworks is ‘metadata-oriented framework’
- Concurrent Web-based (and mobile) Applications are degrading the Object’s own property – Cohesion of metadata.
- For example, you need to keep the properties’ type as same value in DHTML scripts, Server Object, Data Object(Hibernate), and the DDL script as well.
- Centralizing metadata and behaviors – it’s Object-Oriented Manner itself!
- Metaworks automatically synchronizes(or hides the operation) ...
 - Stubs for calling Javascript to the server function (using DWR)
 - DAO object and relational query (not completely, SQL is still good)
 - Keeps intuitive and Sophisticated Programming Model... **Don’t be a dog!**
- **By managing the metadata**



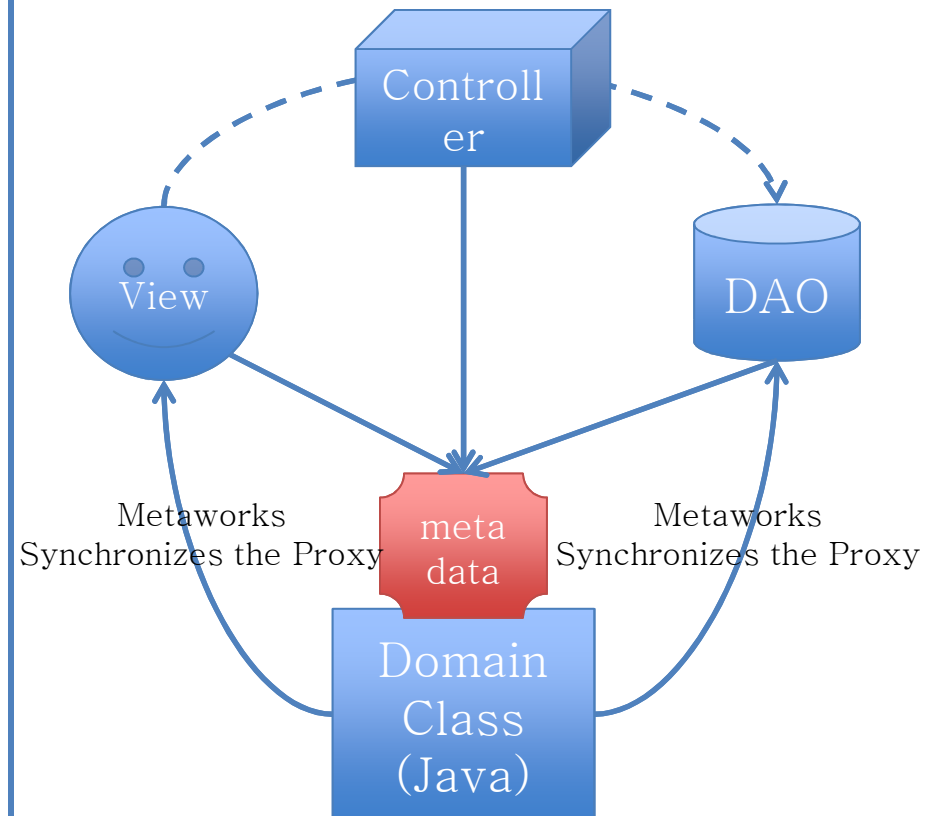
문제:

- 흩어진 메타데이터

General Approach:
Spring MVC, JSON, jQuery, Hibernate



Using Metaworks3:



No more words, Seeing is believing:

#Domain Class (Posting.java)

@Table(name = "fb_posting")

**public class Posting extends MetaworksObject<IPosting>
implements IPosting{**

String document;

@Id

**public String getDocument() {
return document;
}**

**public void setDocument(String document) {
this.document = document;
}**

@ServiceMethod

**public void share() throws Exception{
createDatabaseMe();
}**

#Application (application.html)

<script>

mw3.setContextWhen(mw3.WHEN_EDIT);

mw3.locateObject(
{ document:"",
__className:

"org.metaworks.example.Posting"
},

null,
'newEntry'

);

</script>

...

<div id='newEntry'></div>

#Presentation (Posting.ejs)

<%=fields.document.here() %>

<%

if(mw3.when == mw3.WHEN_EDIT){

%>

<%=methods.share.here()%>

<%

}

%>

#Result

Share your status...

SHARE

"Share your status..."

mysql> select * from fb_posting;

document
Share your status...
Share your status.xxx..
Share your status...

3

FaceBook example

Steps:

1. Recognizing what is (was) 'Object' in your application
 2. Modeling the Objects
 3. Face making and Context-awaring for the Objects
 4. Creating Application with highly-abstracted programming Model – almost natural language!
- Humanity-Recovery !

Step 1.

Recognizing what is (was)
'Object' in your application



1. Recognizing Objects in facebook

The screenshot shows a Facebook group page titled '유엔진 일일 전략 및 회고' (U-engine Daily Strategy and Reflection). The page is viewed in a web browser with multiple tabs open. The left sidebar shows the group's profile and a list of members. The main content area displays a post by '류승호' (Ryu Seung-ho) from 2011-10-06, which includes a list of tasks and a comment by 'Hojun-Harry Sung'. The right sidebar shows a list of documents and a list of members. Various objects are highlighted with colored boxes: blue for person objects (e.g., user avatars, group profile picture), red for posting objects (e.g., post content area, comment input field), green for document objects (e.g., document thumbnails in the right sidebar), and yellow for group objects (e.g., group name in the sidebar, group name in the main header).



Person object



Posting object



Doc object



Group object

1. Recognizing Objects in facebook

1.1. Person objects in different 'Context'

The screenshot shows a Facebook group page for '유엔진 일일 전략 및 회고' (U-engine Daily Strategy and Reflection). The page is viewed in a web browser with multiple tabs open. The left sidebar shows the group's profile picture and name, along with a list of members. The main content area displays a post by '류승호' (Ryu Seung-ho) with a blue box around the profile picture, and a post by 'Hojun-Harry Sung' with a blue box around the profile picture. The right sidebar shows a list of members and a section for '문서(21)' (Documents) with a list of files. The browser address bar shows the URL 'www.facebook.com/groups/157739020951873/?notif_t=group_activity'.



Where: NORMAL



Where: MINIMISED



Where: MEDIUM

1. Recognizing Objects in facebook

1.2. Posting objects in different 'Context'

The screenshot shows a Facebook group interface. The main content area displays a post by '류승호' (Ryu Seung-ho) from 2011-10-06. The post text is highlighted with a blue box. Below the text, there are two more blue boxes: one for the user's profile picture and name, and another for the post's timestamp and location. The right sidebar shows a list of members and documents.



Where: NORMAL
When: EDIT



Where: NORMAL
When: VIEW



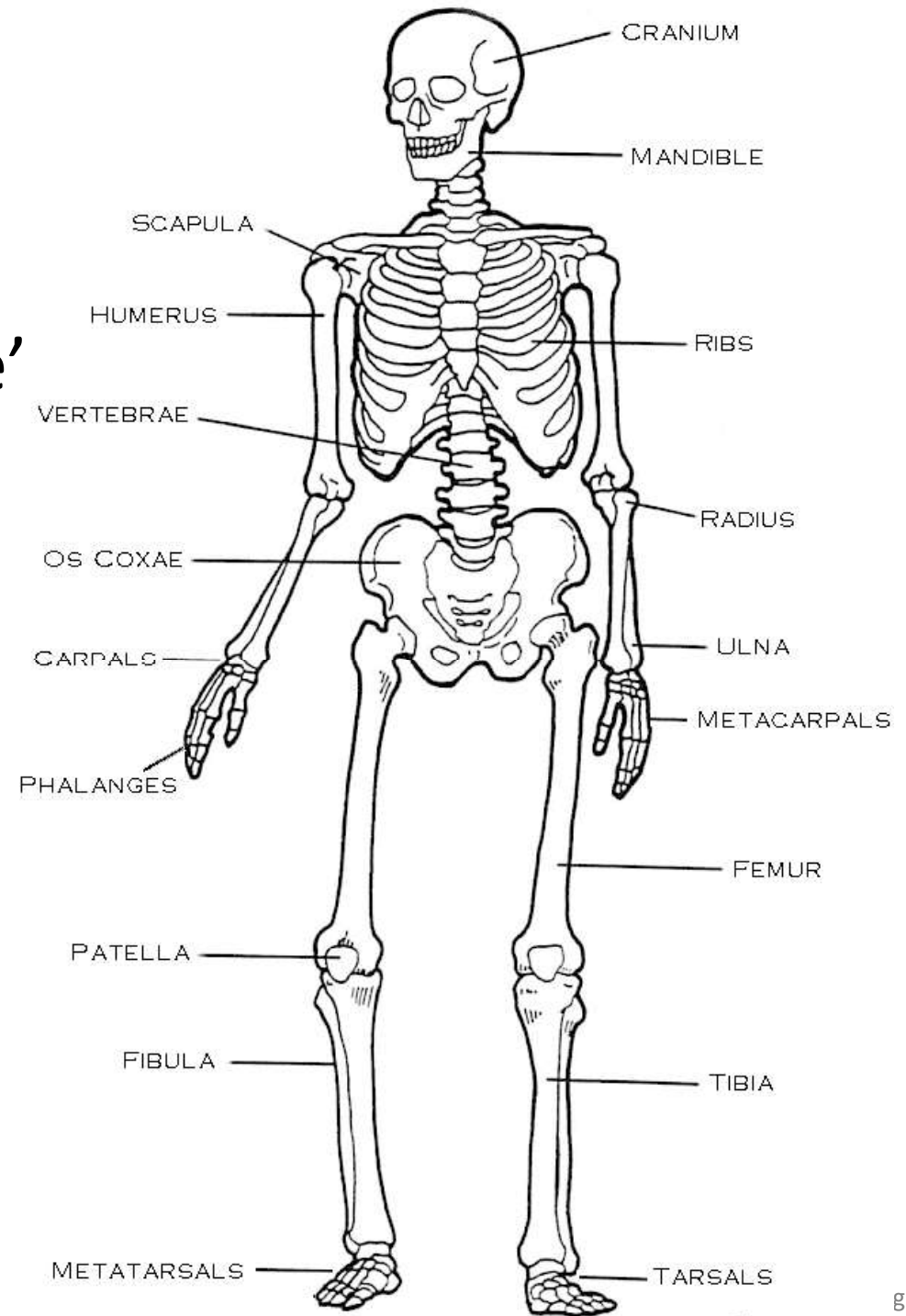
Where: MINIMISED
When: EDIT



www.metaworks3.org

Step 2.

Modeling the 'Cohesive' Objects



2. Modeling Objects - Posting object Design –

2.1. create interface

```
import org.metaworks.dao.IDAO;

public interface IPosting extends IDAO{

    public Person getWriter(); // 작성자
    public void setWriter(Person writer);

    public String getDocument(); // 글
    public void setDocument(String document);

    public boolean isLikelt(); // 좋아요 여부
    public void setLikelt(boolean likelt);

}
```

2. Modeling Objects - Posting object Design –

2.2. add metadata

```
import org.metaworks.dao.IDAO;

@Table(name="Posting")
public interface IPosting extends IDAO{

    public Person getWriter();
    public void setWriter(Person writer);

    @Id
    public String getDocument();
    public void setDocument(String document);

    public boolean isLikelt();
    public void setLikelt(boolean likelt);

    @ServiceMethod
    public void post() throws Exception;

    @ServiceMethod
    public void like() throws Exception;

}
```

2. Modeling Objects – Posting object

2.3. create implementation object

```
public class Posting extends Database<IPosting> implements IPosting{
```

```
    Person writer;
```

```
        public Person getWriter() {  
            return writer;  
        }  
        public void setWriter(Person writer) {  
            this.writer = writer;  
        }  
    }
```

```
    String document;
```

```
        public String getDocument() {  
            return document;  
        }  
        public void setDocument(String document) {  
            this.document = document;  
        }  
    }
```

```
    boolean likelt;
```

```
        public boolean isLikelt() {  
            return likelt;  
        }  
        public void setLikelt(boolean likelt) {  
            this.likelt = likelt;  
        }  
    }
```

```
}
```

2. Modeling Objects - Posting object

2.4. add behaviors

public class Posting extends Database<IPosting> implements IPosting{

Person writer;

```
    public Person getWriter() {  
        return writer;  
    }  
    public void setWriter(Person writer) {  
        this.writer = writer;  
    }
```

String document;

```
    public String getDocument() {  
        return document;  
    }  
    public void setDocument(String document) {  
        this.document = document;  
    }
```

boolean likelt;

```
    public boolean isLikelt() {  
        return likelt;  
    }  
    public void setLikelt(boolean likelt) {  
        this.likelt = likelt;  
    }
```

```
    public void post() throws Exception{  
        createDatabaseMe();  
    }
```

```
    public void like() throws Exception{  
        databaseMe().setLikelt(true); //will automatically synchronize the value  
    }
```

```
}
```



Step 3

Face making and Context-awaring for
the Objects



3. Face Making

3.1. creating 'Face' – Posting.ejs

```
<table>

  <tr>
    <td><%=fields.writer.here() %></td>
    <td><%=fields.document.here() %>

    <%
      if(value.liked){
    %>
    You like this
    <% }else{%>

    <a onclick=<%=methods.like.caller()%>>좋아요</a>
    <}%}%>

    </td>

  </tr>

</table>
```

3. Face Making

3.1. considering the Contexts – WHEN_EDIT

```
<%
    if(mw3.when == mw3.WHEN_EDIT){
        value.document = 'Share your status...';
    }
%>
<table>
    <tr>
        <td><%=fields.writer.here()%></td>
        <td><%=fields.document.here()%>
    <%
        if(mw3.when == mw3.WHEN_EDIT){
    %>
        <%=methods.post.here()%>
    <%
        }

        if(value.likelt){
    %>
        You like this
    <% }else{%>
        <%=methods.like.here()%>
    <%}%>

    <%
        if(mw3.when == mw3.WHEN_EDIT){
    %>
        <input type=button value="save">
    <%
        }else{
    %>
        <input type=button onclick="<%=editFunction%>" value="edit"a>
    <%
        }
    %>
        </td>
    </tr>
</table>
```

Next, Do 1~3 for Person object – Person.java & Person.ejs

#Person.java

public interface Person extends IDAO{

@org.metaworks.Metadata(isKey = true)

public String getName();

public void setName(String name);

public int getAge();

public void setAge(int age);

@org.metaworks.Metadata(face="faces/image.ejs")

public String getPortraitURL();

public void setPortraitURL(String portraitURL);

public boolean isMyFried();

public void setMyFried(boolean isMyFried);

}

#Person.ejs

<%if (!value){%>

Writer is not specified

<%}else{%>

**
**

<%=value.name%>

<%}%>

Step 4

Creating Application with
highly-abstracted
programming Model –

Write a Novel or Poet, not
the alien language



4. Now, create the Application

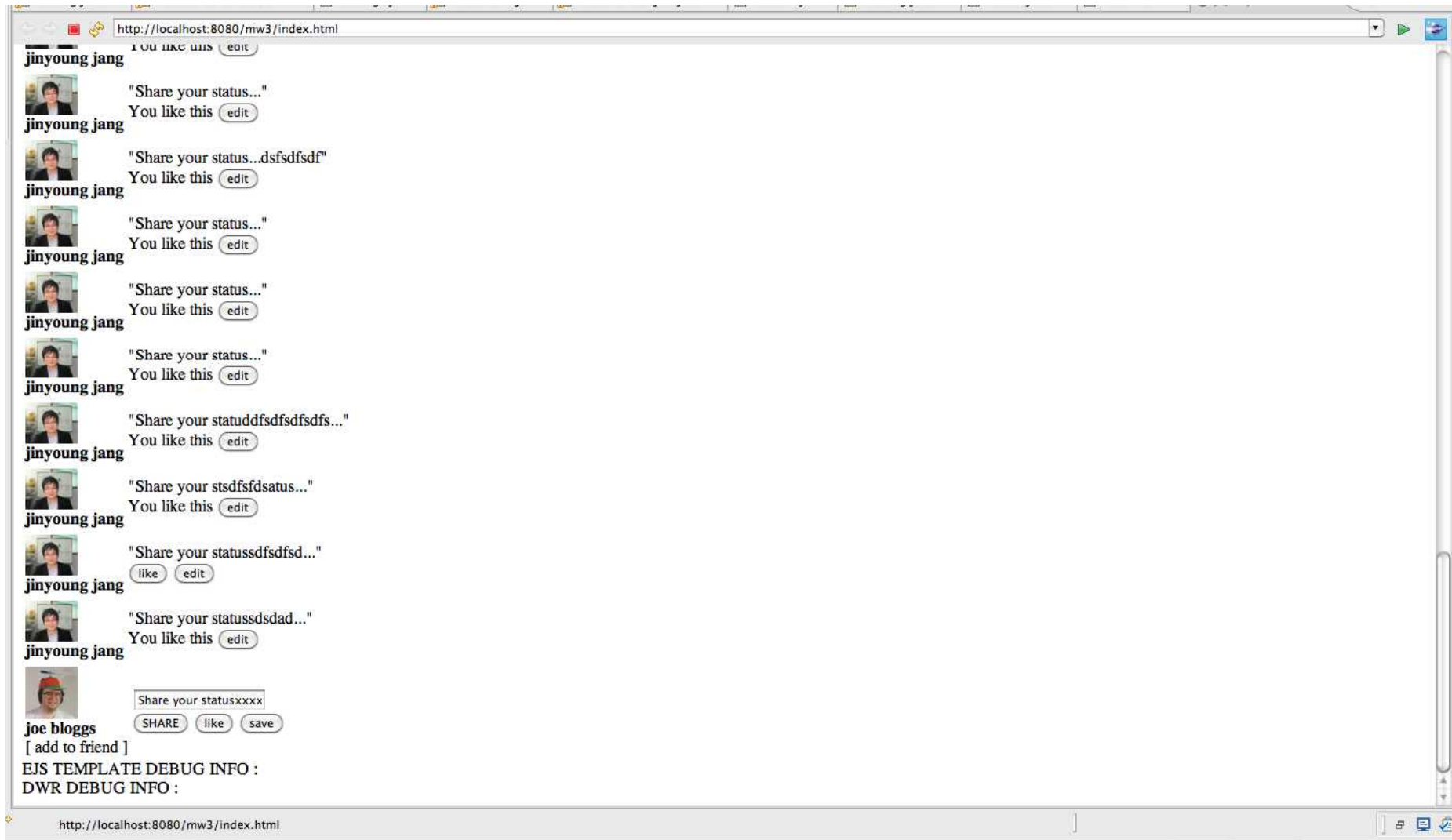
```
<script type="text/javascript">

    lastEntryId = new MetaworksObject(
        {
            document:"",
            writer: loginUser,
            __className: "org.metaworks.example.Posting"
        },
        'newEntry'
    );

</script>

<div id="newEntry"></div>
```

Result

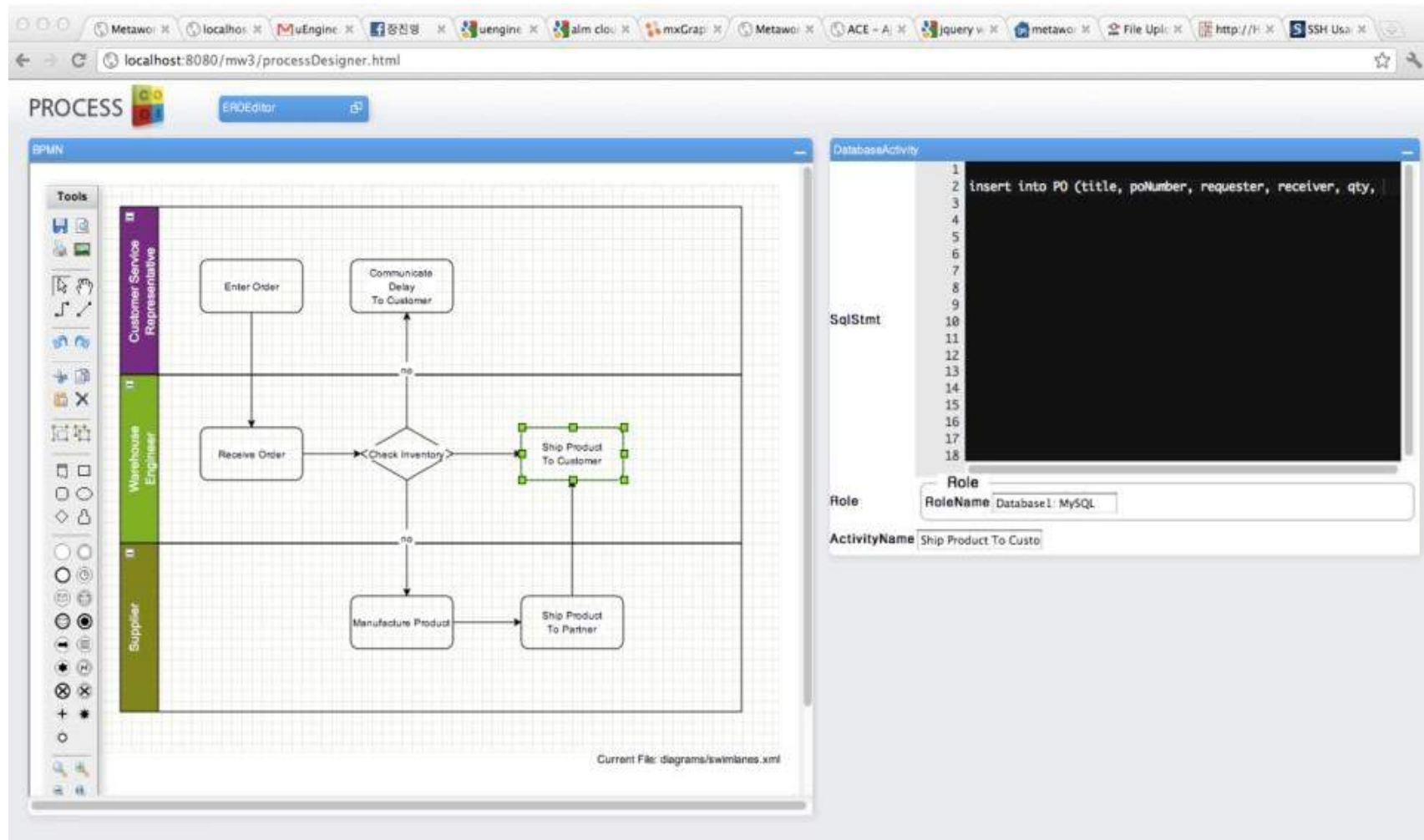


Applications



www.metaworks3.org website Social CMS

Applications



ProcessCodi Cloud IDE

Applications

The screenshot displays the ProcessCodi Enterprise 2.0 Platform interface. The top navigation bar includes the 'PROCESS' logo, a search bar with filters for '이메일연동642', '[이슈]상기 메일을', and '다시 대화를 한번', and a user profile for 'Jinyoung Jang' with options for '도움말' and '로그아웃'.

The left sidebar contains a '네비게이션' (Navigation) menu with sections for '개인중심' (Personal Center), '전략중심' (Strategy Center), '조직중심' (Organization Center), and '기능중심' (Function Center). Below this is a '연락처' (Contacts) section for 'Jinyoung Jang' with a 'Search User' field and a list of contacts including 'Bosang Kim', 'Seunghyeon Kim', 'Sanghyo Song', and 'Suennho Ryu'.

The main content area is divided into two panels. The left panel, titled '개인중심 - 전체 검색' (Personal Center - All Search), shows a table of search results with columns for '참여자' (Participant), '상태' (Status), '제목' (Title), '시작일' (Start Date), and '중요' (Important). The table lists several items, including a search issue, a completed task '[1월성과]재무_하나_90', and a task 'gksrmf'.

The right panel, titled '이메일연동642', shows a detailed view of a workflow. It includes a header with '대화이력' (Conversation History), '프로세스' (Process), '스케줄' (Schedule), '답변' (Reply), and '삭제' (Delete). The main content area displays a 'Human work(Initiator : Jinyoung Jang)' task with a '설계변경' (Design Change) section and buttons for '완료처리' (Complete Processing) and '저장' (Save). Below this, a completed task is shown: 'Human work(Initiator : Jinyoung Jang) - 2011년 10월 07일 (COMPLETED)'.

ProcessCodi Enterprise 2.0 Platform

Conclusion

- No duplicated metadata mapping
e.g. JSON object mapping in Spring Controller and DAO mapping for Hibernate.
- No errors from duplicated metadata source.
- Cohesive User Interface management by EJS-object mapping.
- Intuitive Programming Model – model and metadata-driven, not implementation-driven.

Only thing you have to do is... just forgetting the old manners!

All the Source Code available here:

www.metaworks3.org

it's very early stage now! You are welcomed to participate this Open Source Project.