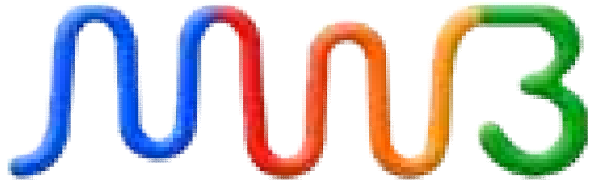
웹기반 AJAX개발을 위한 프레임워크 – 메타웍스3



유엔진 오픈소스 BPM 프로젝트
장진영
jyjang@uengine.org

www.metaworks3.org

Concurrent Web-based Applications :

You are the mother, How about your Object ?

Mother cares All the things?

Do that by himself ?

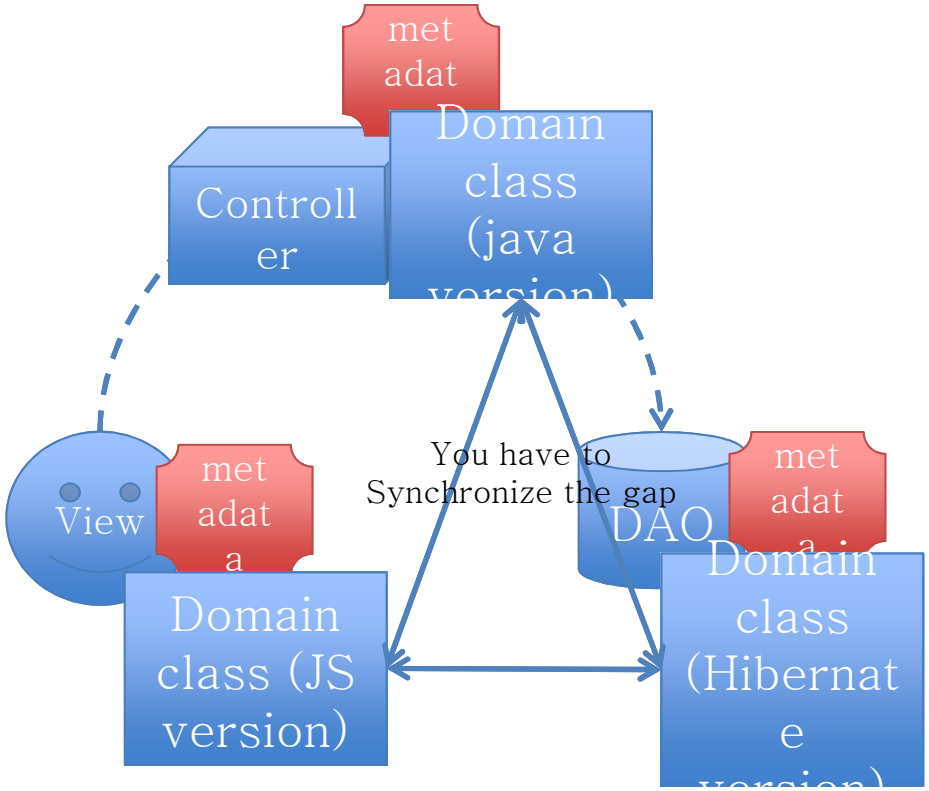
Introducing Metaworks3 –

- 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

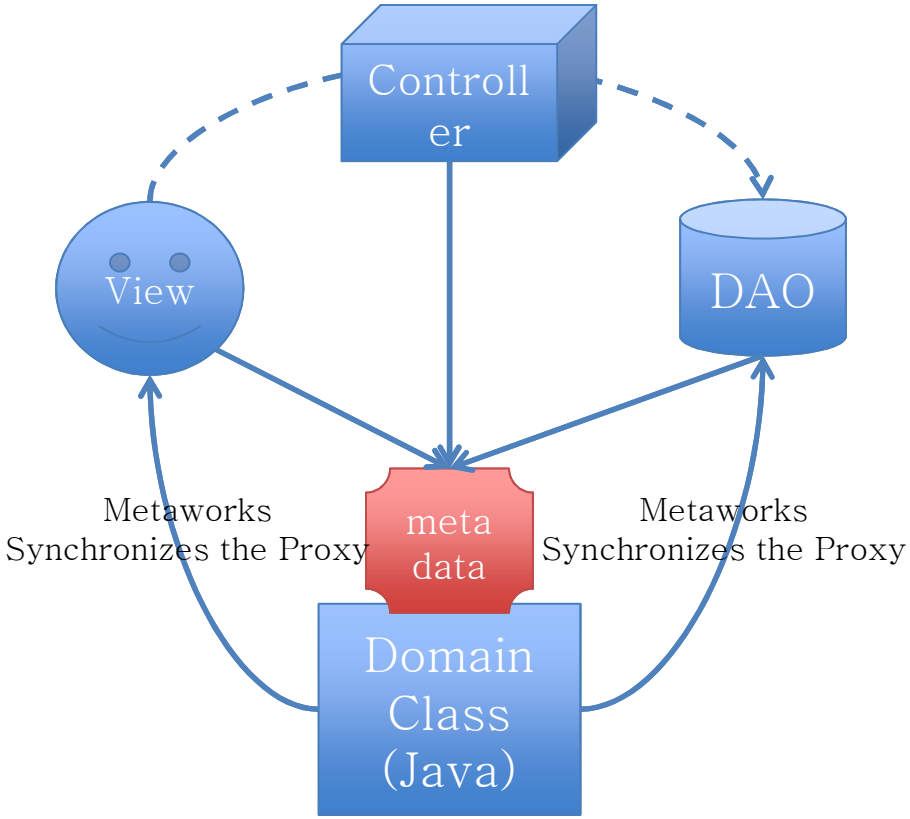
문제:

- 흩어진 메타데이터

General Approach:
Spring MVC, JSON, jQuery, Hibernate



Using Metaworks3:



www.metaworks3.org

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();
        }
}
```

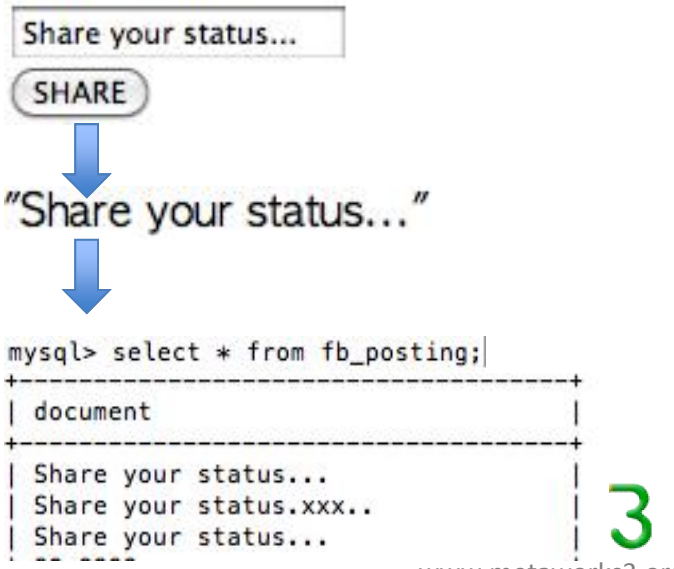
#Application (application.html)

```
<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



FaceBook example

Steps:

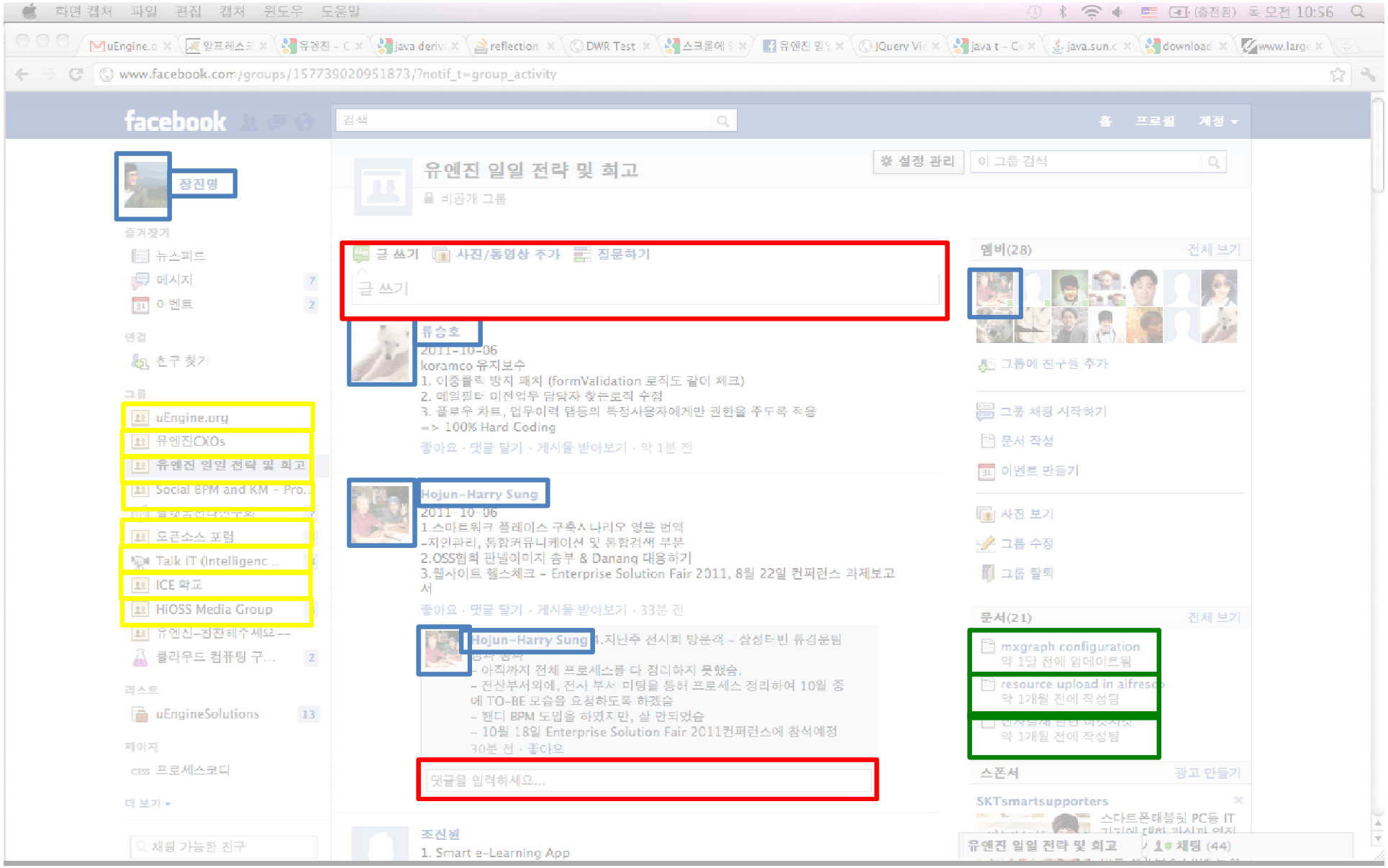
1. Recognizing what is (was) ‘Object’ in your application
2. Modeling the Objects
3. Face making and Context-awaring for the Objects
4. Creating Application with highly-abstracted programming Model – almost natural language!

→ Humanity-Recovery !

Step 1.

Recognizing what is (was)
‘Object’ in your application

1. Recognizing Objects in facebook



Person object Posting object Doc object Group object

1. Recognizing Objects in facebook

1.1. Person objects in different ‘Context’

Where: NORMAL

Where: MINIMISED

Where: MEDIUM

www.metaworks3.org

1. Recognizing Objects in facebook

1.2. Posting objects in different ‘Context’

Where: NORMAL
When: EDIT

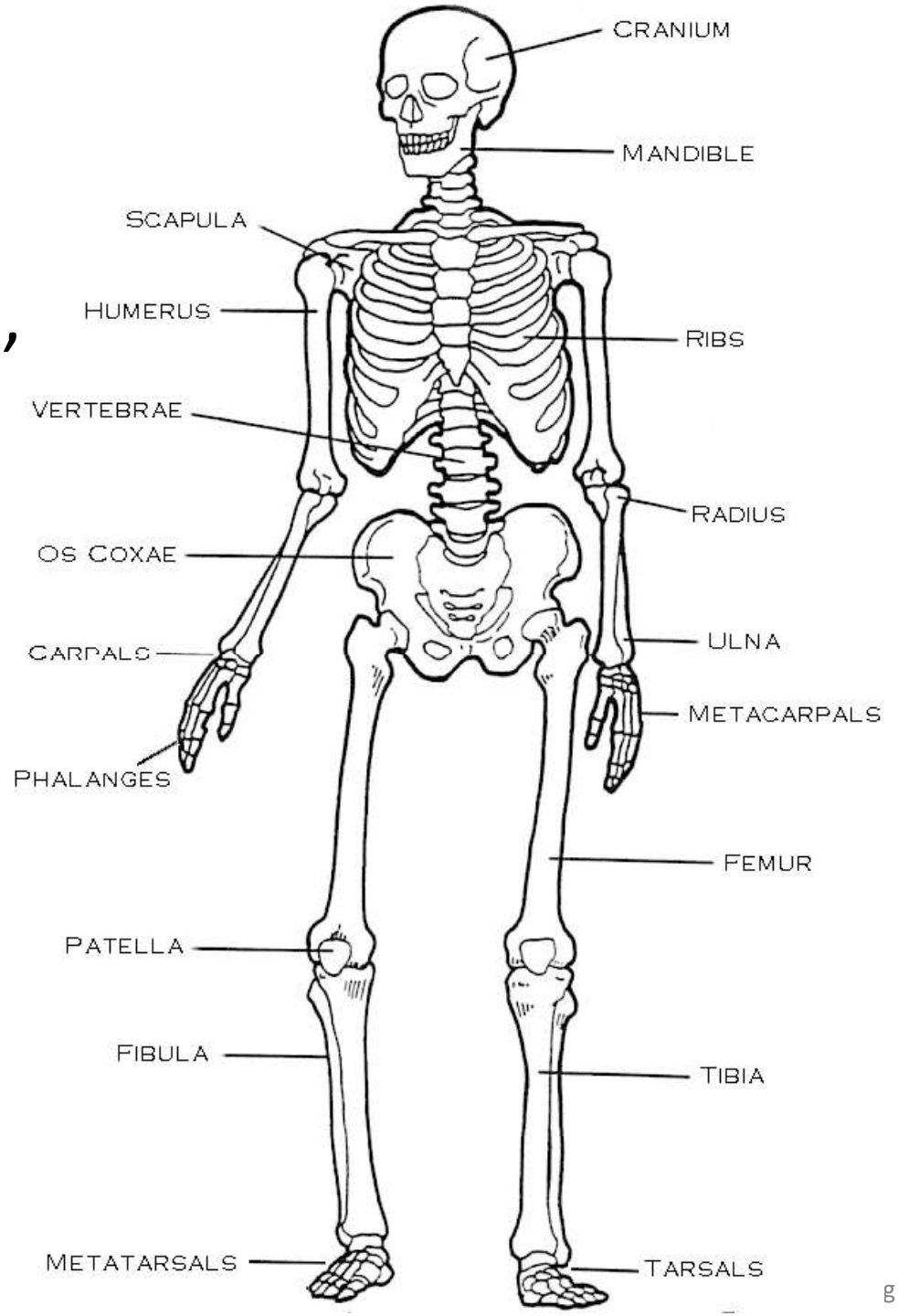
Where: NORMAL
When: VIEW

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 isLikeIt();   // 좋아요 여부
        public void setLikeIt(boolean likeIt);

}
```

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 isLikeIt();
        public void setLikeIt(boolean likeIt);

        @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 likeIt;
                public boolean isLikeIt() {
                        return likeIt;
                }
                public void setLikeIt(boolean likeIt) {
                        this.likeIt = likeIt;
                }
}
```

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 likeIt;
                public boolean isLikeIt() {
                        return likeIt;
                }
                public void setLikeIt(boolean likeIt) {
                        this.likeIt = likeIt;
                }

        public void post() throws Exception{
                createDatabaseMe();
        }
        public void like() throws Exception{
                databaseMe().setLikeIt(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.likeIt){
%>
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.likeIt){
%>
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{%>

<img src="<%=value.portraitURL%>" width=50><br>
<b><%=value.name%></b>

<%}%>
```

Step 4

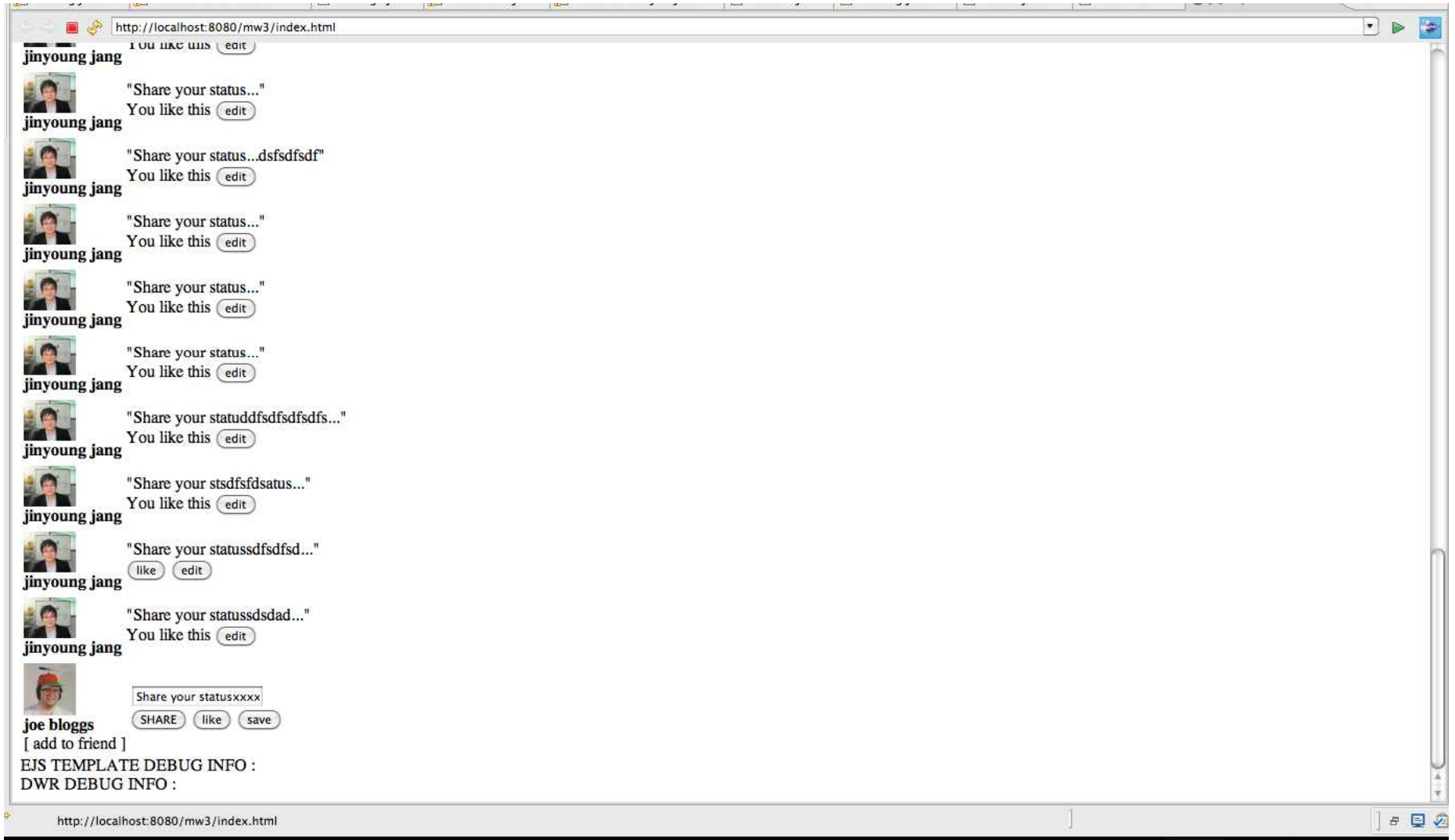
Creating Application with highly-abstracted programming Model –

Write a Novel or Poet, not the alien language !

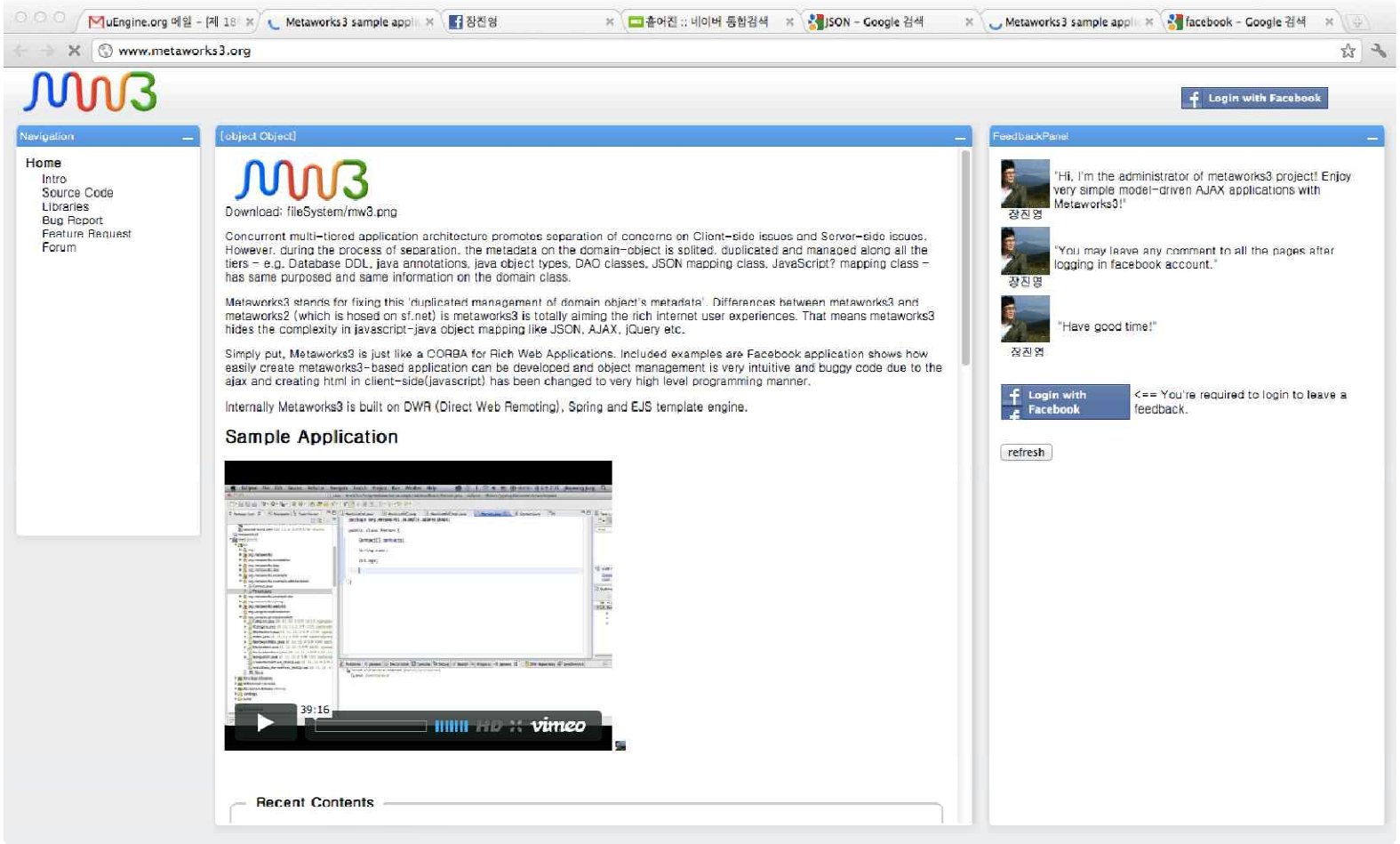
4. Now, create the Application

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


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

                'newEntry'

        );

</script>

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

Result

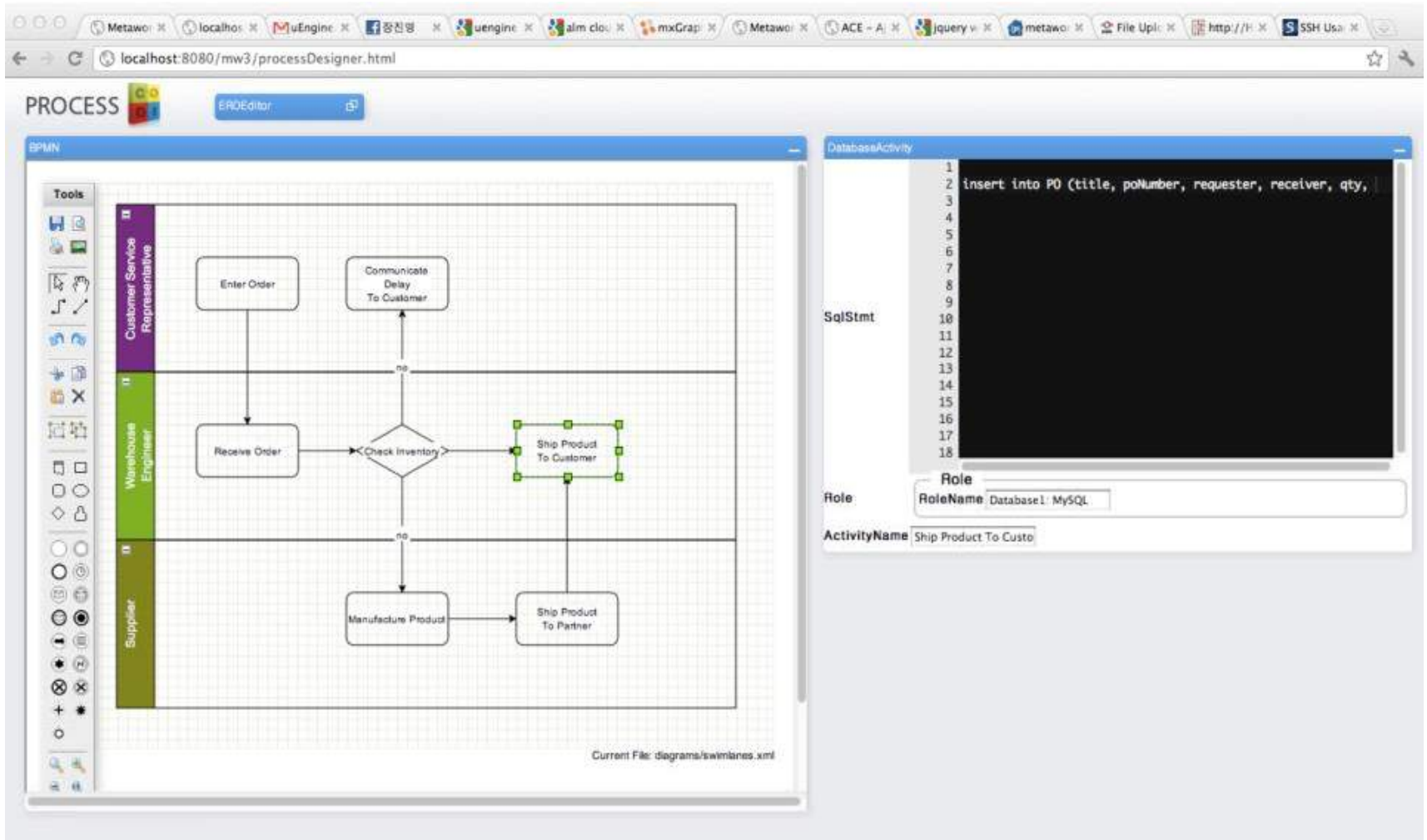

http://localhost:8080/mw3/index.html
jinyoung jang
"Share your status..."
You like this edit
jinyoung jang
"Share your status...dsfsdfsdf"
You like this edit
jinyoung jang
"Share your status..."
You like this edit
jinyoung jang
"Share your status..."
You like this edit
jinyoung jang
"Share your status..."
You like this edit
jinyoung jang
"Share your statuddfsdfsdfsdfs..."
You like this edit
jinyoung jang
"Share your stsdfsfdsatus..."
You like this edit
jinyoung jang
"Share your statussdfsdfsd..."
like edit
jinyoung jang
"Share your statussdsdad..."
You like this edit
jinyoung jang
Share your statusxxxx
SHARE like save
joe bloggs
[add to friend]
EJS TEMPLATE DEBUG INFO :
DWR DEBUG INFO :
http://localhost:8080/mw3/index.html

Applications



www.metaworks3.org website Social CMS

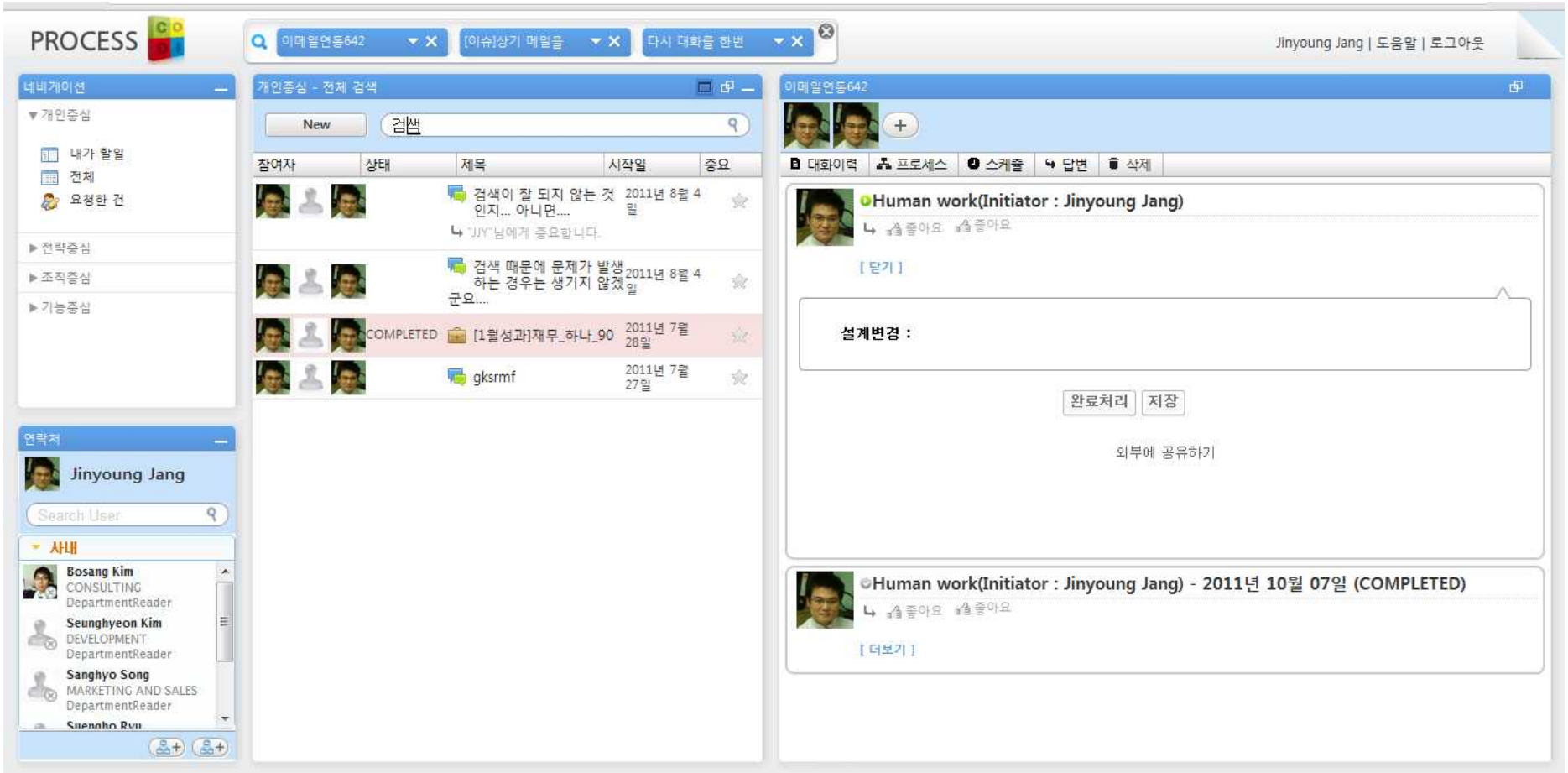
Applications



ProcessCodi Cloud IDE

www.metaworks3.org

Applications



ProcessCodi Enterprise 2.0 Platform

www.metaworks3.org

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.