



Ch'ti JUG



Bienvenue chez le Ch'ti JUG !





Ch'ti JUG

Programme

- Bienvenue chez le Ch'ti JUG
- Le mot du sponsor
- Les JUGs en France
- Java EE 6, qu'est-ce qui nous attend ?
- L'interro
- Le buffet



Ch'ti JUG

Le sponsor de cette session

The logo for ProxiAD, featuring the word "ProxiAD" in a bold, blue, sans-serif font. The letters are slightly italicized. The text is enclosed within a blue, curved, double-lined oval shape that frames the text from above and below.



Les JUGs en France



<%LorraineJUG%>



Ch'ti JUG



Les nouveautés de Java EE 6

Antonio Goncalves





Ch'ti JUG

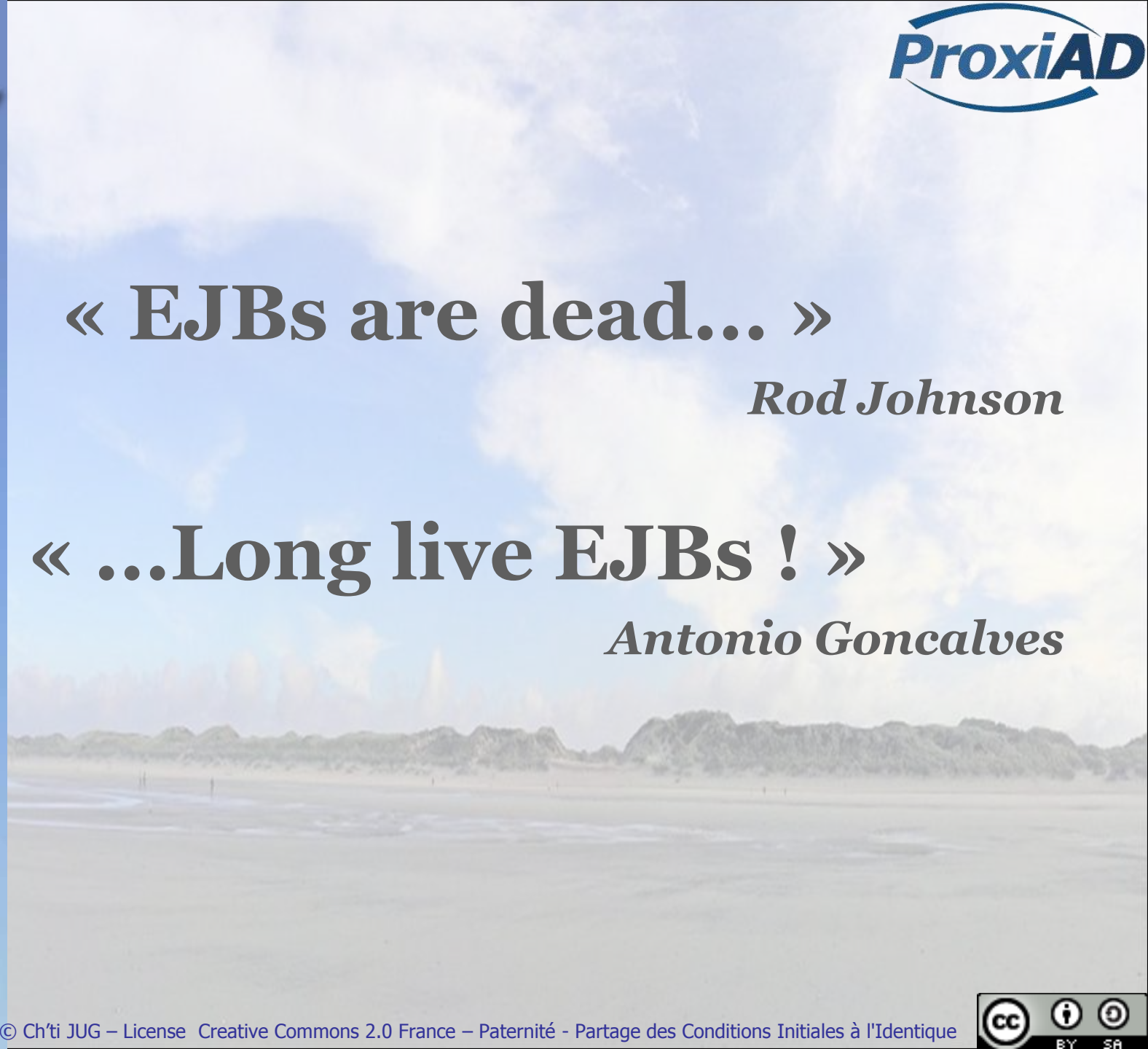


« EJBs are dead... »

Rod Johnson

« ...Long live EJBs ! »

Antonio Goncalves





Ch'ti JUG

Antonio Goncalves



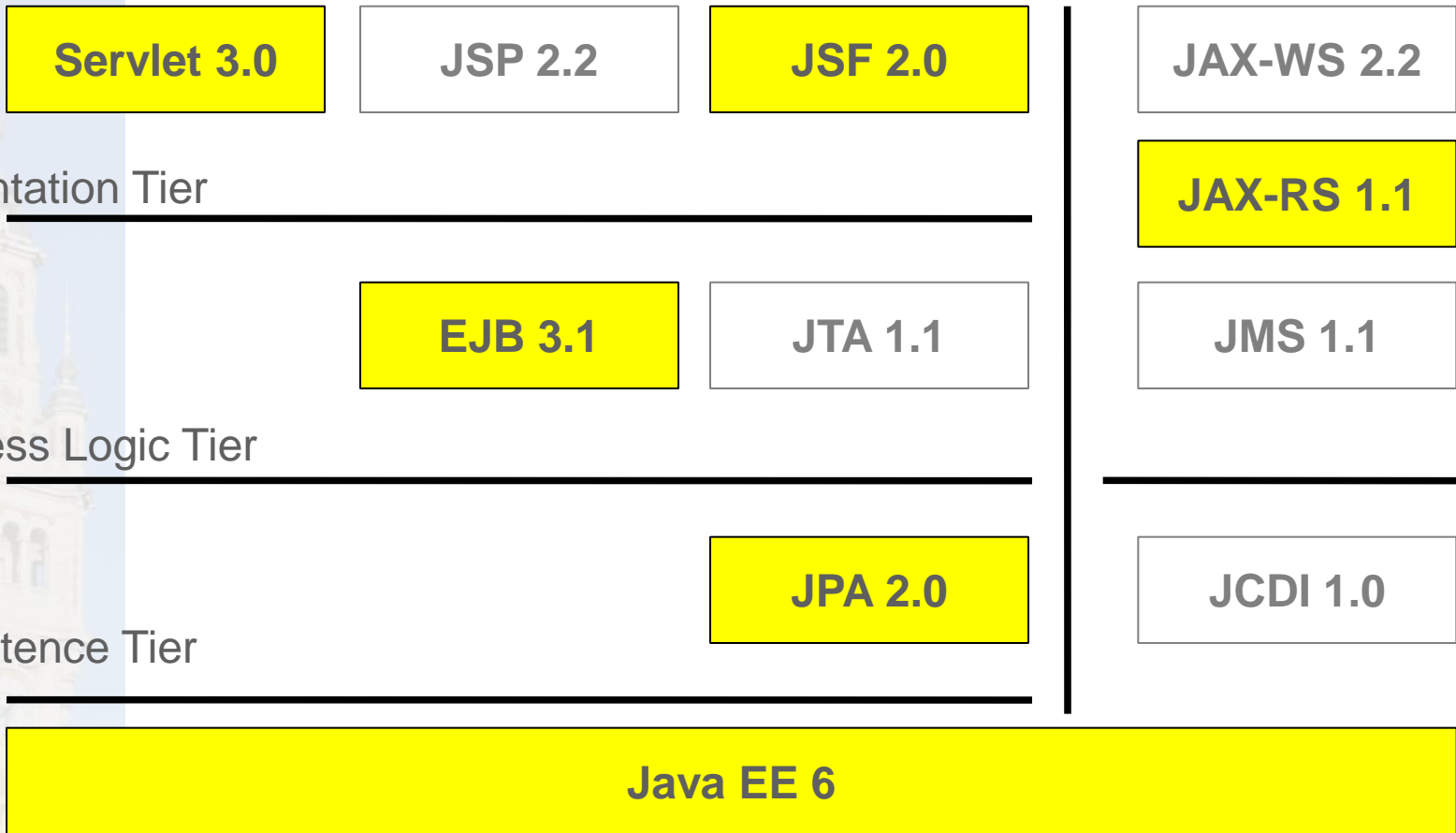
- Software Architect
- Former BEA Consultant
 - Experience with Application Servers
- Java EE 5 author (in French)
- Java EE 6 author (in English)
- JCP Expert Member
 - Java EE 6, EJB 3.1, JPA 2.0
- Co-creator, co-leader of the Paris JUG





Ch'ti JUG

Agenda





Ch'ti JUG

Java EE 6



Servlet 3.0

JSP 2.2

JSF 2.0

Presentation Tier

EJB 3.1

JTA 1.1

Business Logic Tier

JPA 2.0

Persistence Tier

Interoperability Tier

JAX-WS 2.2

JAX-RS 1.1

JMS 1.1

JCDI 1.0

Java EE 6





Ch'ti JUG

A Bit of history



Ease of development (web)

Ease of development

Web Services

Enterprise Application

Robust Scalable
J2EE 1.3

J2EE 1.4

Java EE 5

Java EE 6

Project JPE

J2EE 1.2

Servlet
JSP
EJB
JMS
RMI/IIOP

CMP
JCA

WS
Management
Deployment

Annotations
EJB 3
JPA 1.0
WS-*
JSF

EJB 3.1
JPA 2.0
Servlet 3.0
JSF 2.0
JAX-RS 1.1
JCDI 1.0
Bean Validat°

Web Profile

May 1998

Dec 1999
10 specs

Sept 2001
13 specs

Nov 2003
20 specs

May 2006
23 specs

Q3 2009
~28 specs





Java EE 6 is Richer, Easier, Lighter



- Richer
 - New specifications
- Easier
 - POJO model
 - Less XML...
 - ... even on the web tier
- Lighter
 - EJB Lite
 - Profiles and Pruning





Richer :

~28 specifications



Web Services

JAX-RPC	1.1
JAX-WS	2.2
JAXM	1.0
JAX-RS	1.1
JAXR	1.1
StAX	1.0
Web Services	1.2
Web Services Metadata	1.1

Enterprise

EJB	3.1
JAF	1.1
JavaMail	1.4
JCA	1.6
JMS	1.1
JPA	2.0
JTA	1.1

Web

JSF	2.0
JSP	2.2
JSTL	1.2
Servlet	3.0
Expression Language	1.2

Management, Security and other

JCDI	1.0
JACC	1.1
Common Annotations	1.0
Java EE Application Deployment	1.2
Java EE Management	1.1
Java Authentication Service Provider Interface for Containers	1.0
Debugging Support for Other Languages	1.0
Bean Validation	1.0

+ Java SE 6

JAXB	2.2
JDBC	4.0
JNDI	1.5
RMI	
JMX	
JAAS	
JAXP...	

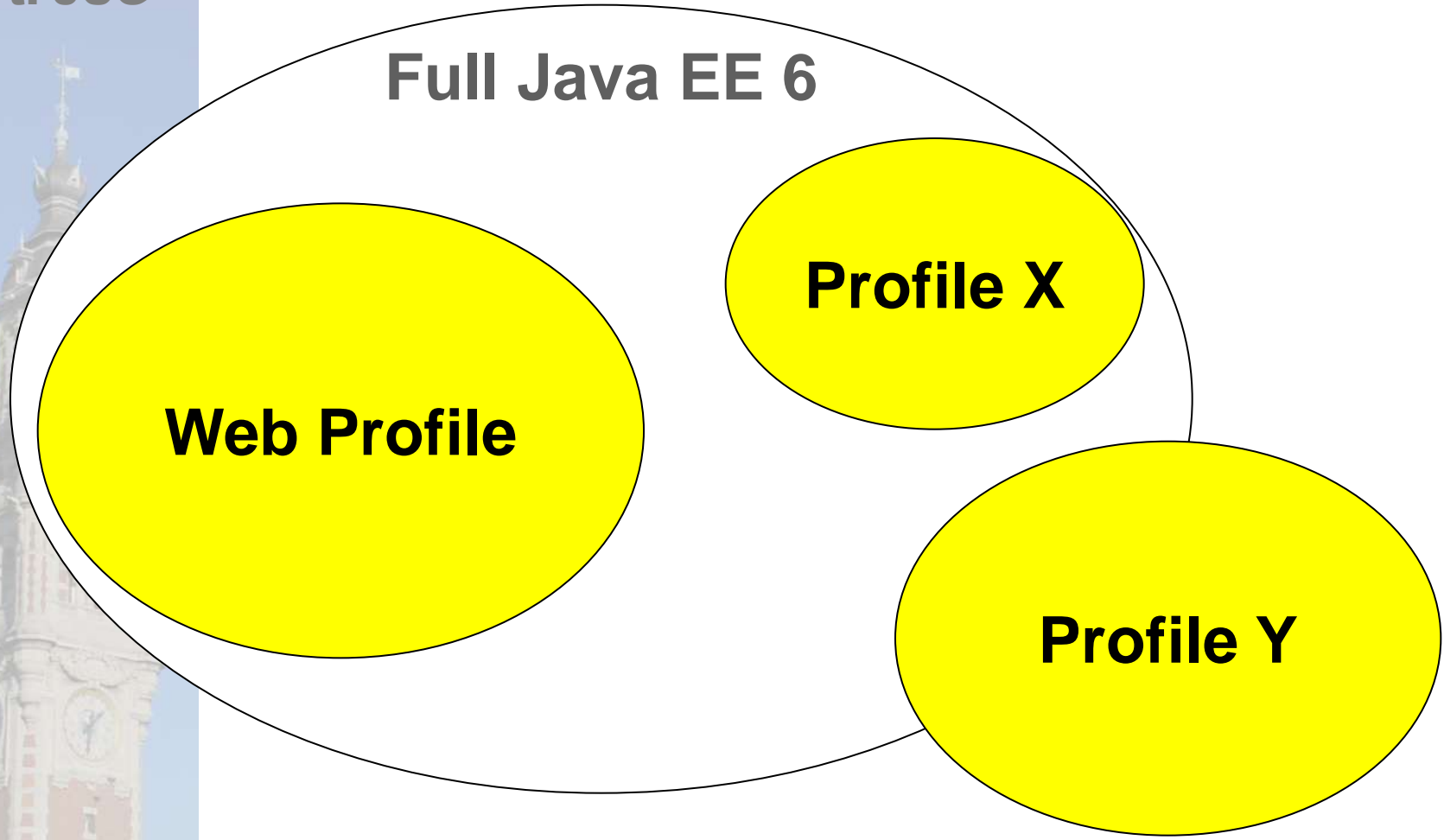




Ch'ti JUG



Lighter : Profiles





Ch'ti JUG



Lighter : Web Profile

- Subset of full platform
 - Focuses on web development
 - Separate specification
 - Others will come
 - Minimal (Servlet/JSP)
 - Portal....
- | | |
|----------|-----|
| Servlet | 3.0 |
| JSP | 2.2 |
| EL | 1.2 |
| JSTL | 1.2 |
| EJB Lite | 3.1 |
| JTA | 1.1 |
| JPA | 2.0 |
| JSF | 2.0 |

« ...you'll see gradual move toward the Web profile »
- *Rod Johnson*





Ch'ti JUG

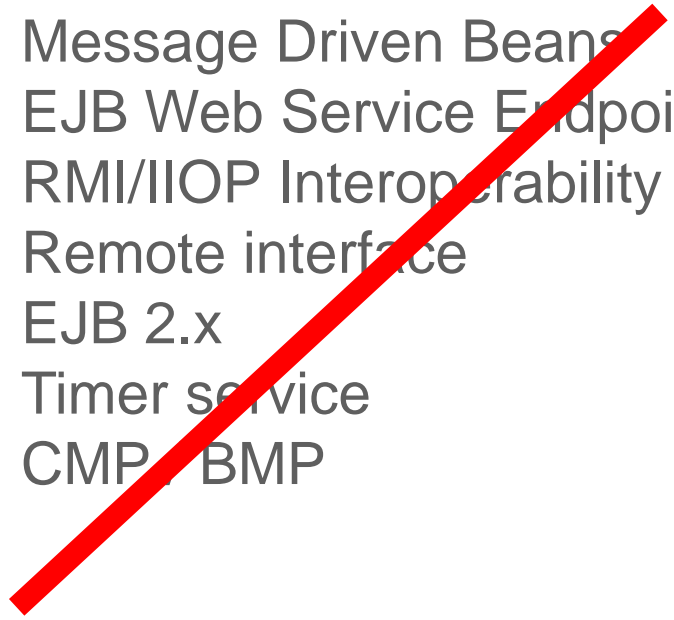
Lighter : EJB Light



- Subset of the EJB 3.1 API
- To be used in Web profile

Local Session Bean
Injection
CMT / BMT
Interceptors
Security

Message Driven Beans
EJB Web Service Endpoint
RMI/IIOP Interoperability
Remote interface
EJB 2.x
Timer service
CMP / BMP





Lighter : Pruning (Soon less specs)



- Makes some specifications optional in next version
- Pruned in Java EE 6
 - Entity CMP 2.x
 - JAX-RPC
 - JAX-R
 - JSR 88 (Java EE Application Deployment)
- Stronger than @Deprecated
- Might disappear from Java EE 7
 - Evolve (or not) separately from Java EE
- Easier for future containers



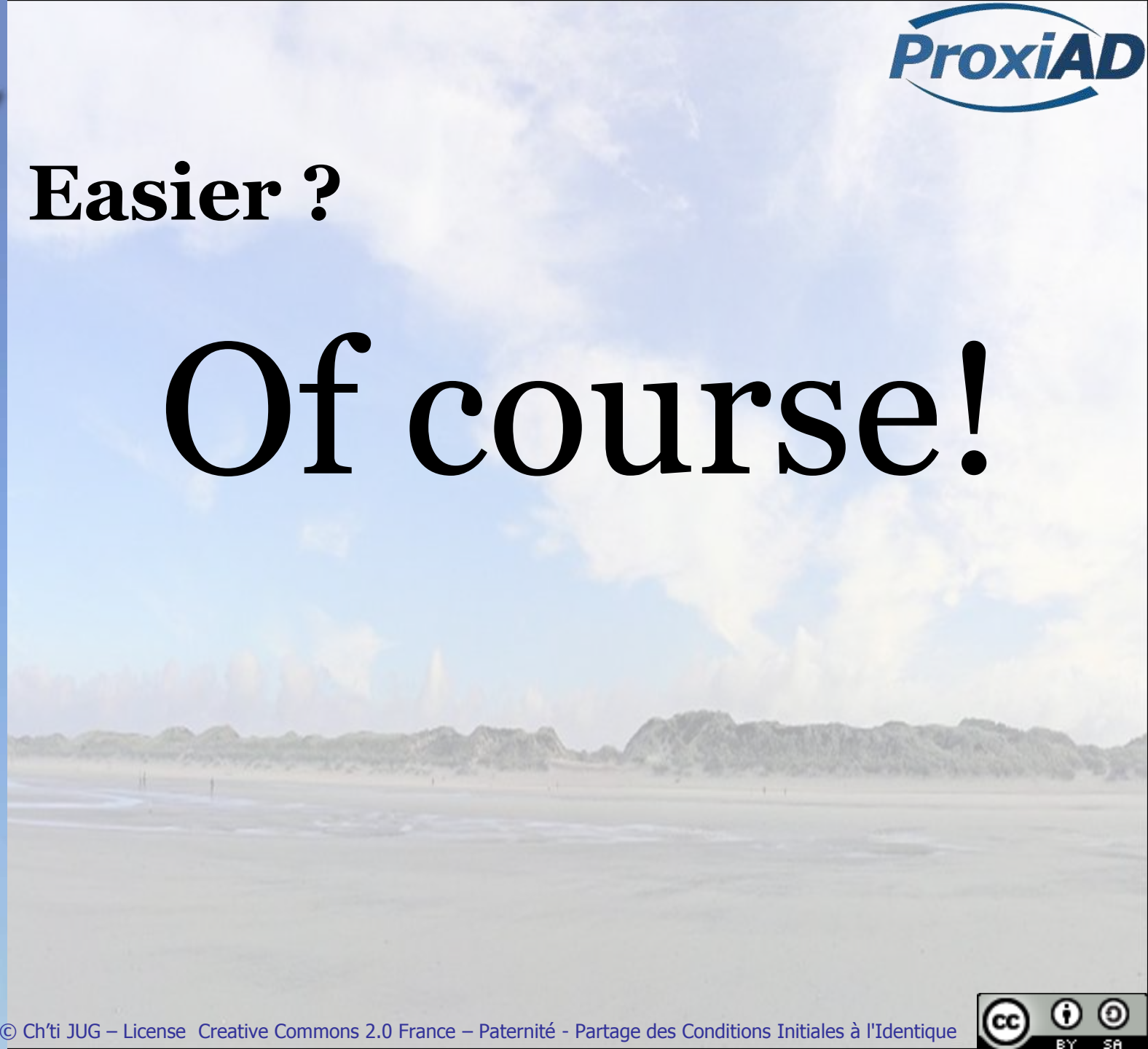


Ch'ti JUG



Easier ?

Of course!





Ch'ti JUG

Servlet 3.0



Servlet 3.0

JSP 2.2

JSF 2.0

Presentation Tier

EJB 3.1

JTA 1.1

Business Logic Tier

JPA 2.0

Persistence Tier

Java EE 6

Interoperability Tier

JAX-WS 2.2

JAX-RS 1.1

JMS 1.1

JCDI 1.0





Ch'ti JUG

Ease of development



- Annotations based programming model
 - @WebServlet
 - @ServletFilter
 - @WebServletContextListener
 - @InitParam
- Deployment descriptors optional (web.xml)
 - Modular





Ch'ti JUG

Ease of development



```
public class MyServlet extends HttpServlet {
    public void doGet (HttpServletRequest req,
                      HttpServletResponse res) {
        ....
    }
}
```

Deployment descriptor (web.xml)

```
<web-app>
  <servlet>
    <servlet-name>MyServlet</servlet-name>
    <servlet-class>samples.MyServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>MyServlet</servlet-name>
    <url-pattern>/MyApp</url-pattern>
  </servlet-mapping>
  ...
</web-app>
```





Ch'ti JUG

Ease of development



```
@WebServlet(urlMappings={ "/MyApp" })
public class MyServlet extends HttpServlet {

    public void doGet (HttpServletRequest req,
                      HttpServletResponse res) {
        . . . .
    }
}
```

web.xml is optional

- Same for filters and listeners





Ch'ti JUG



Extensibility

- Fragments or modular web.xml
 - Logical partitioning of a web application
- Annotations and web fragments are merged

```
<web-fragment>
```

```
<servlet>
```

```
  <servlet-name>myservlet</servlet-name>
```

```
  <servlet-class>samples.MyServlet</servlet-class>
```

```
</servlet>
```

```
<listener>
```

```
  <listener-class>samples.MyListener</listener-class>
```

```
</listener>
```

```
</web-fragment>
```





Ch'ti JUG



Asynchronous support

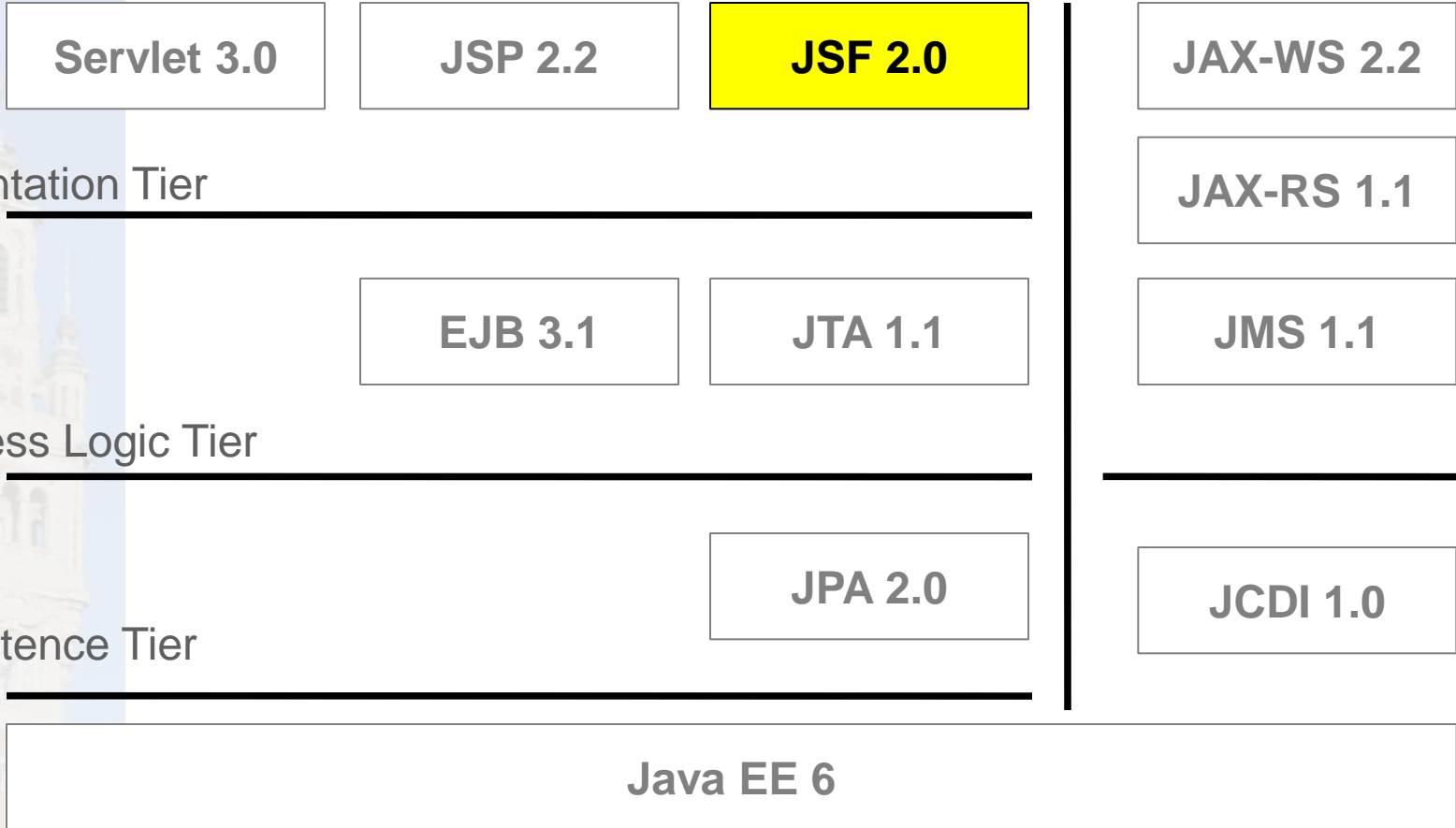
- Servlets have to wait for a response from :
 - Web service
 - JDBC connection
 - JMS message....
- Comet style of programming
- `@WebServlet (asyncSupported = true)`
- New APIs to `ServletRequest` / `ServletResponse`
 - Suspending, resuming, querying the status of the request





Ch'ti JUG

JSF 2.0





Ch'ti JUG

Ease of development



- Annotations
 - @ManagedBean
 - @ApplicationScoped, @SessionScoped...
 - @FacesConverter
 - @FacesValidator
- faces-config.xml optional
- Page declaration language (PDL)
 - Facelets (preferred)
 - JSP (still supported)
- Easier resources management
- Easier way of Component Development
- Ajax support





Ch'ti JUG



Managed bean

```
public class DatabaseUtil {  
  
    private Cities cities;  
    ...  
}
```

faces-config.xml

```
<managed-bean>  
    <managed-bean-name>dbUtil</managed-bean-name>  
    <managed-bean-class>server.DatabaseUtil  
</managed-bean-class>  
    <managed-bean-scope>request</managed-bean-scope>  
    <managed-property>  
        <property-name>cities</property-name>  
        <value>#{cities}</value>  
    </managed-property>  
</managed-bean>
```





Ch'ti JUG



Managed bean

```
@ManagedBean (name="dbUtil")
@ApplicationScoped
public class DatabaseUtil {

    @ManagedProperty (value="#{cities}")
    private Cities cities;

}
```

faces-config.xml is optional

- Same for converters and validators





Ch'ti JUG

Composite component



- No Java code needed
- Use XHTML and JSF tags to create components
- Like Java programming you need :
 - «an interface»
 - «an implementation»





Ch'ti JUG



Composite component

```
<html>
<composite:interface>
  <composite:attribute name="item" required="true"/>
</composite:interface>

<composite:implementation>
  <tr>
    <td>Title :</td>
    <td>
      <h:inputText
value="#{compositeComponent.attrs.item.title}"/>
    </td>
  </tr>
  <tr>
    <td>Description :</td>
    <td>
      <h:inputText
value="#{compositeComponent.attrs.item.desc}" />
    </td>
  </tr>
</composite:implementation>
</html>
```





Ch'ti JUG



Ajax support

- Previous versions had no native Ajax solution
- Ajax support has been specified
- JavaScript library (jsf.js)
 - Several specified JavaScript functions
 - request, response, execute, render...
- Easier integration in your pages





Ch'ti JUG

EJB 3.1



Servlet 3.0

JSP 2.2

JSF 2.0

Presentation Tier

JAX-WS 2.2

JAX-RS 1.1

JMS 1.1

Interoperability Tier

EJB 3.1

JTA 1.1

Business Logic Tier

JPA 2.0

JCDI 1.0

Persistence Tier

Java EE 6





Ch'ti JUG

Easier & Richer



- Optional Local Interfaces
- Singleton
- Asynchronous calls
- Cron-based Timer Service
- Packaging in a war
- Portable JNDI name
- Embeddable Container
- EJB Lite





Ch'ti JUG

Optional Local Interface



- @Local, @Remote
- Interfaces are not always needed
 - Only for local interfaces
 - Remote interfaces are not optional !

@Stateless

```
public class HelloBean {  
  
    public String sayHello() {  
        return "Salut biloute !";  
    }  
}
```





Ch'ti JUG



Singleton

- New component
 - Looks like a stateless / stateful
 - No/local/remote interface
- Follows the Singleton pattern
 - One single EJB per application per JVM
- Use to share state in the entire application
 - State not preserved after container shutdown
- Added concurrency management
 - @ConcurrencyManagement





Ch'ti JUG

Singleton



@Singleton

```
public class CachingBean {  
  
    private Map cache;  
  
    @PostConstruct void init() {  
        cache = ...;  
    }  
  
    public Map getCache() {  
        return cache;  
    }  
  
    public void addToCache(Object key, Object val) {  
        cache.put(key, val);  
    }  
}
```





Ch'ti JUG



Asynchronous calls

- How to have asynchronous call in EJBs ?
- JMS is to send messages not to do asynchronous calls
- Threads are not allowed (don't integrate well)
- `@Asynchronous`
- Method returns `void` or `Future<T>`
 - `java.util.concurrent` package





Ch'ti JUG



Asynchronous calls

```
@Stateless
public class OrderBean {

    public void createOrder() {
        Order order = persistOrder();
        sendEmail(order) ;
    }

    public Order persistOrder() {...}

    @Asynchronous
    public void sendEmail(Order order) {...}
}
```





Ch'ti JUG



Packaging in a war

foo.ear

lib/foo_common.jar

com/acme/**Foo**.class

foo_web.war

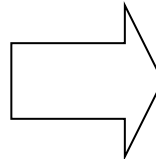
WEB-INF/web.xml

WEB-INF/classes

com/acme/**FooServlet**.class

foo_ejb.jar

com/acme/**FooEJB**.class



foo.war

WEB-INF/classes

com/acme/**Foo**.class

com/acme/**FooServlet**.class

com/acme/**FooEJB**.class





Ch'ti JUG

Portable JNDI Name



- Client inside a container (use DI)

```
@EJB Hello h;
```

- Client outside a container

```
Context ctx = new InitialContext();  
Hello h = (Hello) ctx.lookup(???) ;
```

- Portable JNDI name is specified

```
java:global/env/foo/HelloEJB
```

```
java:global/(app)/(module)/(bean)#(intf)
```



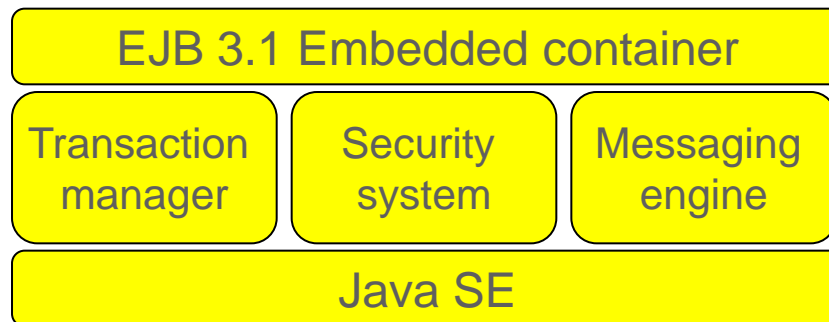


Ch'ti JUG



Embeddable Container

- API allowing to :
 - Initialize a container
 - Get container context
 - ...



- Can run in any Java SE environment
 - Batch processing
 - Simplifies testing
 - Just a jar file in your classpath





Ch'ti JUG

Embeddable Container



```
public class PlaceBidClient {
    public static void main(String[] args) throws Exception {

        EJBContainer container =
            EJBContainerFactory.createEJBContainer();

        Context context = container.getContext();

        Hello h = (Hello)
            context.lookup("java:global/app/foo/HelloEJB");

        h.sayHello;

        container.close();
    }
}
```





Ch'ti JUG



Timer Service

- Programmatic and Calendar based scheduling
 - « Last day of the month »
 - « Every five minutes on Monday and Friday »
- Cron-like syntax
 - second [0..59], minute[0..59], hour[0..23], year
 - DayOfMonth[1..31]
 - dayOfWeek[0..7] or [sun, mon, tue..]
 - Month[0..12] or [jan,feb..]





Ch'ti JUG



Timer Service

```
@Stateless
public class WakeUpBean {

    @Schedule(dayOfWeek="Mon-Fri", hour="9")
    void wakeUp() {
        ...
    }
}
```

- EJB Lite + Timer + Asynch calls + Embeddable Container = Batch processing





Ch'ti JUG

JPA 2.0



Interoperability Tier

Servlet 3.0

JSP 2.2

JSF 2.0

JAX-WS 2.2

Presentation Tier

JAX-RS 1.1

EJB 3.1

JTA 1.1

JMS 1.1

Business Logic Tier

JPA 2.0

JCDI 1.0

Persistence Tier

Java EE 6





Ch'ti JUG

JPA 2.0



- Java Persistent API
- Evolves separately from EJB now
 - JSR 317
- Can also be used in Java SE
- More mappings
 - JoinTable for OneToOne relationship
- Criteria API
- Standard properties in persistence.xml
- Simple Cache API





Ch'ti JUG

Collection of basic types



```
@Entity
Public class Item {

    @ElementCollection
    private Set<String> tags;
}
```

- Mapped in a separate table





Ch'ti JUG

Better Support of Map



```
@Entity
public class Department {
    ...
    @ElementCollection
    public Map<Integer, Employee> employees
    ...
}
```

- Basic types, Objects, Embeddables
- Mapped in a separate table





Ch'ti JUG



Locking Enhancement

```
public enum LockModeType {  
    OPTIMISTIC,  
    OPTIMISTIC_FORCE_INCREMENT,  
    PESSIMISTIC,  
    PESSIMISTIC_FORCE_INCREMENT,  
    NONE  
}
```

- JPA 1.0 only support optimist locking
- Now Pessimist locking
- Multiple places to specify lock
 - Lock, read and lock, read then lock





Ch'ti JUG



Criteria API

- Used to define dynamic queries
- Like JPQL, Criteria API is based on Entities
- Allow the construction of an object-based graph
- Strongly typed
- Uses a metamodel
 - Each entity X has a metamodel class X_





Ch'ti JUG

Criteria API



```
@Entity
public class Customer {
    @Id Integer custId;
    String name;
    Address shippingAddress;
    ...
}
```

```
CriteriaQuery q = qb.create();
Root<Customer> customer = q.from(Customer.class);
q.select(customer.get(Customer_.shippingAddress)).
where(q.equal(customer.get(Customer_.name), "Peter"));
```





Ch'ti JUG

JAX-RS 1.1



Servlet 3.0

JSP 2.2

JSF 2.0

JAX-WS 2.2

Presentation Tier

JAX-RS 1.1

EJB 3.1

JTA 1.1

JMS 1.1

Business Logic Tier

JPA 2.0

JCDI 1.0

Persistence Tier

Java EE 6





Ch'ti JUG



JAX-RS 1.1

- RESTful Services
- POJO and Annotations Based
- Data and functionality are considered resources
- Map HTTP

HTTP	Action	HTTP	Action
GET	Get a resource	PUT	Create or update
POST	Create a resource	Delete	Deletes a resource

- JAX-RS 1.0 has been released





Ch'ti JUG



Hello World

```
@Path("/helloworld")
public class HelloWorldResource {

    @GET
    @Produces("text/plain")
    public String sayHello() {
        return "Hello World";
    }
}

• http://example.com/helloworld
```





Ch'ti JUG

Hello World



Request

```
GET /helloworld HTTP/1.1
Host: example.com
Accept: text/plain
```

Response

```
HTTP/1.1 200 OK
Date: Wed, 12 Nov 2008 16:41:58 GMT
Server: Apache/1.3.6
Content-Type: text/plain; charset=UTF-8
Hello World
```





Ch'ti JUG



MIME Types

```
@Path("/helloworld")
public class HelloWorldResource {

    @GET @Produces("image/jpeg")
    public byte[] paintHello() {
        ...
    }

    @POST @Consumes("text/xml")
    public void updateHello(String xml) {
        ...
    }
}
```





Ch'ti JUG



Parameters

```
@Path("/users/{userId}")
public class UserResource {

    @GET
    @Produces("text/xml")
    public String getUser(@PathParam("userId")
                          String userName) {
        ...
    }
}
```

- <http://example.com/users/Smith123>





Ch'ti JUG

Summary



Servlet 3.0

JSP 2.2

JSF 2.0

Presentation Tier

EJB 3.1

JTA 1.1

Business Logic Tier

JPA 2.0

Persistence Tier

Interoperability Tier

JAX-WS 2.2

JAX-RS 1.1

JMS 1.1

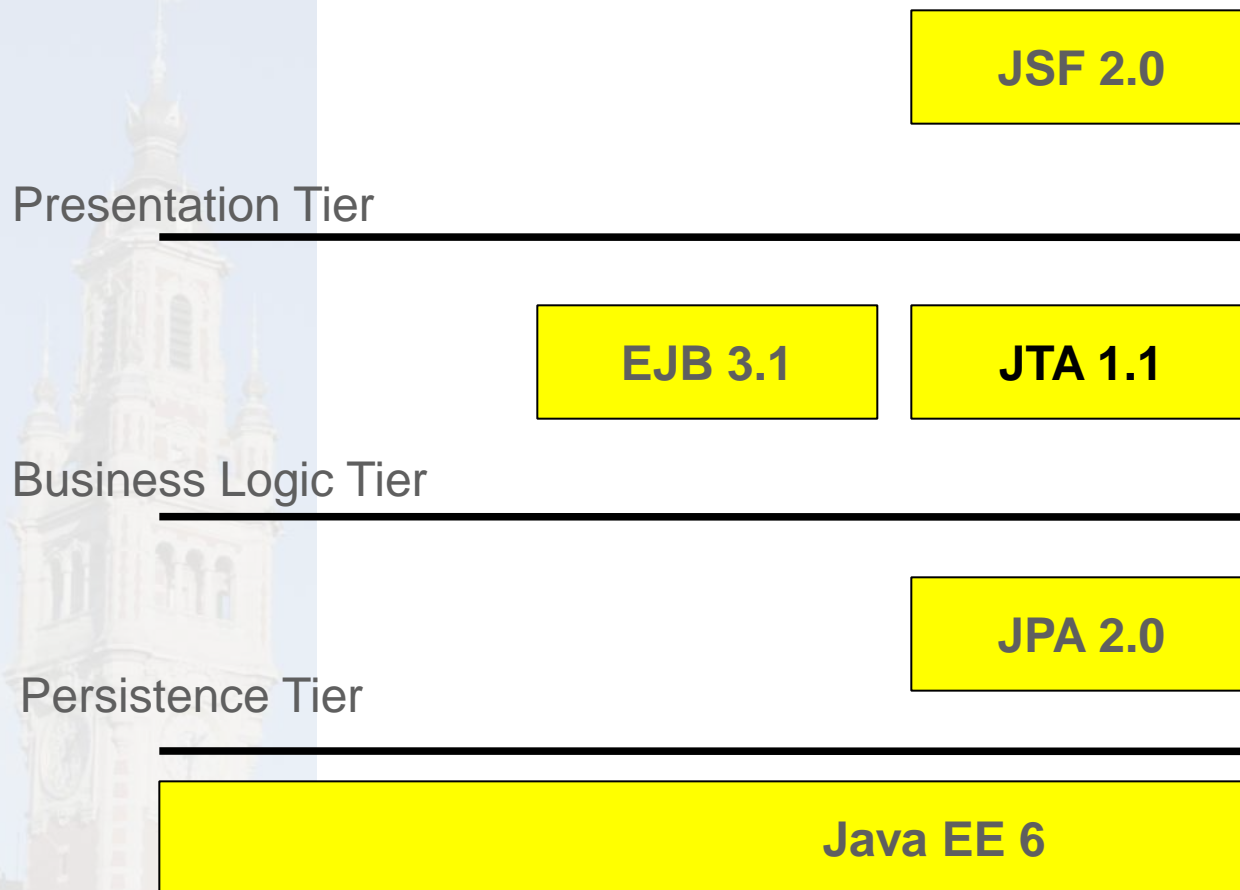
JCDI 1.0

Java EE 6





From simple web application





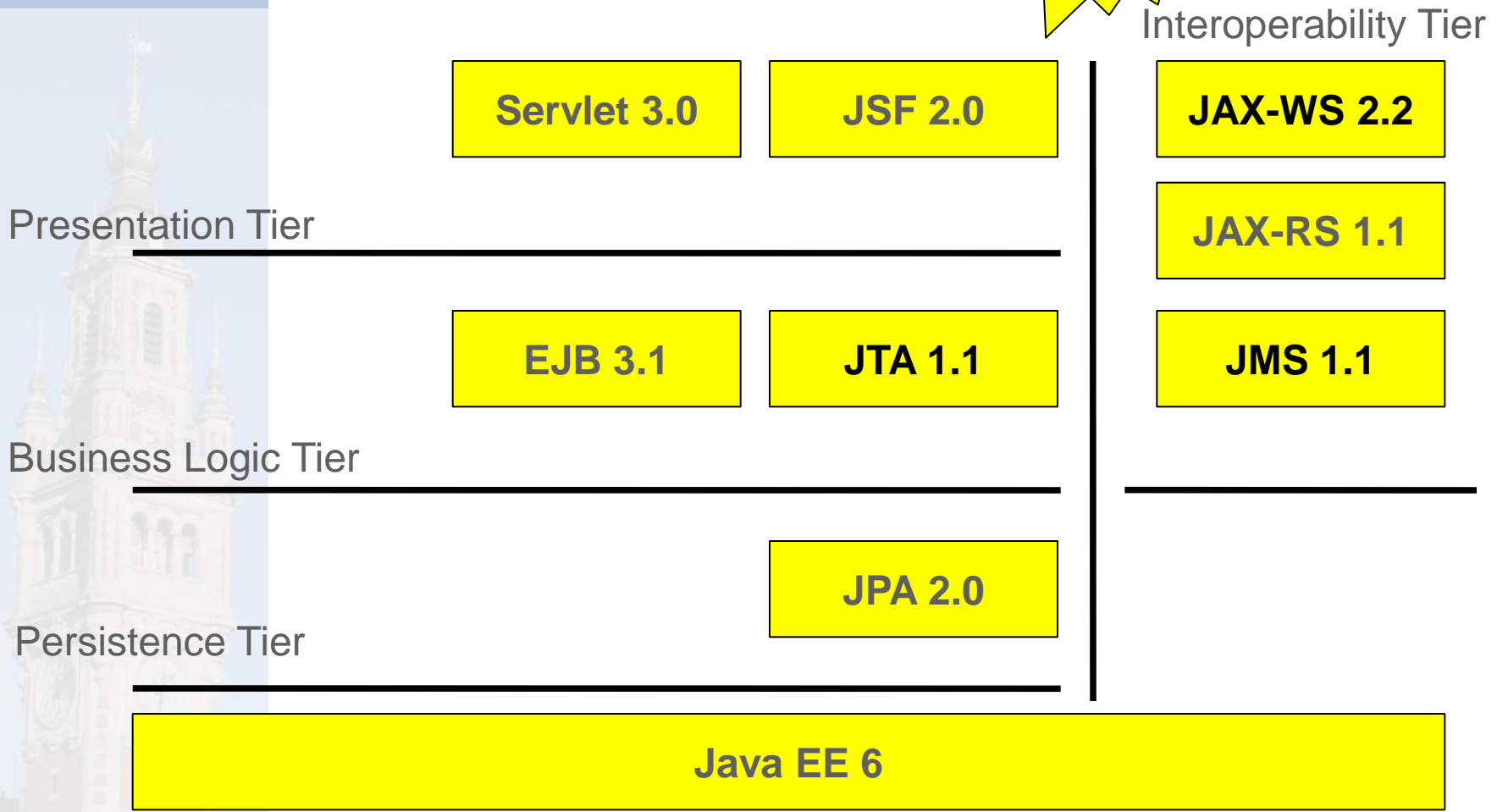
Ch'ti JUG



...to richer ones



ear in
a cluster





Reference Implementations



- All these specs have reference implementations
 - GlassFish V3 : EJB 3.1 and Servlet 3.0
 - EclipseLink : Java Persistence API (JPA 2.0)
 - Jersey : RESTful Web Services (JAX-RS 1.0)
 - Metro : Web Services (JAX-WS 2.2)
 - JBoss Seam : Web Beans 1.0
 - Mojarra : JSF 2.0And they are production ready

Open Source





Ch'ti JUG



Summary

- Java EE 6 is
 - Simpler (POJO, annotation, less XML, Pruning)
 - Richer (more specifications)
 - Lighter (profiles, pruning, EJB lite)
 - Standard (no vendor locking)
 - Robust (10th anniversary)
 - Book out in June 2009
 - **Java EE 6 out in September 2009**

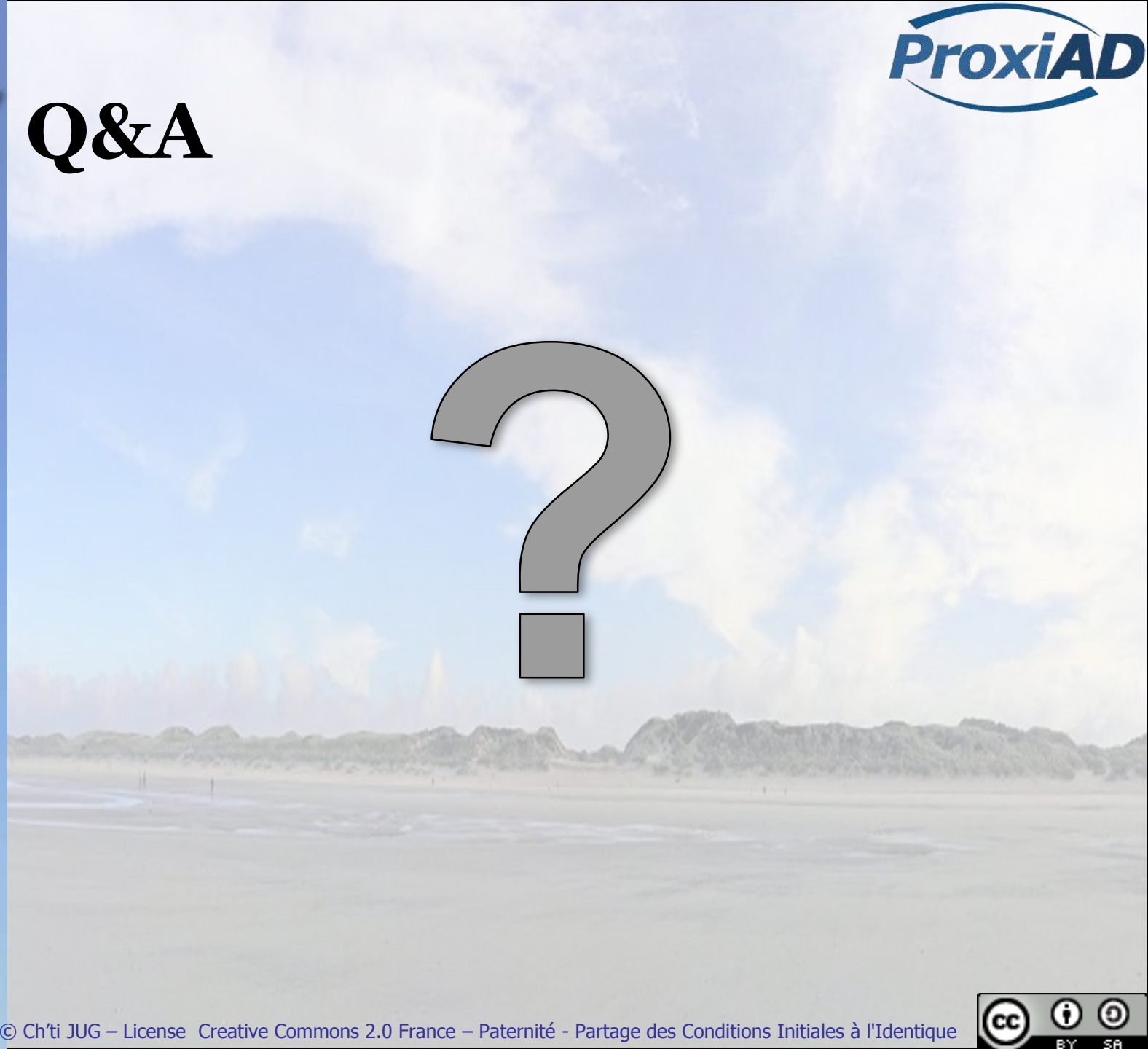
« *Forget the past, look to the future, Java EE 6 is the place to go...* » - Antonio Goncalves





Ch'ti JUG

Q&A



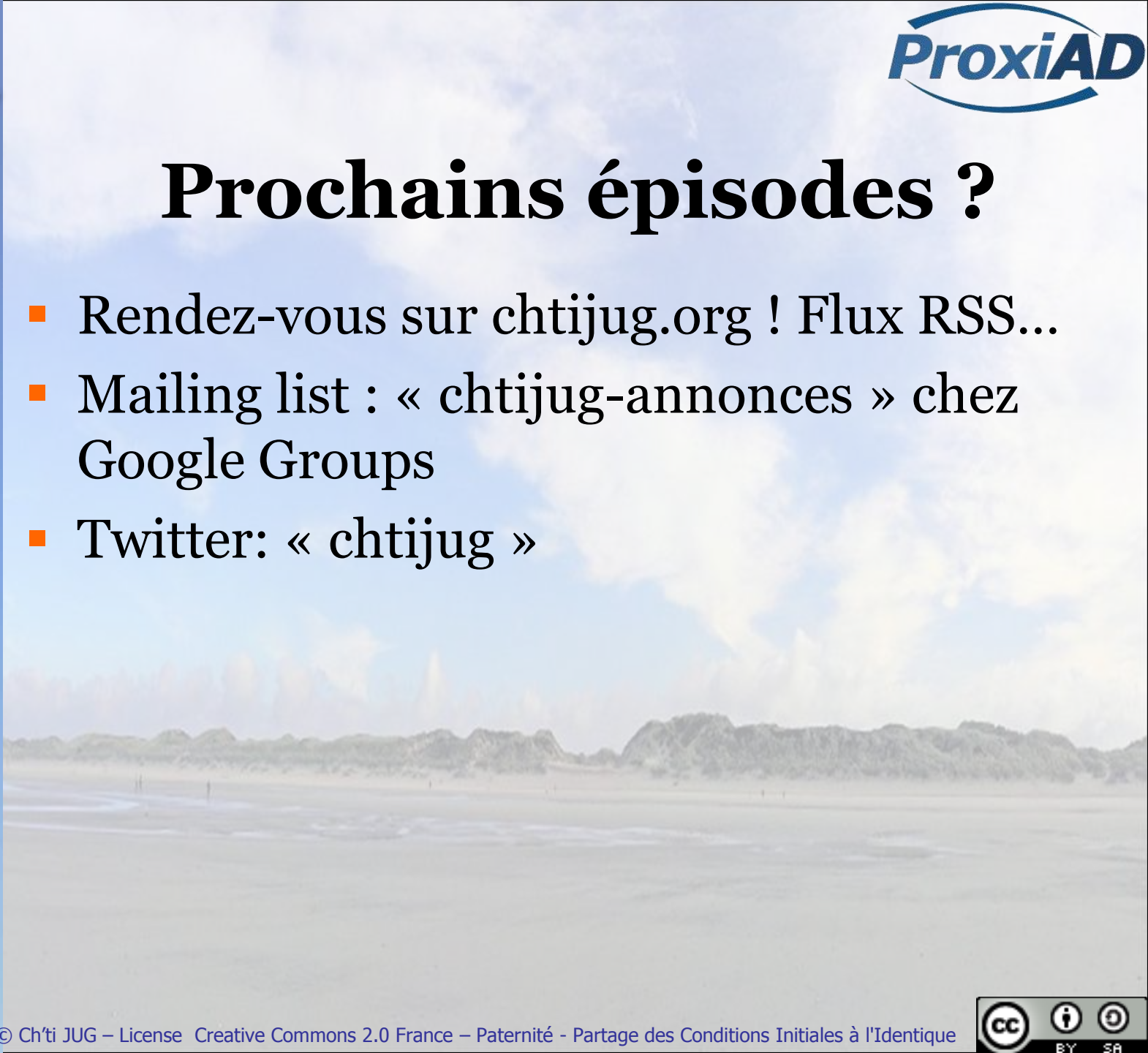


Ch'ti JUG



Prochains épisodes ?

- Rendez-vous sur chtijug.org ! Flux RSS...
- Mailing list : « [chtijug-annonces](#) » chez Google Groups
- Twitter: « [chtijug](#) »

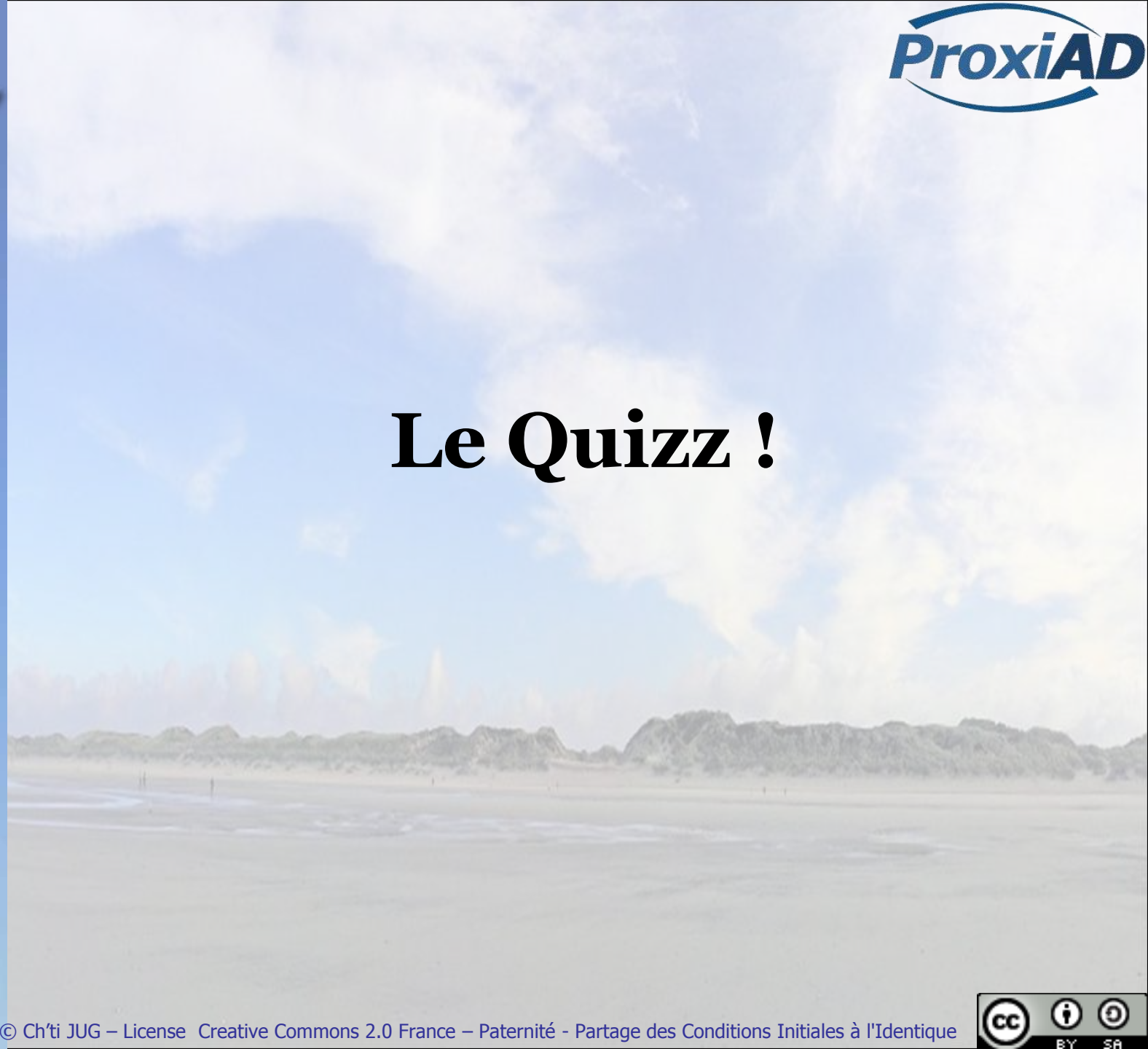




Ch'ti JUG



Le Quizz !





Ch'ti JUG

Servlet 3.0



- Parmi les éléments suivants, lequel est une nouveauté de Servlet 3.0 ?
 1. Prise en charge du JDK 1.6
 2. Un fichier web.xml optionnel
 3. Prise en compte de la locale ch_TI





Ch'ti JUG

EJB 3.1



- Parmi les assertions suivantes concernant les EJB 3.1, laquelle est vraie ?
 1. Pour être appelé en local, je n'ai pas besoin de coder d'interface
 2. Pour être appelé en remote, je n'ai pas besoin de coder d'interface
 3. Je suis obligé de déployer mon EJB dans un JAR séparé du WAR





Ch'ti JUG

JPA 2.0



- Parmi les éléments suivants, lequel est une nouveauté de JPA 2.0 ?
 1. L'utilisation d'annotations
 2. L'API de Criteria
 3. Les locks optimistes





Ch'ti JUG

Les JUGs



- Combien y a-t-il de JUGs en France ?
 1. entre 7 et 8
 2. entre 13 et 15
 3. entre 20 et 23
 4. 42



