What's new in Java EE 6 ?

Antonio Goncalves

Overall presentation

Focuses on news features of Java EE 6

You must know Java EE 5

28 specifications

Thousands of pages

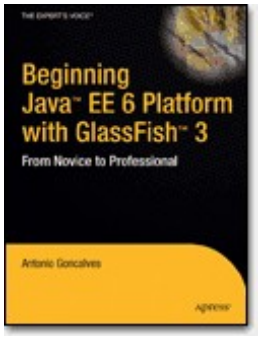
60 slides in 60 minutes

Agenda

- Quick overview
- New concepts
- New features on existing specifications
- New specifications
- Summary

Antonio Goncalves

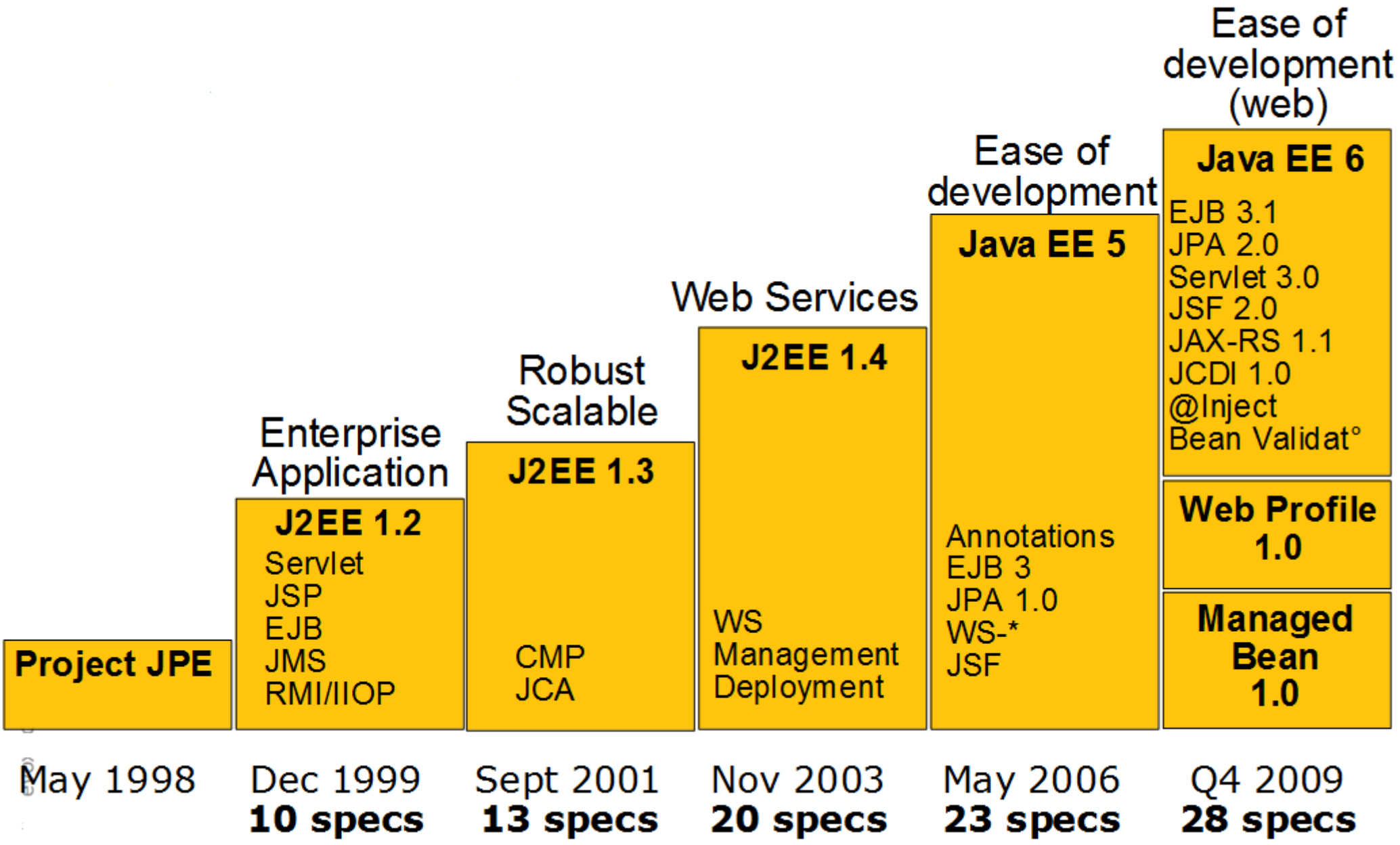
- Freelance software architect
- Former BEA consultant
- Author (Java EE 5 and Java EE 6)
- JCP expert member
- Co-leader of the Paris JUG
- Les Cast Codeurs podcast
- Java Champion
- antonio dot goncalves at gmail dot com



Agenda

- Quick overview
- New concepts
- New features on existing specifications
- New specifications
- Summary

A brief history



Zooming in Java EE 6

Web

JSF	**2.0**
Servlet	**3.0**
JSP	2.2
EL	2.2
JSTL	1.2
Debugging Support	1.0

Enterprise

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

Web Services

JAX-RPC	1.1
JAXM	1.0
JAX-RS	**1.1**
JAXR	1.0
Web Services	1.3
WS Metadata	2.0

Management, Security, Common

CDI (JSR 299)	**1.0**
@Inject (JSR 330)	**1.0**
Bean Validation	**1.0**
Interceptors	**1.1**
Managed Beans	**1.0**
JACC	1.4
Java EE Application Deployment	1.2
Java EE Management	1.1
JASPIC	1.0

+ Java SE 6

JAX-WS	**2.2**
JAXB	2.2
JDBC	4.0
JNDI	1.5
SAAJ	1.3
Common Annotations	**1.1**
RMI	
Java IDL	
JMX	
JAAS	
JAXP	
StAX	

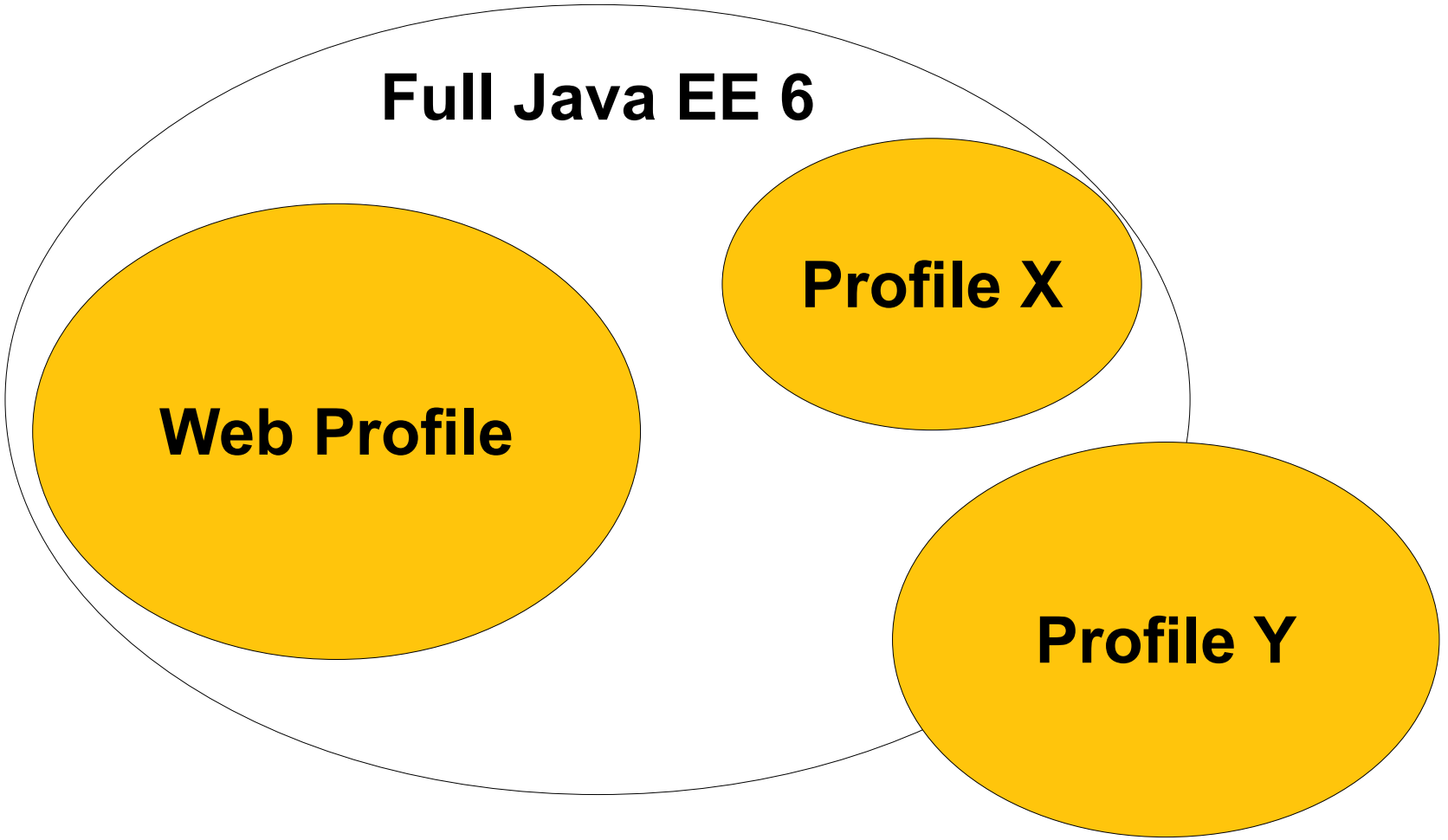
Agenda

- Quick overview
- New concepts
  - Pruning, Profiles, EJB Lite, Portable JNDI names, Managed Beans, Interceptors 1.1
- New features on existing specifications
- New specifications
- Summary

Pruning (Soon less specs)

- Marks specifications optional in next version
- Pruned in Java EE 6
  - Entity CMP 2.x
  - JAX-RPC
  - JAX-R
  - JSR 88 (Java EE Application Deployment)
- Might disappear from Java EE 7
  - Vendors may decide to keeps them...
  - … or offer the delta as a set of modules

Profiles



Web Profile 1.0

- Subset of full platform
- For web development
  - Packages in a war
- Separate specification
- Evolves at its own pace
- Others will come
  - Minimal (Servlet/JSP)
  - Portal....

JSF	2.0
Servlet	3.0
JSP	2.2
EL	2.2
JSTL	1.2
EJB **Lite**	3.1
Managed Beans	1.0
Interceptors	1.1
JTA	1.1
JPA	2.0
Bean Validation	1.0
CDI	1.0
@Inject	1.0

EJB Lite

- Subset of the EJB 3.1 API
- Used in Web profile
- Packaged in a war

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

Portable JNDI names

- Client inside a container (use DI)

```
@EJB Hello h;
```

- Client outside a container

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

- Portable JNDI name is specified

```
java:global/foo/bar/HelloEJB
```

Portable JNDI names

- **`OrderBean`** implements **`Order`** packaged in **`orderejb.jar`** within **`orderpp.ear`**
- `java:`**`global`**`/orderapp/orderejb/OrderBean`
  `java:`**`global`**`/orderapp/orderejb/OrderBean!org.foo.Order`

Usable from any application in the container

- `java:`**`app`**`/orderejb/OrderBean`
  `java:`**`app`**`/orderejb/OrderBean!com.acme.Order`

Fully-qualified interface name

- `java:`**`module`**`/OrderBean java:`**`module`**`/OrderBean!org.foo.Order`

Managed Beans 1.0

- Separate spec shipped with Java EE 6
- Container-managed POJOs
- Support a small set of basic services
  - Injection (`@Resource`...)
  - Life-cycle (`@PostConstruct`, `@PreDestroy`)
  - Interceptor (`@Interceptor`, `@AroundInvoke`)
- Lightweight component model

Managed Beans 1.0

```
@javax.annotation.ManagedBean
public class MyPojo {

  @Resource
  private Datasource ds;

  @PostConstruct
  private void init() {
    ....
  }

  @Interceptors(LoggingInterceptor.class)
  public void myMethod() {...}
}
```

Interceptors 1.1

- Address cross-cutting concerns in Java EE
- Were part of the EJB 3.0 spec
- Now a seperate spec shipped with EJB 3.1
- Can be uses in EJBs...
- … as well as ManagedBeans
- `@AroundInvoke`
- `@AroundTimeout` for EJB timers

Agenda

- Quick overview
- New concepts
- New features on existing specifications
  - JPA 2.0, EJB 3.1, Servlet 3.0, JSF 2.0
- New specifications
- Summary

JPA 2.0

- Evolves separately from EJB now
  - JSR 317
- Richer mappings
- Richer JPQL
- Standard config options
- Criteria API
- ...

Richer mapping

- Collection of embeddables and basic types
  - Not just collection of JPA entities
  - Multiple levels of embeddables
- More flexible support for Maps
  - Keys, values can be one of : entities, embeddables or basic types
- More relationship mapping options
  - Unidirectional 1-many foreign key mappings

Collections of Embeddable Types

```
@Embeddable public class BookReference {
    String title;
    Float price;
    String description;
    String isbn;
    Integer nbOfPage;
    ...
}

@Entity public class ListOfGreatBooks {
    @ElementCollection
    protected Set<BookReference> javaBooks;
    ...
}
```

Richer JPQL

- Added entity type to support non-polymorphic queries
- Allow joins in subquery FROM clause
- Added new reserved words
  - ABS, BOTH, CONCAT, ELSE, END, ESCAPE, LEADING, LENGTH, LOCATE, SET, SIZE, SQRT, SUBSTRING, TRAILING

Criteria API

- Strongly typed criteria API
- Object-based query definition objects
  - (Rather than JPQL string-based)
- Operates on the meta-model
  - Browse the structure of a Persistence Unit
  - Dynamically: `EntityManager.getMetamodel()`
  - Statically:
    Each entity **`x`** has a metamodel class **`x_`**
- `CriteriaQuery` as a query graph

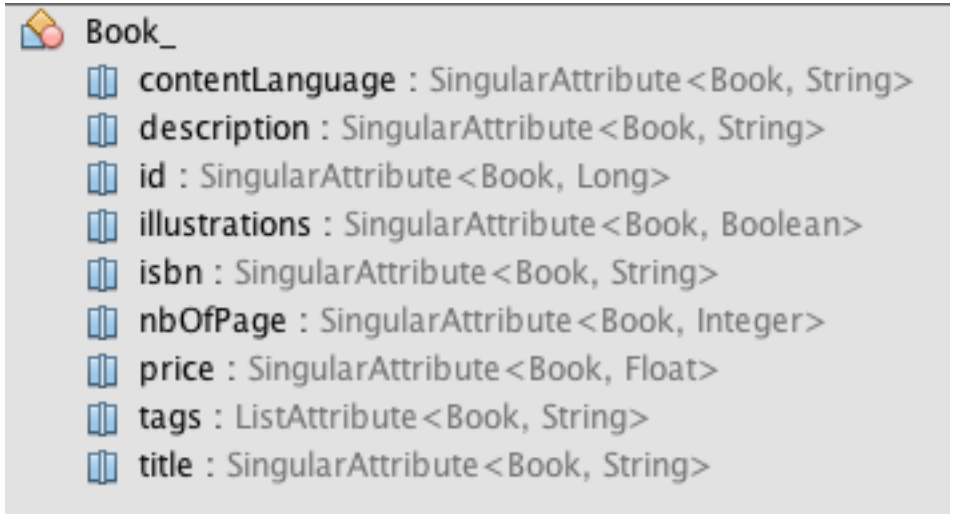
Criteria API

```
EntityManager em = ...;

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<Book> query =
                      cb.createQuery(Book.class);

Root<Book> book = query.from(Book.class);

query.select(book)
     .where(cb.equal(book.get("description"), ""));
```

```
SELECT b

FROM Book b

WHERE b.description IS EMPTY
```

Criteria API (Type-safe)

```
EntityManager em = ...;

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<Book> query =
                      cb.createQuery(Book.class);

Root<Book> book = query.from(Book.class);


query.select(book)
     .where(cb.isEmpty(book.get(Book_.description)));
```

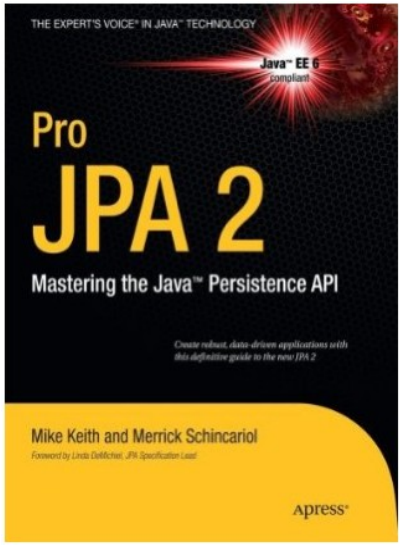


Statically generated
JPA 2.0 MetaModel

Criteria API (Builder pattern)

```
EntityManager em = ...;
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<Book> query =
                      cb.createQuery(Book.class);

Root<Book> book = query.from(Book.class);

query.select(book)
     .where(cb.isEmpty(book.get(Book_.description)))
     .orderBy(...)
     .distinct(true)
     .having(...)
     .groupBy(...);

List<Book> books =
             em.createQuery(query).getResultList();
```

And more...



- `detach()`
- `Join<X,Y>`, `ListJoin`, `MapJoin`
- Orphan removal functionality
  - `@OneToMany(orphanRemoval=true)`
- BeanValidation integration on lifecycle
- Second-level cache API
  - `@Cacheable` annotation on entities
  - `contain(Class,PK)`, `evict(Class,PK)`, ...
- Pessimistic locking

Servlet 3.0

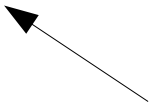
- Ease of development
- Pluggability
- *Asynchronous support*

Ease of development

- Annotations based programming model
  - `@WebServlet`
  - `@WebFilter`
  - `@WebListener`
  - `@WebInitParam`
- Optional `web.xml`
- Better defaults and CoC

A servlet 3.0 example

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

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



web.xml ***is optional***

- Same for `@WebFilter` and `@WebListener`

Pluggability

- Fragments are similar to `web.xml`
- `<web-fragment>` instead of `<web-app>`
  - Declare their own servlets, listeners and filters
- Annotations and web fragments are merged following a configurable order
- JARs need to be placed in `WEB-INF/lib`
- and use `/META-INF/web-fragment.xml`
- Overridden by main `web.xml`

And more...

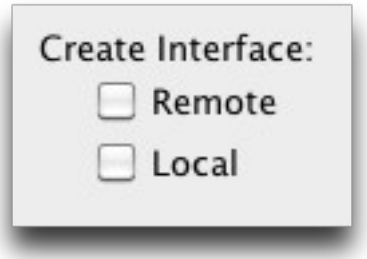
- Async support (Comet-style)
- Static resources in `META-INF/resources`
- Configuration API
  - Add and configure Servlet, Filters, Listeners
  - Add security constraints
  - Using `ServletContext` API
- File upload (similar to Apache File Upload)
- Configure cookie session name
- Security with `@ServletSecurity`

EJB 3.1

- Optional local interface
- Packaging in a war
- Asynchronous calls
- Timer service
- Singleton
- Embeddable container

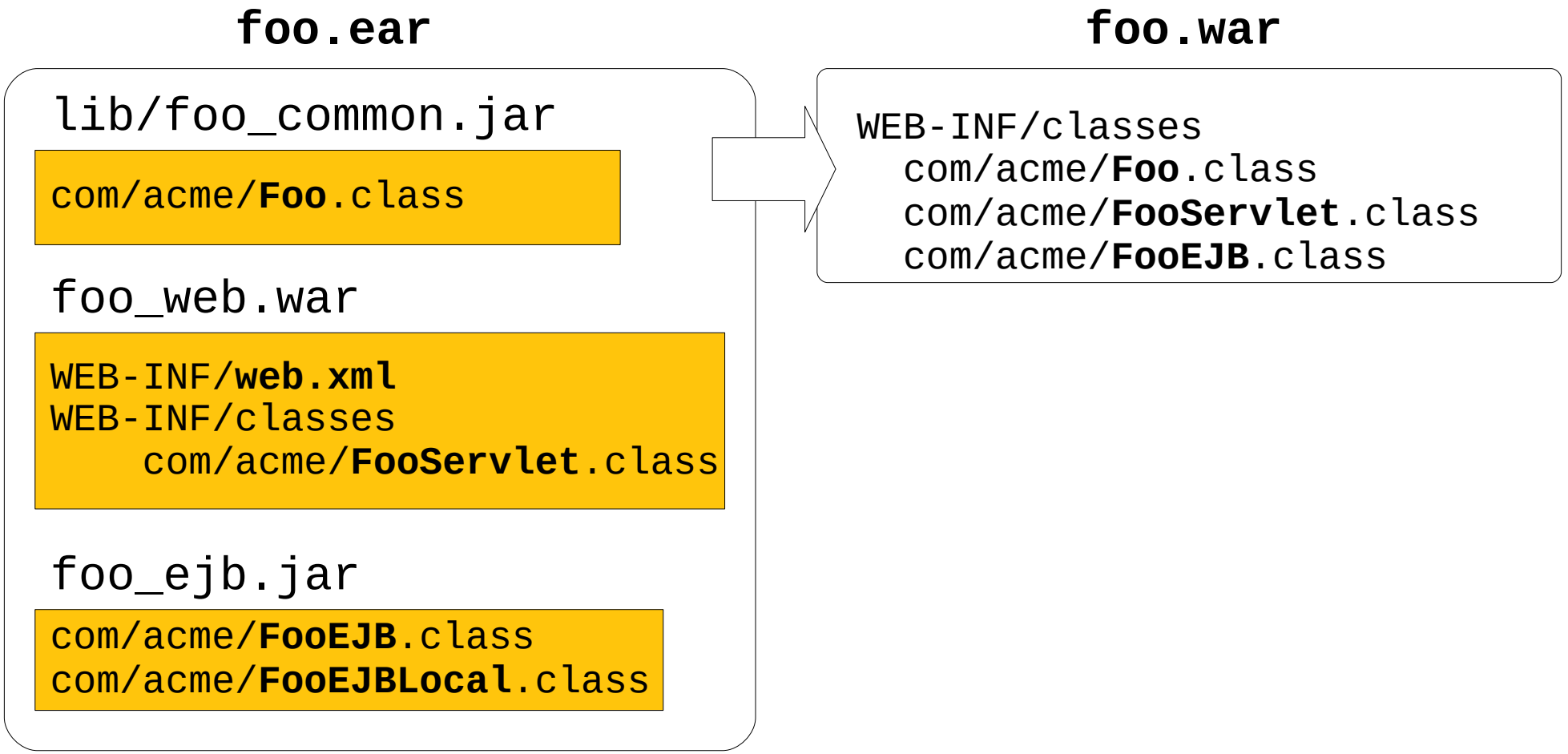
Optional Local Interface

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




```
@Stateless
public class HelloBean {

  public String sayHello() {
    return "Hello world!";
  }
}
```

Packaging in a war



Asynchronous calls

- How to have asynchronous call in EJBs ?
  - JMS is more about sending messages
  - Threads and EJB's don't integrate well
- `@Asynchronous`
  - Applicable to any EJB type
  - Best effort, no delivery guarantee
- Method returns `void` or `Future<T>`
  - `javax.ejb.AsyncResult` helper class : `return new AsyncResult<int>(result)`

Asynchronous calls

```
@Stateless
public class OrderBean {

  public void createOrder() {
    Order order = persistOrder();
    sendEmail(order);  // fire and forget
  }

  public Order persistOrder() {...}

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

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]...
  - **`dayOfMonth`**[1..31]
  - **`dayOfWeek`**[0..7] or [sun, mon, tue..]
  - **`Month`**[0..12] or [jan,feb..]

Timer Service

```
@Stateless
public class WakeUpBean {

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

Singleton

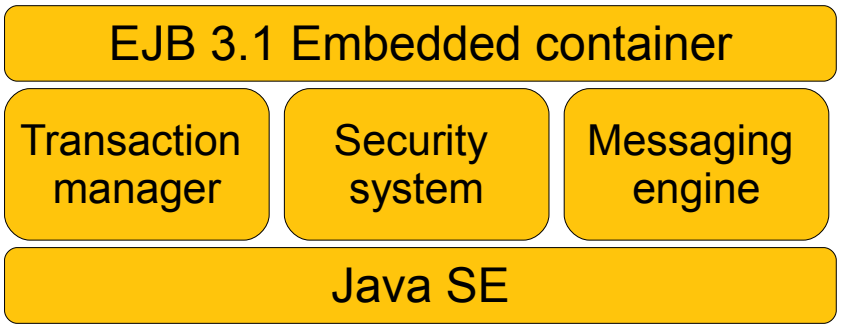
- New component
  - No/local/remote interface
- Follows the Singleton pattern
  - One single EJB per application per JVM
- Used to share state in the entire application
  - State not preserved after container shutdown
- Added concurrency management
  - Default is single-threaded
  - `@ConcurrencyManagement`

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);
  }
}
```

Embeddable Container

- API allowing to :
  - Initialize a container
  - Get container ctx
  - …



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

Embeddable Container

```
public static void main(String[] args){

  EJBContainer container =
        EJBContainer.createEJBContainer();

  Context context = container.getContext();

  Hello h = (Hello)context.lookup("Global_JNDI_Name");

  h.sayHello();

  container.close();
}
```

And more...

- Interceptors and InterceptorBinding
- Singletons can be chained
- Non persistent timer
- `@StatefulTimeout`
- ...

JSF 2.0

- The standard component-oriented MVC framework
- Part of Java EE 5
- Part of Java EE 6 and Web Profile
  - Other frameworks can rely on EE 6 extensibility
- Deserves its 2.0 version number
  - New features, issues fixed, performance focus
- Fully available today in Mojarra 2.0.2
  - Production-quality implementation
  - Part of GlassFish v3

Facelets now preferred VDL

- Facelets (XHTML) as alternative to JSP
  - Based on a generic View Description Language (VDL)
  - Can't add Java code to XHTML page (and “that's a good thing!”™)
- Pages are usable from basic editors
- IDEs offer traditional value-add:
  - Auto-completion (EL)
  - (Composite) Component management
  - Project management, testing, etc...

Setup, configuration

- JSF 2.0 does not mandate Servlet 3.0
  - Servlet 2.5 containers will run JSF 2.0
  - `web.xml` may be optional depending on runtime
- `faces-config.xml` now optional
  - `@javax.faces.bean.ManagedBean`
  - *Not required with JSR 299*
  - Navigation can now belong to the page (`<navigation-rules>` become optional)

JSF Composite Component

- Using JSF 1.x
  - Implement `UIComponent`, markup in renderer, register in `faces-config.xml`, add `tld`, ...
- With JSF 2.0
  - Single file, no Java code needed
  - Use XHTML and JSF tags to create components

```
<html xmlns:cc="http://java.sun.com/jsf/composite">
<cc:interface>
  <cc:attribute ...>
<cc:implementation>
```

  - Everything else is auto-wired

Ajax support

- Inspired by RichFaces, IceFaces, DynaFaces, ...
- Common JavaScript library (`jsf.js`)
  - `request` JavaScript functions captured by `PartialViewContext` for sub-tree processing
  - Client JavaScript updates the DOM

```
<h:commandButton
   onclick="jsf.ajax.request(this,event,{render:'foo'});
            return false;"/>
```

- `<f:ajax>` tag to ajaxify existing pages

```
xmlns:f="http://java.sun.com/jsf/core"
<h:commandButton>
  <f:ajax event="change" execute="myForm" render="foo" />
</h:commandButton>
```

And more...

- Validation delegated to BeanValidation
- Easier resources management
- Better error reporting
- New managed bean scope (View)
- Groovy support (Mojarra)
- Bookmarkable URLs
- Templating : define and apply layouts
- Project stages (dev vs. test vs. production)
- ...

Agenda

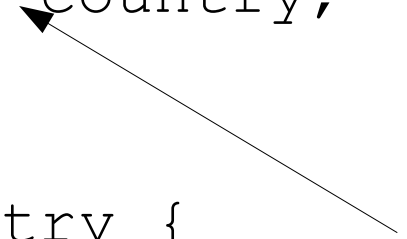
- Quick overview
- New concepts
- New features on existing specifications
- New specifications
  - Bean Validation, JAX-RS, @Inject, CDI
- Summary

Bean Validation 1.0

- Enable declarative validation in your applications
- Constrain Once, Validate Anywhere
  - restriction on a bean, field or property
  - not null, size between 1 and 7, valid email...
- Standard way to validate constraints
- Integration with JPA 2.0 & JSF 2.0

Bean Validation 1.0

```
public class Address {
   @NotNull @Size(max=30,
        message="longer than {max} characters")
   private String street1;
   ...
   @NotNull @Valid
   private Country country;
}

public class Country {
   @NotNull @Size(max=20)
   private String name;
   ...
}
```



request recursive object graph validation

Build your own!

```
@Size(min=5, max=5)
@ConstraintValidator(ZipcodeValidator.class)
@Documented
@Target({ANNOTATION_TYPE, METHOD, FIELD})
@Retention(RUNTIME)
public @interface ZipCode {
    String message() default "Wrong zipcode";
    String[] groups() default {};
}
```

And more...

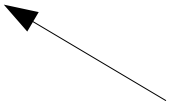
- Group subsets of constraints
- Partial validation
- Order constraint validations
- Create your own
- Bootstrap API
- Messages can be i18n
- Integration with JPA 2.0 and JSF 2.0
- ...

JAX-RS 1.1

- High-level HTTP API for RESTful Services
- POJO and Annotations Based
  - API also available
- Maps HTTP verbs (Get, Post, Put, Delete...)
- JAX-RS 1.0 has been released in 2008
- JAX-RS 1.1 integrates with EJBs
  (and more generally with Java EE 6)

Hello World

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

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



GET http://example.com/helloworld

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: GlassFish v3
Content-Type: text/plain; charset=UTF-8
Hello World
```

Different Mime Types

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

   @GET @Produces("image/jpeg")
   public byte[] paintHello() {
   ...
   @GET @Produces("text/plain")
   public String displayHello() {
   ...
   @POST @Consumes("text/xml")
   public void updateHello(String xml) {
   ...
}
```

Parameters & EJBs

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

   @PersistenceContext
   EntityManage em;

   @GET @Produces("text/xml")
   public String getUser(@PathParam("userId")
                                    String id){

      User u = em.find(User.class, id)
      ...
   }
}
```

And more...

- Different parameters (`@MatrixParam`, `@QueryParam`, `@CookieParam` ...)
- Support for `@Head` and `@Option`
- Inject `UriInfo` using `@Context`
- JAX-RS servlet mapping with `@ApplicationPath("rs")`
- ...

Injection in Java EE 5

- Common Annotation
  - `@Resource`
- Specialized cases
  - `@EJB`, `@WebServicesRef`, `@PersistenceUnit` ...
- Requires *managed* objects
  - EJB, Servlet and JSF Managed Bean in EE 5
  - Also in any Java EE 6's `javax.annotation.ManagedBean`

Injection in Java EE 6

CDI (JSR 299)
&
@Inject (JSR 330)

Inject just about anything anywhere...

...yet with strong typing

The tale of 2 dependency JSRs

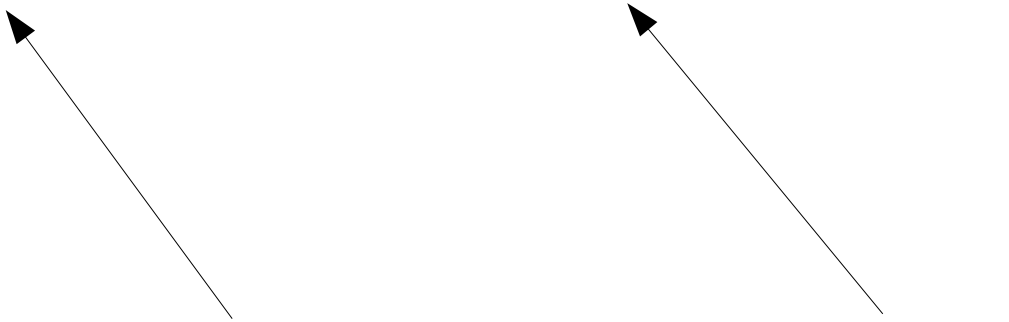
- Context & Dependency Injection for Java EE
  - Born as WebBeans, unification of JSF and EJB
  - “Loose coupling, strong typing"
  - Weld as the reference implementation, others to follow (Caucho, Apache)
- Dependency Injection for Java (JSR 330)
  - Lead by Google and SpringSource
  - Minimalistic dependency injection, `@Inject`
  - Applies to Java SE, Guice as the reference impl.
- Both aligned and part of Java EE 6 Web Profile

`@Inject`

- `javax.inject` package
- `@Inject` : Identifies injectable constructors, methods, and fields
- `@Named` : String-based qualifier (for EL)
- `@Qualifier` : Identifies qualifier
- `@Scope` : Identifies scope annotations
- `@Singleton` : Instantiates once

Injection

```
@Inject Customer cust;
```



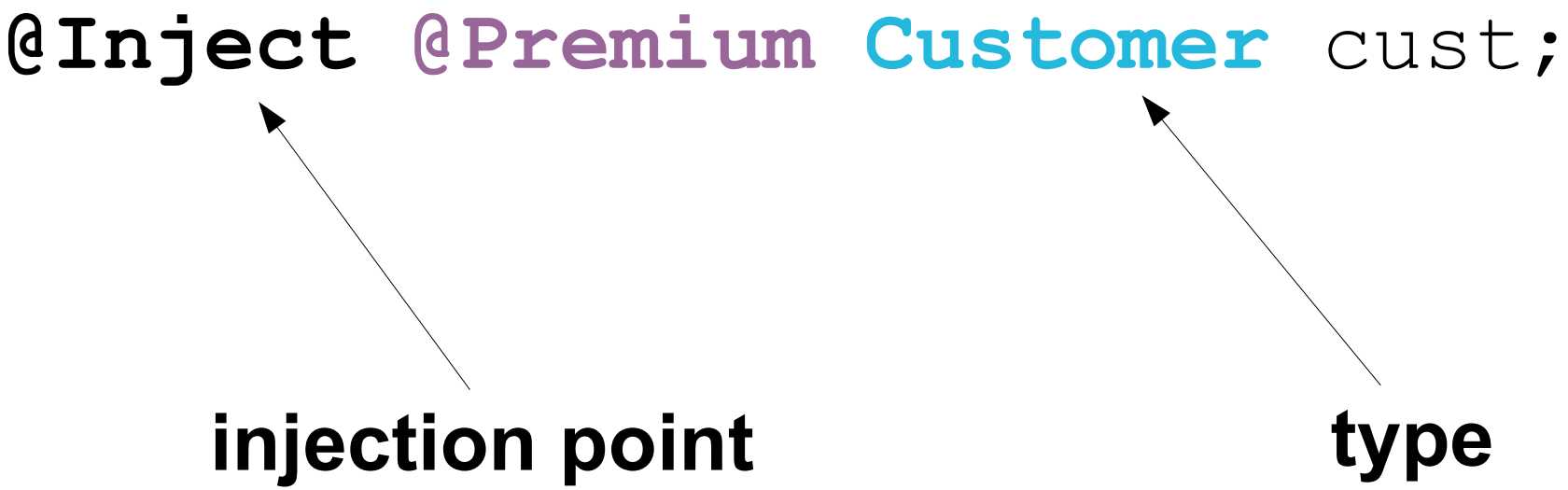
Qualifier Annotation

```
@Target({TYPE,METHOD,PARAMETER,FIELD})
@Retention(RUNTIME)
@Documented
@Qualifier
public @interface Premium {…}
```

```
@Premium  // my own qualifier (see above)
public class SpecialCustomer
                    implements Customer {
    public void buy() {…}
}
```

Injection with qualifier

qualifier (user-defined label)
i.e. « which one? »



Contexts (The 'C' in CDI)

- Built-in “Web” Scopes :
  - `@RequestScoped`
  - `@SessionScoped*`
  - `@ApplicationScoped*`
  - **`@ConversationScoped*`**
- Other Scopes
  - `@Dependent` is the default pseudo-scope for un-scoped beans *(same as Managed Beans)*
  - Build your own `@ScopeType`
- Clients need not be scope-aware

`*`: requires `Serializable` fields to enable passivation

@ConversationScoped

- A *conversation* is :
  - explicitly demarcated
  - associated with individual browser tabs
  - accessible from any JSF request

```
@Named
@ConversationScoped

public class ItemFacade implements Serializable {
  @Inject Conversation conversation;
  ...
  conversation.begin(); // long-running
  ...
  conversation.end();   // schedule for destruction
```

And more...

- Alternatives
  - `@Alternative` annotation on various impl.
- Interceptors & Decorators
  - Loosely-coupled orthogonal (technical) interceptors
  - `@Decorator` bound to given interface
- Stereotypes (`@Stereotype`)
  - Captures any of the above common patterns
- Events
  - Loosely-coupled (conditional) `@Observable` events

Agenda

- Quick overview
- New concepts
- New features on existing specifications
- New specifications
- Summary

Java EE 6 is...

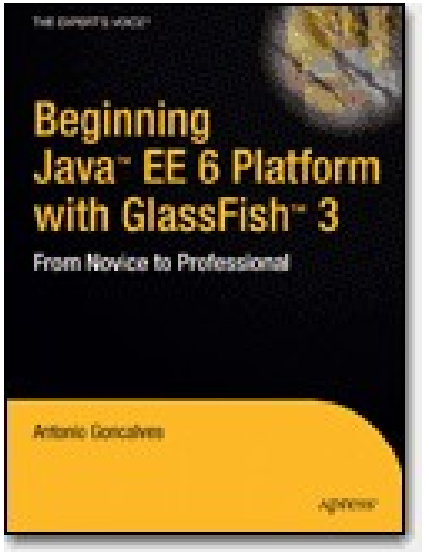
Richer
Ligther
Simpler

This is no science fiction

Java EE 6 and GlassFish v3 shipped final releases on December 10th 2009

Want to know more ?

- Covers most specs
- 450 pages about Java EE 6



"*The best book ever !*" – my sister
"*Nice photo at the back*" – my mum
"*Plenty of pages to draw on*" – my daughter

Thanks for your attention!

Q&A