183.223 Web Application Engineering & Content Management

Leitung: Univ.Prof. DI Dr. Thomas Grechenig

Pitfalls in Java EE

Mark Struberg

Kontakt: teaching@inso.tuwien.ac.at

About me

- Mark Struberg, INSO TU Vienna
- struberg@apache.org
- Apache Software Foundation member
- Committer / PMC for Apache OpenWebBeans, MyFaces, Maven, OpenJPA, BVal, Isis, DeltaSpike, commons, Geronimo, JBoss Arquillian, ...
- Java JCP Expert Group member
- Twitter: @struberg
- github.com/struberg

Agenda

- What fits your project best?
- Java EE spec parts
  - JPA-2, EL-2.2, JSF-2, Servlet-3.0, CDI, ...
- 3rd party Extensions and Libs
- Deployment, Clustering
- Apache Maven
- examples, examples, examples, ...
- Small example stub online available at https://github.com/struberg/lightweightEE

Basic Assumptions

- "If you have a hammer,
  every problem seems to be a nail"
- "Use the right tool for the right job"
- Every design decision has pros and cons!
  - Examples:
  - centralised vs de-centralised
  - statefull vs stateless

What do you like to build?

- Time To Market?
- Stable vs quickly changing project requirements.
- How long is the lifespan of your project?
- DSLs vs generic Programming Languages
- What is your team/partner capable of?
- Who can maintain that stuff?
- "Java is todays Cobol"
- Update Often vs Major Upgrades only

How does your data look like?

- How long is the lifespan of your data?
- Standard **SQL** (MySQL, Oracle, DB/2, PostgreSQL)
- **NoSQL** (Apache CouchDb, MongoDB, Cassandra)
- **BigData** (Apache Hadoop)
- Probably SQL + Lucene Search engine (via Solr)?
- **CAP** Theorem, Brewer:
  Consistency, Availability, Partition tolerance
  only 2 of them can be guaranteed at a time!
  - SQL DBs are 'ACID':
    Atomic, Consistent, Isolation, Durable
  - distributed DBs (NoSQL mostly) are faster which large data but not reliable

Open Source vs Closed Source

- finding failures and debugging
- fixing the bugs
  - workaround
  - fork
  - 'real' fix
- open mailing list archives and bug trackers
- long-time availability
- Why contribute to an Open Source project?
- professional support available for most projects

The License Situation

- Commercial Licenses
- GPL
- LGPL
- MIT
- ALv2
- Do you have a right on the sources at all?
  - 'Project' vs 'Product'

Lightweight Java EE?

- Java EE and lightweight?
- What are the Alternatives?
  - we will discuss alternatives for each part
- Spring?
  - How does a typical Spring App look like?

Java EE parts

- Java EE provides all-in-one
- But EE Components are usable standalone as well!
- Most EE Specs are independent of each other.

EE-6 Environments

- Defined in JSR-316

  http://jcp.org/aboutJava/communityprocess/final/jsr316/index.html
- Full EE-6
- EE-6 Web Profile
- Each server is allowed to add additional parts. But they must pass the TCK...
- EE-7 is just around the corner

EE-6 Full Profile

- EE5 +:
- New parts:
  - CDI (JSR-299)
  - Bean-Validation (JSR-303)
  - JAX-RS (JSR-311)
- + **Major** Upgrades to existing parts!
  - JSF-2, JPA-2, EJB-3.1, Servlet-3.0, EL-2.2, ...

EE-6 Web Profile

- EJB-3.1 light - no remoting, no @Asynchronous
- CDI-1.0
- JSF-2.0
- JPA-2.0
- JSR-303 Bean Validation
- Servlet-3.0 and EL-2.2
- JTA-1.1
- + 'Optional Components' (but hard to get approved)
- **No EAR support!** (some container provide it)

EE-6 DIY...

- Apache Tomcat-7 (or Tomcat-6 + JUEL)
- Apache MyFaces-2.1.x (or Oracle Mojarra)
- Apache OpenWebBeans (or JBoss Weld)
- Apache OpenJPA or JBoss Hibernate or EclipseLink
- Apache BVal (or Hibernate Validator)

EAR Container DIY

- Just package an EAR and unpack into Tomcats 'webapps' folder
- 1 Tomcat → 1 EAR Server
- Tomcat6 and 7 have a SharedClassLoader
  - disabled by default
  - can be enabled in conf/catalina.properties
  - with manually configured webapp Contexts:

```
shared.loader=${catalina.base}/webapps/lib/*.jar
```

  - with auto-deployment enabled:

```
shared.loader=${catalina.base}/applib/*.jar
```

- On Tomcat7 you need to place the shared lib folder outside of webapps because of Servlet-3.0 scanning

What about EE-7?

- CDI-1.1 - maintenance release
- **JMS-2.0** - major upgrade
- JSF-2.2 - maintenance release
- JPA-2.1 - maintenance release
- EJB-3.2 - maintenance release
- Servlet-3.1 - maintenance release
- **JCache** JSR-107 - finally! After 12 years! Or not?
- JJSON, JBatch
- Cloud? - beware!
- multi tenancy? - beware!

?

JPA-2

- Object Relation Mapping
- defined in JSR-317
- currently in process: JPA-2.1 as JSR-338
  - Stored Procedures, CDI support
- @Entity and how it maps to a database
- Relations: @OneToMany, @ManyToOne, etc
- How 'Enhancement' works
- The EntityManager
- performing inserts, updates, queries
- pitfalls, pitfalls, pitfalls, ...

JPA - Available Implementations

- JBoss Hibernate
- EclipseLink as RI
  (formerly Oracle TopLink)
- Apache OpenJPA
  (based on the SolarMetric Kodo codebase)
- Be careful what vendor and version your target environment uses!
- There are also different 'Operation Modes' for each of them..

JPA-2 - The Meta-Idea

- Object Relation Mapping
- Define your Data Model as Java Classes
- Generate the mapping of those Java Classes to their representation in the Database
- Vendor independent Query Language (**JPQL**)
- Alternatives:
  - Relation-Object-Mapping:
    - DODS
    - mybatis
  - Hibernate native
  - JDO (requires no 'relational' persistence)

JPA - Primary Keys and Foreign Keys

- indicated via @Id
- JPA also supports composite primary keys
- via @Embeddable and @EmbeddedId
- via @IdClass (quirks mode: matched by name)
- JPA manages Foreign Keys internally, you will only use the Object Class you link.

JPA - A simple Entity

- A JPA Entity is mapped to one or more DB tables.

```
import javax.persistence.*;
@Entity
public class Address {
  @Id
  @GeneratedValue
  private Long id;

  private String street;
  private int number;

  @Temporal(TemporalType.TIMESTAMP)
  private Date creationDate;

  @Column(length = 2000)
  private String comment;
  // + getter and setter
}
```

JPA - persistence.xml

- META-INF/persistence.xml
- Contains configuration of available PersistenceUnits
- PersistenceUnit contains the list of available Entities
- transaction-type JTA vs RESOURCE_LOCAL

```
<persistence-unit name="myPU" transaction-type="RESOURCE_LOCAL">
```

- On EE Servers:
  - <jta-data-source>, <non-jta-data-source>
- In Java SE *only* via properties (see spec 8.2.1.9)
  - `javax.persistence.jdbc.driver`
    `+ .url, .user, .password`

JPA - Alternative Configuration methods

- JNDI is not always available in Java SE
- The JNDI location for server configured DataSources is **not** specified.
  - Every server places it somewhere else.
  - No way to use a jndi:// URL in persistence.xml in a portable way
- Apache DeltaSpike ConfigurableDataSource + DataSourceConfig

JPA - Relations

- @OneToOne

```
public @Entity class Course {
  @OneToOne
  private Course linkedCourse;
```

- @OneToMany

```
public @Entity class Course {
  @OneToMany(mappedBy=”course”, fetch=EAGER)
  private List<Lecturer> lecturers;
```

- @ManyToOne
- @ManyToMany + @JoinTable
- @ElementCollection for lists of simple types

```
@ElementCollection
@OrderColumn(name = "POSITION")
@CollectionTable(name = "Course_precedingCourses")
private List<String> precedingCourseNumbers;
```

The problem with the Id

- @GeneratedValue @Id is hard to use for **equals()** and hashCode()!
- alternative: @EmbeddedId with UUID

```
java.util.UUID#randomUUID().toString();
```

- If you use generated Ids, always tune your Sequencers!
  - set your defaults in persistence.xml

```
<property name="openjpa.Sequence"
value="class-table(Table=SEQUENCES, Increment=20,
InitialValue=10000)"/>
```

  - configure per Entity via own @SequenceGenerator and @GeneratedValue

JPA - Optimistic Locking

- *Always* use **@Id** and **@Version** in your Entities!
- Internally the following happens:

```
update Xxx set dings="x", version = :oldversion+1
where id=.. and version = :oldversion;
```

- If you use DAO/DTO you need to check the @Version field yourself!
- JPA also knows pessimistic LockModes!

JPA - Entity Inheritance

- JPA supports inheritance for supertype-subtype relations
- Person ← Student, Lecturer, Employee,..
- @Inheritance on the base class
- 3 strategies
  - InheritanceType.SINGLE_TABLE
  - InheritanceType.TABLE_PER_CLASS
  - InheritanceType.JOINED
- @DiscriminatorColumn + @DiscriminatorType + @DiscriminatorValue

JPA - How Enhancement works

- At runtime Entity class might not be used natively
- Instead the Entity Classes might get 'Enhanced' via byte code engineering.
- Setter and Getter will track information about
  - _loaded : which parts of the entity got already lazily loaded
  - _dirty : which parts of the entity got changed via setters
  - safeEquals and only do updates if the content really changed

http://struberg.wordpress.com/2012/01/08/jpa-enhancement-done-right

JPA - Lazy Loading

- Only works on 'attached' Entities (not on 'detached')
- FetchType.LAZY is only a 'hint' and not a requirement
- Use @Basic(fetch=LAZY) for simple fields
  - works in OpenJPA and EclipseLink, gets ignored in Hibernate
- @OneToOne(fetch=LAZY) works only in OpenJPA and EclipseLink
- @ManyToOne(fetch=LAZY) works only in OpenJPA and EclipseLink
- @OneToMany(fetch=LAZY) ok in *all* JPA providers

JPA - Transaction Handling

```
EntityManagerFactory emf
    = Persistence.createEntityManagerFactory(“MyDb”);
EntityManager em = emf.createEntityManager();

em.getTransaction().begin()
try {
  // do some stuff

  em.getTransaction().commit();
}
catch (Exception e) {
  em.getTransaction().rollback();
}
finally {
  em.close();
}
```

JPA - The EntityManager 1/2

- The EntityManager is the central point to deal with JPA
- EntityManager is a 'cache' for instances.
- Managed (type Transaction or Extended) and Unmanaged EntityManagers
- In a CDI app you can use the Unmanaged EntityManager and do the management yourself via CDI Interceptors
- In EJB you only use Managed EMs

JPA - The EntityManager (2/2)

- 1 DB entry is always only 1 Entity instance in memory
- EntityManager allows to control the Transactions
- Not Concurrently usable! (CAP Theorem!)
- *attached* vs *detached* Entities
- EntityManager is NOT Serializable!

JPA - creating DB entries

- Just create a new Entity instance and persist it

```
Customer c = new Customer();
c.setFirstName("John");
c.setLastName("DontKnow");
em.persist(c);
```

- Take care about @OneToMany with CascadeType.PERSIST
  - If you use CascadeType.PERSIST then only em.persist() the parent entity
  - If you don't use it, you need to em.persist() each child yourself

JPA - selects with JPQL

- Find an Entry via it's primary key

```
Customer c = em.find(Customer.class, customerId);
```

- Select Entities via JPQL

```
Query q = em.createQuery(
  "SELECT c FROM Car c WHERE c.vendor=:vend”, Car.class);
q.setParameter("vend", someVendor);
List<Car> customers = q.getResultList();
```

- List for parts of an Entity

```
em.createQuery("SELECT new CarListEntry(c.type, c.vendor) .."
```

- JPQL always works via Prepared Statements thus prevents SQL-Injection
- Attention! A JPQL Select might cause temporary inserts/updates
- Take care about '%' and lists!

JPA - Updating Entities

- Just change an attached Entity. The changes will automatically get written into the database latest on EntityManager#flush().
- You need to use EntityManager#merge() to make the Entity attached if it is not yet managed.
- Attention, the new 'managed' Entity is the return value:

```
c1.setFirstName(“X”);
customer c2 = em.merge(c1);
c1.setFirstName(“Y”);  //     will not change anything
c2.setFirstName(“Z”);  //     this works.
```

JPA - 'Main Entities'

- Be careful when creating bi-directional Relations!
- 'Main Entities' will get touched for Optimistic Locking purpose
  - Most times according to Cascade.PERSIST and 'mappedBy'. See JPA spec 5.2.10.1
- Sometimes it's better to omit 'Entity Navigation' and use a 'Service' call instead

JPA / CDI - @Transactional

- Apache DeltaSpike @Transactional interceptor

```
@ApplicationScoped
@Transactional
public class CustomerServiceImpl {
  private @Inject EntityManager em;

  public boolean isCustomerActive(Long customerId) {
    Customer customer =
      em.find(Customer.class, customerId);
    if (customer != null) {
      customer.isActive();
    }
    return false;
  }
```

JPA EntityManager patterns

- entitymanager-per-invocation
  - default in EJBs (Container Managed Transactions)
  - LazyInitialisationException or NullPointerException
  - requires DTO/DAO pattern for your View!
- entitymanager-per-session
  - Store the EM in the users session
  - Doesn't work in Clusters as an EM is ***not*** Serializable
- entitymanager-per-conversation
  - same problems as with em-per-session
- entitymanager-per-request

JPA - entitymanager-per-request

- em.close() on end of each http request
- Be careful to not accidentally modify your *attached* entities in your View!
- Be careful with ad-hoc Transactions caused by lazy-fetching
- **Selects** might cause updates/inserts!
- In practice not that worse, because all Entities get detached at the end of a request.
- *Explicit* em.merge() before storing in a callback!

JPA / CDI - The EntityManager producer

- Create the injectable EntityManager via a producer method:

```
@ApplicationScoped
public class EntityManagerProducer {

  private EntityManagerFactory entityManagerFactory =
    Persistence.createEntityManagerFactory("CaroLine");

  @Produces @RequestScoped
  public EntityManager createEntityManager() {
    return entityManagerFactory.createEntityManager();
  }

  public void dispose(@Disposes EntityManager em) {
    em.close();
  }
}
```

JPA - Typesafe Query API

- CriteriaQuery via EntityManager#getCriteriaBuilder()
- tree based fluent Java API
- gets almost unreadable for non-trivial queries
- Use own QueryBuilder helper instead for dynamic queries (see CustomerServiceImpl)

JPA - Setup Tips

- Never let the JPA provider touch your DB-Schema!
  - build DDL/Schema upfront and review
  - use maven-sql-plugin in development
  - manage schemaDelta.sql semi-automatic
  - always check your Indices and do explains!
- Track slow Queries via a CDI-Interceptor!
- Attention with version=”2.0” vs version=”1.0” in persistence.xml!

?

JSR-303 Bean Validation

- Applying Constraints via Annotations
- One single framework for all application layers!
- Supports Java SE and EE
- Defines a few standard Constraints
- Defines how to trigger the validation
- Defines how to build own Constraints
- allows validation-groups
- Cross-field validation
- Constraint composition via meta-annotations

JSR-303 - built in Constraints

- @NotNull
- @AssertTrue, @AssertFalse
- @Size (for String length)
- @Max, @Min (for numbers)
- @DecimalMax, @DecimalMin
- @Digits
- @Past, @Future
- @Pattern (regular expression)

JSR-303 - own Constraints

- A Constraint Annotation has at least the following properties:
  - message
  - payload
  - groups
- meta-annotated with javax.validation.Constraint

JSR-303 - Constraint Example

- The Annotation

```
@Constraint(validatedBy={EmailValidator.class})
public @interface Email {
  Class<?>[] groups() default {};
  String message() default "{email.message}";
  Class<? extends Payload>[] payload() default {};
}
```

JSR-303 - Constraint Example

- The Implementation:

```
public class EmailValidator implements
  ConstraintValidator<Email, CharSequence> {
  public boolean isValid(CharSequence value,
      ConstraintValidatorContext context) {
    return EMailValidationUtils.isValid(value);
  }

  public void initialize(Email parameters) {
    // do nothing (as long as Email has no properties)
  }
}
```

JSR-303 - Framework Support

- Bean Validation support in the following modules:
  - JSF-2.x
  - JPA-2.x
    ```
    <validation-mode>AUTO</validation-mode>
    ```
  - EJB-3.1 via
    @Resource ValidatorFactory,
    @Resource Validator
  - CDI-1.0 via
    @Inject ValidatorFactory,
    @Inject Validator
- Define your own messages via a simple MessageBundle ValidationMessages.properties

JSR-303 - weak spots

- Should be even better integrated with JPA
  - @Size currently doesn't affect varchar width
  - @NotNull doesn't affect @Column(nullable=)
  - feature request got opened for JPA-2.1
- Class-level validation doesn't work well in JSF
  - field order in convert values / validate is not defined.
- Validations defined in a superclass or interface can only be overridden via XML configuration!

?

Servlet-3.0 - what's new

- defined in JSR-315
- Annotations, Annotations, Annotations
  - @WebServlet
  - @WebListener
  - @WebFilter
  - @WebInitParam
- asynchronous processing support
- web.xml now optional
- web-fragment.xml
- Programmatic configuration

Servlet-3.0 - web-fragment.xml

- each JAR can contain a META-INF/web-fragment.xml
- only JARs in WEB-INF/lib/ will be scanned!
  - tomcat ignores this rule ;)
- please define a <name> for your fragment
- Can be sorted in web.xml via:
  - <absolute-ordering> + <name> + <others>
    - Attention! if 2 fragments have the same name, later one will get ignored!
- Sorting hints in web-fragment.xml itself:
  - <ordering> + <after> + <before> + <others>

Servlet-3.0 - Performance boost

- Many specs know a 'metadata-complete' property to disable the whole scanning.
- Attention: if defined in web.xml this will also disable the scanning for META-INF/web-fragment.xml files!
- in WEB-INF/web.xml:
  ```
  <web-app version=”3.0” metadata-complete="true">
  ```
- Can be used in META-INF/beans.xml to disable scanning of a single JAR!
- Tomcat7 (since 7.0.27): catalina.properties
  ```
  jarsToSkip=*.jar
  ```
  - also helps with a tomcat performance issue in tc7 with META-INF/resources

Servlet-3.0 - Asynchronous Processing

- Allows better usage of servlet resources if a request gets blocked
- Allows separation of request and response
- All Servlets and Filters in the processing chain must support async processing!
- Thus badly supported yet...
- Attention with ThreadLocals!

Servlet-3.0 - Filters and Listeners

- Filters get added to the request invocation chain
  - doFilter(ServletRequest, ServletResponse, FilterChain)
  - chain.doFilter(request, response)
- Listeners receive callbacks on Servlet events
  - requestInitialized, requestDestroyed
  - sessionInitialized, sessionDestroyed
  - …
- new in Servlet-3.0: ServletContainerInitializer
  - META-INF/services/javax.servlet.ServletContainerInitializer

Servlet-3.0 - Example: LogFilter

- Sets the request encoding
- Performs logging + timing of 'real' requests
- 'NDC' handling: add SessionId and userId to every log message in the request
- Synchronizes on the Session to shield against JPA problems

Servlet-3.0 - the bad side

- increased startup time due to Annotation scanning
- harder to diagnose if something is misconfigured
- Always check what got effectively enabled!

?

EL-2.2

- Main feature: Method Invocations
  `#{bean.getInfoFor(user)}`
  - This is really a huge improvement for <h:datatable>
- Inspired by Seam2 (but without 'outjection')
- Can be plugged into tomcat6 as well (e.g. JUEL):
  http://wiki.apache.org/myfaces/HowToEnableEl22

EL-2.2 - own ELResolver

- e.g. for LocalizedResources

```
MyELResolver extends javax.el.ELResolver {
  public Object getValue(ELContext ..)
  public Class<?> getType(ELContext)
  public void setValue(ELContext..)
}
```

- If an ELResolver is responsible

```
elContext.setPropertyResolved(true); // or false if not...
```

- Register for JSF in faces-config.xml:

```
<application>
  <el-resolver>mycompany.MyELResolver</el-resolver>
```

- Be careful with performance issues!

EL-2.2 - issues

- coercing is not clearly defined
  - vargs, type conversions, ...
  - might be implementation depending
- http://juel.sourceforge.net/guide/issues.html

?

JSF-2 Basics - New Features 1/2

- Facelets as default VDL
- AJAX
- Common jsf.js library
- GET support
- Bookmarkable URLs
- PostRedirectGet
- Composite Components (improved in JSF-2.1)

JSF-2 Basics - New Features 2/2

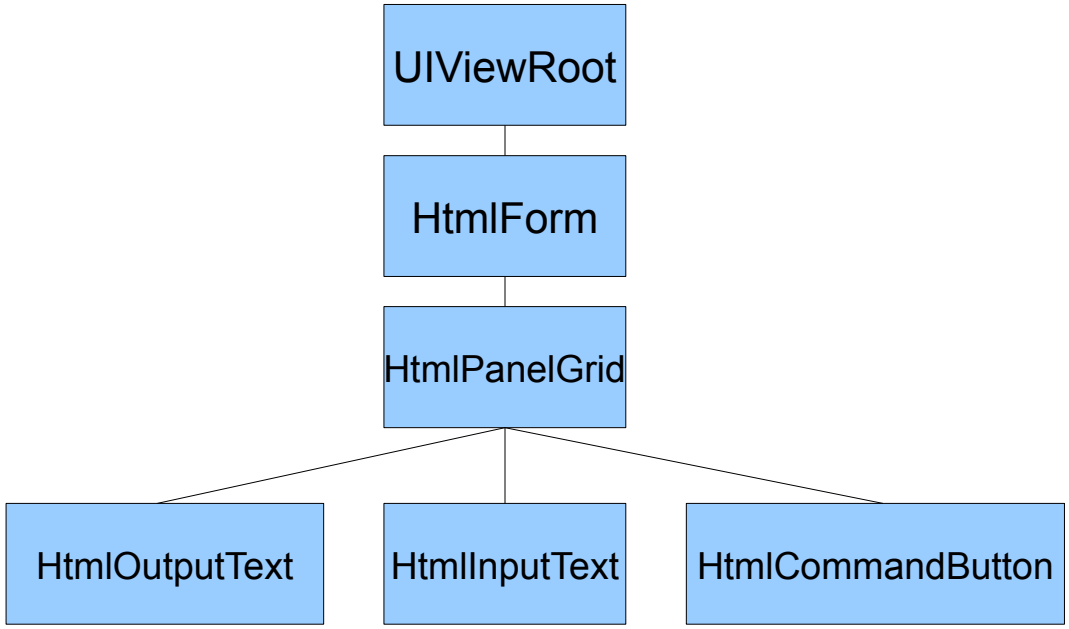
- Implicit Navigation
- No more XML
- Partial State Saving
- System Events
- ProjectStage
- JSR-303 Integration
- ViewScope

JSF-2 - basic setup

- No more JSP!
- Use ***.xhtml as** <servlet-mapping> for FacesServlet
- Increase NUMBER_OF_VIEWS_IN_SESSION
- Tune your JSF Implementation parameters:
  - http://myfaces.apache.org/core21/myfaces-impl/webconfig.html
- use <error-page>
- ProjectStage.Development vs ProjectStage.Production

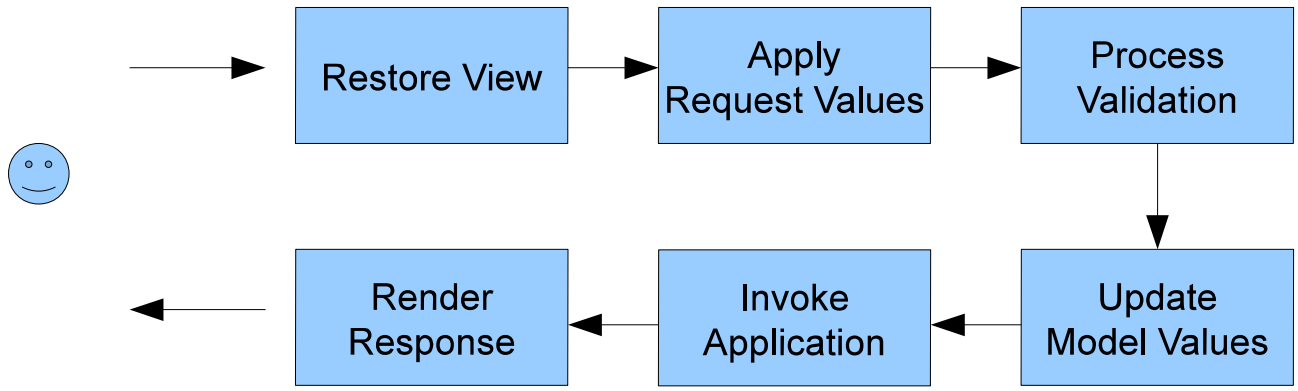
JSF-2 Basics - The Component Tree

- Each Component has Data, Controller, Renderer
- Independent of the VDL



JSF-2 Basics - The JSF Lifecycle

- JSF is component oriented and stateful



JSF-2 Basics - Templating with Facelets

- <ui:composition template=..> allows to define a common template
- place embedded Facelet parts in WEB-INF/templates
- Facelet resources get only looked up in the webapp, not on the classpath.
  - http://ocpsoft.org/opensource/create-common-facelets-jar
- Define a few 'standard' blocks for your app:
  - 'metadata', 'header', 'title', 'content', ...
- Attention: <f:metadata> must be direct child of the View Root!

```
<ui:insert name="metadata"/
```

JSF-2 Basics - Simple skins with Facelets

- Facelet templates are a candidate for implement skinning
- Changing the body style

```
<h:body class="#{(skin==null) ? 'lehre' : skin}
#{settings.locale.language}">
```

- or even swapping CSS

```
<h:head>
  <c:if test=”#{tenant eq 'A'}”>
    <link href="#{resource['css:theme_a.css']}"
     rel="stylesheet" type="text/css" media="all"/>
  </c:if>
  ..
</h:head>
```

JSF-2 - GET support

- View Parameters
- PreRenderViewEvent

```
<f:metadata>
  <f:viewParam name=”userId” value=”#{userForm.id}”/>
  <f:event type=”preRenderView”
           listener=”#{userForm.load}”/>
</f:metadata>
```

- Bean listener method signature:

```
public void load(ComponentSystemEvent ev) {
  ...
}
```

- new components: <h:link>, <h:button> + <f:param>

JSF-2 - MyFaces ResourceHandler

- Default Resource Handling of JSF-2 is bad
  `http://myapp/javax.faces.resource/someimg.png?ln=en`
- Not easily cachable (EL in resource files...)
- Default cache expiry 14 days
- Problematic with JavaScript and CSS updates
- Will get fixed in JSF-2.2 (I hope, no progress so far)
- Solution: Apache MyFaces Commons ResourceHandler

https://cwiki.apache.org/confluence/display/MYFACES/MyFaces+Commons+ResourceHandler

  - Restful URLs, versioning, branding layers

JSF-2 - Back-Button handling

- JSF is stateful, but serializes View into the Session
- Server Side State-Saving vs
  Client Side State-Saving
- Most times the own Data is the problem
- Recommended to disable caching for most pages
  - e.g. via a CacheControlPhaseListener in before-render-response
- PRG + CODI @RestScoped for single pages

```
?faces-redirect=true
```

JSF-2 - Double Submit prevention

- jsf.js Queues serializes AJAX requests, but not normal POST and GET
- in LogFilter and return HTTP-304 (NOT_MODIFIED) for 2nd request for GET
- via JavaScript ondblclick="return -1"
  - not reliable
- via PhaseListener and token consumption in after-restore-view
- delayed post queue
  http://werpublogs.blogspot.co.at/2011/07/apache-myfaces-jsfjs-queue-control.html
- might finally get targeted with JSF-2.2

JSF - Hiding Components

- <... *rendered="#{do.some.eval}"*> is expensive!
  - Gets evaluated up to 6-times per request
- <c:if>, <c:forEach> is problematic on AJAX pages
  - c:if, c:forEach changes the view-tree
  - view-tree must not be changed during view lifetime
  - f:ajax
- Difference between <c:forEach> and <ui:repeat>!
- For larger blocks you can use:

```
<h:panelGroup rendered="#{not empty somebean.list}”>
  <h:outputText...
</h:panelGroup>
```

  - This automatically checks divs/tables

JSF - Resource Text

- Define the following in your faces-config.xml:
  - <locale-config> for setting default Locale for each View
  - <message-bundle> for error messages
  - <resource-bundle> used for i18n text pages (via EL)
- JSF provides a built-in ELResolver for them
- ???missing_text???

JSF - AJAX 1/4

- AJAX: Asynchronous JavaScript And XML
  - submit parts of the page and
  - replace parts of the page
  - without replacing the whole site
- Usually AJAX requires to write tons of JavaScript
  - JSF hides this complexity
- <f:ajax> deprecates old component-lib specific hacks
- It's now possible to mix JSF Component Libs

JSF - AJAX 2/4

- Example page:

```
<h:commandLink value="#{txt.searchOpts}"
        actionListener="#{courseList.switchSearchMode}"
        rendered="#{not courseList.extendedSearchOptions}">

  <f:ajax render="@form globalMessagesPanel"
          execute="searchField" />

</h:commandLink>

<h:panelGroup id="extSearchPanel"
          rendered="#{courseList.extendedSearchOptions}">
  <li>
    <h:outputLabel for="title" value="#{txt.courseTitle}"/>
…
</h:panelGroup>
```

JSF - AJAX 3/4

- Sample backing bean:

```
@ViewScoped
@Named
public class CourseList {
  private boolean extendedSearchOptions = false;
  …
  public void switchSearchMode(ActionEvent e) {
    extendedSearchOptions = !extendedSearchOptions;
    if (extendedSearchOptions && search != null) {
      courseNumber = search;
    } else {
      resetSearchParams();
    }
  }
}
```

JSF - AJAX 4/4

- <f:ajax> attributes
  - event: when should the AJAX trigger
  - execute: list of IDs to execute in the JSF lifecycle, and/or @this (default), @form, @all, @none
  - render: list of IDs to re-render, and/or @this, @form, @all, @none (default)
  - listener: method to invoke for fields, ...

JSF - ELResolver speedup

- If many components access a deep nested structure

```
<h:outputText value=”#{bn.usr.ctrct.addr.firstname}”/>
<h:outputText value=”#{bn.usr.ctrct.addr.lastname}”/>
<h:outputText value=”#{bn.usr.ctrct.addr.zip}”/>
…
```

- you can replace it with a <ui:include>

```
<ui:include src="/addressblock.xhtml">
  <ui:param name="add" value="#{bn.usr.ctrct.addr}"/>
</ui:include>
```

  - in addressblock.xhtml:

```
<ui:composition xmlns...>
  <h:outputText value=”#{add.firstname}”/>
</ui:composition>
```

JSF - the 'immediate' problem

- immediate not only bypasses the validation phase, but also the update-model-values!
- Better to temporarily disable the validation:

```
<f:validateBean disabled="#{!empty param['check']}">
..
<h:commandButton action="#{model.doSearch}">
  <f:param name="check" value="false"/>
</h:commandButton>
```

JSF-2 - Performance

- JSF-2 is actually really fast!
- MyFaces is faster than Mojarra :)
- Faster as wicket-1.5, much faster as wicket-1.6
- As fast as wicket-1.4
- Almost as fast as nacked SpringMVC (which does no validation/conversion, etc)
- partial state saving helped a lot
- tune your JSF parameters!
  http://myfaces.apache.org/core21/myfaces-impl/webconfig.html

JSF-2.2 Preview: Browser Tab handling

- Will come as part of JSF-2.2
- Maintain view-state per Browser Tabs
- Already available through Apache CODI ClientSideWindowHandler
- JSF-2.2 will introduce SPI for handling 'state per window'
- Until then: increment the number of saved states in session

?

CDI

- Contexts and Dependency Injection for Java EE
  - but also usable in Java SE and standalone
- Defined in JSR-299
- Incorporates JSR-330 ('atinject')
- Purely Annotation based
  - no XML configuration out of the box
  - but you can simply write your own
- Can fully replace EJB and Spring as Component Model

CDI - Features

- Typesafe Dependency Injection
- Interceptors
- Decorators
- Events
- SPI for implementing “Portable Extensions”
- Unified EL integration
- Standard Scopes

CDI - Extensions

- JSR-299 defines a comprehensive SPI
- All you can do via Annotations can also be done via Extensions
- CDI Extensions are portable
- and easy to write!
- Are activated by simply dropping them into the classpath
- CDI Extensions are based on java.util.ServiceLoader Mechanism

CDI-Extension Projects

- JBoss Seam3
  - Successor of Seam2
  - But not compatible ...
- Apache MyFaces CODI
  - Based on some ideas of MyFaces Orchestra
  - and lots of new stuff
  - used in many productive projects!
- Apache DeltaSpike
  - the future!
  - Coordinated effort of JBoss and ASF
  - work in progress

?

JSF-2 - Component Libraries

- JSF provides a rich SPI for replacing default functionality. Look for all the Wrapper stuff
- The Component Lib must support JSF-2!
- E.g. PrimeFaces highlights:
  - <p:watermark>
  - <p:dialog>, <p:confirmDialog>
  - <p:calendar> (date-, time-picker, range, …)
  - <p:tooltip>
  - <p:pickList>
  - <p:dataTable> (client and server side sorting/filtering)
  - <p:tabView>, <p:accordionPanel>,..
  - <p:schedule>

AppTricks - Unit Test setup

- Use DeltaSpike CdiContainer to boot CDI
- clean the instances between tests by stopping and restarting all Contexts

Arquillian

- JBoss Testing Effort
- Define @Deployment for your unit test
- Inplace- and Container-tests
- Run your unit tests on the real server
- Needs an Arquillian connector for your container
- Inplace Container leaks Resources into your test.

?

EE-6 Containers

- GlassFish
  - has problems with 'BDA'
  - only use latest version
- JBossAS-7.1.2.Final
  - pretty fine! Actually much better then AS6!
- WebSphere
  - ok, but contains an old OWB version
- WebLogic
  - contains an old Weld version
  - has problems with 'BDA'
- TomEE
  - really fine and fast!

Clustering

- We use 'asymmetric clustering'
- use sticky sessions
- backup away the session to a memcached after each request
- do not replicate the session over to other nodes!
- only restore the session from the memcached if a fail-over happened
- msm can be integrated into OWB:
  `http://code.google.com/p/memcached-session-manager/`

?

Apache Maven

- Apache Maven is a build management suite
- Ant is 'imperative', Maven is 'declarative'
- no libraries in your project anymore
- dependency management
- jar versioning
- convention over configuration – 'best practice' approach
  - default build lifecycle
  - default project structure
- use plugins – don't repeat yourself!

Maven - The default build lifecycle

- validate
- compile
- test
- package
- integration-test
- verify
- install
- deploy
- + pre and post phases most times

Maven - Default directory layout

```
pom.xml
src
    main
        java
        resources
        webapp
    it
    site
    test
        java
        resources
```

Maven - Project Sushi

- how to slice your project modules is very important!
- each jar, war, ear, ... always gets an own module
- creating jars is cheap
- adding jars as dependencies is even cheaper
- separate your project into logical parts
- modules can form a tree

Maven - Plugin Goodies

- maven-help-plugin
- useful utility plugin to get informations about your current build
  - $> mvn help:effective-pom
  - $> mvn help:all-profiles
  - $> mvn help:effective-settings

Maven - Plugin Goodies

- maven-dependency-plugin
  - $> mvn dependency:list | less
  - $> mvn dependency:tree | less
  - $> mvn dependency:copy
  - $> mvn dependency:copy-dependencies
  - $> mvn dependency:go-offline
  - $> mvn dependency:sources
  - $> mvn dependency:unpack

Maven - Plugin Goodies

- versions-maven-plugin
  - helps with showing upgrade paths to new plugins and dependencies
  - $>mvn versions:display-dependency-updates
  - $>mvn versions:display-plugin-updates
  - $>mvn versions:lock-snapshots
  - $>mvn versions:use-releases

Maven - Repository Managers

- mandatory for Companies!
- Multiple Products available:
  - Apache Archiva
  - Sonatype Nexus
  - JFrog Artifactory
- "mirroring" by disabling the 'central' repository
- **Always** have a backup of your repo manager storage
- There have been artifacts disappearing from public repos already...

Maven - Releasing with Maven

- $> mvn release:prepare
  - -> test
  - -> check that all SNAPSHOTS got removed
  - -> checks that no modifications are dirty
  - -> tags in the SCM
- $> mvn release:perform
  - -> does a clean checkout into target/checkout
  - -> runs the whole build
  - -> tests again
  - -> optionally signs the packages
  - -> by default builds the site
  - -> deploys to the <distributionManagement> locations

Documentation

- JSR-299 spec:
  - http://jcp.org/en/jsr/detail?id=299
  - http://jcp.org/en/jsr/detail?id=330
  - http://jcp.org/en/jsr/detail?id=346
- OpenWebBeans:
  - http://openwebbeans.apache.org
  - svn co \
    http://svn.apache.org/repos/asf/openwebbeans/trunk
- CODI
  - https://cwiki.apache.org/confluence/display/EXTCDI
  - svn co \
    http://svn.apache.org/repos/asf/myfaces/extensions/cdi/trunk

Legal Notes

- Apache, OpenWebBeans, MyFaces, CODI, Tomcat, BVal, OpenEJB and OpenJPA are trademarks of the Apache Software Foundation
- Seam3, Weld and JBossAS are trademarks of RedHat Inc
- CanDI is a trademark of Caucho Inc