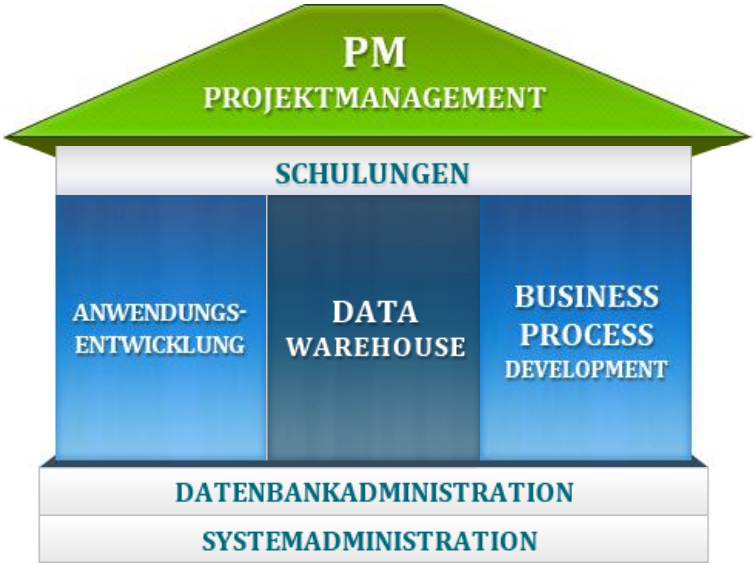
syntegris information solutions

DOAG 2012

Fallstricke bei der Entwicklung einer JEE6 Anwendung

Moritz Lippe

Firmenprofil



- gegründet am 01. September 2000
- Sitz in Neu-Isenburg bei Frankfurt am Main
- Mitarbeiter: 30
- Schwerpunkte:
  - Data Warehouse
  - Business Process Development
  - Anwendungsentwicklung
  - Administration Oracle / Solaris
  - Schulung
- Kontakt:
  Telefon (0 61 02) 29 86 68
  Email info@syntegris.de
  Fax (0 61 02) 55 88 06
- Besuchen Sie unseren Stand!

syntegris information solutions

Inhalt

1. Ein neuer Technologiestack soll her
2. Wie soll das funktionieren?
3. Fallstricke
4. Migration nach JBoss Application Server
5. Fazit

Ein neuer Technologiestack soll her

Die Auswahl eines Frameworks zur Entwicklung einer modernen Webanwendung ist im Java-Bereich alles andere als eine leichte Entscheidung. Es gibt unzählige Open Source und proprietäre Frameworks, aus denen man wählen kann.

- Ziel sollte natürlich immer die passende Technologie für das aktuell anliegende Projekt sein.

Ein neuer Technologiestack soll her

Die Entscheidungskriterien, welche Technologie man einsetzen kann, bestimmen sich häufig aus weiteren Punkten:

- Eine Technologie wurde bereits bei einem vorhandenen Projekt eingesetzt und soll auch für das neue Projekt verwendet werden.
- Die Projektanforderungen sehen eine bestimmte Technologie vor (z.B. im Lastenheft oder in der Ausschreibung vorgegeben).
- Ein Entscheidungsträger macht eine strikte Vorgabe.

Ein neuer Technologiestack soll her

Der Erfolg eines Projektes wird auch maßgeblich durch die Technologie-Auswahl beeinflusst. Hier stellen sich bei der Technologie-Auswahl folgende Fragen:

- Wie komplex und zeitintensiv gestaltet sich die Umsetzung einer Anforderung?
- Kann man alle Anforderungen mit der Technologie abbilden?
- Wie zukunftssicher ist die Technologie?

Ein neuer Technologiestack soll her

- Wie sehen Dokumentation und Support aus?
- Findet man im Internet Beispiele (gerade bei Open Source sehr wichtig)?
- Ist die Technologie im Markt weit verbreitet und findet man dazu qualifizierte Entwickler?

syntegris information solutions

Ein neuer Technologiestack soll her

Java Platform, Enterprise Edition Version 6

- Versprochener Mehrwert: schnellere und einfachere Entwicklung von Enterprise-Anwendungen im Gegensatz zu früheren Versionen
- Erste Vorträge und Artikel zu JEE6 hörten sich vielversprechend an
- JEE 5 Know-how war bereits vorhanden

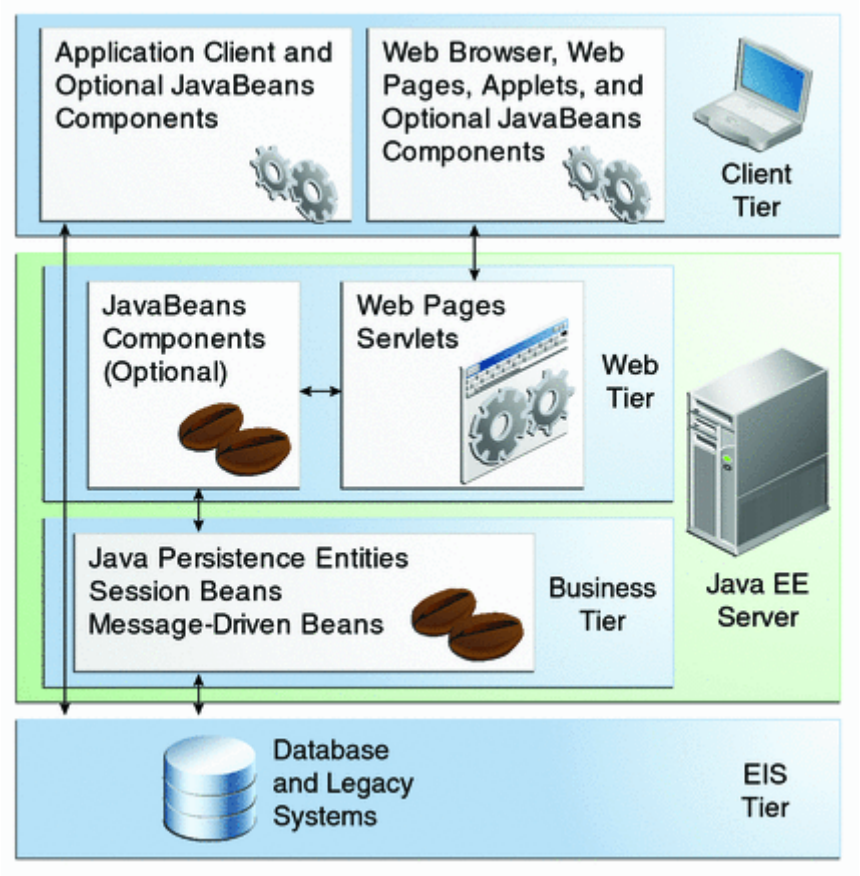
syntegris information solutions

Ein neuer Technologiestack soll her

Technologien (Auszug)

- Java Servlet 3.0 (JSR 315)
- JavaServer Faces 2.0 (JSR 314)
- Contexts and Dependency Injection for Java (JSR 299)
- Bean Validation 1.0 (JSR 303)
- Enterprise JavaBeans 3.1 (JSR 318)
- Java Persistence 2.0 (JSR 317)

Ein neuer Technologiestack soll her



Quelle: The Java EE 6 Tutorial, Distributed Multitiered Applications, Oracle http://docs.oracle.com/javaee/6/tutorial/doc/

syntegris information solutions

Ein neuer Technologiestack soll her

Fragen, die wir uns am Anfang gestellt haben

- Was gibt es für Neuerungen? Sind dies sinnvolle Neuerungen?
- JEE6-Stack oder Spring-Stack?
- Kann man überhaupt schon eine produktive Anwendung entwickeln?
- Welche Application Server unterstützen den neuen Standard?

Wie soll das funktionieren?

Evaluation

Eine Evaluationsphase sollte zeigen, dass die Erwartungen an JEE6 erfüllt werden können.

Eingesetzte Technologien:

- GlassFish 3.1 Server

- ICEfaces als JSF-Komponentenbibliothek

- Hibernate als JPA-Provider

Wie soll das funktionieren?

Erste Probleme

- Oft waren nur Beispiele auf „HelloWorld-Niveau“ zu finden
- Dokumentation fehlte (Beispiel: @Singleton)
- JEE6 Referenzimplementierungen wurden unterschiedlich schnell fertig entwickelt (siehe JSF2 & CDI, @ManagedBean / @Named)
- Debuggen? GlassFish Source-Dateien und JavaDoc waren nur schwer zu finden (lagen nicht im Maven Repository)

Wie soll das funktionieren?

Erste Probleme

- Automatisierte Tests?
- Was sollte man nicht mehr verwenden?
- Hot-Redeployment? Verbuggt mit Eclipse + GlassFish 3.1…

Wie soll das funktionieren?

Hilfe

- Besuch von Konferenzen
- Sehr guter Blog von Adam Bien: http://www.adam-bien.com/roller/abien/
- Fachliteratur (Magazine & Bücher)

syntegris information solutions

Fallstricke

Bei der Entwicklung der JEE6-Anwendung sind wir in einige Fallstricke gelaufen, die nachfolgend erläutert werden:

1. Falscher CDI-Scope
2. CDI und JSF 2
3. Conversation-Scope
4. JSF 2 und Ajax

Falscher CDI-Scope

- Ein Problem war die Verwendung eines „falschen“ Scopes für die @Produces-Methode einer Konfigurationsklasse.
- Die KonfigurationsBean ist eine Singleton-EJB, die mittels der @Produces-Annotation eine Konfiguration für eine JMS-Verbindung zur Verfügung stellt.

Problem: die @Produces-Methode verwendet einen Scope, der im Consumer nicht verwendet werden kann. Dies hatten wir erst durch eine NullPointerException zur Laufzeit gemerkt.

Falscher CDI-Scope

```
@Singleton
@Startup
public class KonfigurationsBean {
...

  @Produces
  @SessionScoped
  public JmsObjectWrapper getJmsObjectWrapper() {
    initRefreshJmsObjectWrapper();
    return jmsObjectWrapper;
  }
}
```

Falscher CDI-Scope

In der Consumer-Klasse wird das Konfigurations-Objekt verwendet. Hier wird mittels der CDI-Annotation @Inject das Objekt injiziert.

```
@Stateless
public class Consumer implements Serializable {

  @Inject
  JmsObjectWrapper jmsObjectWrapper;

...
}
```

CDI und JSF 2

- Mit JSF 2 ist es möglich beliebige Objekte in Listenwerten zu verwenden, indem man einen passenden Converter dazu schreibt, der die Konvertierung String/Objekt realisiert.
- In älteren Versionen konnten immer nur Strings verwendet werden oder man verwendete ein zusätzliches Framework wie etwa JBoss Seam, das einem die Konvertierung erleichterte.
- Als Beispiel wird ein Konverter für eine Liste von Staaten erzeugt. Die Daten kommen dabei aus der Datenbank.

Fallstricke

CDI und JSF 2

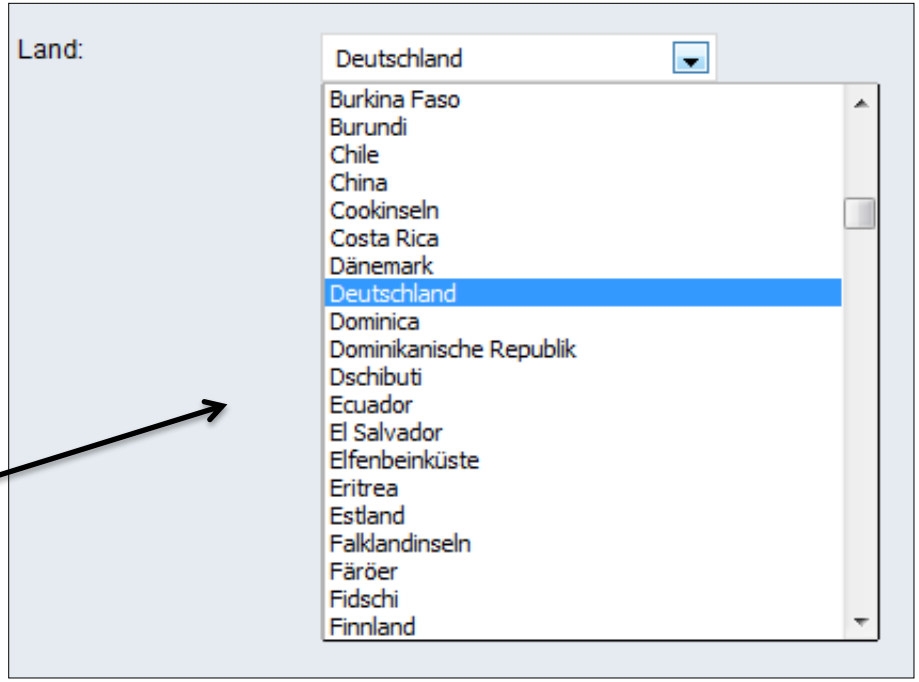
STAMMDATEN_STAATEN

Spalten | Daten | Constraints | Zugriffsberechtigungen | Statistiken | Trigger | Flashback | Abhängigkeiten | Details | Partitionen | Indizes | SQL

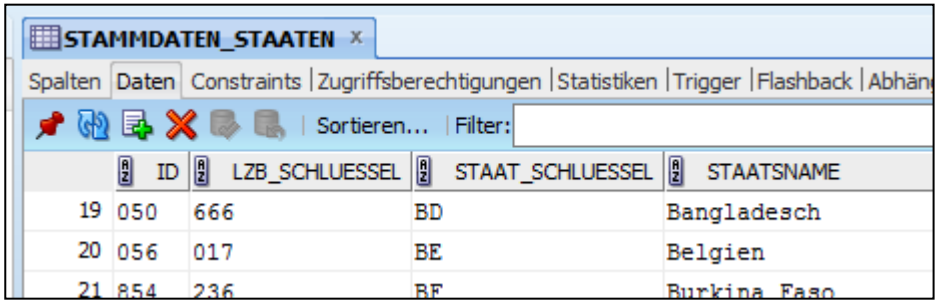
Sortieren... Filter:

	ID	LZB_SCHLUESSEL	STAAT_SCHLUESSEL	STAATSNAME
19	050	666	BD	Bangladesch
20	056	017	BE	Belgien
21	854	236	BF	Burkina Faso
22	100	068	BG	Bulgarien
23	048	640	BH	Bahrain
24	108	328	BI	Burundi
25	204	284	BJ	Benin
26	060	413	BM	Bermuda
27	096	703	BN	Brunei Darussalam
28	068	516	BO	Bolivien
29	076	508	BR	Brasilien
30	044	453	BS	Bahamas
31	064	675	BT	Bhutan
32	074	892	BV	Bouvetinsel
33	072	391	BW	Botswana
34	112	073	BY	Weißrussland
35	084	421	BZ	Belize

Datenbank



xhtml-Seite



CDI und JSF 2

Die Liste an Staaten besteht dabei aus JPA-Entities:

```
@Entity
@Immutable
@Table(name = "STAMMDATEN_STAATEN")
public class StaatenEntity implements Serializable {

  @Id
  @Column(name = "id", nullable = false)
  private String id;
...
}
```

CDI und JSF 2

Die Liste wird in der xhtml-Seite über ein Value-Binding verwendet:

```
<h:selectOneMenu converter="staatenConverter"
                 value="#{antragsteller.adresse.land}">
    <f:selectItems value="#{kundenDaten.landSelectItems}" var="land"
                   itemLabel="#{land.staatsname}" />
</h:selectOneMenu>
```

CDI und JSF 2

Der StaatenConverter muss hierbei einen String in eine Entity umwandeln (und umgekehrt). Eindeutig identifizieren kann man die Entity über das „id“-Feld:

```
@FacesConverter(value = "staatenConverter", forClass =
StaatenEntity.class)
public class StaatenConverter implements Converter {

  @Override
  public String getAsString(FacesContext ctx, UIComponent c, Object v){
    if (v == null) {
      return "";
    }
    return ((StaatenEntity) v).getId();
  }
...
}
```

CDI und JSF 2

Bei der Umwandlung von String zu einem StaatenEntity-Objekt, muss man sich überlegen, wie man das passende Objekt zu dem gegebenen String erhält. Da man die ID hat, kann man über einen passenden Service das Objekt aus der Datenbank auslesen.

- Jetzt könnte man auf die Idee kommen per CDI den Service in den FacesConverter zu injizieren und in der getAsObject()-Methode entsprechend zu verwenden.

CDI und JSF 2

```
@Inject IStaatenDaoService staatenDAOService;

@Override
public Object getAsObject(FacesContext ctx, UIComponent c, String v) {
  if (!StringUtils.isEmpty(v)) {
    return staatenDAOService.findEntityById(v);
  }
  return null;
}
```

Problem: Leider funktioniert CDI innerhalb eines FacesConverter nicht. Auch die Nutzung der EJB-Annotation @EJB nützt hier nichts.

Conversation-Scope

- Eine Neuerung in JEE6 ist der Conversation-Scope.
- Der Conversation-Scope ist kürzer als der Session-Scope aber länger als der Request-Scope. Er ist vor allem für die Abbildung eines Seitenflusses/Wizard gedacht.
- Die Conversation wird mittels eines Request-Parameters identifiziert und kann programmatisch gestartet und gestoppt werden.

```
@Named(KundendatenBean.BEAN_NAME)
@ConversationScoped
public class KundendatenBean extends AbstractBean
```

Conversation-Scope

- Mit JSF 1.x war die Nutzung des Conversation-Scopes über Frameworks wie beispielsweise JBoss Seam oder Apache MyFaces Orchestra möglich.

➢ Mit JEE6 wurde nun ein neuer Conversation-Scope in den Standard integriert.

Problem: bei der Verwendung des Conversation-Scopes merkten wir schnell, dass die Idee zwar gut ist, die Umsetzung jedoch unzureichend.

Conversation-Scope

Der JEE6 Conversation-Scope hat im Gegensatz zu den Conversation-Scopes anderer Frameworks folgende Nachteile:

- Es gibt keine „nested Conversations“, die Kind-Conversations darstellen. Kind-Conversations sind äußerst hilfreich, wenn man komplexere Seitenflüsse realisieren will.
- Es gibt keine Möglichkeit die Conversation an die Navigation anzubinden, wie es z.B. bei JBoss Seam der Fall ist. Dadurch wird das Starten und Stoppen der Conversation umständlich.

JSF 2 und Ajax

- Mit der Version 2 wurde die Ajax-Unterstützung in den JSF-Standard übernommen. Das Standard-Tag f:ajax reichert eine JSF-Komponenten mit Ajax an.
- Eine Standardaufgabe für die Ajax verwendet wird, ist die Neudarstellung einzelner Seitenfragmente. Dieses „Neurendern“ wird durch das JSF-Attribut „render“ realisiert.

JSF 2 und Ajax

Hierbei gibt man die Komponente(n) an, die man nach dem Auslösen des JavaScript-Events neu darstellen will:

```
<h:inputText id="betrag" value="#{konto.betrag}">
  <f:ajax event="change" execute="@this" render="@this otherComponent"/>
</h:inputText>
```

In diesem Beispiel wird neben dem Eingabefeld „betrag“ zusätzlich eine Komponente mit der ID „otherComponent“ neu gerendert, sobald das JavaScript-Event „onchange“ ausgelöst wird.

JSF 2 und Ajax

Hierbei gibt es nun eine Besonderheit, auf die man achten muss:

- Es ist nicht möglich eine Komponente neu darzustellen, die aktuell nicht im Komponentenbaum dargestellt wird. D.h. sobald die Komponente „otherComponent“ eine „rendered“-Bedingung hat (und initial nicht dargestellt wird), muss man ein Vaterelement dieser Komponente neu rendern, das bereits im Komponentenbaum enthalten ist.

JSF 2 und Ajax

Das folgende Beispiel funktioniert also nicht. Die Komponente „otherComponent“ wird nicht korrekt neu dargestellt, sofern die Rendered-Bedingung initial „false“ liefert:

```
<h:inputText id="betrag" value="#{konto.betrag}">
  <f:ajax event="change" execute="@this" render="@this otherComponent"/>
</h:inputText>

<h:outputText id="otherComponent" value="{bean.text}"
         rendered="#{bean.otherComponentRendered} />
```

JSF 2 und Ajax

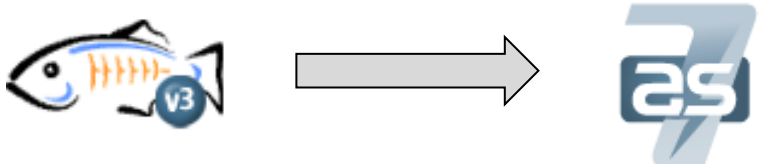
Das Problem behebt man, indem die Komponente eine zusätzliche Vater-Komponente erhält. Nun wird die Vater-Komponente neu dargestellt:

```
<h:inputText id="betrag" value="#{konto.betrag}">
  <f:ajax event="change" execute="@this" render="@this parent"/>
</h:inputText>


<h:panelGroup id="parent">
  <h:outputText id="otherComponent" value="{bean.text}"
          rendered="#{bean.otherComponentRendered} />
</h:panelGroup>
```

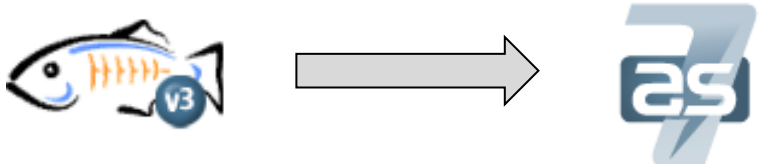
syntegris information solutions

Migration nach JBoss



- Zu Beginn des Projektes gab es eigentlich nur den GlassFish Application Server, der zur Entwicklung und für den Betrieb einer JEE6-Anwendung verwendet werden konnten.
- Seitens des Auftraggebers schon recht früh der Wunsch auf, die Anwendung auf dem JBoss Application Server zu installieren.
- Da dies zu Projektstart nicht möglich war (der JBoss Application Server war zu dem Zeitpunkt noch nicht JEE6 zertifiziert), wurde die Migration nach hinten verschoben. Ein nicht unerhebliches Problem, wie sich am Ende herausstellte…

Migration nach JBoss

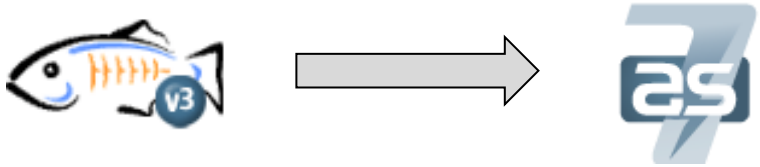


- Der JBoss Application Server ist seit der Version 7.1 Java EE6 zertifiziert.
- Die Implementierungen der einzelnen JEE6 Bestandteile sind größtenteils deckungsgleich mit den Implementierungen, die der GlassFish Application Server nutzt.

Erste Test-Deployments ließen schnell die Hoffnung auf eine problemlose und schnelle Migration schwinden:

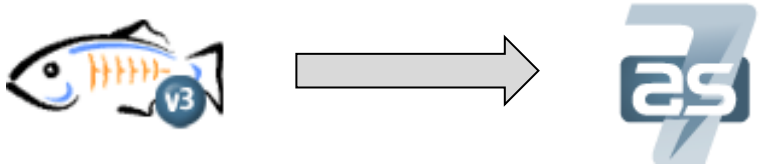
- Der JBoss Application Server hat eine striktere Prüfung der JEE6-Konformität.
- Der JNDI-Lookup wird vom JBoss AS anders realisiert als im GlassFish.

Migration nach JBoss



- Die JSF-Anwendung verweigerte schon auf der ersten Seite ihren Dienst. Kurioserweise funktionierte die Anwendung mit Version 7.1.0, jedoch nicht mit 7.1.1.
  - ➢ Hier half nur ein „Downgrade“ auf Version 7.1.0.
- Die Server sind unterschiedlich „informationsfreudig“, wenn eine Exception im Container geworfen wird.

Migration nach JBoss



- Die Migration der JEE6-Anwendung war zum Zeitpunkt dieses Erfahrungsberichts noch nicht abgeschlossen.
- Zusammenfassend kann man sagen, dass man sich von Anfang an für einen bestimmten Application Server entscheiden sollte oder mit doch recht unterschiedlichen Verhaltensweisen zurechtkommen muss.
- Lehre: Deployment immer von Anfang an auf verschiedenen Application Servern testen!

syntegris information solutions

Fazit

- Eine neue Technologie zu erlernen gehört zum Beruf des Entwicklers und ist für die Entwicklung einer modernen Unternehmensanwendung unerlässlich.
- Bei einem Fazit sollte man bewerten, ob man durch die neue Technologie produktiver geworden ist, die Komplexität beherrschbar ist und ob sich Probleme mit angemessenem Aufwand lösen lassen.
- Zusätzlich kann man natürlich noch weitere Punkte aufführen, die meist einen subjektiven Charakter haben.

Fazit

Positiv

- Die Dokumentation hat sich verbessert.
- Die Entwicklung einer „leichtgewichtigen“ Anwendungsarchitektur ist dank JEE6 sehr einfach.
- Die entwickelte Anwendung läuft performant und stabil. Memory-Leaks sind beispielsweise bisher nicht aufgetreten.
- Die Entwicklungszeit für einen fachlichen Use Case hat sich in Bezug auf JSF reduziert bzw. ist vergleichbar zu einer Entwicklung mit der alten Version 1.2, die zusammen mit einem Zusatzframework eingesetzt wurde.

Fazit

Negativ

- Automatisierte Tests für Container-Managed Komponenten waren für uns nur sehr schwer zum Laufen zu bringen.
- Es gibt Features, die zwar vom Konzept her gut sind, bei denen jedoch die Umsetzung hapert.
- Für einige Features gab es nur unzureichende Beschreibungen und Beispiele.
- Die Fehleranalyse bei Exceptions, die vom Container geworfen werden, ist oft nur recht schwer durchzuführen.

syntegris information solutions

Fazit

Abschließend kann man sagen, dass der neue Java EE 6 Standard ein brauchbares Mittel zur Entwicklung einer komplexen Unternehmensanwendung ist:

- Sinnvolle Neuerungen wurden eingeführt und die Entwicklung geht in vielen Teilen einfacher vonstatten als zuvor.
- Allerdings sollte man sich vor der Verwendung von JEE6 über die Möglichkeiten und Grenzen des Frameworks informieren.
- Wir sind zuversichtlich, dass sich der Standard durchsetzen wird und es viele interessante und erfolgreiche Projekte mit JEE6 geben wird.

Fazit

Persönliches JSF 2 Fazit

- Es gibt so gut wie keine Änderungen im Konzept. JSF ist immer noch eine viel zu bunte Mischung aus Java/EL/xhtml-Syntax.
- Für Neueinsteiger ist diese Kombination nicht immer verständlich und stellt eine nicht zu unterschätzende Lernhürde dar.
- Immer noch recht hoher Aufwand für „wenig“ Anwendung: Der Code, den man für relativ wenig Oberflächenfunktionalität schreiben muss, ist meistens umfangreich.

Persönliches JSF 2 Fazit

- Es gibt immer noch eine starke Abhängigkeit von guten Komponentenbibliotheken und Frameworks.
- Dieser Punkt kann bei gewissen Anforderungen eine richtige Qual sein, wenn man sich auf die Funktionalität dieser Bibliotheken verlässt:
  ICEfaces war im Nachhinein gesehen die absolut falsche Wahl für das Projekt.

Persönliches JSF 2 Fazit

- In einigen Punkten kommt JSF 2 nicht an die Mächtigkeit von JSF 1.2 und dem JBoss Seam Framework heran. So fehlt z.B. die globale Unterstützung von EL-Ausdrücken, eine vergleichbar mächtige EL, komplexe Navigationsregeln und Security-Features.
- Es gibt immer noch zu wenige Beispiele für bestimmte Neuerungen. So finden sich zwar zu Hauf Beispiele für Composite Components – wie man jedoch z.B. optionale Attribute umsetzen kann, wird nirgends erklärt.

Persönliches JSF 2 Fazit

- Ein weiteres Beispiel ist die neue Multi-Field Validierung, die eigentlich sehr praktisch ist, jedoch nur am Rande erwähnt wird.
- Die Dokumentation ist in vielen Teilen unzureichend, missverständlich oder nicht mehr aktuell. Häufig kann man nicht mehr unterscheiden, ob die Dokumentation für JSF 1.x oder 2 gilt.
- Die Entwicklung von JSF 2 schreitet scheinbar schneller voran als die Entwicklung der alten Version. So gibt es bereits gute Ideen für die Version 2.2 und kontinuierliche Fehlerbehebungen für die Version 2.1.x