Deutscher Ring
Wir erledigen das.

Refactoring to Spring
- ein Erfahrungsbericht -

Holger Breitling, C1 WPS GmbH
Torsten Köster, Deutscher Ring

Software Engineering 2007

C1 WPS

Gliederung

Deutscher Ring
Wir erledigen das.

- Ausgangssituation
- Ziele des Refactorings
- Refactorings to Spring

C1 WPS

Ausgangssituation

Deutscher Ring
Wir erledigen das.

- **Zentrale Intranetanwendung „iKontakt“**
  - Besteht seit 2000
- **stürmisch und projektbezogen gewachsen**
- **Sehr hohe Fertigungstiefe**
  - Standardlösungen waren zur Konstruktionszeit nicht verfügbar
- **Droht der Verlust der Beherrschbarkeit?**
  - Hohe Einarbeitungszeit im Entwicklerteam
  - Angst vor grundlegenden Änderungen aufgrund befürchteter Seiteneffekte
- **Sehr begrenzter Rahmen für ein Refactoring**
  - 160 PT insgesamt, 8 Entwickler
  - > 250 DAOs

C1 WPS

Ziele des Refactorings

Deutscher Ring
Wir erledigen das.

Sicherstellung der Beherrschbarkeit

- Erhöhung der Testabdeckung
- Modularisierung entlang fachlicher Einheiten

Reduktion der Fertigungstiefe

- Spring zur Konfiguration und Initialisierung der Anwendung
- Optimale Anbindung von Drittrahmenwerken (Hibernate, Tiles)
- JDBC Connection Pooling

Ausrichtung hin zu einer servicebasierten Architektur

- Zerlegung des Systems in Schichten
- Herauslösen der Datenzugriffsschicht in ein eigenes Projekt

C1 WPS

Versprechen von Spring

Deutscher Ring
Wir erledigen das.

- **Verbesserung der Struktur**
  - Entkopplung durch Dependency Injection
  - Aspektorientierte Programmierung
- **Bessere Testbarkeit**
- **Viele einfach zu verwendende Anschlüsse an relevante Technologien bzw. Frameworks**
  - Persistenz: JDBC, Hibernate, JDO, JPA,…
  - Web: Struts, Tiles, JSF,…
  - Remoting: RMI, Axis, XFire, Burlap, Hessian
  - Weiteres: JTA, JNDI, JMS,…
- **„It‘s all about choice“**
- **Aber: Eine komplette Umstellung auf Spring (auch nur auf die Dependency Injection) war nicht leistbar!**

C1 WPS

Die entscheidenden Fragen:

Deutscher Ring
Wir erledigen das.

Ist es machbar und sinnvoll, eine Anwendung (erst einmal) nur in Teilen auf Spring umzustellen?

Wie geht man dabei am besten vor?

C1 WPS

Erfolgsfaktoren

Deutscher Ring
Wir erledigen das.

Identifizierung der für iKontakt nützlichsten Spring-Features

- Konzentration auf diese Features

Refactorings to Spring

- Führe einen dedizierten Service Locator für den Zugriff auf den Spring-Context ein
- Ersetze das explizite Auslesen von Properties durch Injektion von Properties
- Bringe Spring-gemanagte Ressourcen indirekt über die bestehenden Ressourcenmanager in die Anwendung
- Erhalte explizite Transaktionssteuerung durch direkten Zugriff auf die Spring-Transaktionsmanager aufrecht.

C1 WPS

Für iKontakt relevante Spring-Features

Deutscher Ring
Wir erledigen das.

- JDBC-Template
- JMS-Template
- Connection-Pooling
- Deklarative Transaktionssteuerung
- Deklaratives Caching

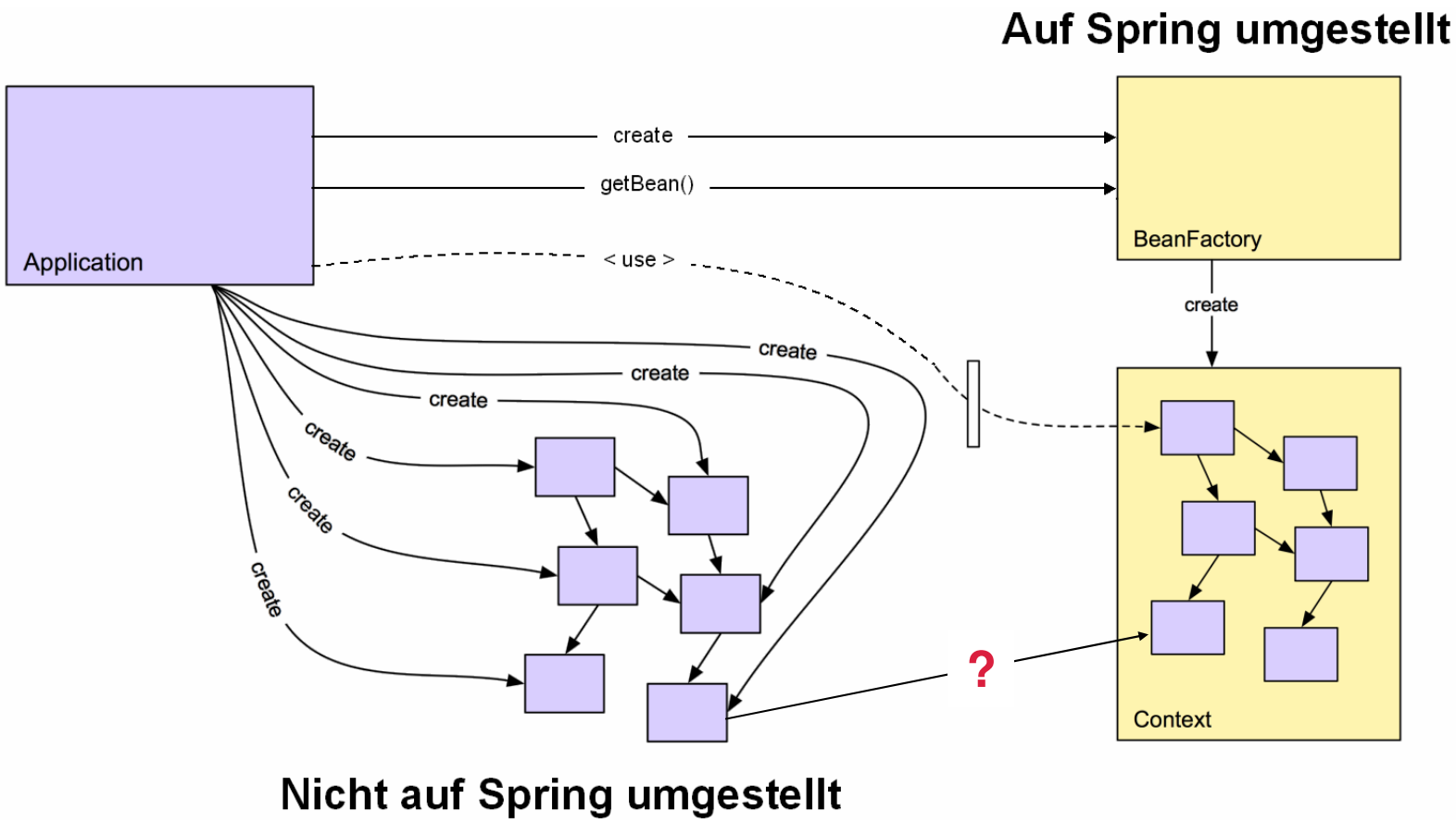
Später (für neue Funktionalität)

- Tiles-Einbindung
- Dependency Injection von Session Beans

C1 WPS

Was wir mit „auf Spring umgestellt“ meinen

Deutscher Ring
Wir erledigen das.



C1 WPS

Vorgehen bei der Einführung

Deutscher Ring
Wir erledigen das.

- **Höchste Priorität hatte das Umstellen der Data Access Objects (DAOs)**
  - Ein Data Access Object nach dem anderen (aber nicht alle) wurde auf Spring umgestellt, d.h. über den Spring Context konfiguriert und erzeugt.
  - DAOs verwenden das Spring JDBC Template und deklarative Transaktionssteuerung.
  - JMS DAOs verwenden eine eigene Erweiterung des Spring JMS Templates (+ dekl. Transaktionssteuerung).
- **Das Team stellte auch Teile der über den DAOs liegenden Serviceschicht auf Spring um.**
- **(Nach dem Refactoring hat das Team *neue Teile* der Präsentationsschicht mit Spring MVC + Tiles entwickelt)**

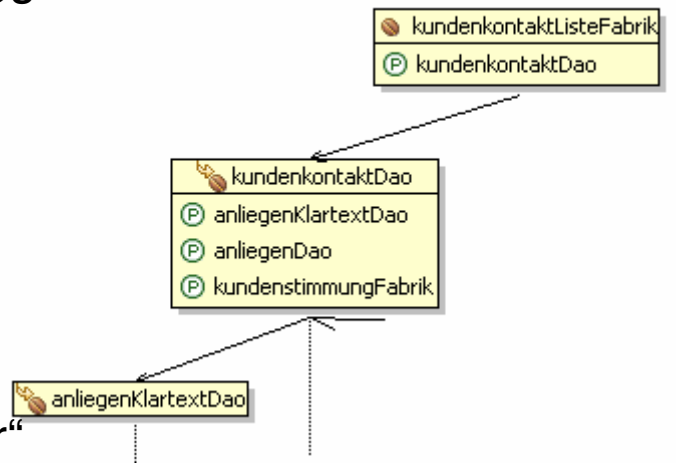
C1 WPS

Refactoring (1): Zentraler Service Locator für Spring

Deutscher Ring
Wir erledigen das.

- **Problem:**
  - Wie greifen nicht umgestellte Services und DAOs auf die Spring Beans zu?

- **Lösung: `DirtySpringContext` als Service Locator**
  - Kapselt den Spring Application Context (als Klassenvariable).
  - Ist ein definierter Übergang zwischen „neuer“ und „alter“ Welt
  - Zugriffe sind statisch erkennbar und damit leicht umzubauen.



C1 WPS

Refactoring (2): Injektion von Properties

Deutscher Ring
Wir erledigen das.

- **Problem: Anwendung wurde über Properties konfiguriert**
  - unübersichtlich, aber der Status Quo.
  - Wie weiter verwenden mit Spring?

- **Lösung: Ersetze das Auslesen von Properties durch Injektion von Properties**
  - Große Verbesserung: Konfigurationsmöglichkeiten werden an den Klassenschnittstellen sichtbar
  - Injektion erfolgt in den Spring-Konfigurationsdateien (XML) durch *Property Placeholder.* Vorteil: Property-Namen werden explizit sichtbar.
  - Festlegung der Properties über Property-Dateien
    - Property-Dateien für Entwicklung, Test, Produktion
    - *nur ein Set von Spring-Konfigurationsdateien* für Entwicklung, Test, Produktion

```
<property name="hostName" value="${jms.connection.hostName}" />
<property name="channel" value="${jms.connection.channel}" />
```

C1 WPS

Refactoring (3): Spring-Management von Ressourcen

Deutscher Ring
Wir erledigen das.

Problem:

- Das Spring Ressourcen-Management (DB-Connections, Message Queues etc.) soll überall genutzt werden, obwohl nur ein Teil der Klassen auf Spring umgestellt wird.

Lösung:

- Baue bestehende Ressourcen-Manager so um, dass diese das Spring-Management für die Ressourcen nutzen.
  - Resultat: Anwendungsweiter Einsatz Spring-gemanagter Ressourcen.

C1 WPS

Refactoring (3): Spring-Management von Ressourcen

Deutscher Ring
Wir erledigen das.

Beispiel JDBC Connection Pooling

`DBConnectionSuite` ist der iKontakt Container für JDBC-Verbindungen

- Via Spring wird ein **`DBConnectionSuiteDelegate`** initialisiert
  - Erbt von **`DBConnectionSuite`** und überschreibt alle Methoden
  - Injektion von gepoolten **`DataSource`**s
  - Leitet alle Anfragen nach JDBC-Verbindungen an die injizierten **`DataSource`**s weiter
  - Injektion des **`DBConnectionSuiteDelegate`** in die zentrale Konfiguration: Nutzung auch durch noch nicht umgestellte DAOs

Ergebnis

- Merkliche Beschleunigung der DB-Zugriffe

```
<!--
    Die DBConnectionSuite. Diese Datenbankverbindung kann von
    noch nicht umgestellten Fabriken genutzt werden und delegiert
    intern an die übergebenen Datenquellen.
-->
<bean id="dbConnectionSuite"
    class="de.deutscherring.common.spring.dao.DBConnectionSuiteDelegate">
    <property name="dataSourceAmadeus" ref="dataSource
    <property name="dataSourceDisp" ref="dataSourceD
    <property name="dataSourceKe
```

C1 WPS

Refactoring (4): Programmatische und deklarative Transaktionssteuerung

Deutscher Ring
Wir erledigen das.

Problem:

- In iKontakt gibt es ein ausprogrammiertes Transaktionshandling.
  - Dieses muss bei nicht auf Spring umgestellten Klassen unverändert weiter so funktionieren.
  - Herausforderung: Zusammenspiel mit dem deklarativen Transaktionsmanagement der umgestellten DAOs.

Lösung:

- Versehe die DAOs mit Transaktionseinstellung **`REQUIRED`**
- Implementiere die Transaktionslogik wie folgt:
  - Hole aus dem **`DirtySpringContext`** den passenden Spring-Transaktionsmanager und eröffne eine neue Transaktion.
  - Verwende das DAO
  - Rufe **`commit()`** oder **`rollback()`** am Transaktionsmanager auf (statt an einer Connection etc.).
- (Erwäge alternativ, die betreffende Klasse nur wegen des Transaktionshandlings doch zu einer Spring Bean zu machen).

C1 WPS

Zusammenfassung

Deutscher Ring
Wir erledigen das.

- **Anwendung während des Refactorings ständig lauffähig**
- **Merkliche Beschleunigung der Anwendung**
- **Regressionstests zeigen Fehler auf**
- **Refakturierte Elemente bilden Grundlage für eine servicebasierte Architektur**

- **Fazit:**
  - Es ist machbar und sinnvoll, eine Anwendung (erst einmal) nur teilweise auf Spring umzustellen.
  - Auch „ein bisschen Spring“ kann eine Anwendung voran bringen.

C1 WPS