

# ColdSpring

## Einführung in das Framework

Präsentation bei der ColdFusion Usergroup

München, 31. Juli 2007

Harry Klein



## Übersicht

- § OOP, CFC, Instanzierung
- § Probleme
- § Lösungsmöglichkeiten
- § ColdSpring
- § IoC / DI
- § ColdSpring XML Konfiguration
- § ColdSpring Arbeitsweise
- § Framework Unterstützung
- § Ressourcen

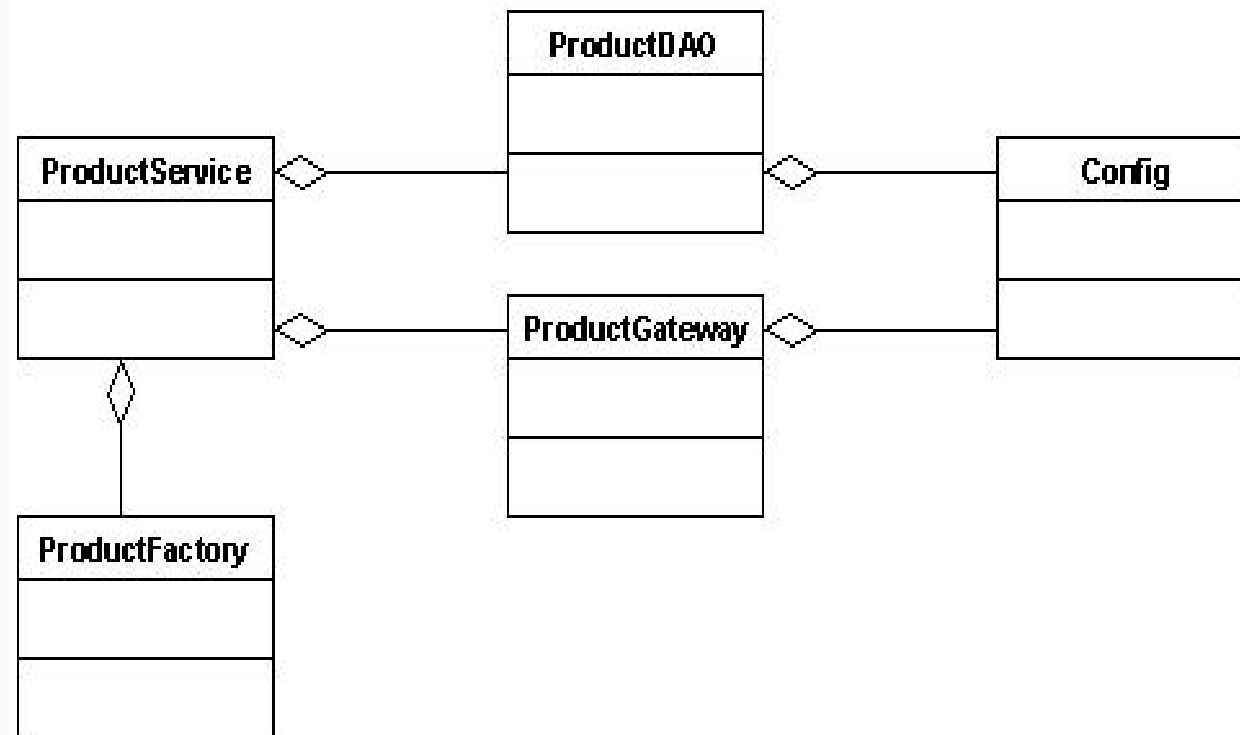
## ColdFusion OOP

### Fragen:

- § Wie viele Zuhörer verwenden CFCs?
- § Wie viele Zuhörer verwenden objektorientierte Frameworks?
- § Verwenden Sie eine hohe Anzahl von komplexen, voneinander abhängigen CFCs?
- § Welche Probleme sind Ihnen bei der Verwendung von CFCs aufgefallen?

## Abhängigkeiten

UML Diagramm



## Erzeugen von Objekten in ColdFusion

```
<cfset oListUtil = createObject( "component",  
    "com.util.listutil" ).init() >
```

2 Typen von Objekten: statefull - stateless

Probleme beim Erzeugen von Objekten:

- § Hardgecodete Pfadnamen wie „com.util.listutil“
- § Schwierigkeiten: Implementierungen austauschen, Objekte verschieben
- § Testing durch Einsatz von „mock“ Objekten aufwendig

## Probleme durch Abhängigkeiten

- § Reihenfolge der Initialisierung
- § Wer ist für die Initialisierung verantwortlich?
  - § Controller
    - § Sollte nicht die Aufgabe eines Controllers sein
  - § Factory
    - § Erzeugt Objekte
    - § Kapselt die Instanzierung
    - § Verwaltet Singletons
    - § Hardgecodete Pfade werden aufgelöst
    - § Abhängigkeiten sollten dem Factory Objekt bekannt sein

## Generische Factories

### § ChiliBeans

§ wurden in ModelGlue verwendet, veraltet

### § LightWire

§ Peter Bell

### § ColdSpring

## ColdSpring

- § Portierung des Java's Spring Frameworks
- § "super-factory-registry" – BeanFactory
- § Inversion of Control (IoC) / Dependency Injection (DI) container
- § Aspect Oriented Programming (AOP) container
- § Erzeugt und verwaltet Objekte
- § Kapselt die Erzeugung & Abhängigkeiten
- § XML-basierte Konfiguration

## ColdSpring als Lösung aller bisherigen Probleme?

- § Keine hardgecodeten Pfade im Appl.code
- § Keine Initialisierungen im Appl.code notwendig
- § Abhängigkeiten werden verwaltet
- § CFCs können leicht verschoben werden
- § Vereinfachtes Testing
  - § z.B. durch Austausch der XML Datei oder Änderung der XML Definitionen

### IoC / DI

- § Eine Abhängigkeit entsteht, wenn ein CFC ein anderes benötigt
- § Die zwei beteiligten Objekte nennt man “Kollaborateure”
- § IOC - Das “Hollywood Prinzip” – ruf mich nicht an, ich werde dich anrufen
- § ColdSpring, als ein IoC container, “injiziert” ein CFC in ein anderes
- § DI = “dependency injection”
  - § ist ein Design Pattern
  - § überträgt die Verantwortung für Objekterzeugung und Verlinkung vom Objekt an eine Factory
  - § kann als eine Implementierung von IoC verstanden werden

## OOP Design Prinzipien

- § Lose Kopplung
  - § DI reduziert Kopplung – keine Erzeugung im CFC
  - § Implementierung der Kollaborateure unwichtig
- § Hohe Kohäsion
  - § CFC erledigt eine Aufgabe richtig
  - § Erzeugung anderer Objekte ist nicht die Aufgabe einer CFC

## ColdSpring Wiring

- § Auflösen von Abhängigkeiten
- § Es gibt zwei Methoden um Beans/CFCs zu “injizieren”:
  - § Constructor-argument Injection – init() Methode
  - § Setter Injection ” - Setter Methode
- § Was ist besser?
  - § Nur Setter Injection erlaubt zirkuläre Abhängigkeiten

## ColdSpring XML Konfiguration

- § bean = CFC
- § id="foo" = Name/Identifizier eines Beans
- § class="path.to.cfc" = CFC Typ
- § Abhängigkeiten in child-tags:
  - § <property/> (Setter-injection)
  - § <constructor-arg/> (Constructor-arg injection)
- § Kinder von <property/> und <constructor-arg/>:
  - § <value/> (String)
  - § <map/> (Struct)
  - § <list/> (Array)
  - § <ref bean="idOfBean"/> referenziert ein Bean
  - § <bean/> inline Definition eines anderen Bean

## ColdSpring Beispiele

- § Beispiel 1:
  - § userService enthält emailService
  - § Constructor-arg injection
- § Beispiel 2:
  - § Services enthalten sich gegenseitig
  - § Setter injection
- § Beispiel 3:
  - § userService enthält emailService
  - § MailServer constructor-arg in emailService (String)

## ColdSpring Arbeitsweise

- § Erzeugen einer XML Konfigurationsdatei, die alle CFCs und deren Abhängigkeiten enthält.
- § Erzeugen einer ColdSpring “BeanFactory”, laden der Bean Definitionen.
- § Die BeanFactory sollte in einem persistent-scope (application oder server) gehalten werden, normalerweise sind alle Beans Singletons.
- § Die BeanFactory liefert Beans über die Methode `getBean(...)`.
  - § Nur ColdSpring hat das Wissen zur Objekterzeugung
  - § ColdSpring löst alle Abhängigkeiten auf
  - § CFCs kennen ColdSpring nicht

## Autowiring

- § ColdSpring kennt die Metadaten aller Beans
- § ColdSpring untersucht den Konstruktor und alle Setter
  - § und erkennt alle Abhängigkeiten automatisch
- § Zwei Möglichkeiten: byName oder byType
- § Kann für alle Beans mit dem default-autowire="" Attribut gesetzt werden
- § oder für ein individuelles Bean mit dem Attribut autowire=""
- § Vorteile:
  - § Geringerer Aufwand, kleineres XML
- § Nachteile:
  - § Übersichtlichkeit geht verloren – welches Objekt wird wo injiziert?

## Autowiring – Demo

§ Beispiel 4:

§ userService enthält emailService

§ `<beans default-autowire="byName">`

§ Beispiel 5:

§ demonstriert Autowire für ein einzelnes Bean - `autowire="byName"`

## ColdSpring Factories

- § Unterstützung für “legacy” Factories über das “factory-method” Attribut.
- § Verwendung eigener Factories (Objekte, die andere Objekte erzeugen) über ColdSpring
- § Factory wird wie ein normales Bean definiert
- § Beans, die über diese Factory (statt über ColdSpring) erzeugt werden sollen, verwenden statt class=""
  - § factory-bean= “idDerFactory”
  - § factory-method= “factoryMethodeDieObjekteErzeugt”

## ColdSpring Factories - Demo

§ Beispiel 6:

§ userService enthält userDao, welches über eine eigene Factory (factory-bean) erzeugt wird

## ColdSpring Framework Support

- § Die meisten populären ColdFusion MVC Frameworks unterstützen ColdSpring
- § Mach-II enthält ein ColdSpring Plugin
- § Fusebox 5 enthält Lexika für ColdSpring
- § ModelGlue 2.0 ist das erste Framework, welches ColdSpring als IoC container intern und extern verwendet
- § In Model-Glue, Mach-II und Fusebox ist auch “autowiring” möglich

## ColdSpring Generierung

- § Illudium
  - § DB auswählen
  - § CFC Path: z.B. “contens30.model.editors.”

## Zusammenfassung

- § OO Applikationen werden im Lauf der Zeit immer größer und komplexer  
– der Aufwand für die Erzeugung und Verwaltung des Lebenszyklus (Abhängigkeiten) steigt
- § Factories können helfen diese Komplexität zu verringern
  - § custom Factories können aber selbst sehr komplex werden
  - § ... und lösen nicht alle Probleme
- § ColdSpring ist eine generische Lösung, die Erzeugung und Auflösung von Abhängigkeiten kapselt
  - § und somit den Applikationscode entschlackt und von diesen Komplexitäten isoliert
- § Die ColdSpring XML Konfigurationsdatei bietet eine sehr gute Übersicht aller Komponenten

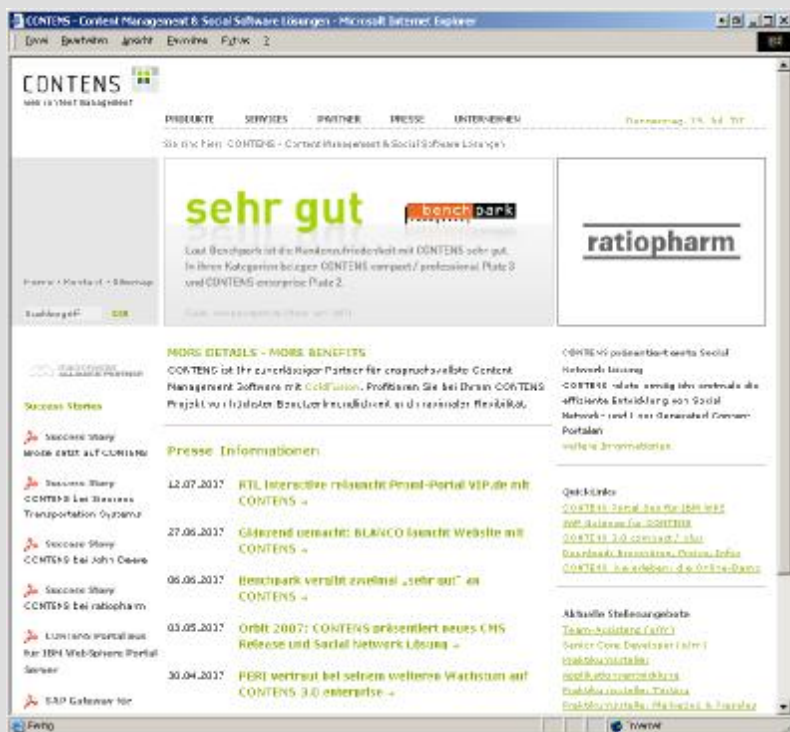
## Weitere Informationen

- § Website: <http://www.coldspringframework.org>
- § Autoren:
  - § <http://www.d-ross.org> (Dave Ross)
  - § <http://cdscott.blogspot.com> (Chris Scott)
- § CVS:
  - § server: [cvs.coldspringframework.org](http://cvs.coldspringframework.org)
- § BugTracker: <http://code.coldspringframework.org>
- § MailingList:
  - § To: [MDaemon@coldspringframework.org](mailto:MDaemon@coldspringframework.org)  
Subject: SUBSCRIBE [coldspring-dev@coldspringframework.org](mailto:coldspring-dev@coldspringframework.org)
  - § <http://www.mail-archive.com/coldspring-dev%40coldspringframework.org/info.html>

## Beispielsapplikationen

- § Feedviewer Applikation (im ColdSpring Download Paket enthalten)
- § MachBlog: <http://www.machblog.org/>
- § AppBooster: <http://kurtwiersma.com/releases/appbooster-beta3.zip>
- § CFUG Attendance Recorder:  
<http://www.daveshuck.com/blog/index.cfm/2007/7/11/Example-MachII-15ColdSpring-application>
- § ColdSpring Petmarket: <http://www.cfpetmarket.com>

# Herzlichen Dank für Ihre Aufmerksamkeit



Für Ihre Fragen stehen wir Ihnen gerne jederzeit zur Verfügung.

CONTENS Software GmbH  
Oettingenstr. 25  
80538 München  
[info@contens.de](mailto:info@contens.de)  
[www.contens.de](http://www.contens.de)