Übung 7 – Architektur-Muster

Methoden des Software Engineering

Christian Kroiß

Agenda

Broker

- Überblick (siehe Folien Vorlesung)
- Kurze Einführung in Reflection
- Beispiel-Implementierung in Java

MVC

- In Swing
- Im Web

Dependency Injection

- Überblick + Google Juice

Reflection

Wikipedia: Reflection (computer programming)

„In computer science, reflection is the process by which a computer program can observe (do type introspection) and modify its own structure and behavior at runtime."

Reflection in Java

- Eigenschaften von Klassen und Methoden analysieren
  - Vererbungsstruktur, implementierte Interfaces
  - Sichtbarkeiten
- Arbeiten mit *Methodenobjekten*
  - Iterieren über die Methoden einer Klasse
  - Suchen einer Methode nach Name + Signatur
  - Aufruf der Methode auf kompatiblen Objekten
- Analysieren von Annotations

Java Reflection Tutorial:
http://docs.oracle.com/javase/tutorial/reflect/index.html

Dynamic Proxy

- **Generell: Proxy-Pattern (http://en.wikipedia.org/wiki/Proxy_pattern)**
- Die Klasse `java.lang.reflect.Proxy` ist Bestandteil der Java-Reflection-API und ermöglich das Erstellen von Objekten, die nach außen hin wie ein gegebenes Interface auftreten, ohne dass dieses Interface zur Compile-Zeit bekannt sein muss.
- Durch Delegation ermöglicht ein Proxy im Prinzip Hinzufügen von Funktionalität zu einer Klasse, ohne diese zu verändern.

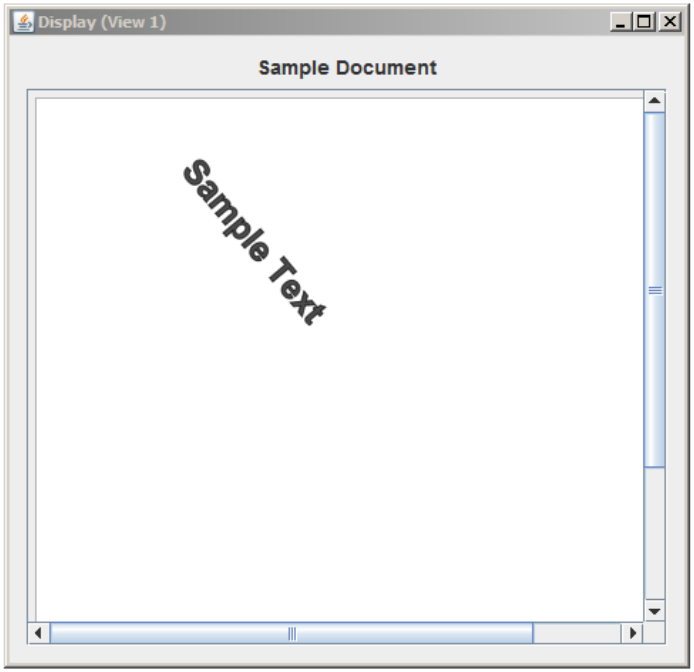
➔ Transparenz

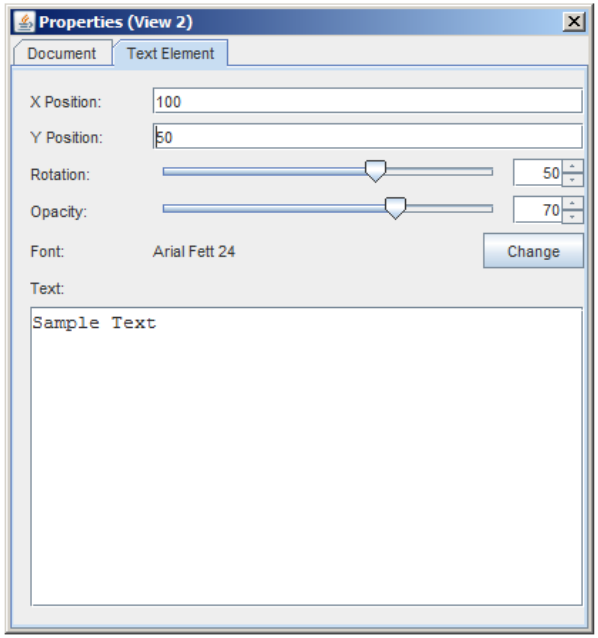
Broker - Beispielimplementierung in Java

- .ZIP-Datei auspacken
- In Eclipse:
  - Import → General → Existing Projects into Workspace
  - Bei den Run-Configurations sollte es jetzt die drei Konfigurationen *BrokerTest*, *ServerTest* und *ClientTest geben.*
    - Auf richtige Reihenfolge achten:Diese immer Starten in der Reihenfolgen
    - *Broker->Server->Client*

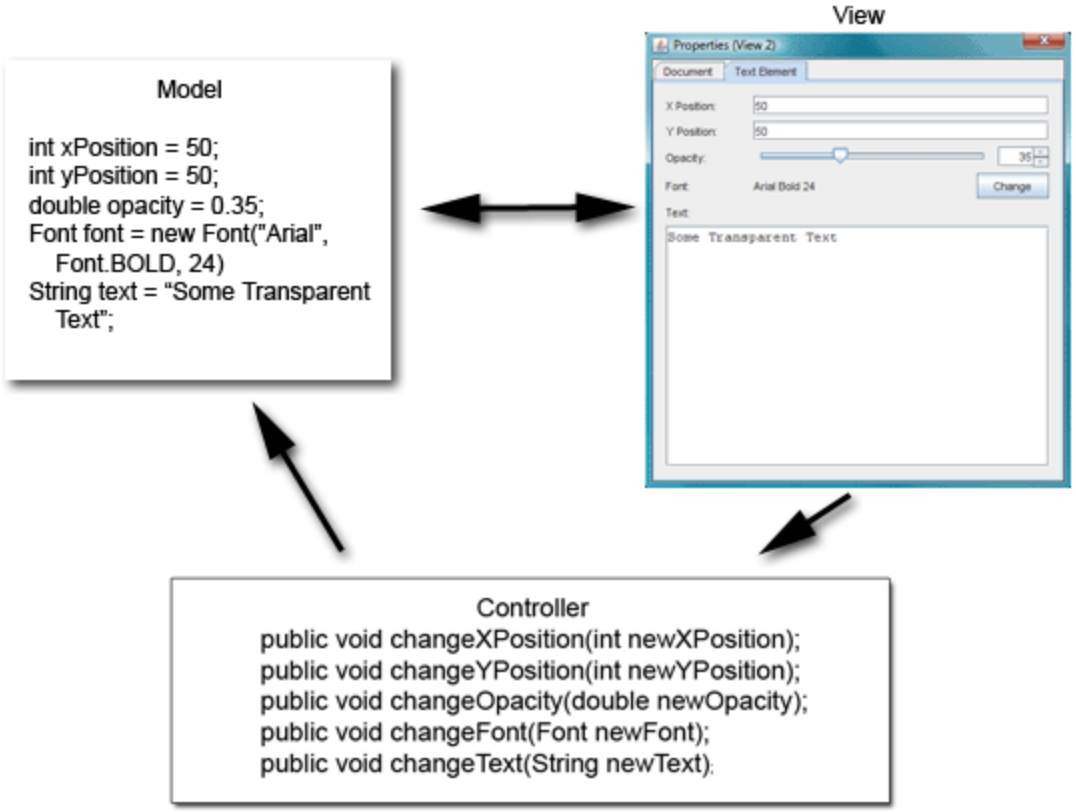
Artikel + Beispiel zum Runterladen:

- http://www.oracle.com/technetwork/articles/javase/index-142890.html



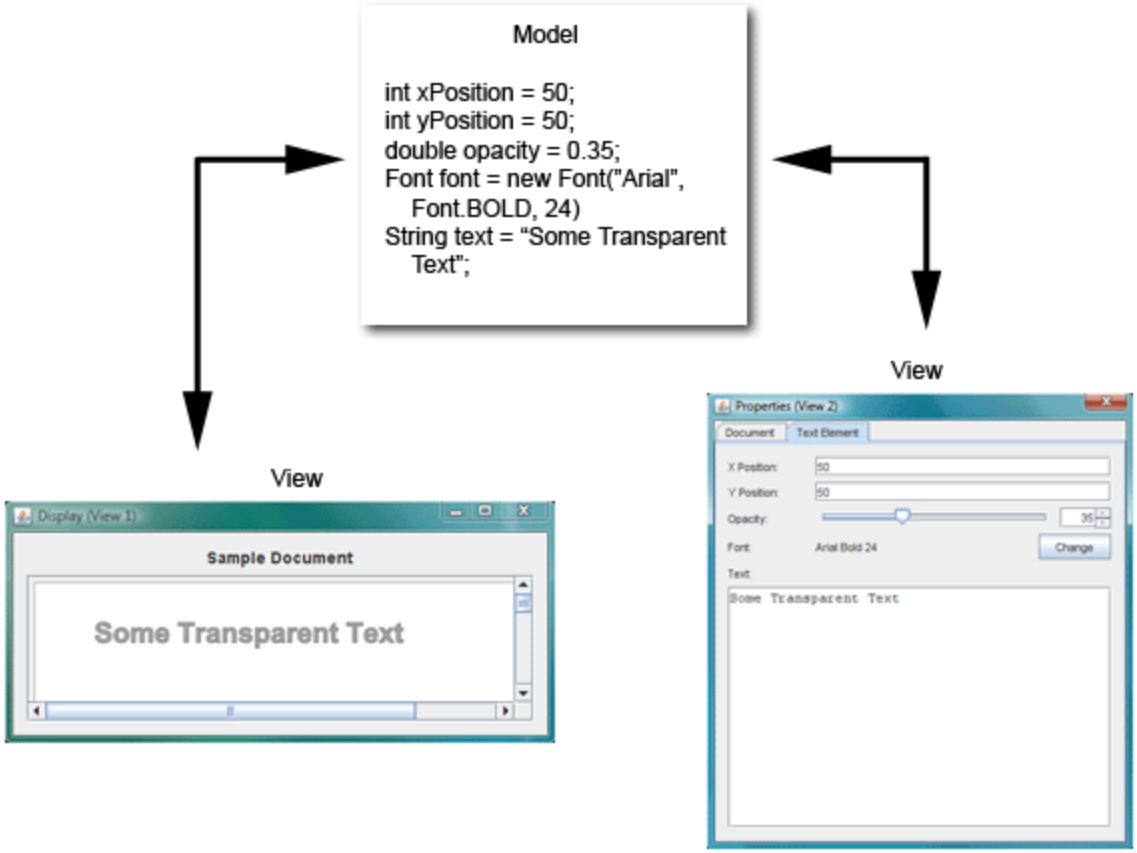


MVC in Swing



MVC Pattern in Swing

MVC in Swing

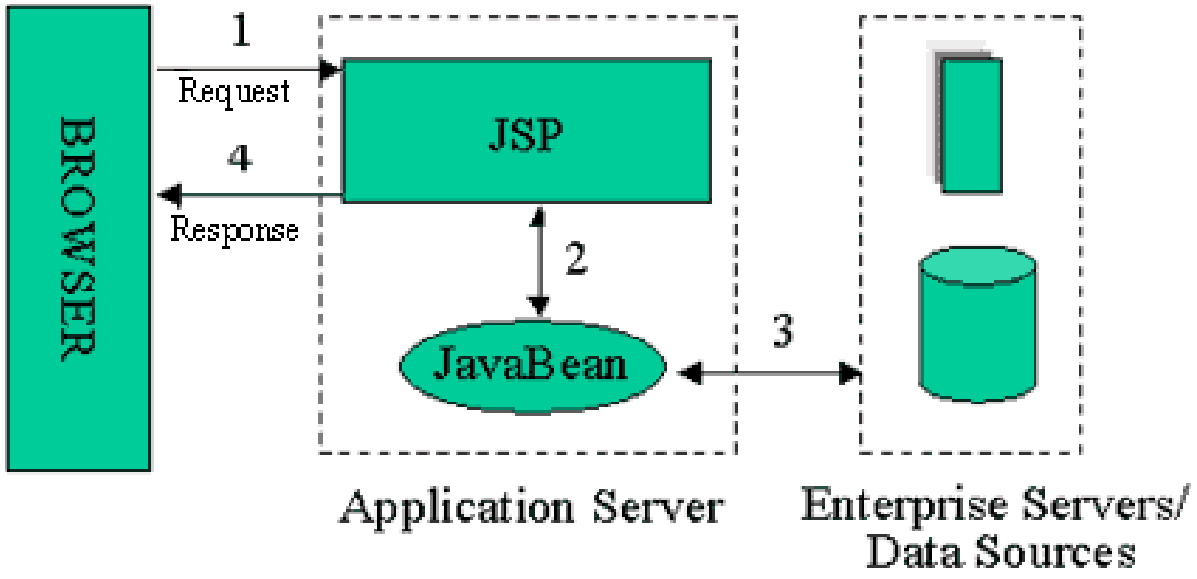


Mehrere Ansichten zu einem Modell

MVC im Web

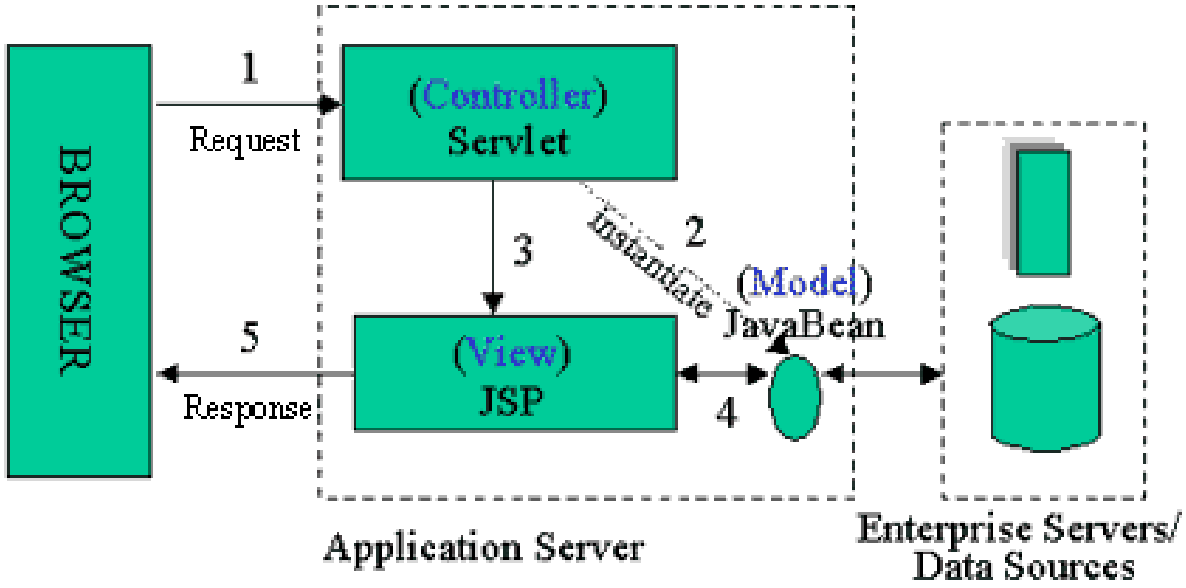
- In klassischen Web-Anwendungen ist MVC etwas anders zu verstehen, da das Update der Views nicht einfach vom Server ausgeführt werden kann.
- Stattdessen erfolgt eine *Navigation* von einer Seite zur nächsten.
- Das MVC-Basisprinzip der Trennung zwischen Modell und Ansicht hat natürlich weiter Bestand.
- In der Java-Welt existierte früher, bevor sich moderne Frameworks wie Java Server Faces oder Wicket durchgesetzt haben, eine Unterscheidung zwischen MVC gemäß Model 1 oder Model 2

Model 1



- Ansicht ist vom Modell (Java Bean) Getrennt .
- Behandlung von Requests + Navigation ist innerhalb der JSP-Seiten realisiert.

Model 2



- In der View (JSP-Seite) erfolgt nur die Darstellung.
- Behandlung des Requests + Auswahl der nächsten View (= Navigation) im Controller-Servlet.