



Triona – Information und Technologie GmbH

TrionaTip - Workshop

Matthias Göldner & Christian Martienssen 23. Januar 2009

Was ist TrionaTip?

- entstanden durch Spring-Workshop (ehemals Euro 2008)
- Tippsystem für die Fußball Europameisterschaft 2008

Erweiterungen

- Tipp-Plattform für verschiedenste Turnierarten
- Erstellung und Verwaltung von eigenen Turnieren

Status

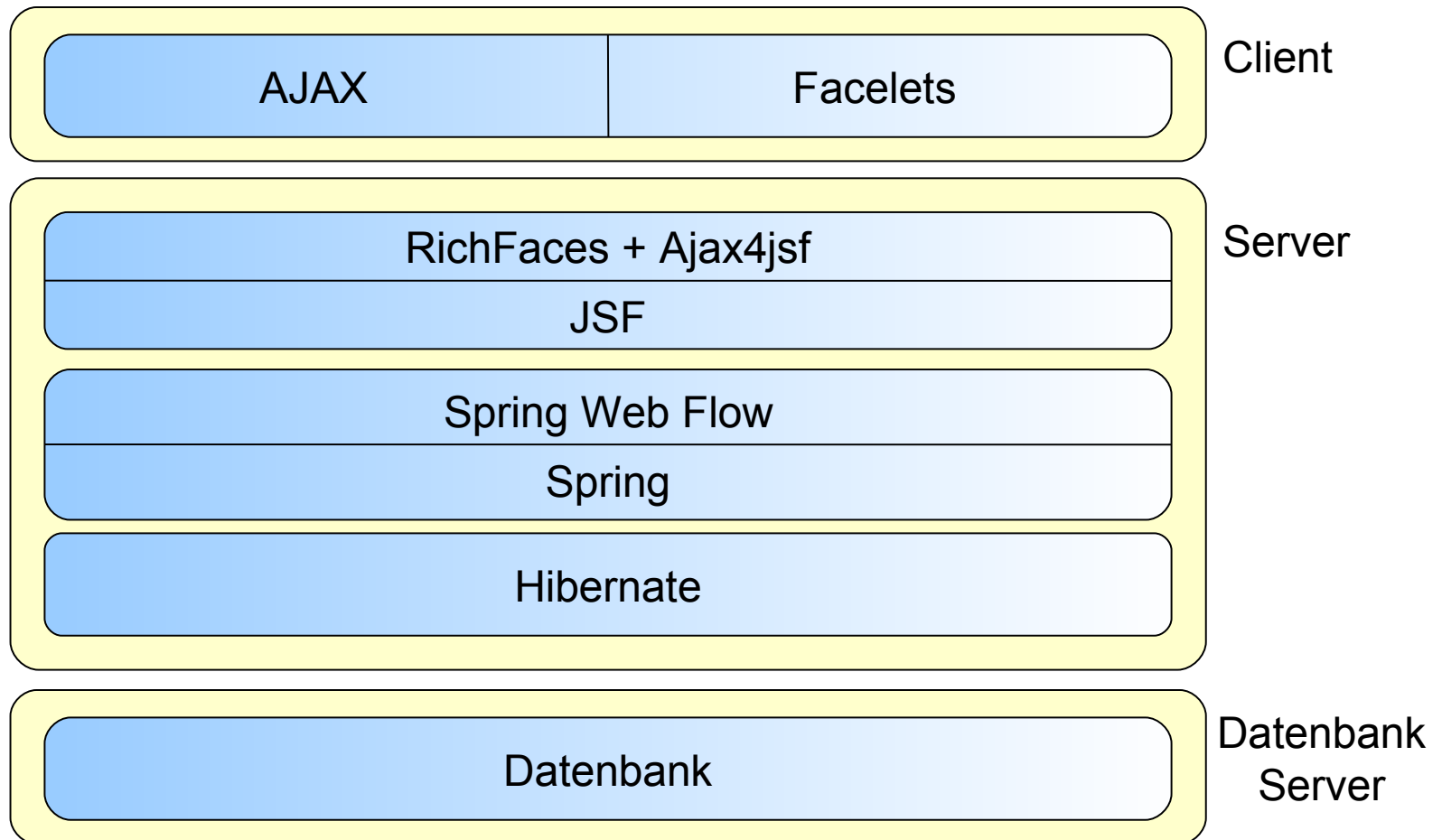
- Design für Turnierverwaltung abgeschlossen
- Entwicklungsphase der GUI-Funktionalitäten

Agenda

- Einleitung
- TrionaTip Architektur
- TrionaTip Praxis



2. TrionaTip Architektur



Java Server Faces (JSF)

Technologiebeschreibung

- Komponenten Framework für Webanwendungen
- stellt konfigurierbare Komponenten bereit
- ermöglicht Event Handling, Datenvalidierung und -konvertierung
- bietet ein Navigationskonzept

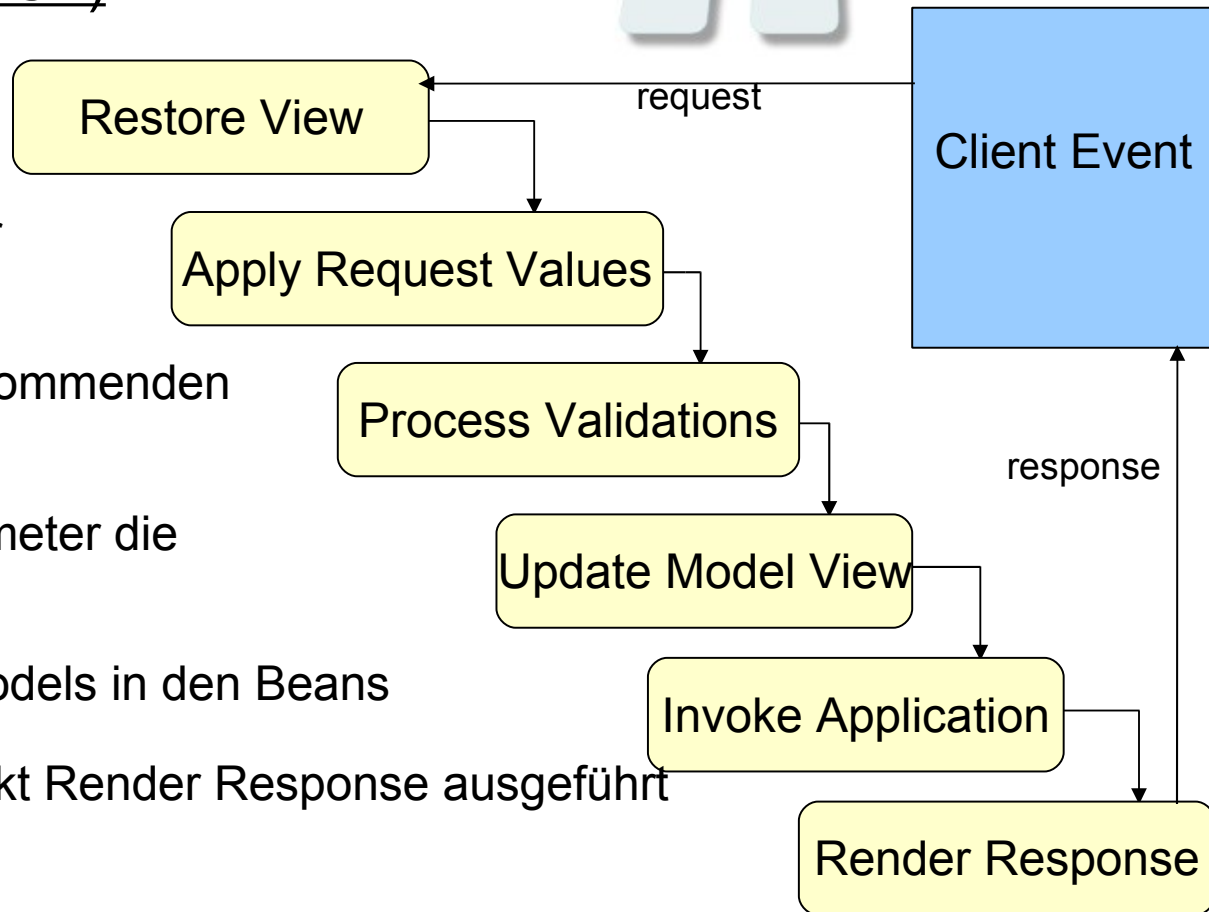
Einsatzgebiete

- Entwicklung von JEE Benutzerschnittstellen

Java Server Faces (JSF)

Der JSF Lifecycle

- der Lifecycle ist fester Bestandteil von JSF
- orientiert sich an einkommenden Requests
- füllt mit Requestparameter die Komponenten
- Aktualisierung des Models in den Beans
- im Fehlerfall wird direkt Render Response ausgeführt



Java Server Faces (JSF)

Vorteile

- Teil des JEE 5 Standards
- komponentenbasierte UI Entwicklung

Nachteile

- Lifecycle von JSF wird immer komplett durchlaufen
- Keine clientseitige Validierung möglich

Alternativen

- MyFaces



Facelets

Technologiebeschreibung

- View-Technologie basierend auf dem JSF-Lifecycle
- Templating-Framework
- Zusammensetzung der JSF-Komponenten in eigenem Komponentenbaum

Einsatzgebiete

- in Verbindung einer JSF Implementierung



Facelets

Vorteile

- einfache Realisierung eines Templates
- Erstellung eigener Komponenten durch Templating
- Anpassung an den JSF-Lifecycle

Nachteile

- eingeschränkte Debuggingmöglichkeiten
- bisher wenig IDE-Support (JBoss HTML Editor)

Alternativen

- Apache Tapestry
- Apache Tiles
- JSP / Servlet

Ajax4jsf

Technologiebeschreibung

- Ajax Framework für JSF
- versteckt die Komplexität der JavaScript Entwicklung
- ist direkt in den JSF Lifecycle integriert

Einsatzgebiete

- in Verbindung mit JSF für Rich Internet Applications

Ajax4jsf

Ajaxifizierung von JSF Komponenten

```
<h:outputLabel for="#{viewScope.text}" view="Eingabetext:">
<h:inputText value="#{viewScope.text}">
  <a4j:support event="onkeyup" reRender="textOutput"/>
</h:inputText>
<h:outputText id="textOutput" value="#{viewScope.text}"/>
```

Weitere Ajax Komponenten

- <a4j:region>, <a4j:form>, <a4j:poll>
- <a4j:commandButton>, <a4j:commandLink>

Ajax4jsf

Vorteile

- keine JavaScript Kenntnisse notwendig für die Ajaxifizierung
- einfaches Einbinden in bestehende Seiten
- reduziert Datenmengen die von dem JSF Lifecycle verarbeitet werden müssen

Nachteile

- kein Standard (ohne JavaScript nicht möglich)
- eigene Komponenten sind schwer zu erstellen

Alternativen

- Woodstock
- DynaFaces



RichFaces

Technologiebeschreibung

- Rich Web Interface Framework
- erweitert JSF Komponenten
- Ajax4jsf Framework integriert
- unterstützt Skinning

Einsatzgebiete

- Rich Internet Application in Kombination mit JSF

RichFaces

Rich Komponenten

- rich:tabPanel, rich:panel
- rich:modalPanel
- rich:toolTip
- rich:dataTable, rich:scrollableDataTable, rich:extendedDataTable
- rich:datascroller

RichFaces

Vorteile

- leichte Integration in Projekte
- bietet erweiterte und ajaxifizierte JSF Komponenten
- Entwicklung von Desktop ähnlichen Anwendungen

Nachteile

- generierter HTML Code sehr unübersichtlich
- Browserunterstützung ist nicht für alle Komponenten gleich gegeben

Alternativen

- ICEfaces



Spring Web Flow

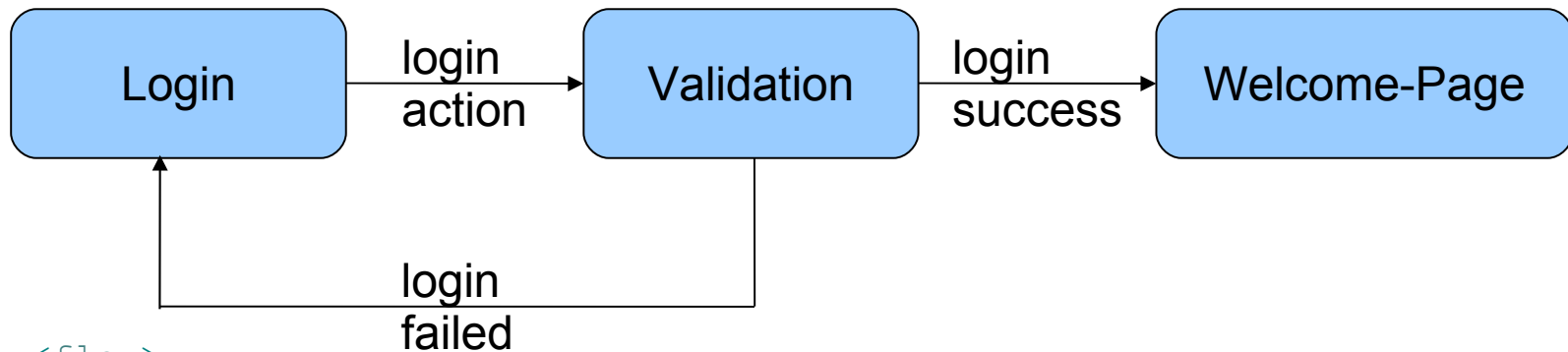
Technologiebeschreibung

- abstrakte Beschreibung von Geschäftsprozessen (Pageflows)
- Einführung von erweiterten Scopes
- weitere Abstraktionsschicht zwischen View und Middle-Tier

Einsatzgebiete

- Definition von Geschäftsprozessen in Spring-Umgebung

Spring Web Flow



```
<flow>
  <view-state id="login">
    <transition on="login-action" to="validation"/>
  </view-state>

  <action-state id="validation">
    <evaluate expression="...validateLogin()"/>
    <transition on="failed" to="Login"/>
    <transition on="success" to="welcome-user"/>
  </action-state>

  <view-state id="welcome-user">
  </view-state>
</flow>
```

Spring Web Flow

Vorteile

- klare und leicht lesbare Darstellung von Geschäftsprozessen
- zusätzlich definierte Scopes entsprechen den benötigten Gültigkeitsbereichen
- unabhängig von der View-Technologie

Nachteile

- keine Möglichkeit zur Bereitstellung global erreichbarer Daten

Alternativen

- Apache Orchestra (MyFaces Sub-Project)
- JBoss Seam
- JSF

Spring

Technologiebeschreibung

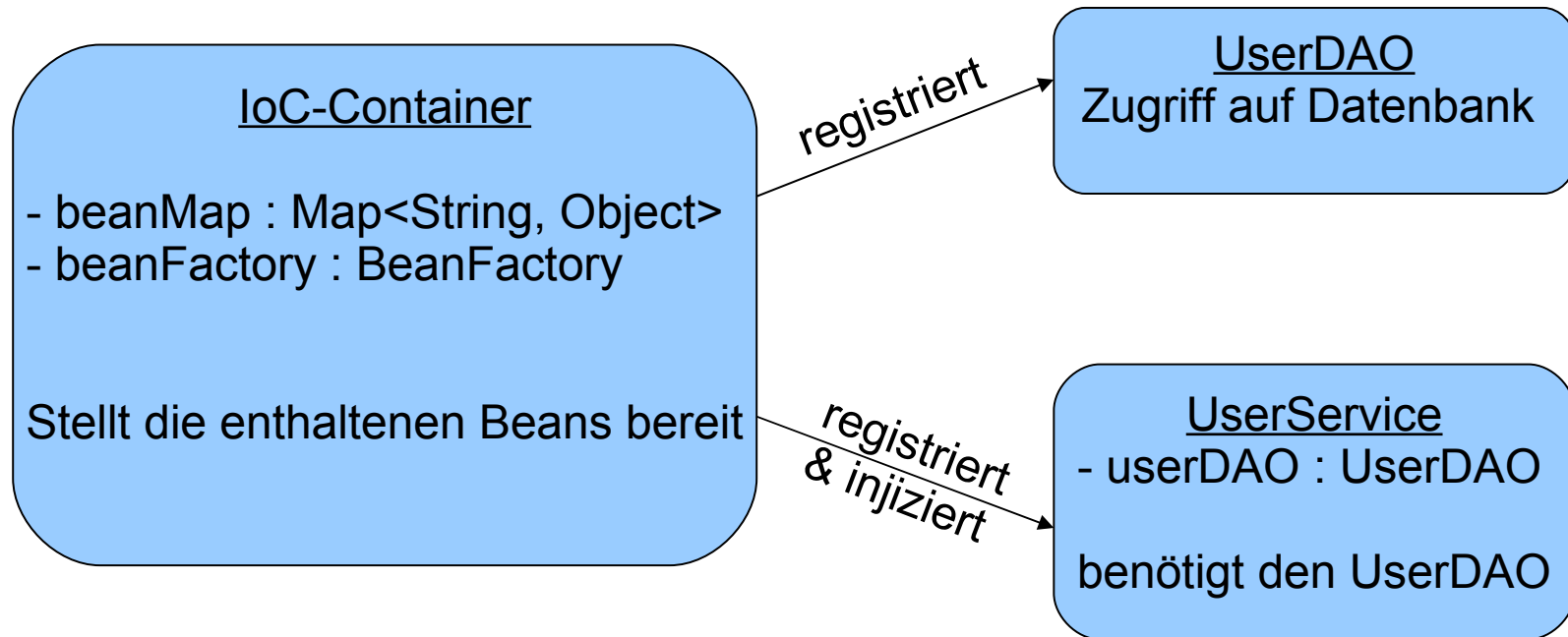
- Inversion of Control (IoC) Container mit Dependency Injection
- Verwendung von POJO's
- bietet viele Module (z.B. AOP, Web MVC, Testing)
- Konfiguration über zentrale XML-Datei und / oder Annotationen

Einsatzgebiete

- als leichtgewichtiger Container für JEE-Anwendungen

Spring

vereinfachte Darstellung eines IoC-Containers



Spring

Vorteile

- Entkopplung von Anwendungslogik und Konfiguration
- leichte Entwicklung durch Verwendung von POJO's
- Kompatibel zu vielen weiteren Frameworks (OR-Mapper, Web Frameworks)

Nachteile

- sehr komplexes Framework

Alternativen

- PicoContainer
- Apache Excalibur
- EJB



Hibernate

Technologiebeschreibung

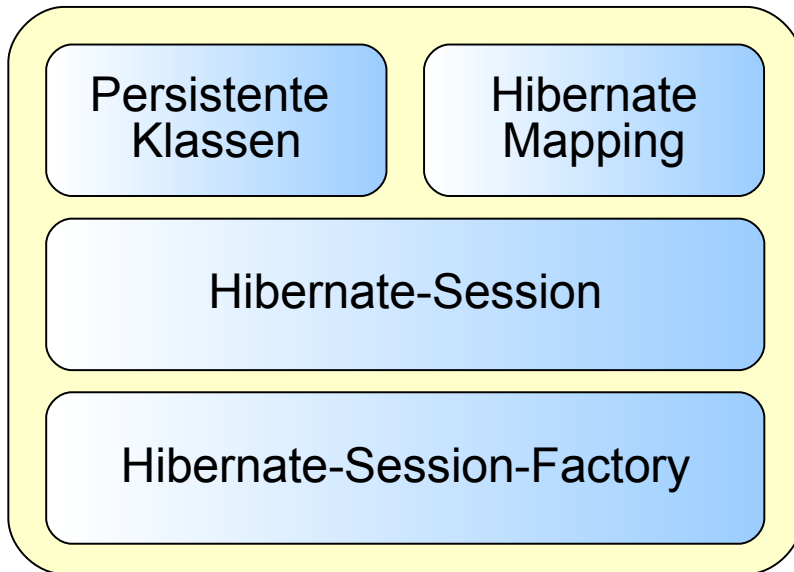
- Objektrelationales Persistenz Framework
- bietet objektorientierte Sicht auf Tabellen und Beziehungen
- breite Datenbankunterstützung
- setzt den JPA Standard um und erweitert diesen

Einsatzgebiete

- Java Anwendungen mit Datenbankbindung

Hibernate

High Level Architektur



Persistente Klassen

- Verwendung von POJOs
- keine Ableitung von spezieller Hibernate Klasse notwendig

Hibernate-Mapping

- XML Mapping oder Annotationen

Hibernate-Session

- Interaktion mit der Datenbank
- nicht thread-safe

Hibernate-Session-Factory

- lädt und hält alle Mappings
- erzeugt die Session
- eine Session-Factory pro Anwendung

Hibernate

Mapping

- Legt fest wie ein Objekt in der Datenbank persistiert werden soll

Wichtige Mapping Annotations

- @Entity, @Table(...)
- @Id, @GeneratedValue(...)
- @Column(...), @Transient
- @OneToOne, @OneToMany, @ManyToMany

Hibernate

Vorteile

- Open Source
- JDBC Anbindung wird von Hibernate übernommen
- Erleichterung durch die Arbeit mit Objekten statt mit JDBC Results
- leichter Austausch der Datenbank möglich

Nachteile

- Performanzverlust durch Automatisierung

Alternativen

- JPA RI
- TopLink

1

2

3 Praxis

3. TrionaTip Praxis

- Turnier-Domainmodell
- Der Weg eines Requests durch TrionaTip
- Turnierkonfigurator

Turnier-Domainmodell

Turnier

- Name
- Anzahl von Runden
- Anzahl von Rundengruppen
- Anzahl von Mannschaften

Tournament
- name: String
- roundGroups: List<RoundGroup>
- rounds: List<Round>
- teams: List<Team>

Turnier-Domainmodell

Runde

- Name
- Ein Rundentyp (Liga, Gruppenphase, ...)
- Anzahl von Spielgruppierungen

Rundengruppe

- Name
- Anzahl von Runden

Round
- matchGroups: List<MatchGroup>
- name: String
- roundType: RoundType

RoundGroup
- name: String
- rounds: List<Round>

Turnier-Domainmodell

Spielgruppierung

- Name
- Anzahl von Spielen

Spiel

- Startzeit
- Ort
- Mannschaft 1 + Mannschaft 2
- Qualifikationsregel
- Ergebnis

MatchGroup
- matches: List<Match>
- name: String

Match
- begin: Date
- location: String
- qualificationRule: QualificationRule
- rankingParticipant 1: RankingParticipant
- rankingParticipant 2: RankingParticipant
- result1: int
- result2: int

Turnier-Domainmodell

Platzhalter für eine Mannschaft

- Punkte
- Team

Team

- Name
- Anzahl von Mitgliedern



RankingParticipant
- globalResult: RankingParticipantResult
- team: Team

Team
- name: String
- teamMembers: List<TeamMember>

Turnier-Domainmodell

Qualifikationsregel

- Herkunft der 1. Mannschaft (Platz und Berechnungseinheit)
- Herkunft der 2. Mannschaft (Platz und Berechnungseinheit)

Berechnungseinheit

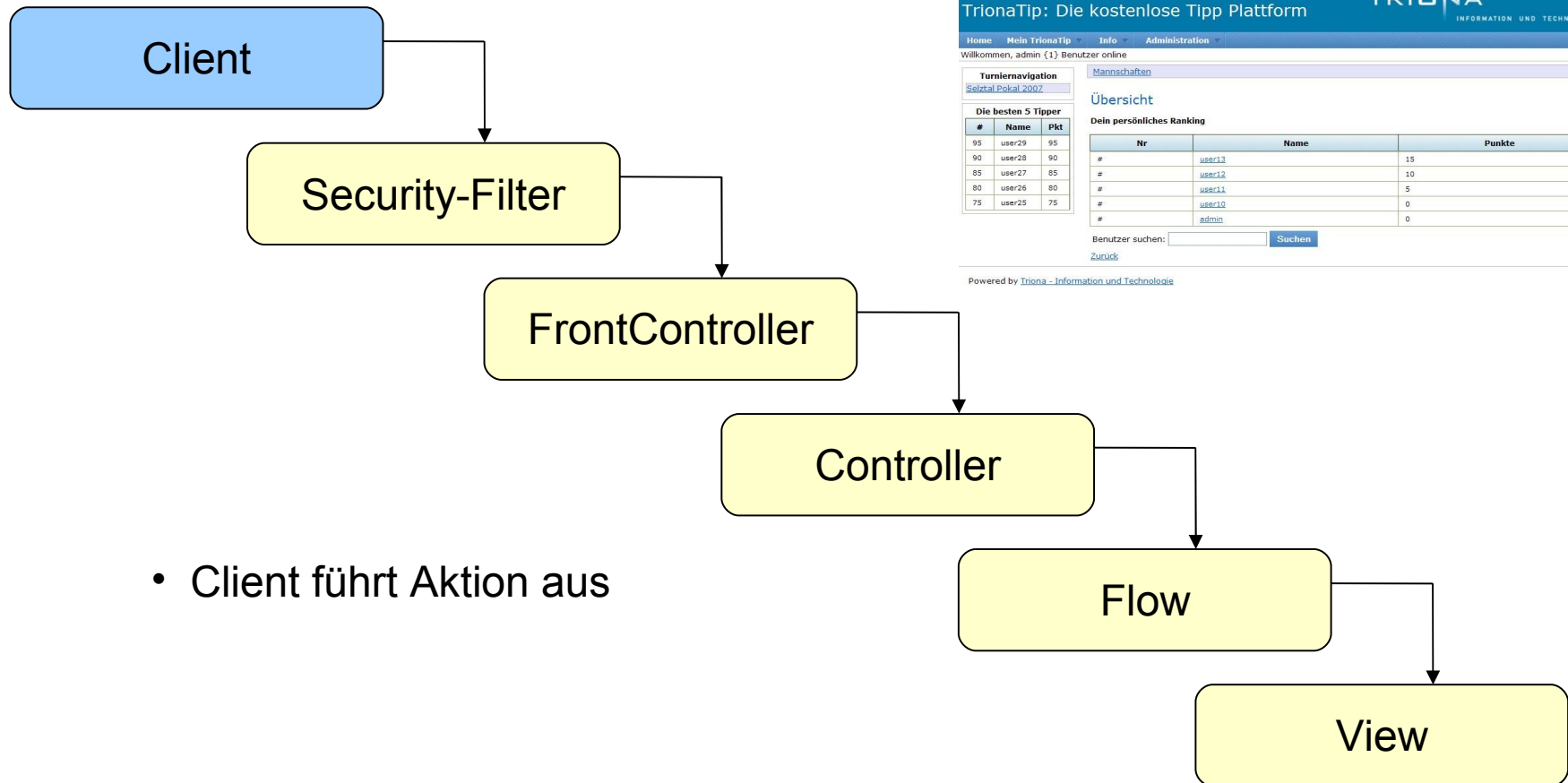
- Liste von Platzhaltern für eine Mannschaft
- boolesches Feld zur Bestimmung



QualificationRule
- participant1Rank: int
- participant1Rankable: AbstractRankingUnit
- participant2Rank: int
- participant2Rankable: AbstractRankingUnit

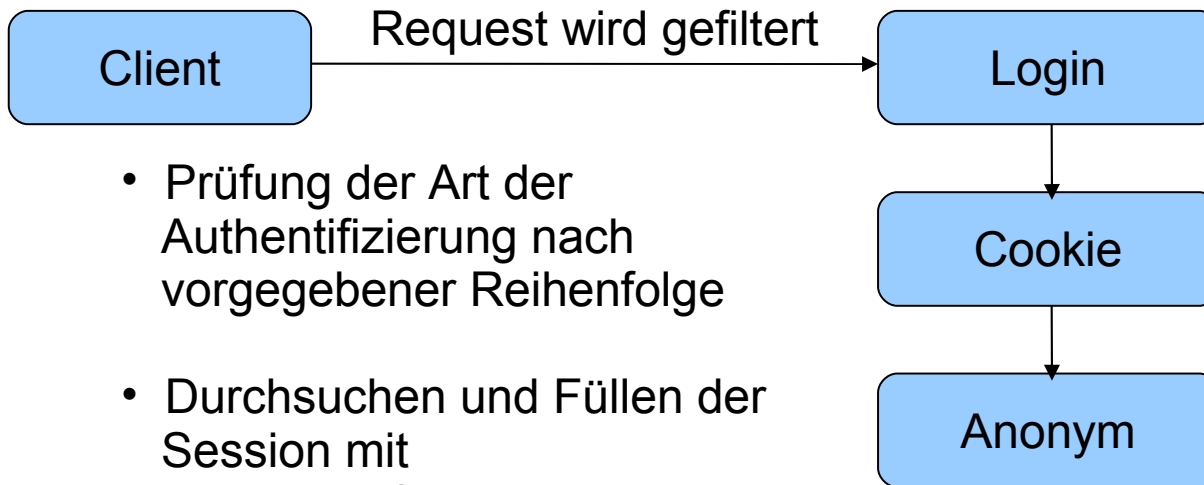
AbstractRankingUnit
- rankable: boolean
- rankingParticipants: List<RankingParticipant>

Der Weg eines Request durch TrionaTip



Der Weg eines Request durch TrionaTip

- Authentifizierung des Clients über mehrere Filter



- Prüfung der Art der Authentifizierung nach vorgegebener Reihenfolge
- Durchsuchen und Füllen der Session mit Benutzerinformationen
- verknüpfen der Authentifizierung mit zugehörigen Benutzerrechten

Client

Security-Filter

FrontController

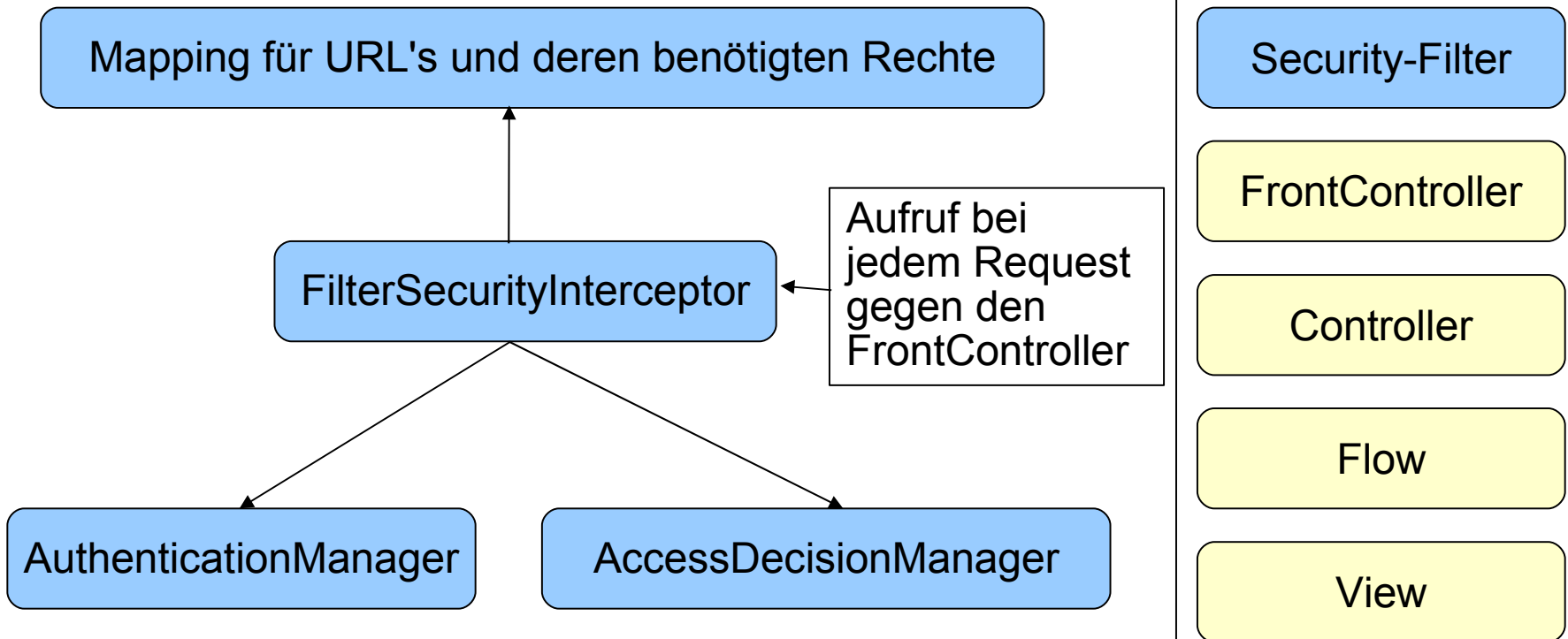
Controller

Flow

View

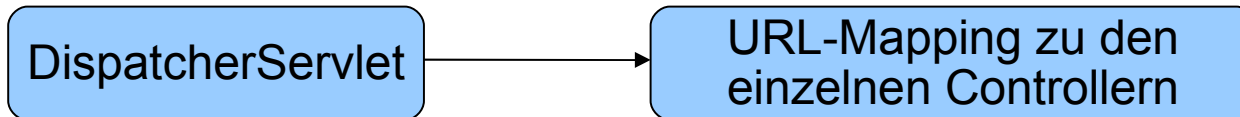
Der Weg eines Request durch TrionaTip

- Abgleich der Benutzerrechte und der benötigten Rechte

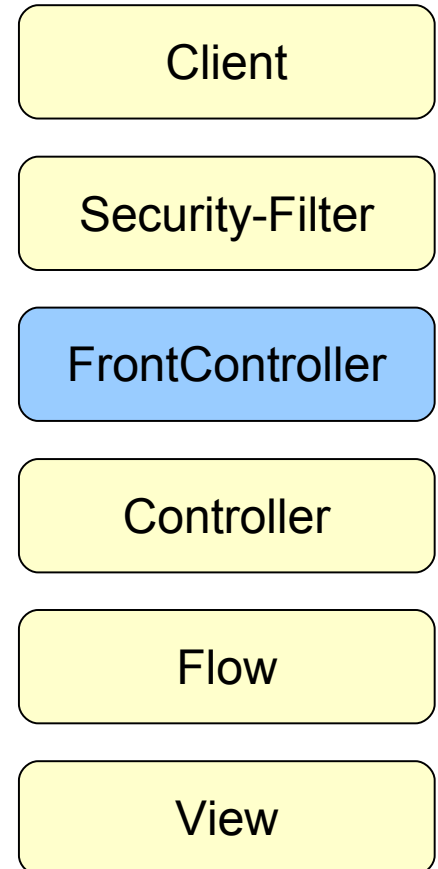


Der Weg eines Request durch TrionaTip

- Standard FrontController ist das DispatcherServlet (im „Spring Web“-Umfeld)
- folgt dem Front Controller Pattern

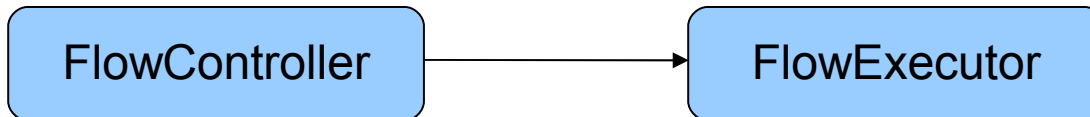


- verteilt Aufgaben an die zuständigen Controller
- in TrionaTip meistens der FlowController



Der Weg eines Request durch TrionaTip

- stellt die gewünschte Funktionalität bereit (z.B. View anzeigen, File Upload entgegennehmen)
- FlowController - Adapter zwischen DispatcherServlet und Flow-Engine



- Erstellung oder Wiederherstellung eines Flows durch FlowExecutor
- Anbindung an Spring Security

Client

Security-Filter

FrontController

Controller

Flow

View

Der Weg eines Request durch TrionaTip

- Konversation zwischen Client und Server
- Einstiegspunkt wird vom FlowExecutor bestimmt
- wird von einer Factory erstellt
- stellt Daten für die View bereit
- Definition von benötigten Rechten

Client

Security-Filter

FrontController

Controller

Flow

View

Der Weg eines Request durch TrionaTip

- Zusammensetzung der View-Komponenten
- Einbinden der bereitgestellten Daten
- definieren von Aktionen, welche der Client ausführen kann

Client

Security-Filter

FrontController

Controller

Flow

View



Quellen:

http://au.sun.com/sunnews/events/2008/techdays/download/Web_2.0_Track/TD_SYD_JSF_McDonald.pdf

http://developers.sun.com/events/techdays/presentations/2007/TD_BOS_AjaxWeb20_McDonald.pdf

<http://facelets.dev.java.net/>

<http://java.sun.com/javaee/javaserverfaces/>

<http://jboss.org/jbossrichfaces/>

<http://martinfowler.com/articles/injection.html>



Quellen:

<http://www.hibernate.org/>

<http://www.ibm.com/developerworks/java/library/j-facelets>

<http://www.ibm.com/developerworks/library/j-jsf2/>

<http://www.springsource.org/>