# Medienprojekt

## Website für die Frauenfussballmannschaft Este06/70


Ellen Schwartau, Minf9888
Julia Menzel, Minf ???

# Gliederung

# Anforderungsbeschreibung

Aufgabenstellung unseres Medienprojektes war eine Website für die Frauenfussballmannschaft des Este06/70.
Diese sollte unter Anderem zur besseren Organisation der Mannschaft selbst, der vereinfachten Präsentation der sportlichen Aktivitäten und Ergebnisse sowie dem Werben neuer Teammitglieder dienen.

Die Website sollte sich aus einem öffentlichen, sowie einem mannschaftsinternen Bereich zusammensetzen. Die genauen Anforderungen waren dabei Folgende:

**Allgemein**

- Login und Registrierungsfunktion

**Öffentlicher Bereich**

- Allgemeine Informationen
  Besucher sollen über die Seite aktuelle Informationen zu der Mannschaft, den Trainern und dem Training abrufen können.
- Weiterführende Links
  Wichtige Seiten sollen über die Mannschaftswebsite schnell zugreifbar gemacht werden.
- Kontaktaufnahme
  Es sollen Möglichkeiten geschaffen werden direkt Kontakt zur Mannschaft oder zu den Trainern aufzunehmen (beispielsweise über E-Mail Versand oder Facebook).
- Bildergalerie
  Eine Bildergalerie zum Upload von Eindrücken aus dem Mannschaftsgeschehen.
- Spielberichte
  Über die Website sollen Spielberichte zu vergangenen Spielen eingesehen werden können.

**Mannschaftsinterner Bereich**

- Forum
  Als Grundlage zur einheitlichen Kommunikation soll ein Forum dienen, in denen Threads gepostet und kommentiert werden können.
- Terminplaner
  Zur besseren Organisation zukünftiger Aktivitäten soll ein Terminplaner, bzw. ein Zeitplan zum Eintragen von Ab-/Anwesenheit, Krankheit oder längerfristigen Verletzungen dienen.
- Dokumentemanager
  Dateien wie eine Mannschaftskassenübersicht oder Formulare sollen über einen Manager hoch- und heruntergeladen werden können.
- Adressbuch
  Die Adressen der Mannschaftsmitgliedern sollen konsistent abruf- und aktualisierbar gemacht werden.

# Zeitplanung

Zeitplanung Soll / Ist, Endtermin

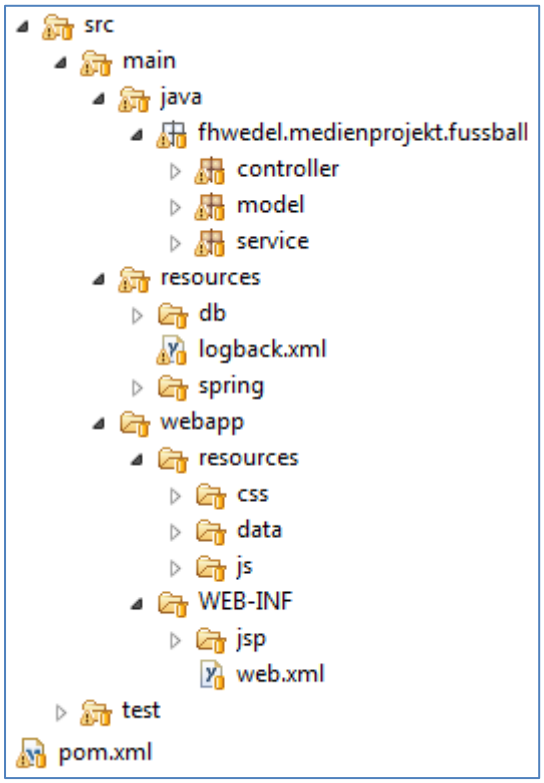
# Projektplanung und Strukturierung

# Aufgabenteilung

# Framework und wichtige Komponenten

**Programmiersprache und Framework**
Zu Beginn des Projektes entschieden wir uns dazu, die Website mit PHP umzusetzen, da uns diese Sprache sowohl aus der Uni, als auch aus dem Arbeitsumfeld bekannt war.
Schließlich nahmen wir jedoch eine Portierung auf Java vor, um unsere Kenntnisse auf eine neue Programmiersprache zu erweitern. Außerdem verwendeten wir das Framework Spring, das einige Features zur Webentwicklung wie Dependency Injection und Datenbanken Templates mitbringt und gestützt auf Java Server Pages (im Folgenden *JSP* genannt) unterstützt.

**Projektstruktur**
Zum besseren Verständnis der Struktur des Programmes, gehen wir hier grob auf den Aufbau des Projektes ein:




Das Projekt gliedert sich in die nebenstehende Ordnerstruktur auf.

Unter `src/main/java` sind die Java Dateien zu finden, die sich in Controller, Model und Service gliedern lassen.
Die Controller behandeln die Web Requests und ermitteln den logischen Namen der View, die zurückgeliefert werden soll, unter Model sind die verschiedenen Datenstrukturen der Anwendung zusammengefasst, bzw. hierin sind die darzustellenden Daten enthalten und die Services beinhalten Klassen, die die Logik ermöglichen (z.B. Datenbankzugriffe).

`Src/main/resources` umfasst die Datenbank aufsetzung und Spring Konfigurationen.

Weitere statische Ressourcen (wie css, javascript oder Dateien wie Bilder und Icons) sind unter `src/main/webapp/resources` zu finden.

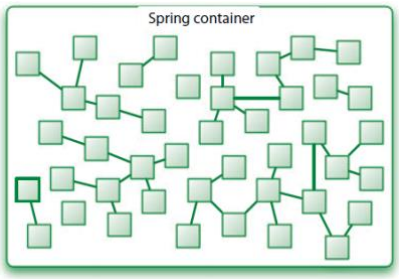
Die eigentliche View in Form von JSPs liegen unter `src/main/webapp/WEB-INF/jsp`. Entsprechende Web-Konfigurationen (z.B. `DispatcherServlet`) sind ebenfalls im `WEB-INF` Ordner in der `web.xml` zu finden.

Für Tests steht der Ordner src/test zu Verfügung.

Dependencies für das Framework können in der pom.xml festgelegt werden, über die die benötigten Ressourcen bereitgestellt werden.

**Wichtige Komponenten**
Das Spring Framework bringt einige mächtige Werkzeuge mit sich, mit denen eine Webanwendung entwickelt werden kann.
Im Folgenden werden kurz einige wichtige Komponenten, die wir verwendet haben genannt und deren Aufgabe erläutert.

In Spring-basierten Anwendungen, werden Objekte über den Spring `Container` verwaltet. Dieser kümmert sich um das Erzeugen und Verwalten, sowie das gegenseitige Verdrahten und organsisiert diese während ihres gesamten Lebenszyklus.
Zum definieren der Objekte in diesem Container wird die `Dependency Injection` verwendet. Diese basiert auf den Deklarationen, die überwiegend in der `src/main/resources/spring/applicationContext.xml` gemacht werden.




**Quelle:**
**„Spring in Action"**
**Third Edition,**
**Craig Walls**

Im `ApplicationContext` können unter Anderem Objekte (im weiteren Bean, gestützt auf die `JavaBeans`, genannt) deklariert, in andere Objekte gebunden und deren Lebenszyklus beeinflusst werden.

Das grundsätzliche Vorgehen kann an diesem Beispiel verdeutlicht werden:

Über das `<bean>` Tag können Objekte mit den Attributen id und class definiert werden. Des weiteren können zum Beispiel Konstruktorvariablen übergeben oder Properties der Klasse gesetzt werden.

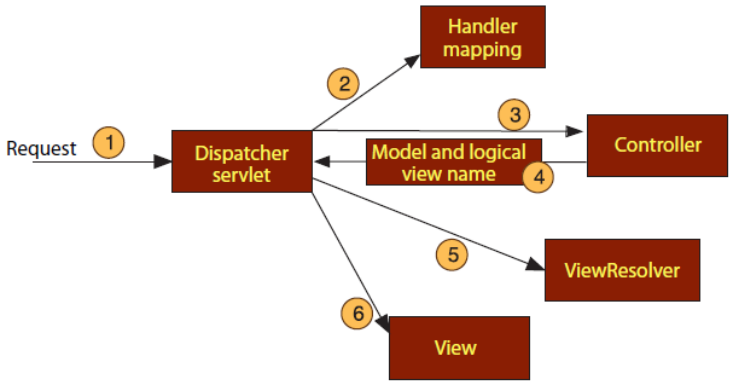
```
<bean   id="namedParameterJdbcTemplate"
        class="org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate">
    <constructor-arg ref="dataSource"/>
 </bean>
<bean   id="DataAccessUsers"
        class="fhwedel.medienprojekt.fussball.service.dataAccess.DataAccessUsers">
   <property name="namedParameterJdbcTemplate" ref="namedParameterJdbcTemplate"/>
</bean>
```

Auch über Annotationen wie `@Controller` können Beans automatisch generiert werden. `@Autowired` sorgt dafür, dass Beans aus dem Container über die id ohne weitere Konfiguration in Properties gemappt werden.

```
@Controller
public class LoginPageController {
    /* ----------------- Klassenvariablen ---------------------- */
    /** Datenbankanbindung */
    @Autowired
    private DataAccessUsers dataAccessUsers;
```

Alternativ kann über die Funktion getBean(<id>) auf die Beans zugegriffen werden. Näheres kann z.B. unter `http://docs.spring.io/spring/docs/2.5.6/reference/beans.html` nachgelesen werden.

Auch beim Bearbeiten von Request und dem Auflösen in JSPs kommen ebenfalls einige Komponenten zum Einsatz. Das unten stehende Diagramm veranschaulicht, wie ein Request mit Spring verarbeitet wird:




Quelle:
„Spring in Action"
Third Edition,
Craig Walls

Ein Request wird zunächst vom `DispatcherServlet` entgegengenommen, dessen Aufgabe es dann ist, den Request an den entsprechenden Controller weiterzuleiten. Hierfür sind die Controller mit den URLs gekennzeichnet, auf die sie hören sollen. Dies geschieht über die Angabe des `@RequestMapping` am Funktionskopf des Controllers. Zur Verdeutlichung ein Beispiel für einen Home-Controller:

```
@RequestMapping(value="/home/", method=RequestMethod.GET)
public String displayHome() {
    return "home";
}
```

Als Ergebnis liefert die Controller-Methode einen String als Indikator für die JSP die geladen werden soll.

Um nun auf die tatsächliche JSP, die geladen werden soll, schließen zu können, können über den ViewResolver Regeln festgelegt werden, wie mit der Antwort des Controllers umgegangen werden soll. In unserem Fall dient ein einfacher Post- und Prefix dazu, den Pfad der JSP zu definieren:

```
<bean class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver">
    <property name="viewResolvers">
        <list>
            <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
                <property name="prefix" value="/WEB-INF/jsp/"/>
                <property name="suffix" value=".jsp"/>
            </bean>
        </list>
    </property>
</bean>
```

Der gelieferte String „home" wird demnach zu `/WEB-INF/jsp/home.jsp` aufgelöst und verweist somit auf die eigentliche Ressource und kann zur Anzeige gebracht werden.

Für die Arbeit mit JSPs lassen sich erweiternde `Tag-Libraries` importieren, mit denen das Arbeiten mit diesen Vereinfacht wird. Es stehen beispielsweise Tags zur Verfügung, mit denen z.B. Schleifen oder Verzweigungen umgesetzt oder Formulareingaben in Objekte gebunden werden können. Ein weiterer Vorteil ist, dass so Java Befehle in prinzipieller Schreibweise von HTML-Elementen dargestellt werden können, was den Aufbau von JSPs auch optisch einheitlicher macht. Die Tag-Libraries, die in unserer Anwendung Gebraucht gefunden haben sind unter `src/main/webapp/WEB-INF/jsp/includes/taglibs_varables.jspf` zu finden.

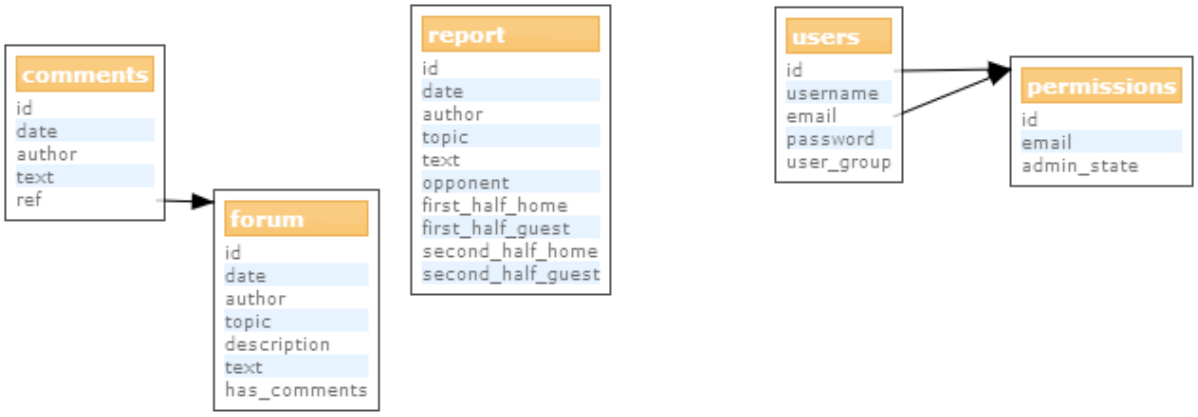
Weiteres zu Spring kann unter `http://spring.io/` bzw. `http://spring.io/docs` nachgelesen werden.

# Datenbankarbeit

Unsere Wahl bezüglich des Datenbankverwaltungsssystems fiel auf MySQL, da dieses uns sowohl aus der Vorlesung als auch aus dem Arbeitsumfeld bereits bekannt war. Auch wegen weiterer Vorteile wie betriebssystemunabhänggkeit, Verfügbarkeit als Open Source System und häufige Anwendung als Basis für dynamische Webanwendungen und eine somit lebhafte Community, war MySQL für unser Projekt gut geeignet. Zudem sind Hosts verfügbar, die Webanwendungen mit Java in Zusammenarbeit mit MySQL unterstützen.

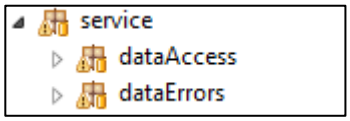
Unsere Datenbank gliedert sich in folgende Tabellen auf:

DATENBANKSCHEMA, AUF FOREIGN KEYS UND CONSTRAINTS EINGEHEN




Aktualisierung unter: file:///C:/Program%20Files%20(x86)/CreateDadabaseDiagram/index.html

Zur Datenbankarbeit mit Java stehen im Package service.dataAccess sowie Service.dataErrors Klassen zum Datenbankzugriff sowie dem Abfangen und Behandeln von Fehlern in den Benutzereingaben zur Verfügung.




Hierbei greifen wir vor allem auf das `NamedParameterJdbcTemplate` zurück, bei dem im Vorteil gegenüber dem `NamedParameterJdbcTemplate` den Vorteil hat, dass die Abfrageparameter über einen zugewiesenen Namen zugreifbar gemacht werden können, die Reihenfolge der Parameter spielt somit keine Rolle.
Die jdbcTemplates arbeiten mit der Templating Methode, das heißt, das nur die eigentlichen Datenbankabfragen geschrieben werden müssen, Verbindungsauf- und -abbau, sowie das Behandeln von Exceptions werden von den Templates übernommen, was die Datenbankarbeit vereinfacht und übersichtlicher macht.

```
/**
 * Liefert einen User ausgehend von einer id.
 * @param   id      int     ID des Users
 * @return  User            ausgelesener User
 */
private User getUserById(int id) {
    final String SQL_SELECT_USER_BY_ID
        = "SELECT * FROM " + Constants.dbUsers + " WHERE (id = :id)";
    /* Name-Wert Paare für Abfrage festlegen */
    Map<String,Object> params = new HashMap<String,Object>();
    params.put("id", id);
    /* User auslesen */
    ArrayList<User> res
        = (ArrayList<User>) this.namedParameterJdbcTemplate.query(
                                SQL_SELECT_USER_BY_ID,
                                params,
                                this.userMapper
                            );
    /* Sichergehen, dass genau ein User gefunden wurde */
    assert (!res.isEmpty())
        : "Über die angegebene id konnte kein User gefunden werden";
    assert (res.size() == 1)
        : "Über die angegebene id konnte kein eindeutiger User gefunden werden.";

    return res.get(0);
}
```

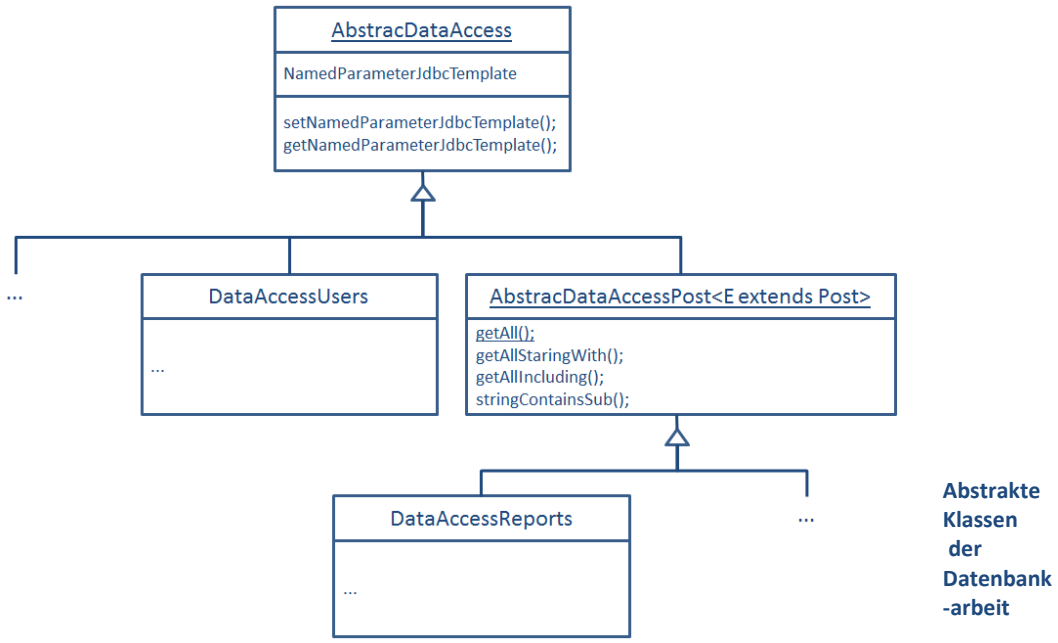
Um die Funktionsweise zu verdeutlichen, im Folgenden ein Beispiel zur Abfrage eines Users über dessen ID (`DataAccesUser.java`):

Die eigentliche SQL Abfrage wird als String

repräsentiert, wobei Variablen, die mit „:“ beginnen, über die `HashMap` ein Wert zugewiesen wird.

Zur Abfrage wird in diesem Beispiel außerdem ein `RowMapper` benötigt, der den Attributen der Klasse User Spalten des Ergebnisses zuweist.

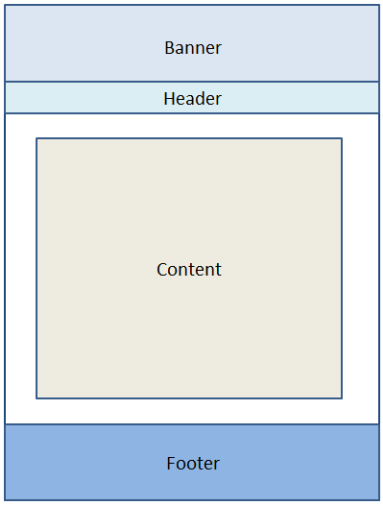
Um nicht in jeder DataAccessKlasse das `NamedParameterJdbcTemplate` setzen zu müssen, extenden alle DataAccess-Klassen die `AbstractDataAccess.java`, die sowohl das jdbcTemplate, als auch die Set- und Get-Methode hierfür implementiert. Auch die `AbstractDataAccessPost<Post>` fasst einige Aufgaben für die erbenden Klassen zusammen:




# Konzept und Implementierung

**Design**

Das Aussehen der Webiste musste sowohl gut bedienbar, als auch übersichtlich und ansprechend sein. Da im Fussballsport sowohl junge als auch ältere Männer und Frauen tätig sind, musste sie dafür ausgelegt werden eine breite Zielgruppe anzusprechen.




Wir entschieden uns für einen Grundaufbau mit Folgenden Bestandteilen:

Am Kopf der Seite sollte ein Banner mit einem Bild passend zum Thema Fussball stehen.

Mit dem Header folgt darauf die Navigation, die je nachdem, ob ein Besucher ein- oder ausgeloggt ist, andere Links zur Verfügung stellt.

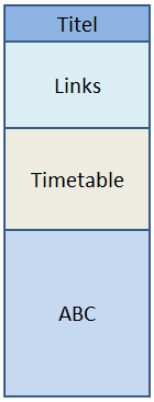
Am Saum der Seite erscheint der Footer mit Kontaktdaten, wichtigen Shortcuts, sowie Login und Registrierfunktion, sollte der Besucher nicht eingeloggt sein.

Zwischen Header und Footer wird der seitenspezifische Content dargestellt.

Um die verschiedenen Seiten mit einem konsistenten Design auszustatten, dieses aber dennoch an die individuellen Bedürfnisse anpassen zu können, entwarfen wir verschiedene Grundlayouts, die dann übergreifend Anwendung finden konnten.

Das Grunddesign lässt sich in zwei Teile gliedern: Der Sidebar und dem eigentlichen Content.

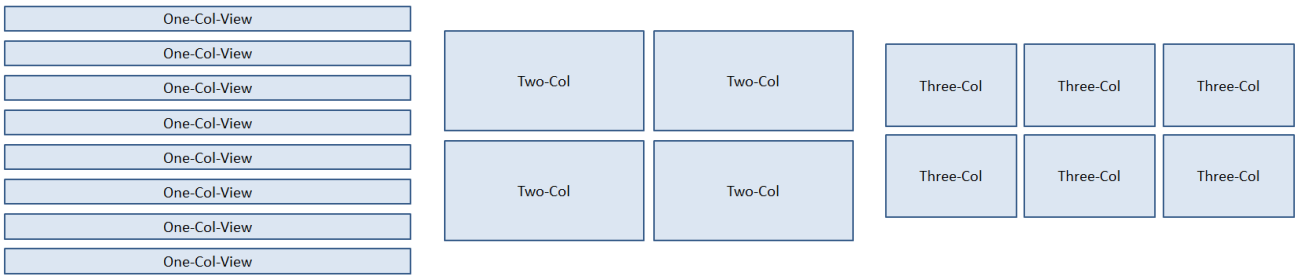
Die Sidebar setzt sich zusammen aus dem Titel der Seite, darunter folgend weitere Links, bzw. Optionen, einem Timetable, bzw. einer alphabetischen Schaltfläche zur Selektion von Content zu einem bestimmten Monat oder Buchstaben.




Die einzelnen Bestandteile sind einzeln in- bzw. exkludierbar, je nach Bedarf kann diese dann entsprechend bereitgestellt werden.

Der Content der Seite steht entweder allein oder neben, bzw. unter der Sidebar.

Der Content selbst kann in drei verschiedenen Layouts organisiert werden – einer ein-, zwei- oder dreispaltigen Einteilung.




Hierbei besteht bei dem einspaltigen Design außerdem die Möglichkeit, den Content per Mausklick ein- oder auszublenden, um für mehr Übersicht zu sorgen.

**Farbgebung**

Als Grundlage zum Design der Seite wählten wir verschiedene Grautöne und die Farbe Grün, zum setzen bestimmter Akzente als Anspielung auf den Fussballrasen.

**Responsive Design**

Aufgrund der Vielzahl an verschiedenen Devices und den damit einhergehenden unterschiedlichen Display-Maße und Auflösungen war eine weitere Anforderung an das Design die Kompatibilität mit möglichst vielen Formaten, auf denen die Website angezeigt werden könnte.

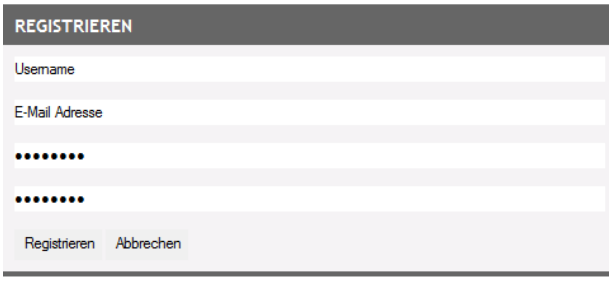
Hierfür setzten wir vor allem relative Größenangaben und Media-Queries ein, mit denen wir zum Beispiel auf das Floating Verhalten, das Ein- oder Ausblenden von Bildern oder Größen- und Abstandsverhältnisse der Elemente Einfluss nahmen.

Im Folgenden ein Beispiel des Verhaltens des Three-Col-Views in Abhängigkeit zur Bildschirmauflösung (von links nach rechts 850px Breite, 760px Breite, 605px Breit, 400px Breite):


Zunächst wird das Design wie gewollt dreispaltig angezeigt, anschließend werden die Bilder weggelassen und auf ein zwei-spaltiges Design geswitcht, bis der Inhalt nur noch einspaltig in gleicher Höhe, anschließend einspaltig mit automatischer Höhe dargestellt wird.

**Login und Registrierung**

Die Registrierung läuft in einem mehrstufigen Verfahren ab - Zunächst muss die Email-Adresse, mit der ein User sich registrieren möchte, durch einen Administrator zugelassen werden. Erst anschließend kann sich ein User mit entsprechender E-Mail einmalig anmelden. Bei Vergabe der Registrierungserlaubnisse können Administratorrechte vergeben werden. Diese Rechte können über einen Administrator auch später bearbeitet werden.

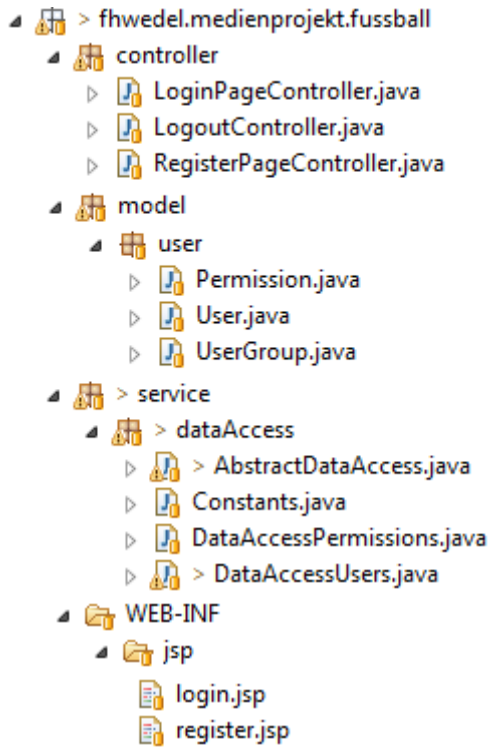




Das Registrierungsforumlar besteht aus vier Pflichtfeldern für den Usernamen, die Email-Adresse, welche der abgespeicherten in Datenbank entsprechen muss sowie zwei Passwortfeldern, die gleich sein müssen.
Nach der Registrierung ist der User automatisch angemeldet .
Der Login erfolt nach Registrierung über den angegebenen Usernamen und das Passwort.
Nach erfolgreicher Anmeldung wird der User auf die Landingpage für eingeloggte Besucher weitergeleitet (diese enthält Shortcuts zu wichtigen Seiten für angemeldete Benutzer, wie Adressbuch, Forum usw.).

Bei beiden Formularen wird auf Labels verzichtet und stattdessen mit Placeholdern gearbeitet, die angeben, was in das jeweilige Feld einzutragen ist.

Zu erreichen sind Registrierung und Login über Schaltflächen im Footer, sowie im Header. Ist ein User eingeloggt, erscheinen an deren Stelle Links für den Logout.




Die Logik auf drei grundlegenden Klassen aufgebaut:

- `Permission.java` repräsentieren die Registrierungszulassungen
- `User.java` bilden die Grundlage für die User
- Und `UserGroup.java` beinhaltet die möglichen Rollen, die ein User innehaben kann (genaueres siehe: Zugriffsrechte und Editieren von Content)

Die Datenbankarbeit wird übernommen von den DataAccess-Klassen `DataAccessPermissions.java` und `DataAccesUser.java`.

Die involvierten Controller sind der `LoginPageController.java`, `LogoutController.java` und der `RegisterPageController.java`.

Die Darstellung für den Login und die Registrierung ist über die `login.jsp` und die `register.jsp` definiert. Die `register.jsp` enthält außerdem die Schaltflächen zum hinzufügen neuer, bzw. eine Übersicht über aktuelle Permissions, die dort auch bearbeitet werden können. Sichtbar sind die zuletzt genannten Bestandteile nur für Administratoren.

**Forum und Spielberichte**

...

**Adressbuch**

**...**

**Galerie**

**...**

**Zugriffsrechte und Editieren von Content**

Die Website gliedert sich in einen öffentlichen und einen authorisierten Bereich, der je nach Login-Status der Besuchers zugreifbar ist. Die öffentlichen Seiten (siehe Anforderungen) sind für alle sichtbar, wohingegen nur angemeldete Nutzer den authorisierten Bereich der Website einsehen dürfen. Dazu zählen zum Beispiel das Forum oder das Adressbuch.

Zusätzlich besitzen die User der Seite verschiedene Rechte – sie lassen sich aufteilen in Administratoren und normale User.
Administratoren (gehören zur UserGroup `USER_GROUP_ADMIN`) besitzen gegenüber normalen Usern die Rechte, Content zu verfassen, ihn zu editieren oder zu löschen und neue Registrierungszulassungen zu vergeben.
User ohne Administratorrechte (sie gehören zu UserGroup `USER_GROUP_NO_ADMIN`) können ausschließlich ihre eigenen Daten (wie z.B. ihre Adresse) und selbst verfasste Kommentare zu Foreneinträgen löschen.
Eine spezifischere Einteilung der User sahen wir als nicht nötig an, da die Userzahl der Fussballseite relativ überschaubar bleibt.

Fehlt noch: UMSETZUNG USERBEZOGENER CONTENT

**Eingabeüberprüfung**

...

# Fazit