STW2 Projekt SS08:

SWTKal Webclient

Ein webbasierte JSP/Servlet Client für die SWTKal-Schnittstelle

Projektteilnehmer:

Markus Balsam
Dennis Böing
Markus Bollenberg
Michael Blitzner
Sebastian Kroll (Projektleiter)
Stephan Osterkamp
Christian Schleiffer
Dominik Seifert

Inhalt

1. Kurzübersicht

In unserer SWT2-Projektgruppe haben wir einen Webbasierten Client für die SWTKal Schnittstelle erstellt. Das Projekt umfasst die Spezifikation des Leistungsumfanges, die Festlegung der zu nutzenden Anwendungsstruktur, sowie die Implementierung des Clients.

Als Programmiersprache haben wir JavaServerPages bzw. Servlets verwendet. Diese generieren HTML und versenden es, zusammen mit JavaScripts und Grafiken von einem ApplicationServer, an alle anderen Netzwerkteilnehmer.

Die grafische Gestaltung wurde in CSS und mit Grafiken realisiert. Es wurde besonderen Wert darauf gelegt, das Design vom Inhalt zu trennen.

Diese unterschiedlichen Aufgaben und Themengebiete konnten passend auf die 8 Projektmitglieder verteilt werden. Zusätzlich wurde die Aufgabenverteilung durch Softwaretechnische Case-Werkzeuge, wie z.B. einem Versionskontrollsystem unterstützt.

2. Verwendete Programmier- und Laufzeitumgebung

1. Servertechnologien

- Apache Tomcat 6.0.16
- MySQL Server 5.0.45 Community Edition
- MySQL GUI Tools 5

2. Programmiersprachen

- Java SDK (JDK 1.6)
    - JSP
    - Servlets
    - Javascript
- HTML 4
- CSS 1/2/3

3. Verwendete Schnittstellen

- SWTKal Server
    - SimpleServer, während der Entwicklung
    - JPAServer, während der Testphase
        - Java Persistence API (Hibernate Version 3)
        - MySQL Connector (JDBC Version 5.0.4)

4. Casewerkzeuge

- Eclipse Platform (J2EE) 3.3.2
    - Subclipse 1.2.4
    - Sysdeo Tomcat Plugin 3.2.1
    - Web Standard Tools
- SVN Server
- e-Le@rning zur Kommunikation und Koordination, sowohl intern als auch extern
- Mozilla Firefox 2
    - Firebug Extension (Javascript und CSS Debugging)

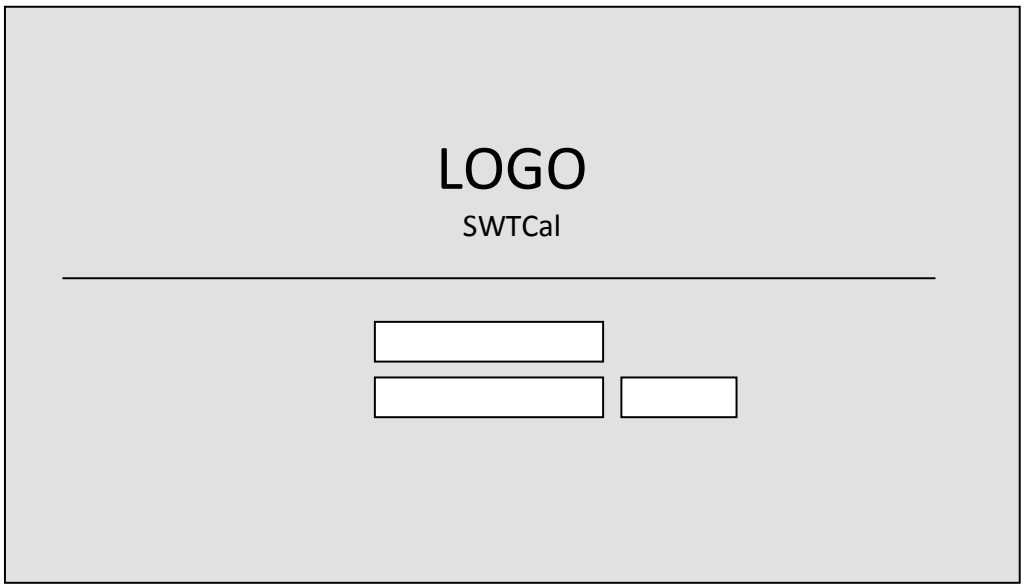
3. Spezifizierter Funktionsumfang

Der Client bietet eine gewisse Grundfunktionalität, sodass er eingeschränkt nutzbar ist. Zur Festlegung dieser Funktionalität werden im Folgenden verschiedene Sichten beschrieben. Beispielhaft sind darunter mögliche Strukturierungen der Oberfläche skizziert.

1. Loginmaske

Die Loginmaske bietet in unserer Ausführung keinerlei weitere Funktionalität, bis auf das Anmelden selbst. Es besteht weder die Möglichkeit neue Konten anzulegen, noch eine gewisse Basisfunktionalität als Gast zu nutzen.

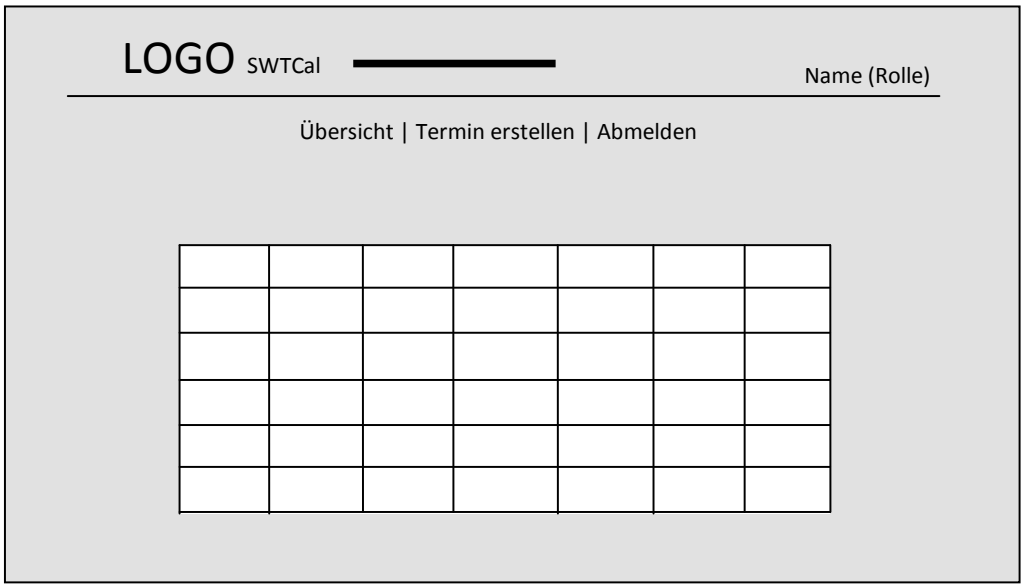
An optischen Elementen werden wir uns daher auf zwei Eingabefelder für Name und Passwort, sowie auf eine Schaltfläche zum Anmelden beschränken.



2. Übersicht

Die Übersicht soll als eine Art Startseite dienen, welche direkt nach erfolgreichem Anmelden angezeigt wird.

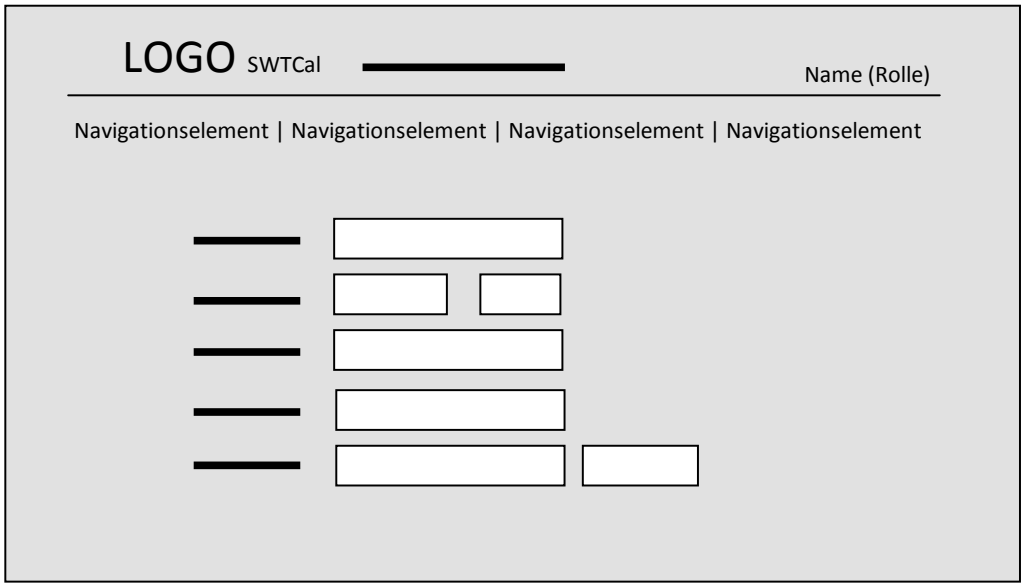
Hier findet man eine Navigationsleiste, welche auch in allen anderen Sichten (mit Ausnahme der Loginmaske) zu finden sein wird. Diese bietet die Möglichkeit einen neuen Termin zu erstellen, sich abzumelden oder aus einer anderen Sicht zurück auf die Übersicht zu gelangen. Ebenfalls sollte ersichtlich sein, als welcher Nutzer und in welcher Rolle man angemeldet ist.



Im restlichen Bildbereich zeigt die Übersicht die aktuelle Woche in gewohnter Form als Tabelle an. Termine sind in dieser Tabelle pro Tag chronologisch geordnet und mit der Anfangszeit und einem kurzem Text versehen. Dies können, je nach Platz, die ersten Worte/Zeilen der Terminbeschreibung selber sein.

3. Neue Termine eingeben / bestehende Termine bearbeiten

Da diese beiden Aufgaben recht verwandt sind, was die Eingabemasken angeht, werden wir hier in beiden Fällen dieselbe Struktur verwenden. Dennoch halten wir diese Seite für jeden Anwendungsfall doppelt im System vor, hauptsächlich aus Gründen der Aufgabenverteilung.



Beim Neuanlegen eines Termins, wären die Eingabefelder der Maske leer, lediglich ein Auswahlfelder, wie z.B. Terminkategorien oder ähnliche könnten bereits Standardwerte anzeigen. Möchte man einen bestehenden Termin ändern, so wird die Maske mit den aktuellen Werten ausgefüllt und bietet die Möglichkeit diese zu ändern.

Eine Komfortfunktionalität für diese Sicht, wird eine browser-seitige Verifikation der eingegebenen Daten sein. So wird es nicht nötig sein Kontakt zum Server aufzunehmen, um fehlerhafte eingaben zu erkennen. Das verkürzt zum einen die Reaktionszeit für den Anwender, zum anderen verringert es die Last für den Server.

4. Projektstand

In der recht kurzen Projektphase des Semesters haben wir den Webclient zuerst komplett entworfen und dann begonnen diesen zu implementieren. Da die Teilaufgaben unterschiedlich aufwendig und waren, wurden Teile bereits recht schnell vollkommen funktionsfähig fertiggestellt. Als zeitaufwendigste, und auch komplizierteste Aufgabe hat sich die Übersichtsseite mit dem Wochenkalender herausgestellt. Wir haben bis zu letzt versucht eine möglichst fehlerfreie Version zu erstellen.

Eine Woche vor Semesterende haben aber auch wir die Entwicklung eingestellt und sind mit ein paar kleinen Fehlern im Programm verlieben.

Es folgt eine Übersicht über die Implementierte Funktionalität und die noch bestehenden Probleme. Stand ist die die SVN-Revision #112 von Donnerstag, dem 25.06.2008

Implementiert, bisher fehlerfrei	Bestehende Fehler / Probleme
Session/Anmeldung ☑ Als bestehender Nutzer einloggen ☑ Eingeloggt bleiben (Session) ☑ Manuelles abmelden	**Session/Anmeldung** ☒ Wenn die Session abläuft während man Termine bearbeitet/erstellt kann es zu einer Null-Pointer-Exception kommen
Wochenübersicht ☑ Termine farblich hervorgehoben ☑ Terminüberschneidungen ☑ Termine über mehrere Tage ☑ Termine anwählbar ☑ Wochen vor/zurück blättern	**Wochenübersicht** ☒ Bei Terminen, die über Wochengrenzen hinweg laufen kann es Darstellungsprobleme geben ☒ Beim schnellen blättern durch die Wochen kann ein Serverfehler auftreten
Termine Erstellen/Bearbeiten ☑ Termin erstellen ☑ Termin ändern ☑ Termin löschen ☑ Autokorrektur bei der Eingabe	
Allgemeines ☑ Anbindung an JPA Server oder SimpleServer	**Allgemeines** ☒ Lange Wartezeit beim Einloggen auf JPA Server

5. Softwarestruktur

Der Webclient für den SWTKal-Server wird mit Java Webtechnologien implementiert. Wir haben uns in unserem Projekt auf die Verwendung von JavaServerPages und Servlets beschränkt. Je nach Funktionalität haben wir diese für unterschiedliche Aktionen verwendet.

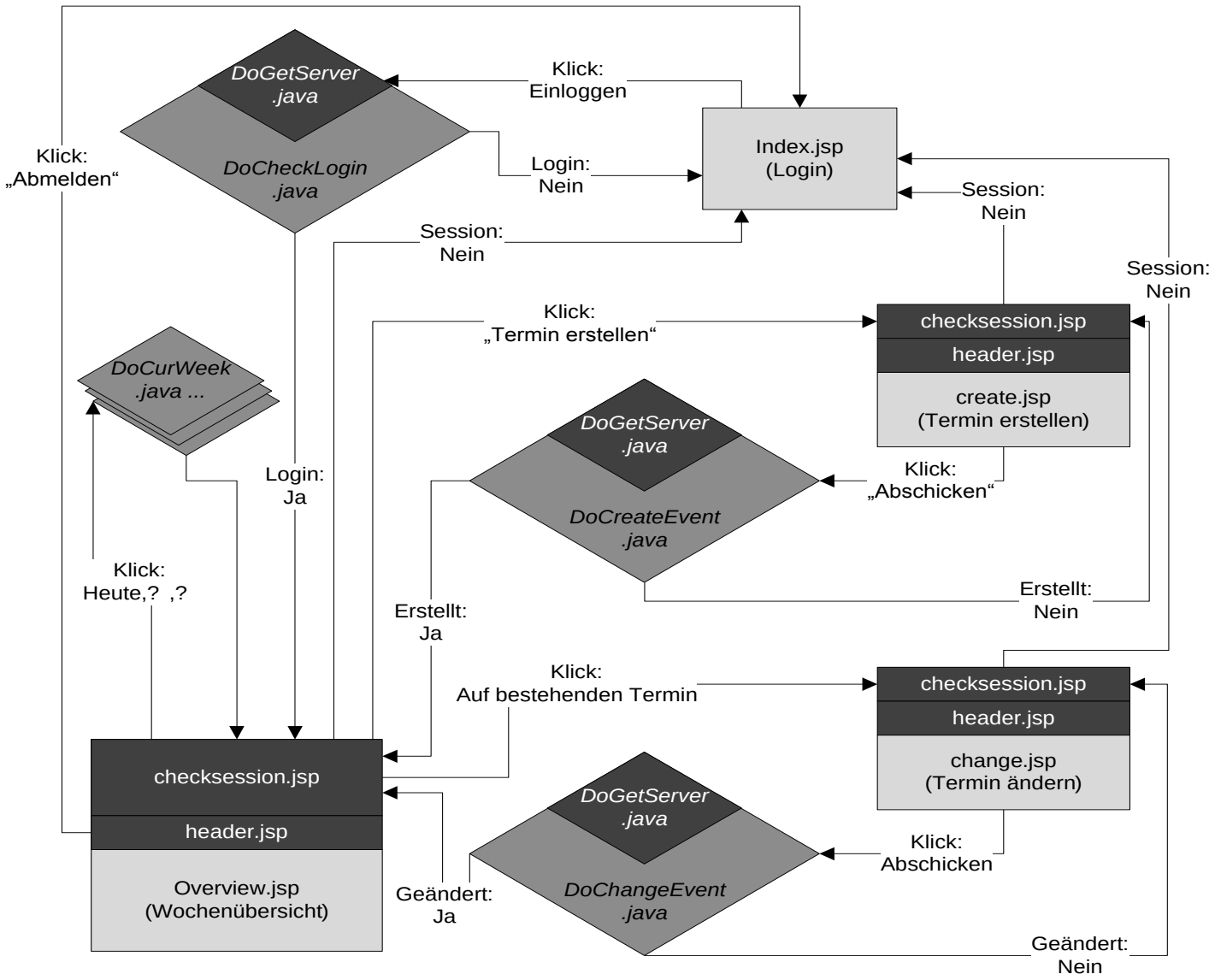
Für die Oberflächengestaltung bieten sich die JSPs an, da sie im Grunde genommen einfache HTML Seiten sind, welche um Java-Bereiche erweitert werden. Für Aufgaben, die nur im Hintergrund laufen, haben wir Servlets eingesetzt, da hier keine Browserausgaben nötig sind und wir uns hier auf die reine Java Funktionalität beschränken können.

Die Oberflächengestaltung selbst ist in zwei Bereiche getrennt. Die JavaServerPages, und teilweise auch die Servlets, generieren das HTML nur als Grundgerüst, welches an den Browser geschickt wird. Hier werden Inhalte und Strukturen erzeugt. Die grafische Ausgestaltung durch Farben und die Positionierung wird von Cascading Style Sheets erzeugt. Durch diese Trennung von Design und Inhalt können redundante Formatierungen vermieden werden und alle Oberflächen können durch das Einbinden einer CSS-Datei gleichzeitig und gleichförmig verändert werden.

Auf den Oberflächen create.jsp und change.jsp werden zusätzlich JavaScripts verwendet. Diese prüfen bereits auf Seiten des Clients ob die eingaben Korrekt sind und können auf eventuell fehlende Angaben hinweisen. Dies kann ohne eine Verbindung zum Server geschehen, was den Server dauerhaft entlasten kann.

Auf der folgenden Seite ist ein Flussdiagramm zu finden, welches die Übergänge und die Zusammenhänge (Includes) der unterschiedlichen JSPs und Servlets verdeutlicht.

Die rechteckigen Formen sind Oberflächen, welche durch JSPs implementiert wurden. Falls noch andere JSPs benötigt werden, sind diese als dunkle Rechtecke darüber dargestellt. Funktionalität, die nur im Hintergrund abläuft und in Form von Servlets ausgeführt wird, ist als Raute dargestellt. Falls für einen Übergang zwischen zwei Zuständen Bedingungen erfüllt sein müssen, oder diese als Resultat einer Benutzeraktion geschehen, sind diese am jeweiligen Pfeil kurz erklärt. Die Servlets „DoCurWeek“, „DoPrevWeek“ und „DoNextWeek“ wurden aufgrund von nahezu identischer Funktionalität im Diagramm optisch zusammengefasst.


DoGetServer
.java
DoCheckLogin
.java
Klick:
Einloggen
Klick:
„Abmelden“
Login:
Nein
Index.jsp
(Login)
Session:
Nein
Session:
Nein
Session:
Nein
Klick:
„Termin erstellen“
checksession.jsp
header.jsp
create.jsp
(Termin erstellen)
DoCurWeek
.java ...
DoGetServer
.java
DoCreateEvent
.java
Klick:
„Abschicken“
Login:
Ja
Klick:
Heute,? ,?
Erstellt:
Ja
Erstellt:
Nein
Klick:
Auf bestehenden Termin
checksession.jsp
header.jsp
change.jsp
(Termin ändern)
checksession.jsp
header.jsp
Overview.jsp
(Wochenübersicht)
DoGetServer
.java
DoChangeEvent
.java
Klick:
Abschicken
Geändert:
Ja
Geändert:
Nein

6. Aufgabenverteilung

Unsere Projektgruppe bestand aus insgesamt 8 Mitgliedern, was eine Aufgabenverteilung nicht nur begünstigt sondern auch nötig macht. Die Verteilung selbst ergab sich fast komplett von selbst. Nach der Einarbeitungsphase und den Projektvorträgen, in welchen die verschiedenen Mitglieder bereits in unterschiedlichen Themengebieten Erfahrungen gesammelt hatten, hat man sich in 2er Teams organisiert und so 4 Aufgabenbereiche aufgestellt.

1. Sessionverwaltung, Erstellen der Projektstruktur, Projektleitung

Markus Bollenberg, Sebastian Kroll

Zur Beginn der Projektarbeit hat dieses Team damit begonnen, eine Grundlage zu schaffen auf deren Basis die Teams mit ihren Arbeiten beginnen konnten. Diese Grundlage beinhaltet das Bereitstellen einer Arbeitsumgebung (Tomcat, Eclipse), sodass alle Projektmitglieder auf der gleichen Basis entwickeln konnten und alle Zugang zum Versionsmanagement hatten.

Nachdem das Grundgerüst der Oberflächen erstellt wurde, hat dieses Team die Umsetzung in ein lauffähiges Webprojekt vorgenommen. Es wurden Dummy-Servlets erstellt, welche ab diesem Zeitpunkt im Versionsmanagement zur Verfügung standen und mit deren Implementierung begonnen werden konnte.

Ab diesem Zeitpunkt konnten die unterschiedlichen Gruppen größtenteils nebeneinander, also ungestört von den anderen, an ihren Aufgaben arbeiten. Auf Wunsch war es aber auch jederzeit möglich, den aktuellen Projektstand aus dem Versionsmanagement abzurufen.

Wir konnten nun damit beginnen die Sessionverwaltung ins Projekt zu übernehmen, welche im Rahmen der Seminarvorträge bereits vorbereitet wurde. Das beim Login erzeugte Sessionobjekt wurde dann in den anderen Gruppen benutzt, z.B. um den aktuellen Benutzer und das aktuelle Datum für die Oberflächendarstellung erreichbar zu machen.

2. Wochenübersicht, Terminausgabe

Michael Blitzner, Stephan Osterkamp

Diese Gruppe hat sich hauptsächlich mit der Darstellung der Wochenübersicht beschäftigt.

Als Grundlage haben wir die overview.jsp benutzt, die auch schon von Christian, Markus und Sebastian erstellt, bzw. genutzt wurde.

Durch unsere Arbeit werden die Termine dynamisch in die einzelnen Zellen der Tabelle gefüllt. Nur zu Stunden, in denen auch ein Termin beginnt oder endet steht dessen Kurzinformation.

Zwischen Begin und Ende dieser Termine haben wir die Felder markiert, um die Zeiträume anzuzeigen, in denen die Termine fortwähren.

Von der Wochenübersicht lassen sich die Termine auch bearbeiten. Zu diesem sind die Terminüberschriften gleichzeitig Links zum Bearbeitungsformular. Zur Weitergabe der Informationen werden die Datenbank internen IDs der Termine genutzt. Nach Bearbeitung oder wenn ein neuer Termin eingestellt wurde, gelangt man automatisch zur Wochenübersicht der betroffenen Woche zurück.

3. Einfügen / Ändern von Terminen, allgemeine Verbesserungen

Markus Balsam, Dennis Böing

Die Hauptaufgabe des Teams bestand darin das funktionale Gerüst des Projekts in Form von Servlets und JSPs zu erstellen. Zu Beginn des Projekts wurde ausschließlich gegen die Schnittstelle des „SimpleServers“ programmiert, welcher keinen Zugriff auf eine Datenbank hatte, sondern die Objekte nur im lokalen Arbeitsspeicher ablegen konnte. Dies hatte den Vorteil, mögliche Fehler bei der Kommunikation mit der Datenbank auszuschließen und sich somit voll auf die Entwicklung der geplanten Funktionalitäten zu konzentrieren.

Zunächst hat sich das Team damit beschäftigt das Erstellen und Speichern von Terminen in dem entsprechenden Servlet (DoCreateEvent.java) zu programmieren. Gleichzeitig wurde aber auch die Kommunikation zwischen JSPs und Servlets (und umgekehrt) realisiert. Als diese Funktionalität gegeben war, wurde nicht mehr mit dem „SimpleServer“, sondern mit einem „JPA Server“ gearbeitet. Dadurch war es nun möglich, (Termin-)Objekte in einer Datenbank zu speichern, und es hatte nun auch jeder Termin eine eindeutige ID. Als dann schließlich noch die Methode *getTermin(int id)* für den „JPA Server“ zur Verfügung stand, die den Zugriff auf Terminobjekte über deren ID ermöglichte, konnte damit begonnen werden, Funktionen wie das Ändern und Löschen von Terminen zu implementieren.

Zusätzlich hat sich das Team um kleinere Optimierungen gekümmert und Funktionen, wie das Vor- und Zurückblättern in der Wochenübersicht, realisiert.

Das Team konnte relativ unabhängig von den anderen Gruppen arbeiten, nutzte aber zum Testen der Funktionalität immer wieder den aktuellen Fortschritt des Oberflächenteams und des Teams für die Terminausgabe. Außerdem wurde die Arbeit des Sessionverwaltungsteams in den Code eingebunden.

4. Erzeugung des Grundgerüst, Oberflächendesign, clientseitiges Scripting

Christian Schleiffer, Dominik Seifert

Diese Gruppe hat sich ausschließlich mit den Projektteilen beschäftigt, welche auf der Clientseite, also im Browser zu sehen sind.

Direkt zu Beginn der Implementierungsphase haben wir, nach den vorher festgelegten Strukturen, die verschiedenen Sichten in HTML erstellt. Hier wurden noch keine genauen Designaspekte festgelegt, lediglich das Grundgerüst wurde Strukturiert und wiederkehrende Elemente wurden in Designklassen eingeteilt. Auf diese Weise können sie per CSS adressiert werden. Als das Grundgerüst fertiggestellt war, konnten die anderen Gruppen beginnen, die leeren Container mit dem Javacode zu füllen.

Gleichzeitig haben wir begonnen, eine Komfortfunktionalität zu implementieren. Auf den beiden Masken „Termin erstellen“ und „Termin ändern“ können nun Plausibilitätsüberprüfungen, auf Seiten des Clients, erfolgen. Außerdem werden Felder bereits während der Eingabe korrekt formatiert. Die Umsetzung erfolgte in Javascript.

Nachdem das Projekt weiter fortgeschritten war, konnten wir dann, unabhängig von den anderen Gruppen, am Design arbeiten. Dank einer CSS Datei haben wir Form, Farbe und Grafiken festgelegt. Jeder Projekteilnehmer konnte jederzeit den aktuellen Stand vom SVN-Server laden und so das aktuelle Design testen.

7. Bewertung der eingesetzten Technologien

Die in diesem Projekt eingesetzten Programmiersprachen waren für alle Gruppenmitglieder Neuland. Ein paar von uns haben bereits mit Webtechnologien gearbeitet und konnten durch ihre Kenntnisse in HTML und CSS den guten Projektfortschritt unterstützten. Bezüglich der Oberflächengestaltung haben wir also auf uns bereits bekannte Technologien gesetzt, obwohl wir auch hier andere Innovative Oberflächenprogrammierung hätten nutzen können. Das hätten z.B. XUL oder Flash sein können. Da aber alle Gruppenmitglieder bereits mit JSPs, Servlets und Java allgemein genug Einarbeitungsaufwand hatten, haben wir darauf verzichtet.

Grundsätzlich unterscheiden sich die Art und Weise, wie Java die Webprogrammierung umsetzt, nicht von der uns schon bekannten Art, wie PHP dies macht. Jedoch war es anfänglich schwer einen Sinn in der Trennung von JSP und Servlet zu sehen, da es diese in PHP nicht gibt. Sowohl Oberflächen als auch Hintergrundaktivität finden sich in denselben Script-Dateien. Sind dort Funktionalitäten zu realisieren, welche keinerlei Ausgaben für Oberflächen erzeugen, so können diese zwar in Klassen implementiert sein, werden allerdings nicht vollständig anders behandelt wie es bei Java ist. Hier fallen, die in der Entwicklungsphase unterschiedlichen JSPs, und Servlets, nach dem kompilieren und deployen ebenfalls auf Servlets zurück. Das deutet daraufhin, das die Anfänge der Java Webprogrammierung ausschließlich in Servlets erfolgten und erst im Nachhinein aus Komfortgründen das Konzept der JSPs hinzugefügt worden ist.

Gäbe es in Java nur die Funktionalität der JSPs, so wär der Unterschied zu PHP längst nicht so groß.

Eine weitere Hürde, die es am Anfang zu bewältigen galt, war ein lauffähiges Script bzw. Servlet zu erstellen, welches eine korrekte Ausgabe im Browser erzeugt. Im Unterschied zu PHP reicht es bei Java nicht die unkompilierten Skriptdateien im Arbeitsordner des Webservers zu platzieren. Diese müssen in kompilierter Form als Java-Class vorliegen. Zusätzlich müssen eventuelle Abhängigkeiten als JAR-Pakete dem Applikations-Server verfügbar gemacht werden. Da gibt es jedoch viele Möglichkeiten diese zu platzieren und nicht immer sind alle Möglichkeiten gleich, was die Funktionalität angeht. Vergleicht man dies mit Erfahrungen in PHP, so scheint es dort, durch die zentrale Konfigurationsschnittstelle (PHP.ini unter Windows) und die, auf dem Dateisystem basierenden Includepfade, etwas simpler gelöst zu sein.

Die Komplexität, welche die Einarbeitung in Java Webprogrammierung erschwert, bietet jedoch auch einige Vorteile gegenüber PHP. Die Nutzung von kompilierten Scripts erhöht die Performanz der Anwendung, hebt sich jedoch bezüglich Ladezeiten auf, da in Java-Applikations-Servern die einzelnen JSPs bzw. Servlets ohnehin nur einmal geladen werden und dann im Speicher verbleiben. Es besteht jeweils nur eine Instanz eines Skripts, welches sich um alle Anfragen kümmert bis der Server, respektive der Servlet-Container gestoppt wird. In PHP wird jedes Skript bei jedem Aufruf neu geladen, interpretiert und ausgeführt. Letzteres vereinfacht die Programmierung in der Testphase, da so Änderungen schnell ohne Deploy und Übersetzung überprüft werden können, reduziert jedoch ohne weitere Caching Technologien die Leistung. Doch auch für PHP gibt es von der Firma ZEND Werkzeuge, die PHP-Skripte optimieren und für den Webserver vorübersetzen.

Das angesprochene Deploy haben wir während unserer Implementierungsphase durch einen Trick umgangen. Das Projektverzeichnis unserer Programmierumgebung haben wir direkt in das Arbeitsverzeichnis des Applikations-Servers gelegt. Daher genügte das Ändern und Speichern der Quelltexte. Das Kompilieren wurde durch das Case-Werkzeug übernommen. Zwischen Änderung am Quelltext und dem sichtbaren Ergebnis sind also in Java mehr Zwischenschritte notwendig. Diese können jedoch durch automatisiert werden.

Einen großen Vorteil, den Java bietet, ist die mögliche Anbindung an bereits bestehende und auch in Java implementierte Software. In großen Projekten können bestehende Zugriffsklassen ohne Probleme genutzt werden. In PHP wären hierfür Adapter notwendig, welche eventuell sogar speziell für eine Plattform entwickelt werden müssten. In Java entfällt dies, einmal durch die Plattform-Unabhängigkeit selbst und zweitens Aufgrund der Tatsache, dass man sich in einer bereits bekannten Programmiersprache befindet und so den bestehenden Code nutzen kann.

Ein relativ undurchsichtiges Problem, in welches wir während der Entwicklungsphase liefen, wurde durch die Java Persistenz Schicht (JPA) hervorgerufen. An einer Stelle haben wollten wir einer Session variablen den Wert eines Objektes geben, welches von der Persistenzschicht verwaltet wurde. Jedoch wurde hier nicht, wie aus anderen Programmiersprachen gewohnt, der Inhalt kopiert, sondern scheinbar nur eine Referenz kopiert. Hat man in diesem Sessionobjekt nun gearbeitet, so hat die JPA Schicht diese Änderungen sofort bis in das Datenbankbackend weitergereicht, was zu unerwarteten Resultaten führte.

Nach langem Debuggen konnten wir den Fehler eingrenzen und durch ein einfaches duplizieren des Objekts, das Problem lösen. Dabei wird ein neues Objekt erzeugt, welches zwar die gleichen Inhalte hat, aber nicht dasselbe Objekt ist. Gaben wir die Referenz auf dieses Objekt weiter, welches sich nicht mehr im Rahmen der JPA Schicht befand, war der Fehlerbeseitigt.

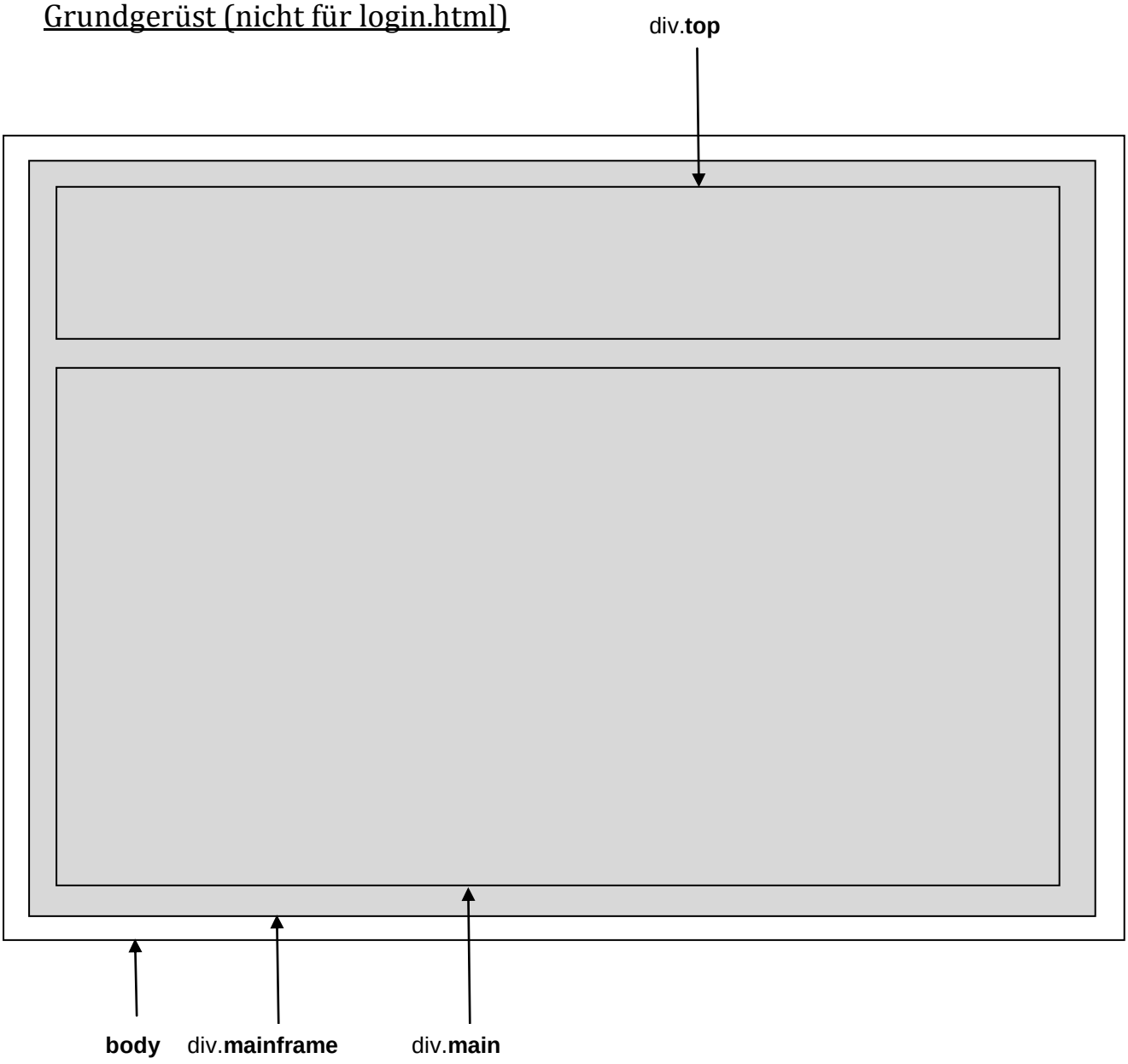
Grundsätzlich ist die Java Persistenz Schicht einer mächtigen Schnittstelle, die einem Projekt sicherlich eine komfortablere Implementierung erlaubt, jedoch muss das Konzept vollkommen verstanden werden, um es fehlerfrei nutzen zu können.

Abschließend bleibt zu sagen, dass die anfängliche Abneigung allein auf der höheren Komplexität und der somit längeren Einarbeitungsphase beruhte. Für kleine Projekte die schnell umgesetzt werden müssen, empfiehlt sich PHP, da es mit geringerem Aufwand einsetzen lässt. Für Großprojekte, die an bestrehende Strukturen angebunden werden sollen, sind die Java Webtechnologien sicherlich im Vorteil. Doch auch PHP kann erfolgreich in Großprojekten eingesetzt werden. Beispiele sind das U.S. amerikanische Facebook, oder das deutsche StudiVZ.

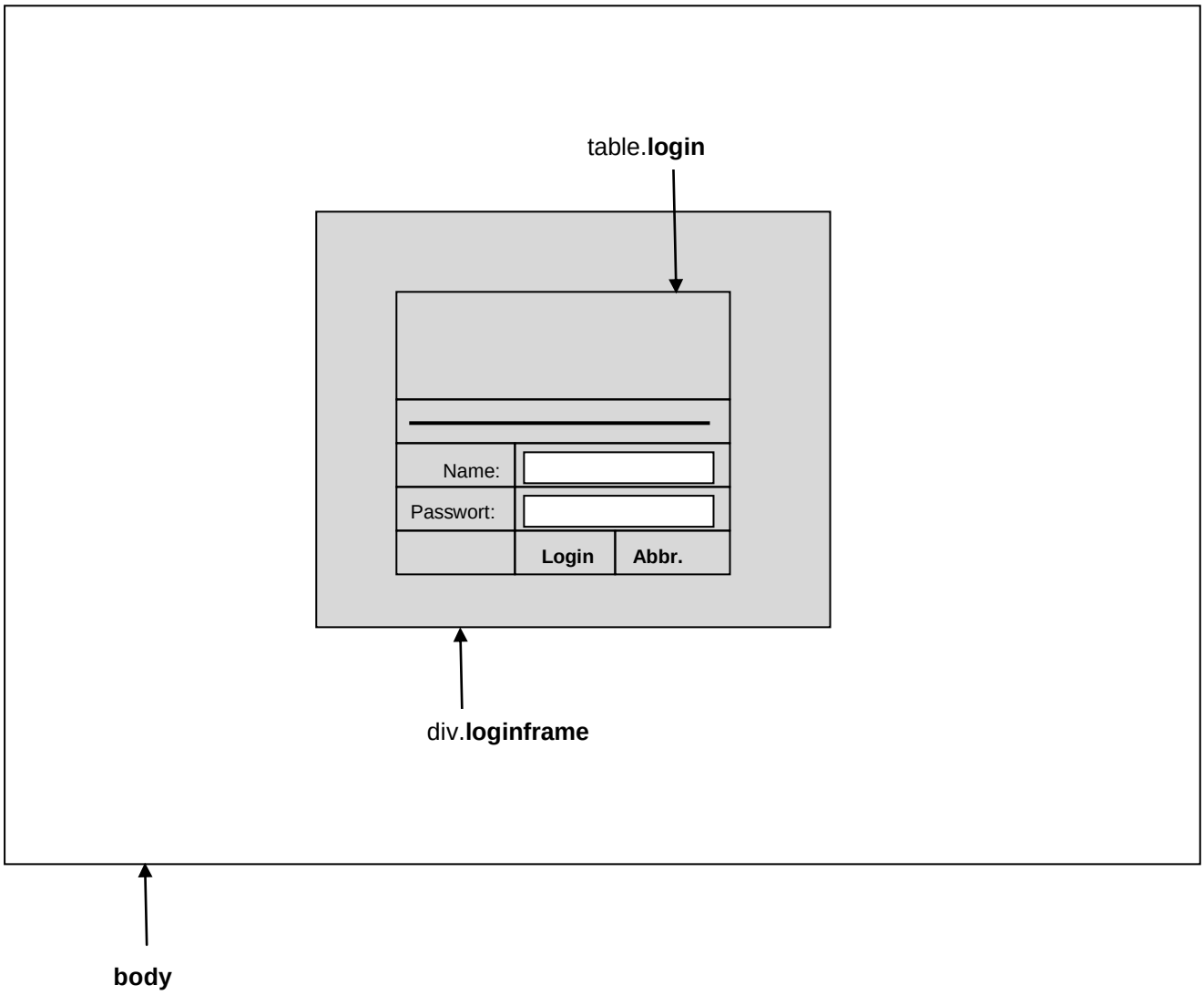
8. Anhang

1. CSS-Klassen

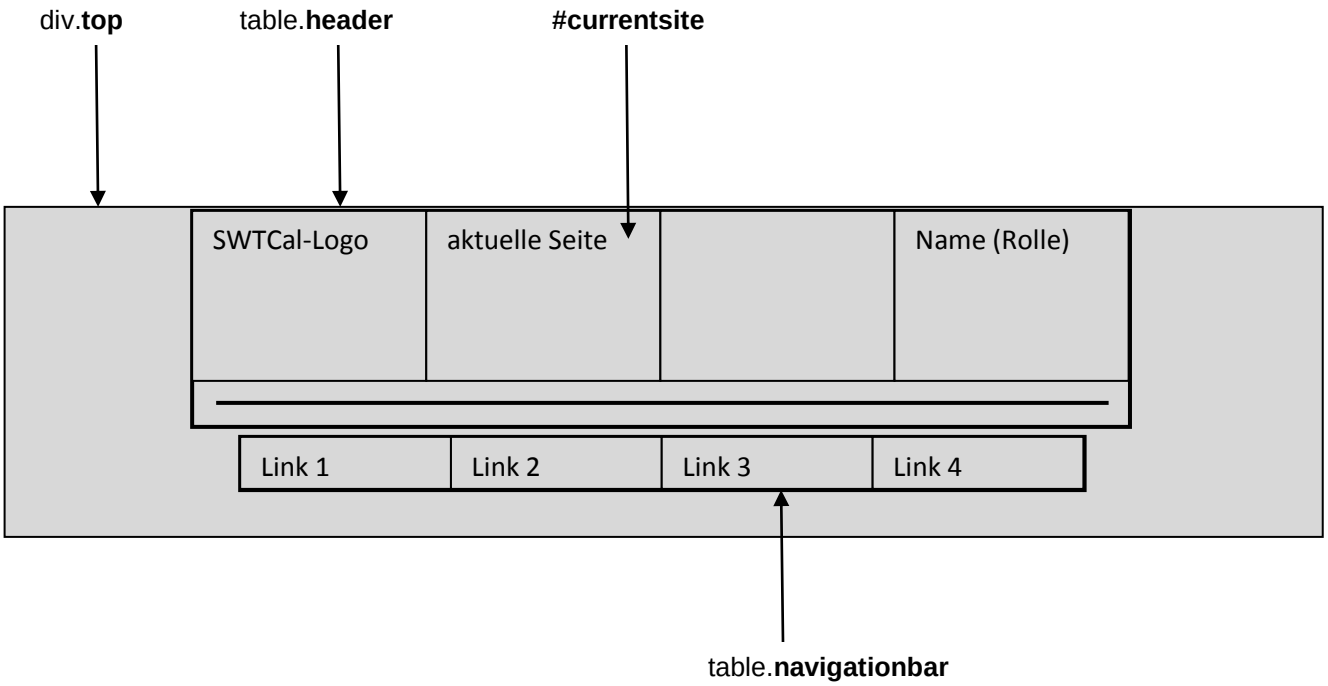
Grundgerüst (nicht für login.html)



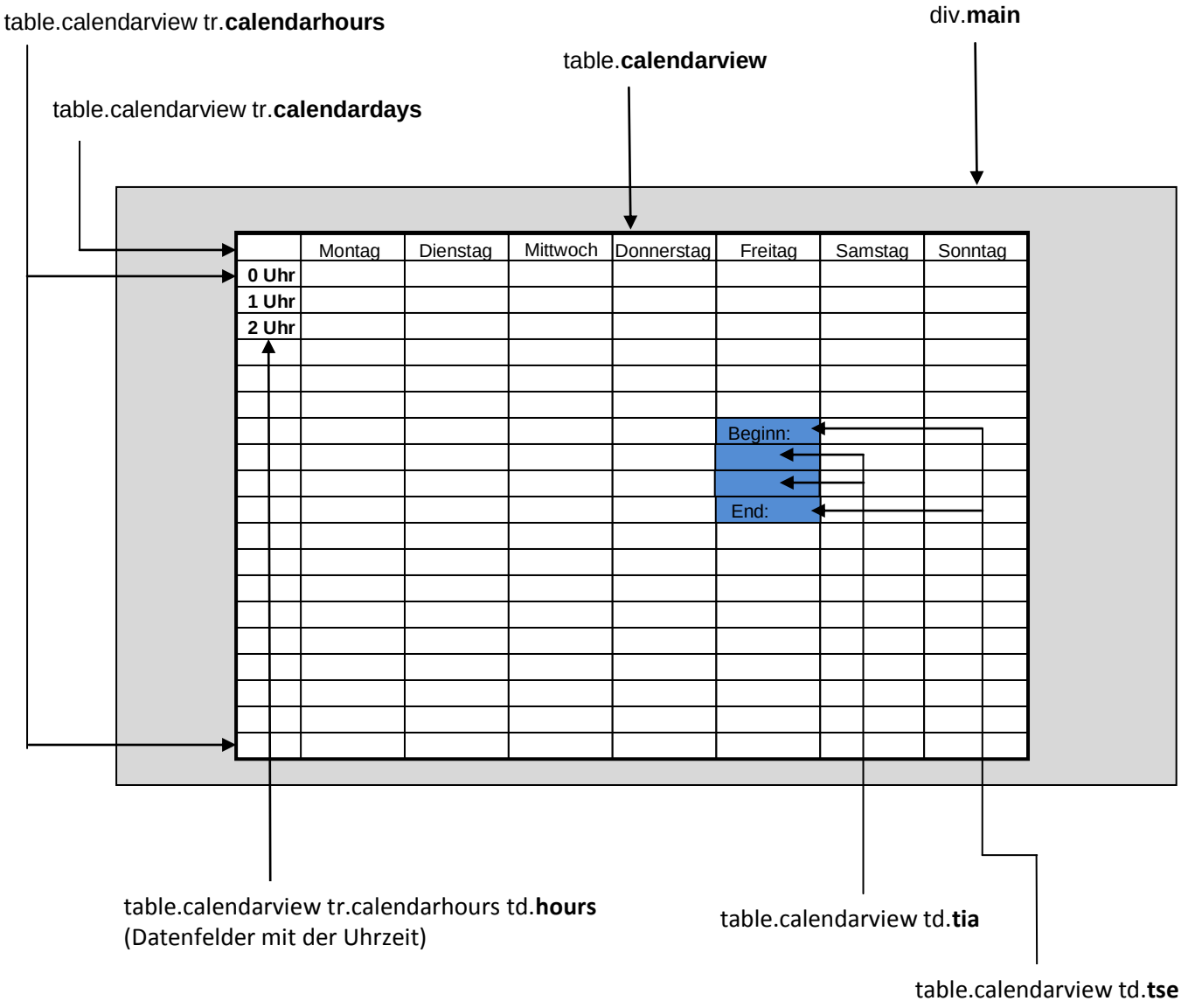
Login.html



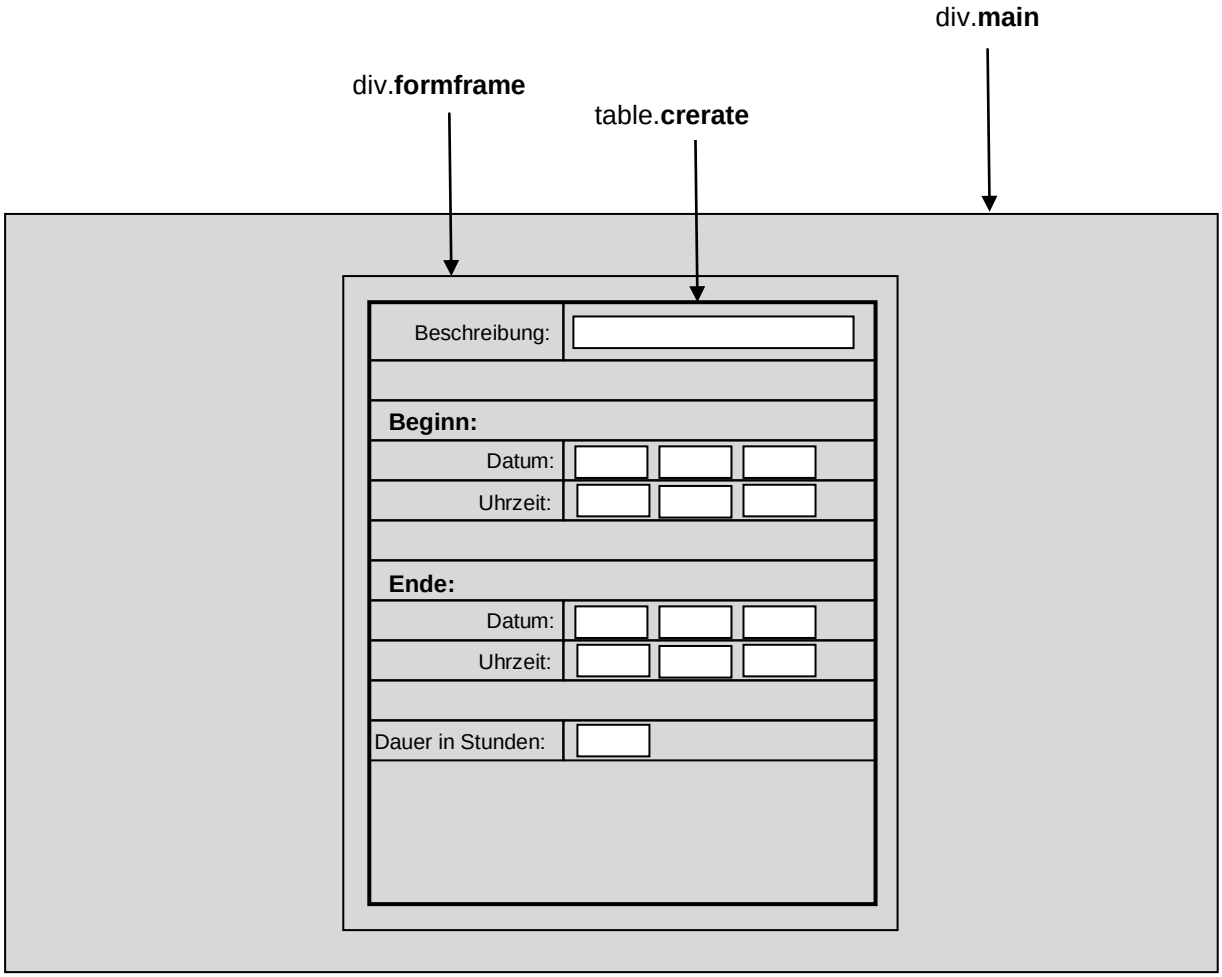
Header mit Navigationsleiste
(bei allen Seiten identisch, nicht für login.html)



Overview.html



Create.html


div.main
div.formframe
table.crerate
Beschreibung:
Beginn:
Datum:
Uhrzeit:
Ende:
Datum:
Uhrzeit:
Dauer in Stunden: