BUCH (Prof. Dr.-Ing. Jörg J. Buchholz)

Veranstaltungsplan Wintersemester 2016/2017

< Zeiten Bemerkungen Langnamen Alle Wochen 40 41 42 43 44 45 46 47 48 49 50 51 02 03 04

	Mo	Di	Mi	Do	Fr
1. Block 08:00 - 09:30	RTFR ILST_3_1 SI 52	INFO KW 40, 41, 42 M 203a	MATH ILST_1_1 KW 40, 42, 44, 46, 48, 50, 02, 04 SI 52		
2. Block 09:45 - 11:15					
3. Block 11:30 - 13:00			MATH ILST_1_2 KW 41, 43, 45, 47, 49, 51, 03 SI 52		
4. Block 13:30 - 15:00			MATH ILST_1_1 || ILST_1_2 SI 52		
5. Block 15:15 - 16:45					
6. Block 17:00 - 18:30					
7. Block 18:45 - 20:15					
8. Block 20:30 - 22:00					

Einsatzplan Wintersemester 2016/2017

Modul	Verband	Raum	Bemerkung	SWS	Faktor	Tag	Beginn	Ende	Kalenderwochen
INFO		M 203a	Nur in den ersten 3 Wochen von 9 - 12 Uhr	4	0,2	Di	09:00	12:00	40, 41, 42
RTFR	ILST_3_1	SI 52		8	1	Mo	08:00	14:00	Alle Wochen
MATH	ILST_1_1 || ILST_1_2	SI 52	Halb angerechnet, zwei Verbände	2	0,5	Mi	13:30	15:00	Alle Wochen
MATH	ILST_1_1	SI 52	Nur in geraden Wochen	6,67	0,53	Mi	08:00	13:00	40, 42, 44, 46, 48, 50, 02, 04
MATH	ILST_1_2	SI 52	Nur in ungeraden Wochen	6,67	0,47	Mi	08:00	13:00	41, 43, 45, 47, 49, 51, 03

Einsatzplan ausblenden, um Veranstaltungsplan zu drucken

Planungsstand: 11.04.2017 13:23:02

plan

Lehrveranstaltungsplanwebdarstellung

Jörg J. Buchholz

17. April 2017

Inhaltsverzeichnis

Teil I

Bedienungsanleitung

1 Einführung

plan (**P**rofessionelle **L**ehrveranstaltungs**an**zeige) ist ein Programm zur detaillierten und übersichtlichen Darstellung von Lehrveranstaltungsplänen im Internet mit folgenden Eigenschaften:

- Dozentinnen-[1], Semesterverbands- und Raumpläne
- Wahlweise Darstellung von Veranstaltungsanfangs- und -endzeiten
- Wahlweise Darstellung von Bemerkungen
- Wahlweise Darstellung von Abkürzungen oder Langnamen
- Wochen- und Gesamtpläne
- Darstellung von Blockungen und Zusammenlegungen
- Wahlweise farbliche Unterscheidung einzelner Lehrveranstaltungen
- Darstellung von Sperrungen
- Wahlweise zusätzliche Anzeige des Einsatzplans während der Planungsphase
- Darstellung der Semesterwochenstunden und gegebenenfalls der Anrechnungsfaktoren für die Lehrdeputatsabrechnung im Einsatzplan

plan liest direkt die Datenbanken des Planungsprogrammes daVinci [1] und stellt die darin enthaltenen Informationen für Dozentinnen und Studentinnen auf einer Webseite dar. Mit ein paar Schnittstellenanpassungen kann plan auch die Ausgaben anderer Planungsprogramme übernehmen.

[1] Dieses Dokument verwendet das generische Femininum. Männer sind automatisch mitgemeint.

2 Anwendung des Programmes

In diesem Kapitel wollen wir die typische Vorgehensweise im Umgang mit plan kennenlernen.

2.1 Webseite

Nachdem wir auf der Einstiegsseite `https://m-server.fk5.hs-bremen.de/plan` (Abbildung 2.1) die gewünschte Fakultät bzw. Abteilung,

Abbildung 2.1: Auswahl der Fakultät bzw. der Abteilung

auf der folgenden Seite (Abbildung 2.2) das gewünschte Sommer- bzw. Wintersemester

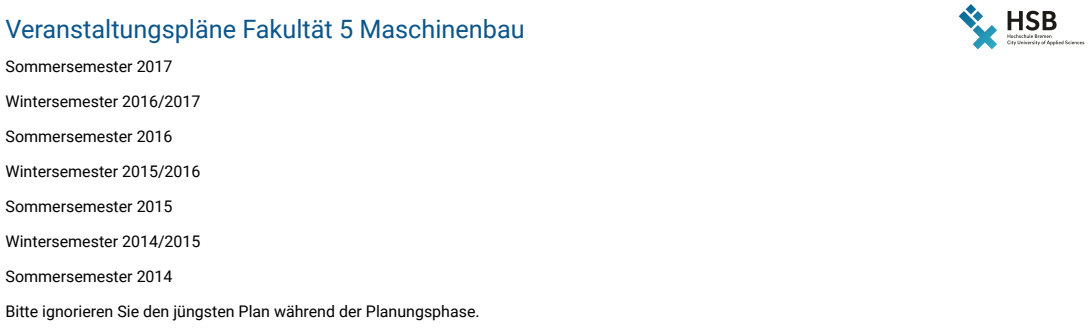
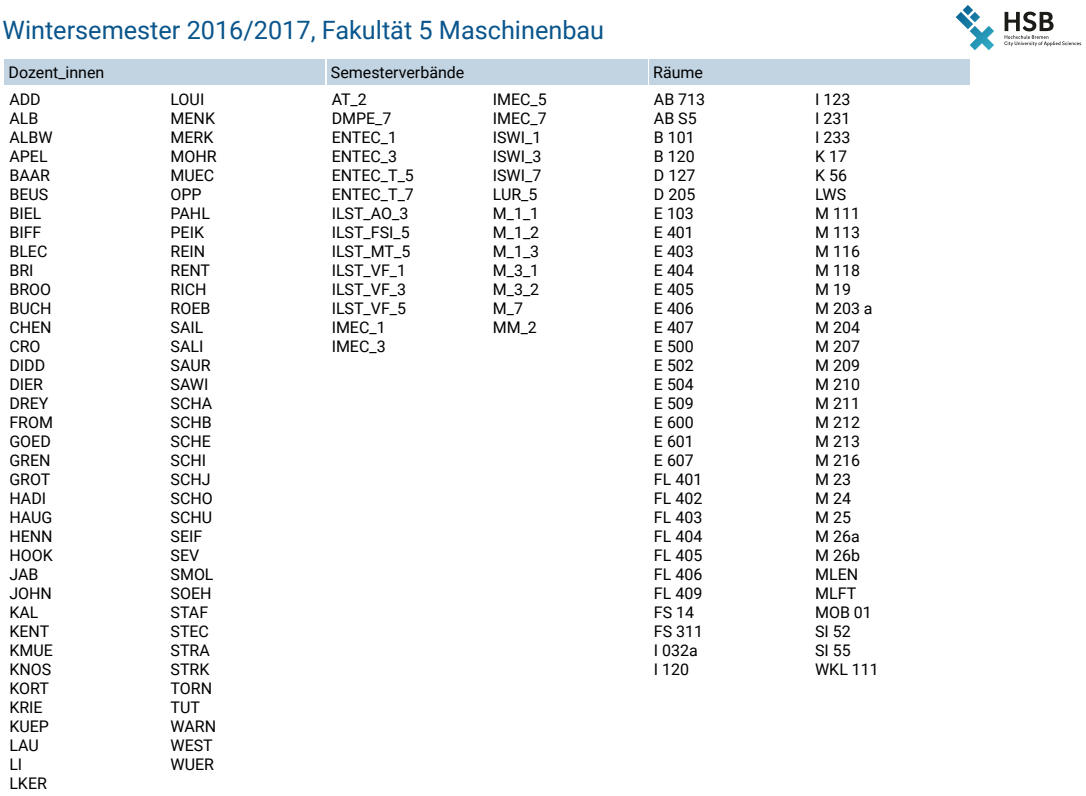


Abbildung 2.2: Auswahl des Sommer- bzw. Wintersemesters

und auf der nächsten Seite (Abbildung 2.3)



Wintersemester 2016/2017, Fakultät 5 Maschinenbau

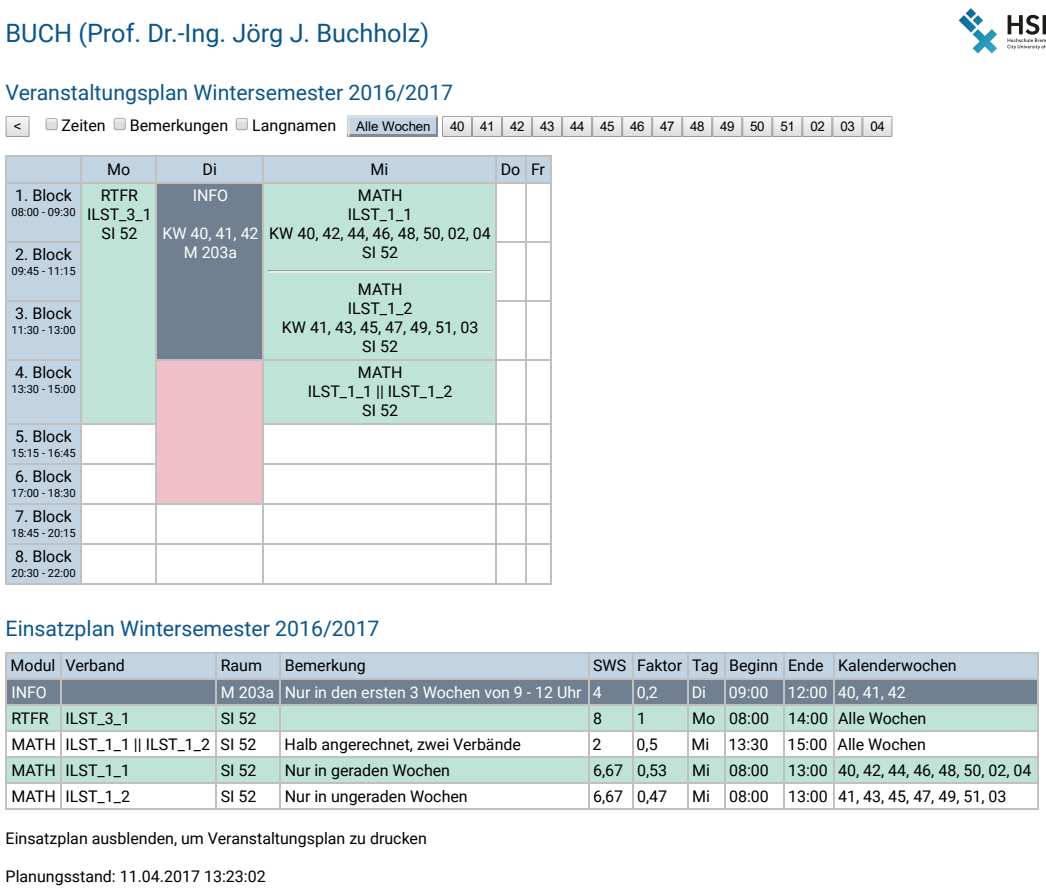
HSB Hochschule Bremen City University of Applied Sciences

Dozent_innen		Semesterverbände		Räume	
ADD	LOUI	AT_2	IMEC_5	AB 713	I 123
ALB	MENK	DMPE_7	IMEC_7	AB S5	I 231
ALBW	MERK	ENTEC_1	ISWI_1	B 101	I 233
APEL	MOHR	ENTEC_3	ISWI_3	B 120	K 17
BAAR	MUEC	ENTEC_T_5	ISWI_7	D 127	K 56
BEUS	OPP	ENTEC_T_7	LUR_5	D 205	LWS
BIEL	PAHL	ILST_AO_3	M_1_1	E 103	M 111
BIFF	PEIK	ILST_FSI_5	M_1_2	E 401	M 113
BLEC	REIN	ILST_MT_5	M_1_3	E 403	M 116
BRI	RENT	ILST_VF_1	M_3_1	E 404	M 118
BROO	RICH	ILST_VF_3	M_3_2	E 405	M 19
BUCH	ROEB	ILST_VF_5	M_7	E 406	M 203 a
CHEN	SAIL	IMEC_1	MM_2	E 407	M 204
CRO	SALI	IMEC_3		E 500	M 207
DIDD	SAUR			E 502	M 209
DIER	SAWI			E 504	M 210
DREY	SCHA			E 509	M 211
FROM	SCHB			E 600	M 212
GOED	SCHE			E 601	M 213
GREN	SCHI			E 607	M 216
GROT	SCHJ			FL 401	M 23
HADI	SCHO			FL 402	M 24
HAUG	SCHU			FL 403	M 25
HENN	SEIF			FL 404	M 26a
HOOK	SEV			FL 405	M 26b
JAB	SMOL			FL 406	MLEN
JOHN	SOEH			FL 409	MLFT
KAL	STAF			FS 14	MOB 01
KENT	STEC			FS 311	SI 52
KMUE	STRA			I 032a	SI 55
KNOS	STRK			I 120	WKL 111
KORT	TORN				
KRIE	TUT				
KUEP	WARN				
LAU	WEST				
LI	WUER				
LKER					

Abbildung 2.3: Auswahl der Dozentin, des Semesterverbandes oder des Raumes

die Dozentin, den Semesterverband oder den Raum ausgewählt haben, öffnet sich die in Abbildung 2.4 dargestellte Webseite, auf der oben der Veranstaltungsplan[1] und unten der Einsatzplan angezeigt wird.

[1]Die in Abbildung 2.4 dargestellten Pläne des Autors sind rein fiktiv und dienen ausschließlich Demonstrationszwecken.



BUCH (Prof. Dr.-Ing. Jörg J. Buchholz)

Veranstaltungsplan Wintersemester 2016/2017

< Zeiten Bemerkungen Langnamen Alle Wochen 40 41 42 43 44 45 46 47 48 49 50 51 02 03 04

	Mo	Di	Mi	Do	Fr
1. Block 08:00 - 09:30	RTFR ILST_3_1 SI 52	INFO KW 40, 41, 42 M 203a	MATH ILST_1_1 KW 40, 42, 44, 46, 48, 50, 02, 04 SI 52		
2. Block 09:45 - 11:15					
3. Block 11:30 - 13:00			MATH ILST_1_2 KW 41, 43, 45, 47, 49, 51, 03 SI 52		
4. Block 13:30 - 15:00			MATH ILST_1_1 || ILST_1_2 SI 52		
5. Block 15:15 - 16:45					
6. Block 17:00 - 18:30					
7. Block 18:45 - 20:15					
8. Block 20:30 - 22:00					

Einsatzplan Wintersemester 2016/2017

Modul	Verband	Raum	Bemerkung	SWS	Faktor	Tag	Beginn	Ende	Kalenderwochen
INFO		M 203a	Nur in den ersten 3 Wochen von 9 - 12 Uhr	4	0,2	Di	09:00	12:00	40, 41, 42
RTFR	ILST_3_1	SI 52		8	1	Mo	08:00	14:00	Alle Wochen
MATH	ILST_1_1 || ILST_1_2	SI 52	Halb angerechnet, zwei Verbände	2	0,5	Mi	13:30	15:00	Alle Wochen
MATH	ILST_1_1	SI 52	Nur in geraden Wochen	6,67	0,53	Mi	08:00	13:00	40, 42, 44, 46, 48, 50, 02, 04
MATH	ILST_1_2	SI 52	Nur in ungeraden Wochen	6,67	0,47	Mi	08:00	13:00	41, 43, 45, 47, 49, 51, 03

Einsatzplan ausblenden, um Veranstaltungsplan zu drucken

Planungsstand: 11.04.2017 13:23:02

Abbildung 2.4: Darstellung des Veranstaltungsplans und des Einsatzplans auf der plan-Webseite

Die Gesamtüberschriftszeile besteht neben dem Logo der Hochschule Bremen aus dem Kürzel der Dozentin[2], in Klammern gefolgt von ihrem Titel, ihrem Vor- und ihrem Nachnamen. In der Folgezeile kann die Planerin eine Bemerkung über die Dozentin ausgeben lassen; bei Lehrbeauftragten kann dies beispielsweise die hauptamtliche Betreuerin der Lehrbeauftragten sein.

Unter der Veranstaltungsplanüberschrift, in der das gewählte Semester angegeben ist, gibt es mehrere Auswahlfelder, mit denen wir den Detailgrad der Pläne und die darzustellenden Wochen festlegen können.

[2] Auf den Seiten der Semesterverbände und der Räume stehen hier natürlich die entsprechenden Abkürzungen und Langnamen der Verbände bzw. Räume.

2.1.1 Veranstaltungsplan

Zeilen und Spalten

Im Veranstaltungsplan bestehen die Spalten aus den Wochentagen[3] und die Zeilen aus den von der Planerin festgelegten Blöcken[4]. Im Plan werden die einzelnen Veranstaltungen als rechteckige Felder dargestellt, die sich über mehrere Blöcke (Zeilen), nicht aber über mehrere Tage (Spalten) erstrecken können.

Farben

Die Standardhintergrundfarbe einer Veranstaltung ist ein Grünton; Planerinnen können aber für jedes Modul eine eigene Farbe wählen (beispielsweise das graue[5] Modul INFO in Abbildung 2.4).

Sperrungen

Sperrungen, an denen keine Lehre möglich ist, werden in einem roten Farbton angezeigt (beispielsweise Dienstag, 4. bis 6. Block in Abbildung 2.4).

Einträge

Üblicherweise gehören zu einer Veranstaltung der Name des Moduls, der teilnehmende Semesterverband, der Raum, in dem die Veranstaltung stattfindet und die Dozentin. So handelt es sich beispielsweise am Montag in Abbildung 2.4 um die Veranstaltung RTFR (Regelungstechnik und Flugregler) im Semesterverband ILST_3_1 (Luftfahrtsystemtechnik, Semester 3, Verband 1) im Raum SI 52 (Rechnerraum, SI-Gebäude, Neustadtswall). Alle drei Angaben können fehlen, wenn die Planerin sie noch nicht eingetragen hat. Beispielsweise ist in der INFO-Veranstaltung am Dienstag Vormittag in der dritten Zeile noch kein Semesterverband vorhanden.

Kalenderwochen

Wenn die Planerin festgelegt hat, dass eine Veranstaltung nicht jede Semesterwoche stattfinden soll, sondern beispielsweise nur in ungeraden Wochen, in der ersten Semesterhälfte oder auch nur einmalig, werden die Kalenderwochen, in denen die Veranstaltung stattfindet, in der dritten Zeile des Veranstaltungsfeldes aufgelistet. So findet die INFO-Veranstaltung in Abbildung 2.4 beispielsweise nur in den ersten drei Wochen (40, 41, 42) des Wintersemesters statt.

[3] Üblicherweise beginnt die Woche am Montag und endet am Freitag. Die Planerin kann aber auch andere Wochenanfangs- und -endtage festlegen.

[4] Die allgemein übliche Standardgröße eines Blockes beträgt zwei Semesterwochenstunden, also $2 \cdot 45$ Minuten = 90 Minuten. Planerinnen können aber beliebige andere Blockgrößen und -zeiten definieren.

[5] Wenn eine Farbe zu dunkel wird, wählt plan automatisch weiße statt schwarzer Schrift.

Mehrere Semesterverbände

Am Mittwoch Vormittag sind in Abbildung 2.4 zwei Veranstaltungen gleichzeitig eingetragen, die aber nicht miteinander kollidieren und die durch eine nicht ganz durchgezogene Trennungslinie voneinander getrennt sind: In geraden Kalenderwochen (40, ..., 04) nimmt der Semesterverband ILST_1_1 teil, in ungeraden Kalenderwochen (41, ..., 03) der Verband ILST_1_2.

Im vierten Block am Mittwoch nehmen die beiden Semesterverbände nicht abwechselnd, sondern gleichzeitig im gleichen Raum teil. Die beiden Verbände sind daher durch zwei kurze senkrechte Striche voneinander getrennt: ILST_1_1 || ILST_1_2.

Ebenso kann die Planerin festlegen, dass von einer Veranstaltung gleichzeitig mehrere Räume belegt werden oder dass (beispielsweise in einem Semesterverbandsplan) gleichzeitig mehrere Dozentinnen lehren.

Zeiten

Es gibt an der Hochschule Bremen Organisationseinheiten, die sich bei der Veranstaltungsplanung strikt an die einmal definierten Blockgrenzen halten; andere Planerinnen müssen beispielsweise auf persönliche Randbedingungen von Lehrbeauftragten Rücksicht nehmen und Veranstaltungen auch außerhalb der Blockgrenzen beginnen oder enden lassen. Daher gibt es über dem Veranstaltungsplan das Auswahlfeld Zeiten, das nach dem Anklicken[6] für jede Veranstaltung in der ersten Zeile ihre Anfangs- und Endzeit einblendet (Abbildung 2.5).

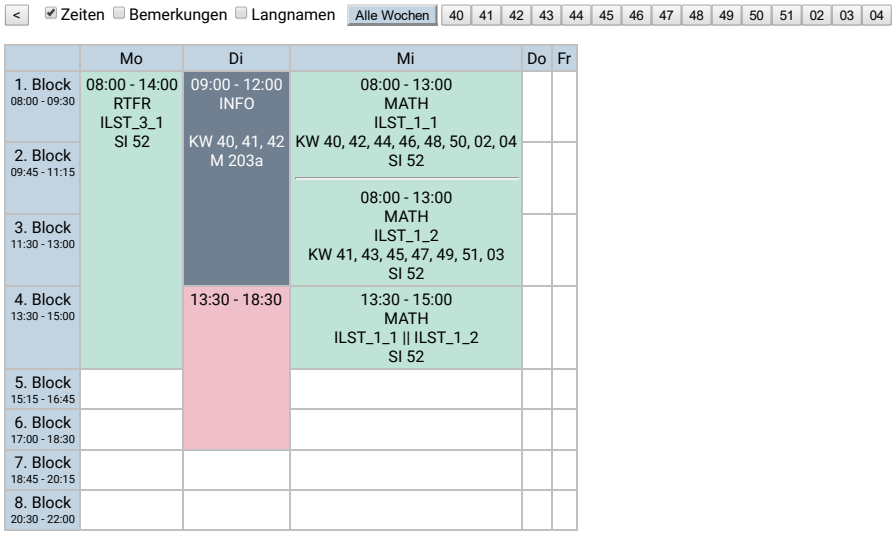


Abbildung 2.5: Anzeige von Anfangs- und Endzeiten in den ersten Zeilen der Veranstaltungen

[6] Der Zustand aller Auswahlfelder bleibt auch denn erhalten, wenn wir weitere Pläne aufrufen; durch die Verwendung von Cookies wird er sogar dann wieder restauriert, wenn wir innerhalb von 30 Tagen das Programm ein weiteres Mal aufrufen.

So erkennen wir beispielsweise, dass die RTFR-Veranstaltung am Montag nicht bis zum Blockende um 15 Uhr dauert, sondern eine Stunde früher endet. Auch INFO am Dienstag beginnt und endet nicht an Blockgrenzen. Durch die Anzeige der Anfangs- und Endzeiten wird außerdem nochmals deutlicher, dass die beiden Veranstaltungen am Mittwoch Vormittag nicht nacheinander, sondern gleichzeitig (aber in unterschiedlichen Wochen) stattfinden.

Bemerkungen

Nach dem Anklicken des Auswahlfeldes Bemerkungen wird in der letzten Zeile einer Veranstaltung gegebenenfalls eine Bemerkung angezeigt, die die Planerin einer einzelnen Veranstaltung hinzufügen kann (Abbildung 2.6).

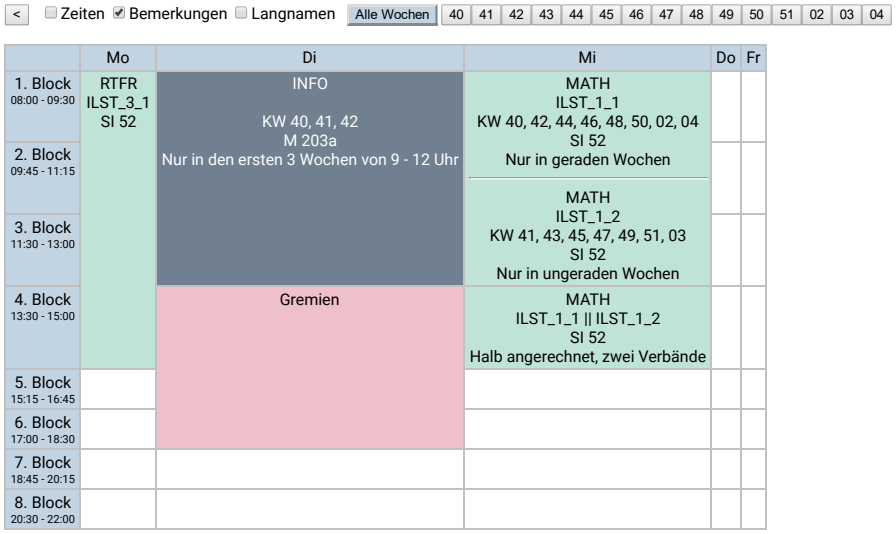


Abbildung 2.6: Anzeige von Bemerkungen in den letzten Zeilen der Veranstaltungen

Durch das Auswahl der Bemerkungen wird beispielsweise auch der Grund für die Sperrung am Dienstag Nachmittag angezeigt.

Langnamen

Traditionell nutzt ein Teil der Planerinnen der Hochschule Bremen in ihren Veranstaltungsplänen Kürzel (beispielsweise RTFR als Modulnamen), um die Pläne kompakt und übersichtlich zu halten, andere Planerinnen argumentieren, dass Nutzerinnen lieber die ausgeschriebene Variante des Modulnamens (Regelungstechnik und Flugregler) in den Plänen sehen möchten, um keine Abkürzungen „auswendig lernen" zu müssen. plan bietet daher die Möglichkeit, durch Anklicken des gleichnamigen Auswahlfeldes, Langnamen statt der Kürzel für Modulnamen, Semesterverbände und Räume anzuzeigen (Abbildung 2.7).

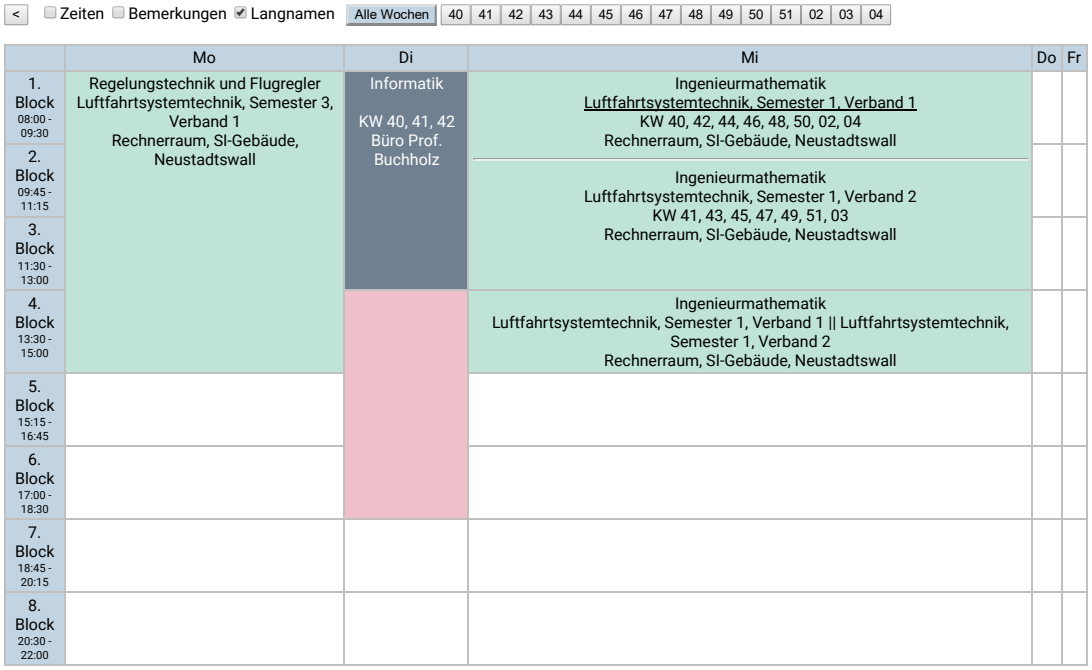


Abbildung 2.7: Anzeige von Langnamen der Module, Verbände und Räume

Selbstverständlich können wir auch alle Auswahlfelder beliebig miteinander kombinieren. Wir können also als Nutzerin feinstufig einstellen, welche Informationen wir in unserer persönlichen Plandarstellung sehen möchten.

Tooltips

Zusätzlich zu der Möglichkeit, explizit Kürzel oder Langnamen anzuzeigen, wird die jeweils andere Variante als Tooltip dargestellt. So wird beispielsweise der Langname eines Moduls in einem kleinen Extrafenster angezeigt, wenn wir mit dem Cursor über dem Modulkürzel verharren.

Wochenauswahl

Beim Start des Programmes wird außerhalb des Veranstaltungszeitraumes anfänglich immer der Gesamtplan angezeigt, der auch alle Veranstaltungen beinhaltet, die nur in bestimmten Wochen stattfinden. In Abbildung 2.4 erkennen wir das daran, dass die Schaltfläche Alle Wochen ausgewählt (blau hinterlegt) ist. Durch Anklicken der Schaltfläche 42 wählen wir die 42. Kalenderwoche aus, so dass nur noch die Veranstaltungen angezeigt werden, die in der entsprechenden Woche stattfinden (Abbildung 2.8). Während des Veranstaltungszeitraumes wird automatisch der Veranstaltungsplan der aktuellen Woche angezeigt. Die aktuelle Woche ist in der Liste während des Veranstaltungszeitraumes etwas hervorgehoben. Außerdem wird bei Einzelwochen in der Tabellenüberschrift für jeden Wochentag zusätzlich sein Datum in der ausgewählten Woche angezeigt.

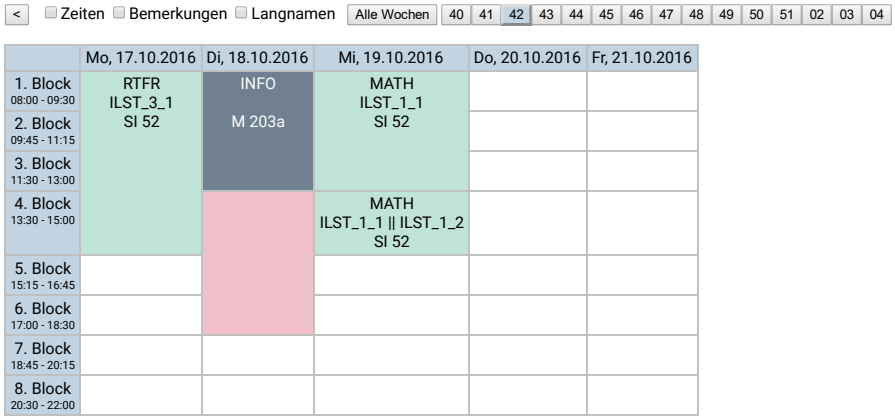


Abbildung 2.8: Anzeige der in der 42. Kalenderwoche stattfindenden Veranstaltungen

Im Vergleich mit Abbildung 2.4 wird deutlich, dass am Mittwoch Vormittag die Veranstaltung im Verband ILST_1_2, die ja nur in den ungeraden Wochen stattfindet, jetzt korrekterweise nicht mehr angezeigt wird. Sie wird natürlich (statt der Veranstaltung im ILST_1_1) wieder angezeigt, wenn wir die Schaltfläche der 43. Kalenderwoche drücken (Abbildung 2.9).

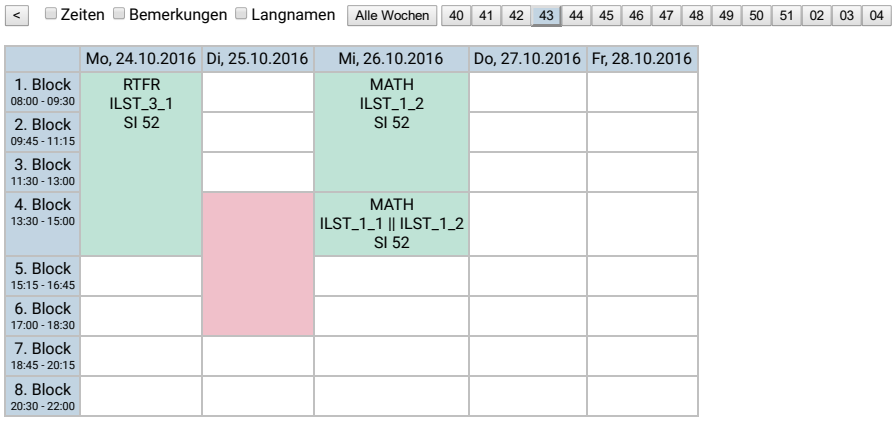


Abbildung 2.9: Anzeige der in der 43. Kalenderwoche stattfindenden Veranstaltungen

Zudem fehlt jetzt am Dienstag die INFO-Veranstaltung, die ja nur in den ersten drei Wochen des Semesters stattfinden soll.

Verlinkungen

Verbands- und Raumnamen sind als Hyperlinks ausgeführt und leiten beim Anklicken auf die jeweils entsprechenden Webseiten weiter. Auf diese Weise können wir beispielsweise sehr schnell feststellen, ob ein Semesterverband oder Raum vielleicht auch zu anderen Zeiten frei ist.

Zusatzbemerkung

Unter dem Veranstaltungsplan kann die Planerin weitere Zusatzinformationen, beispielsweise Farbzuordnungen, ... ausgeben (Abschnitt 3.4). In Abbildung 2.4 ist dies nicht der Fall.

2.1.2 Einsatzplan

Unter dem Veranstaltungsplan finden wir in Abbildung 2.4 den Einsatzplan, in dem die Planerinnen schon am Anfang der Planungsphase – wenn die tatsächlichen Veranstaltungszeiten noch nicht feststehen und der Veranstaltungsplan daher noch leer ist – darstellen können, welche Dozentinnen in welchen Verbänden welche Module lesen werden (Abbildung 2.10).

Einsatzplan Wintersemester 2016/2017

Modul	Verband	Raum	Bemerkung	SWS	Faktor	Tag	Beginn	Ende	Kalenderwochen
INFO		M 203a	Nur in den ersten 3 Wochen von 9 - 12 Uhr	4	0,2	Di	09:00	12:00	40, 41, 42
RTFR	ILST_3_1	SI 52		8	1	Mo	08:00	14:00	Alle Wochen
MATH	ILST_1_1 || ILST_1_2	SI 52	Halb angerechnet, zwei Verbände	2	0,5	Mi	13:30	15:00	Alle Wochen
MATH	ILST_1_1	SI 52	Nur in geraden Wochen	6,67	0,53	Mi	08:00	13:00	40, 42, 44, 46, 48, 50, 02, 04
MATH	ILST_1_2	SI 52	Nur in ungeraden Wochen	6,67	0,47	Mi	08:00	13:00	41, 43, 45, 47, 49, 51, 03

Abbildung 2.10: Einsatzplan

Auch im Einsatzplan werden von der Planerin explizit zugewiesene Modulfarben verwendet. Veranstaltungen ohne zugewiesene Farbe werden zur besseren Lesbarkeit abwechselnd weiß und grün dargestellt.

Semesterwochenstunden

Außerdem sehen wir im Einsatzplan in der Spalte SWS exakt, wie viele Semesterwochenstunden à 45 Minuten eine Veranstaltung umfasst. So wird die INFO-Veranstaltung am Dienstag im Veranstaltungsplan zwar über drei Blöcke hinweg angezeigt, da ihre Anfangs- und Endzeiten (9 bis 12 Uhr) nicht genau auf Blockgrenzen liegen. Faktisch[7] dauert sie aber nur insgesamt 3 Zeitstunden, was den im Einsatzplan ausgewiesenen 4 SWS entspricht.

Faktor

Dass die INFO-Veranstaltung nur in den ersten drei Semesterwochen stattfindet, wird im Einsatzplan in der Spalte Faktor als 3 von 15 Wochen $= \frac{3}{15} = \frac{1}{5} = 0{,}2$ angezeigt. Die

[7] Für die spätere Lehrdeputatsabrechnung ist wichtig, dass für eine Veranstaltung, die über Blockgrenzen hinweg verplant ist, tatsächlich die gesamte Zeitdauer der Veranstaltung berücksichtigt wird, dass die Pausen zwischen den Blöcken also als Veranstaltungszeit gezählt werden. Die MATH-Veranstaltung am Mittwoch Vormittag wird daher als 5 Zeitstunden = 6,67 SWS und nicht als 3 Blöcke = 6 SWS gezählt.

beiden Faktoren der MATH-Veranstaltung am Mittwoch Vormittag unterscheiden sich, da es im Semesterzeitraum eine gerade Woche mehr als ungerade Wochen gibt.

Statt der automatischen Berechnung kann die Planerin auch selbst jeder Veranstaltung einen eigenen Anrechnungsfaktor[8] zuweisen. Der Faktor der MATH-Veranstaltung am Mittwoch Nachmittag wurde explizit von der Planerin auf 0,5 gesetzt, beispielsweise um zu gewährleisten, dass modulbezogene Übungen nur zur Hälfte angerechnet werden.

Langnamen

Das Anklicken des Auswahlfeldes Langnamen über dem Veranstaltungsplan bewirkt, dass auch im Einsatzplan die Namen der Module, Verbände und Räume nicht abgekürzt, sondern ausgeschrieben werden (Abbildung 2.11).

Einsatzplan Wintersemester 2016/2017

Modul	Verband	Raum	Bemerkung	SWS	Faktor	Tag	Beginn	Ende	Kalenderwochen
Informatik		Büro Prof. Buchholz	Nur in den ersten 3 Wochen von 9 - 12 Uhr	4	0,2	Di	09:00	12:00	40, 41, 42
Regelungstechnik und Flugregler	Luftfahrtsystemtechnik, Semester 3, Verband 1	Rechnerraum, SI-Gebäude, Neustadtswall		8	1	Mo	08:00	14:00	Alle Wochen
Ingenieurmathematik	Luftfahrtsystemtechnik, Semester 1, Verband 1 || Luftfahrtsystemtechnik, Semester 1, Verband 2	Rechnerraum, SI-Gebäude, Neustadtswall	Halb angerechnet, zwei Verbände	2	0,5	Mi	13:30	15:00	Alle Wochen
Ingenieurmathematik	Luftfahrtsystemtechnik, Semester 1, Verband 1	Rechnerraum, SI-Gebäude, Neustadtswall	Nur in geraden Wochen	6,67	0,53	Mi	08:00	13:00	40, 42, 44, 46, 48, 50, 02, 04
Ingenieurmathematik	Luftfahrtsystemtechnik, Semester 1, Verband 2	Rechnerraum, SI-Gebäude, Neustadtswall	Nur in ungeraden Wochen	6,67	0,47	Mi	08:00	13:00	41, 43, 45, 47, 49, 51, 03

Abbildung 2.11: Einsatzplan mit Langnamen

Die beim Veranstaltungsplan beschriebenen Tooltips und Verlinkungen funktionieren im Einsatzplan ebenfalls.

Ausblenden

Unter dem Einsatzplan gibt es eine Zeile

Einsatzplan ausblenden, um Veranstaltungsplan zu drucken

die – Nomen est omen – nach Anklicken den Einsatzplan ausblendet, so dass nur noch der Veranstaltungsplan einschließlich aller Überschriften sichtbar ist (Abbildung 2.12) und so in den meisten Fällen auf ein DIN A4-Blatt im Querformat passt.

[8] Wenn eine Fakultät – aus welchen Gründen auch immer – dies so wünscht, kann der Anrechnungsfaktor im Einsatzplan auch ganz ausgeblendet werden.

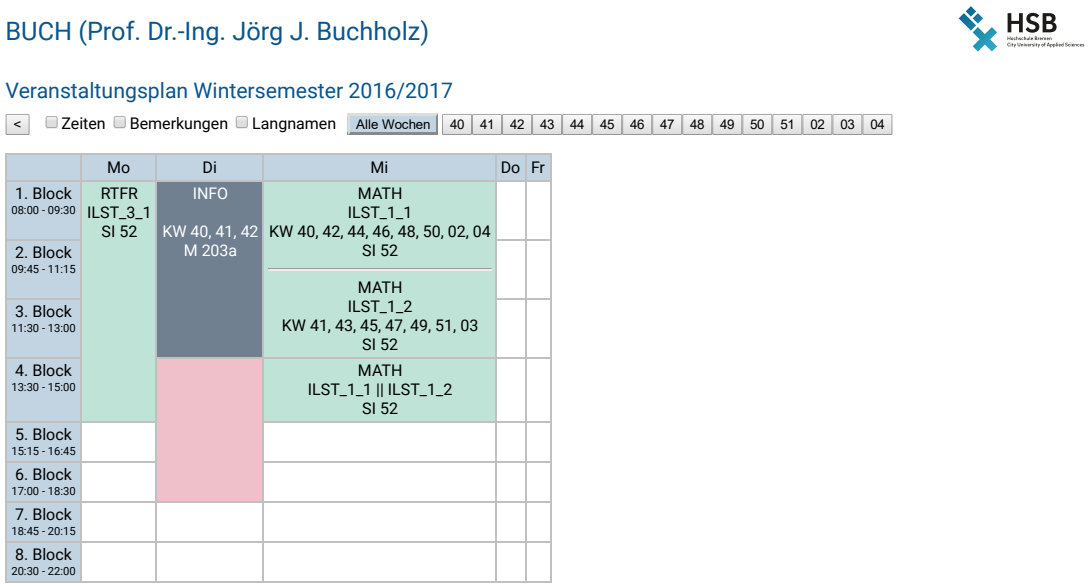


Abbildung 2.12: Einsatzplan ausgeblendet

Planungsstand

Als letzte Zeile wird auf der Seite der aktuelle Planungsstand, also das Datum und die Uhrzeit der letzten Änderung angezeigt:

Planungsstand: 11.04.2017 13:23:02

Dies betrifft allerdings immer den Stand der Gesamtplanung, so dass aus einem neuen Planungsstand nicht direkt auf eine Änderung des aktuell angezeigten Planes geschlossen werden kann.

3 Hinweise für Planerinnen

In diesem Kapitel wollen wir ein paar nützliche und notwendige Hinweise für die sinnvolle Verwendung des Planungsprogrammes daVinci [1] geben, um die daraus resultierenden Pläne korrekt mit plan darstellen zu können.

3.1 Zeitrahmen

Theoretisch könnten Planerinnen in daVinci unter Extras/Zeitrahmen für jeden Semesterverband und jede Dozentin einen eigenen Zeitrahmen definieren. In der Praxis scheint es aber momentan auszureichen, für jede Fakultät bzw. Abteilung nur den Standardzeitrahmen mit seinen Zeiten und Bezeichnungen festzulegen (Abbildung 3.1) und darzustellen.

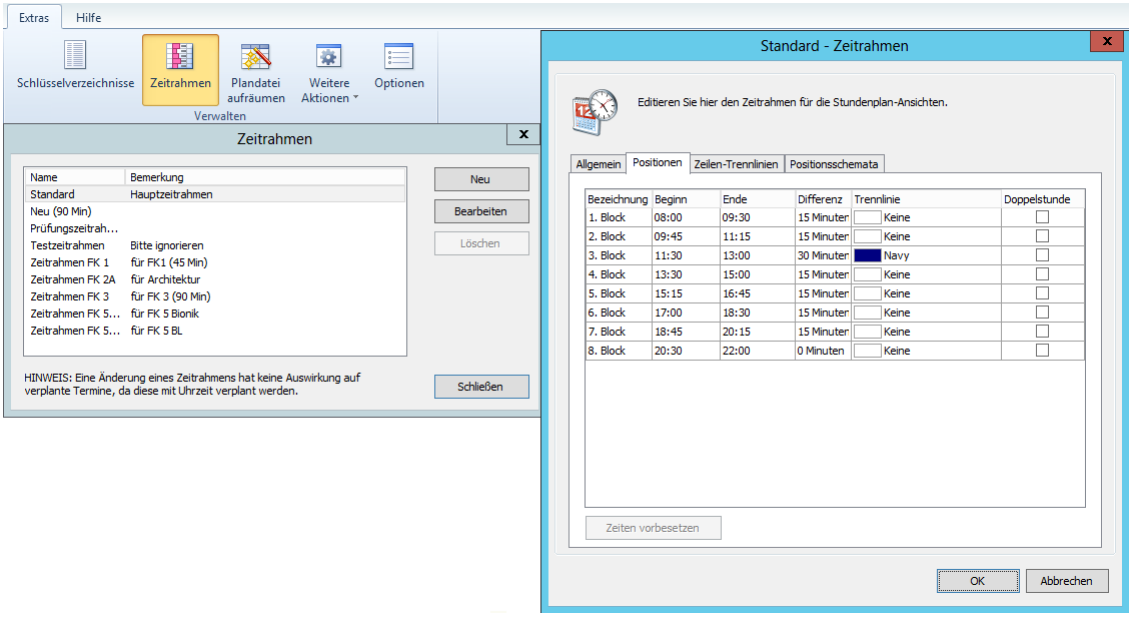


Abbildung 3.1: Standardzeitrahmen festlegen

Andere Zeitrahmen als den Standardzeitrahmen ignoriert die vorliegende Version von plan.

3.2 Faktor

Wie in Abbildung 2.10 dargestellt und in Abschnitt 2.1.2 erläutert, kann die Planerin den Faktor, mit dem eine Veranstaltung für eine Dozentin später bei der Lehrdeputatsabrechnung angerechnet wird, entweder aus der Anzahl der Kalenderwochen automatisch von daVinci berechnen lassen oder ihn explizit vorgeben. Für die Vorgabe des Faktors markiert die Planerin die Veranstaltung in der Veranstaltungsliste von daVinci, entfernt unter Start/Veranstaltung bearbeiten/Zeitdetails den Haken bei Automatisch berechnen und legt den Lehrerfaktor[1] fest (Abbildung 3.2).

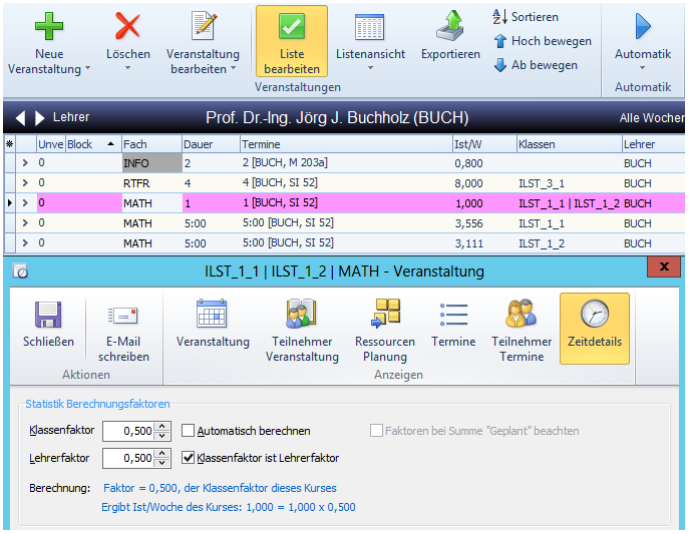


Abbildung 3.2: Anrechnungsfaktor festlegen

3.3 Gemeinsame Veranstaltungen

daVinci bietet die Möglichkeit, eine Veranstaltung gleichzeitig in mehreren Semesterverbänden, bei mehreren Dozentinnen oder in mehreren Räumen stattfinden zu lassen. Üblicherweise markiert die Planerin dazu die Veranstaltung in der Veranstaltungsliste von daVinci und trägt unter Start/Veranstaltung bearbeiten/Teilnehmer Veranstaltung gegebenenfalls mehrere Verbände, Dozentinnen und Räume ein (Abbildung 3.3).

[1] Der Klassenfaktor wird von plan nicht ausgewertet und ist daher egal.

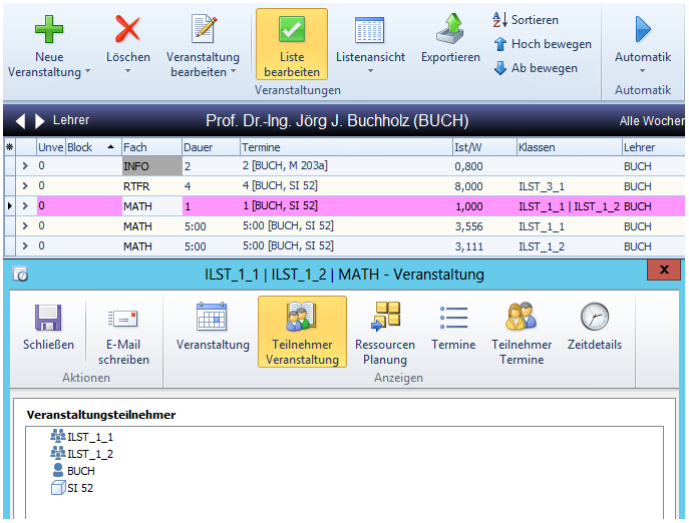


Abbildung 3.3: Gemeinsame Veranstaltungen

Alternativ kann die Planerin – nachdem sie die Schaltfläche Liste bearbeiten aktiviert hat – direkt in der Veranstaltungsliste beispielsweise in der Spalte Klassen mehrere Verbände mit Kommata getrennt eintragen. daVinci und plan trennen die Verbände dann mit einem bzw. zwei senkrechten Strichen (Abbildung 3.3 und Abbildung 2.4). Als letzte, nicht zu empfehlende Möglichkeit kann die Planerin mehrere Veranstaltungen in der Spalte Block mit der gleichen Blocknummer versehen. Die gleichzeitig stattfindenden Veranstaltungen werden dann, wie in Abbildung 2.5 am Mittwoch Vormittag dargestellt, mit einem nicht ganz durchgezogenen waagerechten Strich voneinander getrennt untereinander dargestellt.

3.4 Bemerkungen

Unterhalb der Hauptüberschrift der jeweiligen Webseite kann die Planerin eine kurze Bemerkung zur jeweiligen Dozentin, zum Verband oder zum Raum ausgeben lassen. Sie trägt diese Bemerkung einfach in den entsprechenden Stammdaten in der Spalte Bemerkung ein.

Zusätzlich kann die Planerin auf den Seiten der Dozentinnen und Verbänden zwischen dem Veranstaltungsplan und dem Einsatzplan einen weiteren Bemerkungsblock ausgeben lassen, beispielsweise, um die im Veranstaltungsplan benutzten Farben zu erklären.

Die Bemerkungen werden als HTML [2] formatiert ausgegeben, so dass die Planerin einfache HTML-Tags verwenden kann, um beispielsweise Überschriften, Schriftfarben und -arten oder sogar Hyperlinks zu definieren. So wird die folgende (beispielhafte aber sinnlose) Zwischenbemerkung

```
<h4>Eine kleine Kurzgeschichte</h4>
```

```
Lorem ipsum dolor sit amet, consectetur adipisici elit,
sed eiusmod tempor incidunt ut labore et
<font color="#FF0000">dolore</font>
magna aliqua.
Ut enim ad minim veniam,
quis nostrud exercitation ullamco
<a href="http://prof.red">laboris</a>
nisi ut aliquid ex ea commodi consequat.
Quis aute iure reprehenderit in voluptate
velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint obcaecat cupiditat non proident,
sunt in
<b>culpa</b>
qui officia deserunt mollit anim id est laborum.
```

als eigenständiger Absatz mit einer zur übrigen Webseite passenden Überschrift, einem roten Wort, einem Hyperlink und einem fett gedruckten Wort zwischen Veranstaltungsplan und Einsatzplan ausgegeben (Abbildung 3.4).

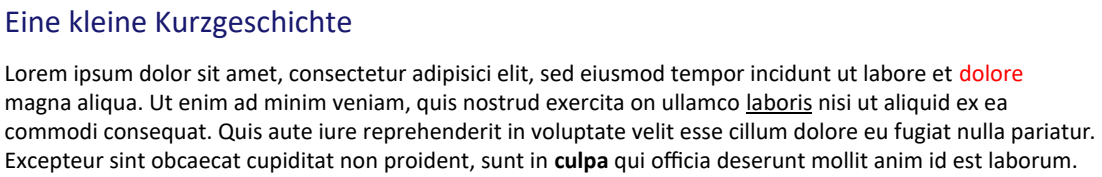


Abbildung 3.4: Zwischenbemerkung

Die Zwischenbemerkungen tragen die Planerinnen – in Ermangelung sinnvoller bezeichneter Spalten – bei den Dozentinnen und Verbänden in deren Stammdaten in der Spalte Schule ein.

3.5 Sperrungen

Unter Start/Zeitpräferenzen kann die Planerin in daVinci unter anderem Sperrzeiten eintragen, an denen für bestimmte Dozentinnen, Verbände oder Räume keine Lehre stattfinden soll (Abbildung 3.5).

Abbildung 3.5: Zeitpräferenzen und Sperrungen

plan stellt dabei nur Sperrungen der Kategorie 1 in der Farbe Lavender Blush – gegebenenfalls mit Bemerkung – dar (Abbildung 2.6), sodass die Planerin für interne Sperrungen, die nicht im Plan erscheinen sollen, einfach eine andere Kategorie verwenden kann.

Teil II

Unter der Haube

4 Blockschaltbild

Abbildung 4.1 zeigt das plan-Blockschaltbild

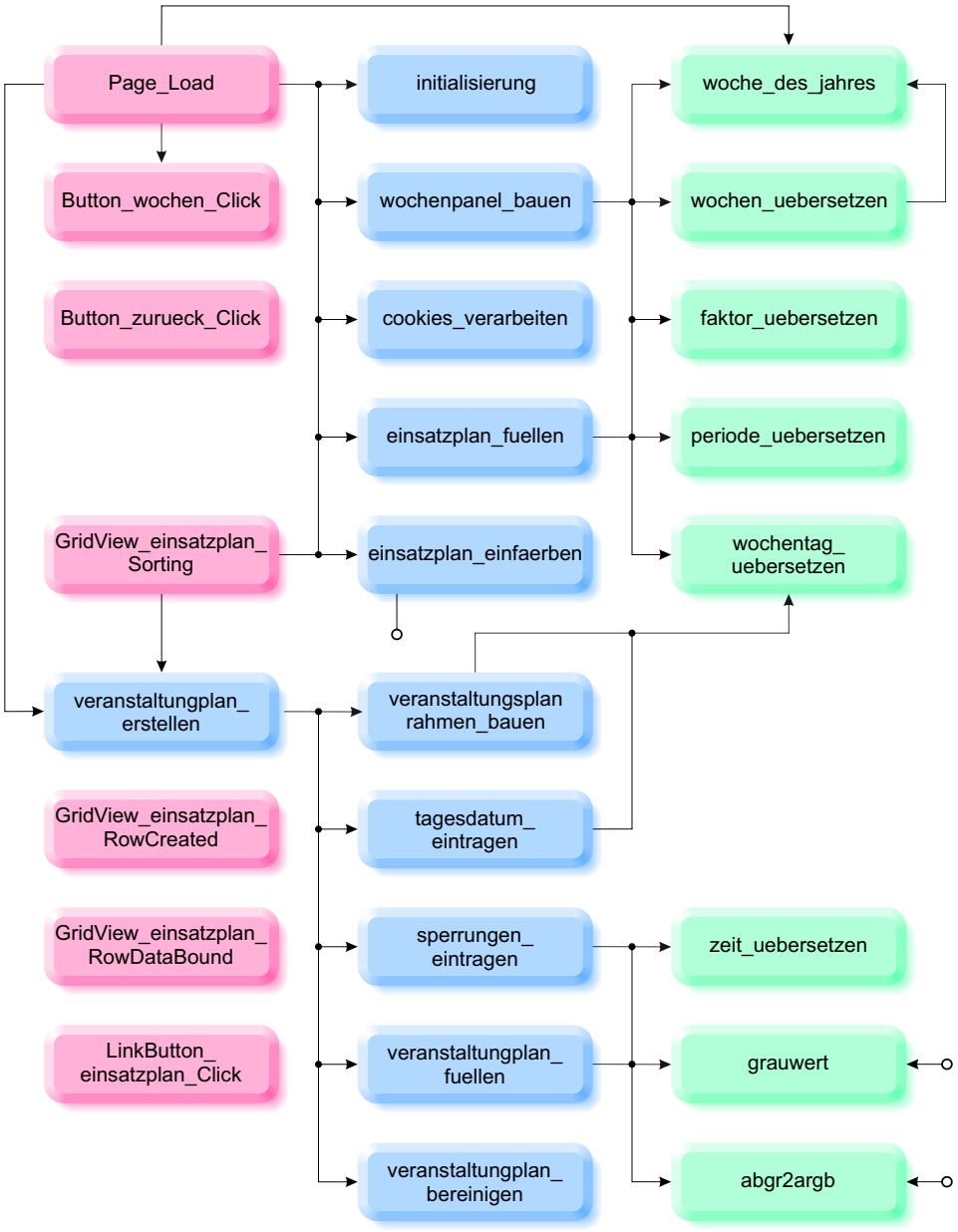


Abbildung 4.1: Blockschaltbild

wie es für die Seiten dozent.aspx, verband.aspx und raum.aspx verwendet wird. Die Seiten team.aspx, semester.aspx und auswahl.aspx (Abschnitt 2.1) sind im Vergleich zu dozent.aspx so trivial, dass wir sie hier nicht beschreiben.

Die roten Blöcke beinhalten Unterprogramme, die vom Laufzeitsystem aufgerufen werden, wenn die Webseite geladen wurde, Schaltflächen gedrückt wurden, ... Die roten Blöcke rufen dann die Unterprogramme in den blauen Blöcke auf, die wiederum die Hilfsfunktionen in den grünen Blöcken benutzen.

Wenn die Webseite geöffnet wird, ruft Page_Load die Unterprogramme initialisierung, in dem wir die Daten der aktuellen Dozentin lesen und die entsprechenden Überschriften schreiben, wochenpanel_bauen, in dem wir die Zeile mit den Wochenschaltflächen aufbauen, cookies_verarbeiten, in dem wir Cookies lesen und daraus den Anfangszustand der Auswahlfelder festlegen, einsatzplan_fuellen, in dem wir den Einsatzplan erzeugen und füllen und einsatzplan_einfaerben, in dem wir die Farben der einzelnen Module setzen, auf.

einsatzplan_einfaerben verwendet dabei die Hilfsfunktionen abgr2argb, in der wir die von daVinci benutzte Farbdarstellung in die von Visual Studio genutzte Farbkodierung umwandeln und grauwert, in der wir berechnen, ab wann wir weiße statt schwarzer Schrift verwenden sollten.

Beim ersten Laden der Seite verwendet Page_Load außerdem das Hilfsprogramm woche_des_jahres, um die aktuelle Kalenderwoche zu ermitteln und emuliert durch Aufruf von Button_wochen_Click gegebenenfalls ein Drücken der aktuellen Wochenschaltfläche. Weitere Seitenaufrufe verzweigen dann zum Unterprogramm veranstaltungsplan_erstellen, in dem der Veranstaltungsplan erneut erzeugt und mit Leben gefüllt wird.

Das Unterprogramm wochenpanel_bauen verwendet die Funktionen wochen_uebersetzen, in der wir die von daVinci verwendet Wochendarstellung in eine leichter interpretierbare Variante umwandeln und woche_des_jahres, in der wir die Kalenderwoche eines bestimmten Datums ermitteln. Die Funktion wochen_uebersetzen nutzt auch selbst die Funktion woche_des_jahres.

Im Unterprogramm einsatzplan_fuellen nutzen wir ebenfalls die Funktionen wochen_uebersetzen und woche_des_jahres und außerdem die Funktionen periode_uebersetzen, um die zu einer Periode gehörenden Wochen herauszufinden, wochentag_uebersetzen, um aus der von daVinci verwendeten numerischen Kodierung des Wochentags lesbare Wochentagesabkürzungen zu erzeugen und faktor_uebersetzen, um leere Faktoren als 1 darzustellen.

Das Unterprogramm veranstaltungsplan_erstellen nutzt als erstes veranstaltungsplanrahmen_bauen, um den äußeren Rahmen des Veranstaltungsplans zu erzeugen, tagesdatum_eintragen, um Datumsangaben zu den Wochentagen hinzuzufügen, wenn eine einzelne Woche ausgewählt wurde, sperrungen_eintragen, um eventuell vorhandene Sperrungen in den Veranstaltungsplan einzutragen, veranstaltungsplan_fuellen, um die Informationen des Einsatzplans in den Veranstaltungsplan zu übertragen und veranstaltungsplan_bereinigen, um nachträglich gleiche Blöcke zusammen zu fassen.

Sowohl veranstaltungsplanrahmen_bauen als auch tagesdatum_eintragen verwenden dabei die Hilfsfunktion wochentag_uebersetzen.

Sowohl sperrungen_eintragen als auch veranstaltungsplan_fuellen nutzen die Hilfsfunktion zeit_uebersetzen, in der wir ermitteln, in welchen Block des Zeitrahmens ein bestimmter Zeitpunkt fällt. veranstaltungsplan_fuellen verwendet außerdem die beiden Hilfsfunktionen abgr2argb und grauwert.

Ein Klick auf die zum Unterprogramm Button_zurueck_Click gehörige Schaltfläche verzweigt direkt zurück zur Auswahlseite (Abbildung 2.3).

Das Systemunterprogramm GridView_einsatzplan_Sorting wird immer dann vom Laufzeitsystem aufgerufen, wenn die Nutzerin eine neue Sortierreihenfolge im Einsatzplan angefordert hat. Nach dem Sortieren müssen die Zeilen des Einsatzplans mit einsatzplan_einfaerben erneut mit ihrer richtigen Hintergrundfarbe versehen werden.

GridView_einsatzplan_RowCreated verwenden wir, um zum richtigen Zeitpunkt die Hilfsspalten des Einsatzplans zu verstecken. GridView_einsatzplan_RowDataBound wird genutzt, um bestimmte Spalten des Einsatzplans als aktiven HTML-Code zu interpretieren.

Die einzige Funktion von LinkButton_einsatzplan_Click ist es, den Einsatzplan unsichtbar zu machen, wenn die Nutzerin auf die entsprechende Schaltfläche geklickt hat.

5 dozent.aspx

Wir haben die Seite, die die Pläne der Dozentin anzeigt, als Active Server Page [3] im ASP.NET Framework angelegt. Dazu gehört die eigentliche ASPX-Seite, die wir in diesem Kapitel beschreiben und außerdem die den aktiven Code beinhaltende Seite `dozent.aspx.vb`.

Eine ASPX-Seite beginnt üblicherweise mit drei (umbrochenen) Zeilen, die sie ein wenig von einer klassischen HTML-Seite unterscheidet:

```
<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="dozent.aspx.vb" Inherits="dozent" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
```

Danach folgt wie gewohnt der Kopf mit dem Server-Attribut

```
<head runat="server">
```

der Titel der Seite

```
  <title>Dozent_in</title>
```

und der Link zur Seite `plan.css` mit den Kaskadierende Stilvorlagen:

```
  <link
    rel="stylesheet"
    type="text/css"
    href="plan.css" />
```

Zusätzlich laden wir noch den von Google freundlicherweise bereitgestellten Font Roboto [4], der dem im Corporate Design der Hochschule Bremen festgelegten Brix Sans ziemlich nahe kommt:

```
  <link
    rel="stylesheet"
    type="text/css"
    href="https://fonts.googleapis.com/css?family=Roboto">
</head>
```

Der Körper der Seite

```
<body>
```

besteht aus einer auf dem Server ausgeführten Form

```
  <form id="form1" runat="server">
```

in der wir die drei Label für den Dozentinnennamen (als Überschrift dritter Ordnung)

```
    <h3>
      <asp:Label
        ID="Label_dozent"
        runat="server">
      </asp:Label>
    </h3>
```

für den Dozentinnenkommentar

```
    <asp:Label
      ID="Label_kommentar"
      runat="server">
    </asp:Label>
```

und für den Namen des Veranstaltungsplanes (als Überschrift vierter Ordnung) ausgeben:

```
    <h4>Veranstaltungsplan
      <asp:Label
        ID="Label_veranstaltungsplan"
        runat="server">
      </asp:Label>
    </h4>
```

Außerdem positionieren wir das Logo der Hochschule Bremen auf der Webseite ganz rechts in die obere Ecke

```
    <img
      src="HSB_Horizontal_4C.svg"
      alt="Logo der Hochschule Bremen"
      width="150"
      class="rechts_oben">
```

Als nächstes folgt das Panel mit den Auswahlfeldern für Optionen und Wochen

```
    <asp:Panel
      ID="Panel_wochen"
      runat="server">
```

auf dem wir neben einer Zurück-Schaltfläche

```
      <asp:Button
        ID="Button_zurueck"
        runat="server"
        Text="<"
        ToolTip="Zurück zur Auswahlseite" />
       
```

die drei Optionen für die Anzeige der Zeiten

```
      <asp:CheckBox
        ID="CheckBox_zeiten"
        runat="server"
        AutoPostBack="True"
        Text="Zeiten" />
```

der Bemerkungen

```
      <asp:CheckBox
        ID="CheckBox_bemerkungen"
        runat="server"
        AutoPostBack="True"
        Text="Bemerkungen" />
```

und der Langnamen darstellen:

```
      <asp:CheckBox
        ID="CheckBox_langnamen"
        runat="server"
        AutoPostBack="True"
        Text="Langnamen" />
```

Durch ein Leerzeichen getrennt folgt rechts daneben die Schaltfläche Alle Wochen:

```
       
      <asp:Button
        ID="Button_alle_wochen"
        runat="server"
        Text="Alle Wochen"
        ToolTip="Alle Wochen" />
    </asp:Panel>
```

Die Auswahlfelder für die anderen Wochen erstellen wir später dynamisch.

Mit einer Zeile Abstand folgt die Tabelle, die später den Veranstaltungsplan beinhalten wird:

```
    <br />
    <asp:Table
      ID="Table_veranstaltungsplan"
      runat="server"
```

```
      BorderColor="Silver"
      GridLines="Both"
      EnableViewState="False">
    </asp:Table>
```

Nach der Legende, in der die Planerin Zusatzinformationen über die Pläne eintragen kann

```
    <asp:Label
      ID="Label_legende"
      runat="server"
      Text="">
    </asp:Label>
```

folgt das Panel des Einsatzplanes

```
    <asp:Panel
      ID="Panel_einsatzplan"
      runat="server">
```

auf dem wir nach der Überschrift (vierter Ordnung)

```
      <h4>Einsatzplan
        <asp:Label
          ID="Label_einsatzplan"
          runat="server">
        </asp:Label>
      </h4>
```

das GridView darstellen, das später den Einsatzplan beinhalten wird:

```
      <asp:GridView
        ID="GridView_einsatzplan"
        runat="server"
        AllowSorting="True"
        BorderColor="Silver"
        CellPadding="5"
        CellSpacing="-1"
        UseAccessibleHeader="False">
        <AlternatingRowStyle
          BackColor="#BFE3D6" />
        <HeaderStyle
          BackColor="#C2D4E2"
          HorizontalAlign="Left" />
      </asp:GridView>
```

Nach einer weiteren Leerzeile folgt dann noch die Schaltfläche, mit der die Nutzerin den Einsatzplan ausblenden kann, um den Veranstaltungsplan zu drucken:

```
      <br />
      <asp:LinkButton
        ID="LinkButton_einsatzplan"
        runat="server">
        Einsatzplan ausblenden, um Veranstaltungsplan zu drucken
      </asp:LinkButton>
```

Abschließend geben wir noch den Datum und Uhrzeit der letzten Änderung der Planungsdatei aus:

```
      <br />
      <br />
      Planungsstand:
      <asp:Label
        ID="Label_planungsstand"
        runat="server">
      </asp:Label>
    </asp:Panel>
  </form>
</body>
</html>
```

5.1 dozent.aspx.vb

Den aktiven, dynamischen Code (Code-Behind) der Dozentinnenseite, den wir im Folgenden beschreiben werden, haben wir in Visual Basic geschrieben und in der Datei `dozent.aspx.vb` abgelegt.

5.1.1 Globale Variablen

Ja, globale Variablen sind schlechter Programmierstil, aber sie machen das Leben so viel einfacher, wenn Informationen zwischen mehreren Unterprogrammen ausgetauscht werden sollen. Wir leisten uns daher den dekadenten Luxus, hier ein paar überall verfügbare Variablen zu erklären:

```
Shared davinci_xml As XElement

Shared dozent_id As String

Shared aktuelle_kalenderwoche As String

Shared aktuelles_wochendatum As String

Shared faktor_ausblenden As Boolean
```

```
Dim DataTable_einsatzplan As New System.Data.DataTable
```

5.1.2 Page_Load

Das als „Hauptprogramm" fungierende

```
Protected Sub Page_Load (
  ByVal sender As Object,
  ByVal e As System.EventArgs)
  Handles Me.Load
```

wird aufgerufen, wenn die Webseite dozent.aspx angefordert wird und ruft seinerseits die Unterprogramme initialisierung, wochenpanel_bauen, cookies_verarbeiten, einsatzplan_fuellen und einsatzplan_einfaerben auf:

```
initialisierung()

wochenpanel_bauen()

cookies_verarbeiten()

einsatzplan_fuellen()

einsatzplan_einfaerben()
```

Wenn die Seite zum ersten Mal aufgerufen wird

```
If Not Page.IsPostBack Then
```

definieren wir, welche Woche(n) im Plan dargestellt werden. Außerhalb des Plangeltungszeitraums soll die Schaltfläche Alle Wochen ausgewählt sein:

```
  Dim schaltflaeche = Button_alle_wochen)
```

Um zu überprüfen, ob wir uns momentan gerade im Planungszeitraum befinden, starten wir eine Schleife über alle Steuerelemente des Panels:

```
  For Each panel_control In Panel_wochen.Controls
```

Wenn nun die Wochennummer des aktuellen Datums der Aufschrift der gerade betrachteten Wochenschaltfläche entspricht

```
    If TypeOf panel_control Is Button And
      woche_des_jahres(Now) = panel_control.Text Then
```

wählen wir die aktuelle Wochenschaltfläche aus:

```
      schaltflaeche = panel_control

    End If

  Next
```

Anschließend emulieren wir einen Klick auf die gefundene Schaltfläche, um sicherzustellen, dass die entsprechende Woche dargestellt werden:

```
  Button_wochen_Click(schaltflaeche, EventArgs.Empty)
```

Wenn die Seite dann wiederholt mit neuen Einstellungen aufgerufen wird

```
Else
```

stellen wir nur den Veranstaltungsplan neu dar:

```
  veranstaltungplan_erstellen()

End If

End Sub
```

5.1.3 initialisierung

Im Unterprogramm

```
Protected Sub initialisierung()
```

lesen wir Eigenschaften der aktuellen Dozentin aus der Datenbank und stellen sie auf der Webseite dar. Dazu lesen wir als erstes das aktuelle Team und das aktuelle Semester, das die Nutzerin beim Aufruf der Seite mit übergeben muss:

```
Dim aktuelles_team = Request.QueryString("team")

Dim aktuelles_semester = Request.QueryString("semester")

Dim dozent_code = Request.QueryString("code")
```

Wenn eine der drei Angaben nicht vorhanden ist

```
If aktuelles_team = Nothing Or
  aktuelles_semester = Nothing Or
  dozent_code = Nothing Then
```

kann das Programm nicht weiter arbeiten und verzweigt zurück auf die Team-Auswahlseite:

```
  Response.Redirect("team.aspx")

End If
```

Wenn alle Angaben vorhanden sind, laden wir die zentrale XML-Datei des Programmes, in der Informationen über die Teams und Semester stehen

```
Dim plan_xml = XElement.Load(MapPath("plan.xml"))
```

und lesen daraus die aktuellen Namen der daVinci-Datei und des Semesters und die Information, ob der Anrechnungsfaktor für dieses Teams angezeigt oder ausgeblendet werden soll:

```
Dim semesters =
  From semester In plan_xml.<team>.<semester>
  Where semester.Parent.<code>.Value = aktuelles_team And
  semester.<code>.Value = aktuelles_semester
  Select
  datei = semester.<datei>.Value,
  semester_name = semester.<name>.Value,
  faktor_ausblenden = semester.Parent.<faktor_ausblenden>.Value
```

Die Entscheidung über das Anzeigen des Anrechnungsfaktors übertragen wir in eine logische globale Variable, die wir später in `einsatzplan_fuellen` verwenden werden:

```
faktor_ausblenden =
  Convert.ToBoolean(semesters.FirstOrDefault.faktor_ausblenden)
```

Aus dem Dateinamen basteln wir den Gesamtpfad der Datei zusammen:

```
Dim aktuelle_datei = MapPath(
  "plaene\" &
  semesters.FirstOrDefault.datei &
  ".davinci")
```

Wenn die Datei – aus welchen Gründen auch immer – nicht existiert, geht es ebenfalls zurück zur Team-Auswahlseite:

```
If Not FileIO.FileSystem.FileExists(aktuelle_datei) Then

  Response.Redirect("team.aspx")

End If
```

Wenn die Datei aber gefunden wurde, laden wir sie:

```
davinci_xml = XElement.Load(aktuelle_datei)
```

und lesen die relevanten Informationen der aktuellen Dozentin:

```
Dim dozenten =
  From dozent In davinci_xml.<Teacher>.<Items>.<Item>
  Where dozent.<Code>.Value = dozent_code
  Select
  id = dozent.@ID,
  surname = dozent.<Surname>.Value,
  first_name = dozent.<FirstName>.Value,
  title = dozent.<Title>.Value,
  kommentar = dozent.<Comments>.Value,
  legende = dozent.<SchoolNumber>.Value
```

Wenn keine Dozentin mit dem angeforderten Namen gefunden wurde

```
If dozenten.Count = 0 Then
```

springen wir zurück auf die Auswahlseite der Dozentinnen:

```
  Response.Redirect("auswahl.aspx")

End If
```

Als nächstes speichern wir die dozentinnenspezifischen Informationen in etwas ansprechenderen Variablen[1] zwischen

```
dozent_id = dozenten.First.id

Dim dozent_first_name = dozenten.First.first_name

Dim dozent_surname = dozenten.First.surname

Dim dozent_title = dozenten.First.title

Dim dozent_kommentar = dozenten.First.kommentar

Dim dozent_legende = dozenten.First.legende
```

und schreiben den Namen des Semesters in die Überschrift des Veranstaltungsplans und des Einsatzplans (Abbildung 2.4):

```
Label_veranstaltungsplan.Text =
  semesters.FirstOrDefault.semester_name

Label_einsatzplan.Text =
  semesters.FirstOrDefault.semester_name
```

Wenn die Planerin einen Dozentinnentitel eingetragen hat

[1] Die Variable `dozent_id` muss hier nicht nochmals deklariert werden, da sie schon in Abschnitt 5.1.1 zur globalen Variablen erklärt wurde.

```
If Not IsNothing(dozent_title) Then
```

hängen wir als Trennzeichen zwischen Titel und Vornamen ein Leerzeichen an den Titel:

```
  dozent_title &= " "

End If
```

Auch ein vorhandener Vorname bekommt ein Leerzeichen angehängt:

```
If Not IsNothing(dozent_first_name) Then

  dozent_first_name &= " "

End If
```

Jetzt können wir die Gesamtüberschrift der Dozentin bestehend aus seinem Kürzel und seinem Gesamtnamen in Klammern ausgeben:

```
Label_dozent.Text = dozent_code &
  " (" &
  dozent_title &
  dozent_first_name &
  dozent_surname &
  ")"
```

Gegebenenfalls füllen wir noch die Felder „Kommentar" und „Legende":

```
Label_kommentar.Text = dozent_kommentar

Label_legende.Text = dozent_legende
```

Abschließend lesen wir den aktuellen Planungsstand aus der Planungsdatei

```
Dim planungsstand As DateTime = davinci_xml.<Changed>.Value
```

und stellen ihn dar:

```
Label_planungsstand.Text = planungsstand

End Sub
```

5.1.4 wochenpanel_bauen

Im Unterprogramm

```
Private Sub wochenpanel_bauen()
```

erzeugen wir die Zeile mit den Auswahlfeldern für die Wochen (Abbildung 2.5). Dazu lesen wir als erstes das Startdatum

```
Dim startdatum As DateTime =
  davinci_xml.<Settings>.<TimetablePeriodFrom>.Value
```

und das Enddatum des aktuellen Planes aus der im Unterprogramm initialisierung geladenen und als globale Variable verfügbaren Plandatei davinci_xml:

```
Dim enddatum As DateTime =
  davinci_xml.<Settings>.<TimetablePeriodTo>.Value
```

Um Veranstaltungsausfallwochen (Ostern, Weihnachten, ...) in der Liste der auswählbaren Wochen nicht darzustellen, lesen wir außerdem die Liste der tatsächlichen Veranstaltungswochen und übersetzen[2] sie mit der Hilfsfunktion wochen_uebersetzen in eine leichter interpretierbare Form:

```
Dim unterrichtswochen =
  wochen_uebersetzen(davinci_xml.<Settings>.<LessonWeeks>.Value)
```

Als nächstes beginnen wir die Schleife über alle darzustellenden Wochen. Dazu setzen wir die Laufvariable auf das Anfangsdatum

```
Dim datum As DateTime = startdatum
```

und lassen sie bis zum Enddatum laufen:

```
While datum <= enddatum
```

In der Schleife erzeugen wir für jede Woche eine neue Schaltfläche

```
  Dim Button_woche As New Button
```

und definieren das Unterprogramm, das aufgerufen wird, wenn die Nutzerin auf die Schaltfläche klickt:

```
  AddHandler Button_woche.Click, AddressOf Button_wochen_Click
```

Aus dem aktuellen Datum ermitteln wir mit der Hilfsfunktion woche_des_jahres seine Kalenderwoche und schreiben sie als Text auf die Schaltfläche:

```
  Button_woche.Text = woche_des_jahres(datum)
```

Um die Feiertagswochen auszublenden, untersuchen wir, ob die aktuelle Kalenderwoche in der Liste der stattfindenden Wochen enthalten ist:

```
  If unterrichtswochen.Contains(Button_woche.Text) Then
```

Wenn dies der Fall ist, schreiben wir – da die meisten Planerinnen mit der Angabe der Kalenderwochen erst einmal wenig anfangen können – das Start- und Enddatum der aktuellen Kalenderwoche in den Tooltip der Schaltfläche, der immer dann in einem kleinen Extrafenster angezeigt wird, wenn der Mauszeiger etwas länger auf der Schaltfläche verweilt

[2] Intern wird die Liste als eine Zeichenkette aus Nullen und Einsen geführt.

```
    Button_woche.ToolTip = datum & " - " & datum.AddDays(6)
```

und fügen die neue Schaltfläche (rechts) an die schon vorhandenen Schaltflächen an:

```
    Panel_wochen.Controls.Add(Button_woche)
```

Wenn die gerade erzeugte Schaltfläche der aktuellen Woche entspricht

```
    If woche_des_jahres(Now) = Button_woche.Text Then
```

dann heben wir den Text dieser Schaltfläche etwas hervor:

```
      Button_woche.Font.Bold = True

    End If

  End If
```

Da wir uns gerade in einer `While`-Schleife befinden, müssen wir anschließend noch ihre Laufvariable manuell um eine Woche weiter setzen:

```
  datum = datum.AddDays(7)

End While

End Sub
```

5.1.5 cookies_verarbeiten

Im Unterprogramm

```
Private Sub cookies_verarbeiten()
```

überprüfen wir beim ersten Aufruf der Seite

```
If Not Page.IsPostBack Then
```

ob es aus einem vorherigen Aufruf Cookies gibt, in denen der Zustand der Auswahlfelder Zeiten, Bemerkungen und Langnamen abgespeichert wurde. Wenn[3] dies der Fall ist, stellen wir diese Zustände in den entsprechenden Auswahlfeldern dar:

```
  Try

    CheckBox_zeiten.Checked =
      Request.Cookies("plan")("zeiten")
```

[3] Da wir an dieser Stelle ein bisschen faul sind, sparen wir uns die zur Vermeidung von Fehlermeldungen eigentlich notwendigen aber lästigen Abfragen der Existenz der einzelnen Cookies und kapseln das Lesen der Cookies in einem `Try`-Block, der die potenziellen Fehlermeldungen sicher abfängt.

```
    CheckBox_bemerkungen.Checked =
      Request.Cookies("plan")("bemerkungen")

    CheckBox_langnamen.Checked =
      Request.Cookies("plan")("langnamen")

  Catch ex As Exception

  End Try
```

Auf diese Weise muss eine Nutzerin ihre Lieblingsauswahl nur einmalig treffen und bekommt auch beim nächsten Aufruf der Seite die Pläne wieder in der gewünschten Form angezeigt.

Wenn die Seite erneut aufgerufen wird, beispielsweise, weil die Nutzerin eines der Auswahlfelder angeklickt und damit geändert hat

```
Else
```

speichern[4] wir den[5] (neuen) Zustand der Auswahlfelder in Cookies ab:

```
  Response.Cookies("plan")("zeiten") =
    CheckBox_zeiten.Checked

  Response.Cookies("plan")("bemerkungen") =
    CheckBox_bemerkungen.Checked

  Response.Cookies("plan")("langnamen") =
    CheckBox_langnamen.Checked
```

Abschließend geben wir der Cookie-Kollektion noch eine – rein willkürliche aber vermutlich angemessene – Lebensdauer von 30 Tagen:

```
  Response.Cookies("plan").Expires = Now.AddDays(30)

End If

End Sub
```

5.1.6 einsatzplan_fuellen

Nach den Vorbereitungen können wir in

[4] Man achte hier auf die vielleicht etwas kontraintuitive Tatsache, dass Cookies im `Response`-Objekt abgespeichert, aber aus dem `Request`-Objekt gelesen werden.

[5] Auch hier könnten wir natürlich genau analysieren, welche Felder denn wirklich geändert wurden, um den Speicheraufwand zu reduzieren, indem wir nur die geänderten Felder neu abspeichern. Bei drei Feldern mag man uns die genutzte quick-and-dirty-Lösung aber vielleicht verzeihen.

```
Private Sub einsatzplan_fuellen()
```

alle Veranstaltungen in den Einsatzplan eintüten und diesen darstellen. Dazu lesen[6] wir als erstes das aktuelle Team und das aktuelle Semester aus den Parametern des Seitenaufrufs

```
Dim aktuelles_team = Request.QueryString("team")

Dim aktuelles_semester = Request.QueryString("semester")
```

Den Einsatzplan stellen wir in einem GridView dar, das uns das Bereitstellen einer Sortiermöglichkeit sehr stark vereinfacht. Wie bei GridViews üblich, trennen wir Daten und Anzeige und speichern die anzuzeigenden Daten in einer Datentabelle (DataTable) ab. Dazu erzeugen wir als erstes die[7] Spalten der Tabelle:

```
DataTable_einsatzplan.Columns.Add("farbe")

DataTable_einsatzplan.Columns.Add("tag_integer")

DataTable_einsatzplan.Columns.Add("Modul")

DataTable_einsatzplan.Columns.Add("Verband")

DataTable_einsatzplan.Columns.Add("Raum")

DataTable_einsatzplan.Columns.Add("Bemerkung")

DataTable_einsatzplan.Columns.Add("SWS")

If Not faktor_ausblenden Then

   DataTable_einsatzplan.Columns.Add("Faktor")

End If

DataTable_einsatzplan.Columns.Add("Tag")

DataTable_einsatzplan.Columns.Add("Beginn")

DataTable_einsatzplan.Columns.Add("Ende")

DataTable_einsatzplan.Columns.Add("Kalenderwochen")
```

Wir lesen alle Veranstaltungen der aktuellen Dozentin aus der Plandatei

[6] Da wir dies in mehreren Unterprogrammen machen, könnten wir darüber nachdenken, auch die Variablen des QueryStrings einmalig einzulesen und dann als globale Variablen zur Verfügung zu stellen.

[7] Als kleine Besonderheit wird die Spalte mit dem Anrechnungsfaktor nicht erstellt, wenn die entsprechende Fakultät diesen – aus welchen Gründen auch immer – nicht anzeigen möchte.

```
Dim dozenten_veranstaltungen =
  From dozenten_veranstaltung
  In davinci_xml.<Teacher>.<Items>.<Item>.<Events>.<Items>.<Item>
  Where dozenten_veranstaltung.Parent.Parent.Parent.@ID =
  dozent_id
```

und beginnen eine Schleife über alle diese Veranstaltungen:

```
For Each dozenten_veranstaltung In dozenten_veranstaltungen
```

Da in der Liste der Veranstaltungen der aktuellen Dozentin nur die ID des zu der Veranstaltung gehörigen Ereignis aufgeführt ist, müssen wir in die Liste der Ereignisse (`Events`) wechseln, um dort die Eigenschaften der Veranstaltung zu finden:

```
  Dim veranstaltung =
    From vveranstaltung In davinci_xml.<Events>.<Items>.<Item>
    Where vveranstaltung.@ID = dozenten_veranstaltung.@ID
```

Jede Veranstaltung kann an unterschiedlichen Terminen stattfinden. Wir ermitteln daher alle Termine der aktuellen Veranstaltung

```
  Dim termine = veranstaltung.<Times>.<Items>.<Item>
```

und beginnen eine Schleife über alle diese Termine:

```
  For Each termin In termine
```

Für jeden Termin erzeugen wir eine neue Zeile der Tabelle

```
    Dim zeile = DataTable_einsatzplan.NewRow()
```

hängen die neue Zeile an die Tabelle an

```
DataTable_einsatzplan.Rows.Add(zeile)
```

und füllen im Folgenden dann die einzelnen Spalten der neuen Zeile mit Leben. Als erstes wollen wir den Modulnamen in seine Spalte eintragen. Dazu beschaffen wir in der Liste der Module (`Subjects`) das[8] zu der aktuellen Veranstaltungs-ID gehörige Modul:

```
    Dim fach =
      From ffach In davinci_xml.<Subjects>.<Items>.<Item>
      Where ffach.@ID = veranstaltung.<Subject>.@ID
```

Wir müssen jetzt überprüfen, ob die Nutzerin Langnamen oder Kürzel der Modulnamen sehen möchte. Wenn sie also das Auswahlfeld für Langnamen markiert hat

```
    If CheckBox_langnamen.Checked Then
```

[8] Jeder Veranstaltung können mehrere Dozentinnen, Semesterverbände, Räume oder Termine zugeordnet sein; es gibt aber immer nur genau ein Modul pro Veranstaltung.

schreiben wir das Kürzel in den Tooltip (`title`) und den Langnamen[9]

in das Tabellenfeld selbst:

```
      zeile("Modul") = "<div title=""" &
        fach.<Code>.Value & """>" &
        fach.<Name>.Value & "</div>"
```

Wenn die Nutzerin mehr an den Kürzeln interessiert ist

```
    Else
```

kommt der Langname in den Tooltip und das Kürzel wird angezeigt:

```
      zeile("Modul") = "<div title=""" &
        fach.<Name>.Value & """>" &
        fach.<Code>.Value & "</div>"

    End If
```

Die von der Planerin gewünschte Modulfarbe schreiben wir erst einmal in eine Hilfsspalte der Tabelle

```
    zeile("farbe") = fach.<Color>.Value
```

die wir in `GridView_einsatzplan_RowCreated` später unsichtbar machen. Wir haben das Einfärben der Tabellenzeilen in das Unterprogramm `einsatzplan_einfaerben` ausgelagert, das diese Farbspalte dann verwendet, da wir die Zeilen ja nach einem Sortiervorgang immer wieder neu einfärben müssen.

Als nächstes bestimmen wir alle Semesterverbände, die an der aktuellen Veranstaltung teilnehmen

```
    Dim alle_verbaende = veranstaltung.<Classes>.<Items>.<Item>
```

und beginnen eine Schleife über alle diese Veranstaltungen:

```
    For Each ein_verband In alle_verbaende
```

Auch hier müssen wir wieder in die Liste der Verbände selbst wechseln, um an die Eigenschaften des aktuellen Verbandes heran zu kommen:

```
      Dim verband =
        From vverband In davinci_xml.<Classes>.<Items>.<Item>
        Where vverband.@ID = ein_verband.@ID
```

Wenn es mehr als einen Verband gibt, möchten wir die einzelnen Verbandsnamen mit zwei senkrechten Strichen (und jeweils einem Leerzeichen vorher und nachher) trennen.

[9] Da wir hier HTML-Code (<div ...) nutzen, müssen wir in `GridView_einsatzplan_RowDataBound` später noch explizit dafür sorgen, dass der HTML-Code in den entsprechenden Spalten auch tatsächlich interpretiert wird. Üblicherweise ist das in einem GridView nicht der Fall.

Dieser Trenner soll aber natürlich nicht vor dem ersten Verband (und auch nicht nach dem letzten Verband) eingefügt werden. Wir überprüfen daher, ob im aktuellen Feld schon ein Verband eingetragen ist

```
      If Not IsDBNull(zeile("Verband")) Then
```

und fügen den Trenner nur dann ein, wenn dies der Fall ist:

```
        zeile("Verband") &= " || "

      End If
```

Wenn die Nutzerin die Langnamen der Verbände sehen möchte

```
      If CheckBox_langnamen.Checked Then
```

verwenden wir – wie in der Modulspalte – das Kürzel des Verbandes als Tooltip und seinen Langnamen als Texteintrag. Darüber hinaus kodieren[10] wir den Verbandsnamen auch als Hyperlink, so dass die Nutzerin nach Anklicken direkt auf die Seite des Verbands geleitet wird:

```
        zeile("Verband") &=
        "<a title=""" &
        verband.<Code>.Value &
        """ href=""https://m-server.fk5.hs-bremen.de/plan/
        verband.aspx?code=" &
        verband.<Code>.Value &
        "&semester=" &
        aktuelles_semester &
        "&team=" &
        aktuelles_team &
        """>" &
        verband.<Name>.Value &
        "</a>"
```

Wenn die Nutzerin nur die Verbandskürzel sehen möchte

```
      Else
```

packen wir den Langnamen in den Tooltip und das Kürzel ins Textfeld und beenden die Schleife über alle Verbände:

```
        zeile("Verband") &=
        "<a title=""" &
        verband.<Name>.Value &
        """ href=""https://m-server.fk5.hs-bremen.de/plan/
        verband.aspx?code=" &
        verband.<Code>.Value &
```

[10] Die hier verwendete Zeichenkettenzusammenstückelung ist sicherlich etwas unübersichtlich, ist aber syntaktisch korrekt und erfüllt ihren Zweck.

```
        "&semester=" &
        aktuelles_semester &
        "&team=" &
        aktuelles_team &
        """>" &
        verband.<Code>.Value &
        "</a>"

      End If

    Next
```

Die Spalte der Räume füllen wir auf die gleiche gerade erläuterte Weise wie die der Verbände. Wir lesen alle zum aktuellen Termin gebuchten Räume

```
    Dim alle_raeume = termin.<Rooms>.<Items>.<Item>
```

und durchlaufen alle diese Räume:

```
    For Each ein_raum In alle_raeume
```

Wir besorgen uns die Informationen des aktuellen Raumes

```
      Dim raum =
        From rraum In davinci_xml.<Rooms>.<Items>.<Item>
        Where rraum.@ID = ein_raum.@ID
```

und fügen einen Trenner ein, wenn schon ein Raum vorhanden ist:

```
      If Not IsDBNull(zeile("Raum")) Then

        zeile("Raum") &= " || "

      End If
```

Abhängig davon, ob die Nutzerin Kürzel oder Langnamen bevorzugt, schreiben wir die entsprechenden Tooltip und Hyperlinks:

```
      If CheckBox_langnamen.Checked Then

        zeile("Raum") &=
        "<a title=""" &
        raum.<Code>.Value &
        """ href=""https://m-server.fk5.hs-bremen.de/plan/
        raum.aspx?code=" &
        raum.<Code>.Value &
        "&semester=" &
        aktuelles_semester &
        "&team=" &
        aktuelles_team &
```

```
        """>" &
        raum.<Name>.Value &
        "</a>"

      Else

        zeile("Raum") &=
        "<a title=""" &
        raum.<Name>.Value &
        """ href=""https://m-server.fk5.hs-bremen.de/plan/
        raum.aspx?code=" &
        raum.<Code>.Value &
        "&semester=" &
        aktuelles_semester &
        "&team=" &
        aktuelles_team &
        """>" &
        raum.<Code>.Value &
        "</a>"

      End If

    Next
```

Als nächstes wollen wir die Bemerkungen eintragen. Dabei gibt es die Besonderheit zu beachten, dass sowohl die Veranstaltung selbst als auch jeder ihrer Termine eine eigene Bemerkung besitzen können. Wir tragen[11] also als erstes die Bemerkung der Veranstaltung in die entsprechende Tabellenspalte ein:

```
    zeile("Bemerkung") = veranstaltung.<Comments>.Value
```

Wenn (und nur wenn) jetzt zusätzlich auch noch der Termin eine Bemerkung besitzt, fügen wir einen Trenner zwischen beiden Bemerkungen ein:

```
    If Not (IsNothing(veranstaltung.<Comments>.Value) Or
      IsNothing(termin.<Comment>.Value)) Then

      zeile("Bemerkung") &= " || "

    End If
```

Jetzt können wir (zusätzlich) die Bemerkung des Termins eintragen:

```
    zeile("Bemerkung") &= termin.<Comment>.Value
```

Im nächsten Schritt wollen wir die Spalte der Semesterwochenstunden füllen. Dazu lesen wir die von daVinci berechnete Dauer des Termins in Minuten

[11] Wenn die Planerin keine Bemerkung eingetragen hat, bleibt das Feld einfach leer. Eine weitere Abfrage ist dabei nicht nötig.

```
    Dim dauer As Double = termin.<SetDuration>.Value
```

teilen sie durch die Dauer einer „Stunde” (45 Minuten) und tragen sie mit zwei Nachkommastellen ein:

```
    zeile("SWS") = Math.Round(dauer / 45, 2)
```

Beim Anrechnungsfaktor müssen wir zwei Dinge beachten. Als erstes müssen wir überprüfen, ob die Fakultät den Faktor überhaupt anzeigen lassen möchte:

```
    If Not faktor_ausblenden Then
```

Außerdem müssen wir berücksichtigen, dass ein Faktor von eins von daVinci überhaupt nicht eingetragen wird. Wir verwenden daher die Hilfsfunktion `faktor_uebersetzen`, um auch in diesem Fall einen sinnvollen Wert mit zwei Nachkommastellen eintragen zu können:

```
      Dim faktor As Double =
        faktor_uebersetzen(veranstaltung.<TeacherFactor>.Value)

      zeile("Faktor") = Math.Round(faktor, 2)

    End If
```

Zum Eintragen des Start- und Endzeitpunkts einer Veranstaltung deklarieren wir zwei Variablen der Klasse Datum/Zeit

```
    Dim start_punkt As DateTime

    Dim end_punkt As DateTime
```

und untersuchen als erstes, ob die Planerin für den aktuellen Termin überhaupt schon Zeiten eingetragen hat.

```
    If Not termin.<Start>.Value = Nothing Then
```

Wenn dies der Fall ist, wandeln[12] wir die Einträge aus der Plandatei in Datum/Zeit-Angaben um

```
      start_punkt = termin.<Start>.Value

      end_punkt = termin.<End>.Value
```

und tragen sie in die entsprechenden Spalten ein:

```
      zeile("Beginn") =
        start_punkt.ToUniversalTime.ToString("HH:mm")
```

[12] daVinci verwendet in seiner Plandatei Zeiteinträge der Form `1899-12-30T13:30:00.000Z`, wobei das Datum offensichtlich egal ist und die Zeitangabe durch das Z am Ende als Zuluzeit alias koordinierte Weltzeit (UTC) alias Greenwich Mean Time (GMT) kenntlich gemacht ist.

```
      zeile("Ende") =
        end_punkt.ToUniversalTime.ToString("HH:mm")
```

Dabei müssen wir – um die eine Stunde Zeitverschiebung zur Lokalzeit zu kompensieren – mit Hilfe der Methode `ToUniversalTime` berücksichtigen, dass die Zeiten als UTC gekennzeichnet sind. Außerdem interessieren uns die Sekunden und ihre Bruchteile in der Uhrzeit nicht.

Beim Eintrag des Wochentages des Termins gilt es, Einzeltermine und wiederkehrende Termine zu unterscheiden. Bei einem Einzeltermin

```
      If veranstaltung.<Reoccur>.Value = "0" Then
```

gibt es keine Angabe des Wochentags (`Weekday`). Dafür ist dort aber ein konkretes Datum im Start- bzw. Endwertes eingetragen, das über die entsprechenden Methoden in eine Ganzzahl umgerechnet werden kann, die den Wochentag des Datums charakterisiert. Wir speichern diese Zahl in einer (später versteckten) Spalte ab, da wir sie später beim Füllen des Veranstaltungsplans auch noch verwenden werden:

```
        zeile("tag_integer") =
          start_punkt.DayOfWeek.ToString("d")
```

Wenn es sich aber um einen periodischen[13] Termin handelt

```
      Else
```

steht der Wochentag als Ganzzahl im entsprechenden Feld des Termins:

```
        zeile("tag_integer") = termin.<Weekday>.Value

      End If
```

Da wir in der Tagesspalte des Einsatzplans keine Zahlen, sondern Abkürzungen wie Mo, Di, Mi, ... sehen möchten, verwenden wir die Hilfsfunktion `wochentag_uebersetzen`, um die Ganzzahlen in Wochentagsabkürzungen zu übersetzen:

```
      zeile("Tag") = wochentag_uebersetzen(zeile("tag_integer"))

    End If
```

Bei der Eintragung der Kalenderwochen müssen wir sogar vier verschiedene Fälle unterscheiden. Wenn es sich bei dem Termin um einen Einzeltermin handelt

```
    If veranstaltung.<Reoccur>.Value = "0" Then
```

dann können wir die (einzige) Kalenderwoche direkt mit Hilfe der Funktion `woche_des_jahres` aus dem im Startzeitpunkt angegebenen Datum berechnen und in die passende Spalte eintragen:

[13] daVinci bezeichnet auch Termine, die nur in einer einzigen Woche stattfinden, als periodisch, wenn sie nicht explizit als Einzeltermine ausgewiesen wurden.

```
      zeile("Kalenderwochen") = woche_des_jahres(start_punkt)
```

Wenn es sich hingegen um einen wiederkehrenden Termin handelt, bei dem die Planerin eine Periode eingetragen hat

```
    ElseIf Not termin.<Period>.@ID = Nothing Then
```

müssen wir zuerst mit Hilfe von `periode_uebersetzen` herausfinden, welche Wochen zu der Periode gehören und diese dann mittels `wochen_uebersetzen` in lesbare Form umwandeln:

```
      zeile("Kalenderwochen") =
        wochen_uebersetzen(
        periode_uebersetzen(termin.<Period>.@ID))
```

Wenn beim Termin keine Periode eingetragen ist, könnte aber die Veranstaltung selbst eine Periode besitzen. In diesem Fall

```
    ElseIf Not veranstaltung.<Period>.@ID = Nothing Then
```

lesen wir die Kalenderwochen natürlich aus der Periode der Veranstaltung:

```
      zeile("Kalenderwochen") =
        wochen_uebersetzen(
        periode_uebersetzen(veranstaltung.<Period>.@ID))
```

Wenn es sich nicht um einen Einzeltermin handelt und weder der Termin noch die Veranstaltung eine Periode besitzen

```
    Else
```

sind die Kalenderwochen direkt bei der Veranstaltung eingetragen:

```
      zeile("Kalenderwochen") =
        wochen_uebersetzen(veranstaltung.<Weeks>.Value)

    End If

  Next

Next
```

Schließlich müssen wir nur noch die Verbindung zwischen der gerade erstellten Tabelle und dem GridView, in dem sie dargestellt werden soll, herstellen:

```
GridView_einsatzplan.DataSource = DataTable_einsatzplan

GridView_einsatzplan.DataBind()

End Sub
```

5.1.7 einsatzplan_einfaerben

Das Unterprogramm

```
Protected Sub einsatzplan_einfaerben()
```

dient dazu, die von der Planerin für einzelne Module vorgegebenen Farben auch im Einsatzplan als Hintergrundfarben der entsprechenden Zeilen anzuzeigen. Wir haben das Einfärben in ein eigenes Unterprogramm ausgelagert, da wir es nach jedem Sortieren erneut aufrufen müssen, da die Farben nicht an die Zeilen gebunden sind und sonst nach dem Sortieren verloren gehen würden.

In einer Schleife über alle Zeilen des GridViews

```
For i_zeile = 0 To GridView_einsatzplan.Rows.Count - 1
```

lesen wir die in einsatzplan_fuellen in einer unsichtbaren Spalte abgelegte Farbe aus:

```
  Dim farbe_in_abgr =
    GridView_einsatzplan.Rows(i_zeile).Cells(0).Text
```

Als nächstes überprüfen[14] wir, ob die Planerin dem Modul der Veranstaltung überhaupt explizit eine Farbe zugeordnet hat:

```
  If Not farbe_in_abgr = " " Then
```

Wenn die Veranstaltung eine eigene Farbe bekommen soll, müssen wir die Farbe, die daVinci in der Reihenfolge Alpha-Blau-Grün-Rot ausgibt, erst noch mit der Hilfsfunktion abgr2argb in die von ASP.NET verwendetet Farbdarstellung der Reihenfolge Alpha-Rot-Grün-Blau umwandeln:

```
    Dim farbe_in_argb = abgr2argb(farbe_in_abgr)
```

Jetzt können wir die Farbe als Hintergrund der aktuellen Zeile verwenden:

```
    GridView_einsatzplan.Rows(i_zeile).BackColor = farbe_in_argb
```

Wenn die Planerin eine zu dunkle Hintergrundfarbe gewählt hat, ist es sinnvoll, als Schriftfarbe weiß statt schwarz zu verwenden. Dazu berechnen wir in der Hilfsfunktion grauwert einen der Farbempfindlichkeit unserer Augen angepassten Grauwert der Farbe und überprüfen, ob dieser eine bestimmte Schwelle unterschreitet:

```
    If grauwert(farbe_in_argb) < 186 Then
```

Wenn die Farbe also zu dunkel ist, setzen wir die Schriftfarbe der aktuellen Zeile auf weiß statt schwarz:

```
      GridView_einsatzplan.Rows(i_zeile).ForeColor = Color.White
```

[14] Offensichtlich wird in eine „leere" Zelle eines GridViews automatisch ein geschütztes Leerzeichen (non-breaking space) eingetragen.

Ärgerlicherweise wird die Schriftfarbe der in den Verbands- und Raumspalten verwendeten Hyperlinks nicht automatisch mit angepasst. Wir fügen daher in den entsprechenden Spalten

```
      For i_spalte = 3 To 4
```

in den Hyperlinks explizit die Farbe Weiß als Schriftfarbe ein:

```
        GridView_einsatzplan.Rows
          (i_zeile).Cells(i_spalte).Text =
          GridView_einsatzplan.Rows
          (i_zeile).Cells(i_spalte).Text.Replace _
          ("href", "style=""color:white"" href")

      Next

    End If

  End If

Next

End Sub
```

5.1.8 veranstaltungsplan_erstellen

Nachdem wir jetzt den Einsatzplan fertig gestellt haben, tragen wir die darin enthaltenen Daten in den Veranstaltungsplan ein. Dieser muss erst einmal erzeugt und dann mit Leben gefüllt werden. Das Unterprogramm

```
Private Sub veranstaltungplan_erstellen()
```

stellt dabei das Rahmenprogramm dar, das den äußeren Rahmen des Veranstaltungsplans mit Blockzeiten und Wochentagsabkürzungen erzeugt

```
veranstaltungsplanrahmen_bauen()
```

bei Einzelwochen die Datumsangaben der einzelnen Tage der aktuellen Woche einträgt

```
tagesdatum_eintragen()
```

etwaige Sperrungen darstellt, an denen die Dozentin keine Veranstaltungen haben möchte

```
sperrungen_eintragen()
```

die Daten aus dem Einsatzplan in den Veranstaltungsplan überträgt

```
veranstaltungsplan_fuellen()
```

und abschließend identische Zellen zusammenfasst:

```
veranstaltungsplan_bereinigen()

End Sub
```

5.1.9 veranstaltungsplanrahmen_bauen

Anders als das GridView des Einsatzplanes, in dem wir ja insbesondere das Sortieren der Zeilen erlauben möchten, können wir den Veranstaltungsplan als klassische statische Tabelle ausführen. Im Unterprogramm

```
Protected Sub veranstaltungsplanrahmen_bauen()
```

erzeugen wir die Kopfzeile der Tabelle mit den Wochentagsangaben und ihre erste, linke Spalte mit den Zeitangaben der einzelnen Blöcke. Dazu leeren wir als erstes bei jedem Aufruf die gesamte Tabelle, um spätere Eintragsverdopplungen zu vermeiden:

```
Table_veranstaltungsplan.Rows.Clear()
```

Wir lesen den Tag, mit dem laut Plan die Woche beginnt[15]

```
Dim start_tag As Integer =
  davinci_xml.<Settings>.<WeekStart>.Value
```

und den Tag, mit dem sie endet

```
Dim end_tag As Integer =
  davinci_xml.<Settings>.<Weekend>.Value
```

als Ganzzahlen ein und berechnen daraus die Anzahl der Wochentage:

```
  Dim n_tage = end_tag - start_tag + 1
```

Als nächstes erzeugen wir die Kopfzeile der Tabelle

```
Dim kopfzeile As New TableHeaderRow
```

legen fest, dass ihre Einträge zentriert dargestellt werden

```
kopfzeile.HorizontalAlign = HorizontalAlign.Center
```

färben sie in blau ein

```
kopfzeile.BackColor = Color.FromArgb(194, 212, 226)
```

und fügen sie in der Tabelle ein:

```
Table_veranstaltungsplan.Rows.Add(kopfzeile)
```

In der Kopfzeile erzeugen wir die Eckzelle links oben

[15] Interessanterweise heißt der erste Knoten wie zu erwarten <WeekStart>, der zweite aber tatsächlich <Weekend> mit einem kleinen „e".

```
Dim erste_zelle As New TableCell
```

und fügen sie als leere Zelle in die Kopfzeile ein:

```
kopfzeile.Cells.Add(erste_zelle)
```

Für die Einträge der Wochentage starten wir eine Schleife vom ersten bis zum letzten Tag:

```
For i_tag As Integer = start_tag To end_tag
```

In jedem Schleifendurchlauf erzeugen wir für jeden Tag eine neue Zelle

```
  Dim tag_zelle As New TableCell
```

füllen sie mit der Abkürzung des jeweiligen Wochentag, die wir mit der Hilfsfunktion `wochentag_uebersetzen` ermitteln

```
  tag_zelle.Text = wochentag_uebersetzen(i_tag)
```

und fügen die Zelle in der Kopfzeile ein:

```
  kopfzeile.Cells.Add(tag_zelle)

Next
```

In der ersten Tabellenspalte jeder Zeile möchten wir die Blockanfangs- und -endzeiten eintragen. Dazu lesen wir die Blockdefinitionen des Standardzeitrahmens[16] ein

```
Dim zeiten =
  From zeit
  In davinci_xml.<Settings>.<TimeFrame>.<Rows>.<Items>.<Item>
  Where zeit.Parent.Parent.Parent.<Code>.Value = "Standard"
  Select
  start = zeit.<Start>.Value,
  ende = zeit.<End>.Value,
  name = zeit.<Name>.Value
```

deklarieren Zeitpunktvariablen für den Start- und Endzeitpunkt eines Blockes

```
Dim start_punkt As DateTime

Dim end_punkt As DateTime
```

und beginnen eine Schleife über alle Blöcke:

```
For i_block = 1 To zeiten.Count
```

Für jeden Block erzeugen wir eine neue, zentrierte, am oberen Rand ausgerichtete Zeile

[16] In der aktuellen plan-Version ignorieren wir anforderungsgemäß alle Zeitrahmen außer dem Standardzeitrahmen.

```
  Dim zeile As New TableRow

  zeile.HorizontalAlign = HorizontalAlign.Center

  zeile.VerticalAlign = VerticalAlign.Top
```

Für die Blockeinträge der ersten Spalte erzeugen wir eine neue blaue Zelle

```
  Dim block_zelle As New TableCell

  block_zelle.BackColor = Color.FromArgb(194, 212, 226)
```

Den Start- und Endzeitpunkt des Blockes wandeln wir in die vorher deklarierten Zeitpunktvariablen um:

```
  start_punkt = zeiten(i_block - 1).start

  end_punkt = zeiten(i_block - 1).ende
```

Den Blocknamen speichern wir zur weiteren Verarbeitung zwischen:

```
  Dim block_name = zeiten(i_block - 1).name
```

Nur, wenn die Planerin tatsächlich einen Blocknamen eingetragen hat

```
  If Not block_name = Nothing Then
```

trennen wir den Blocknamen und die Zeiten mit einem Zeilenumbruch:

```
    block_name &= "<br>"

  End If
```

Schließlich basteln wir den Text der Zelle aus dem Blocknamen und den beiden Zeiten zusammen

```
  block_zelle.Text =
    block_name &
    "<small>" &
    start_punkt.ToUniversalTime.ToString("HH:mm") &
    " - " &
    end_punkt.ToUniversalTime.ToString("HH:mm") &
    "</small>"
```

und fügen die Zelle in die aktuelle Zeile ein:

```
  zeile.Cells.Add(block_zelle)
```

Für die übrigen Spalten jeder Zeile

```
  For i_tag = 1 To n_tage
```

fügen wir leere Zellen

```
    Dim leere_zelle As New TableCell
```

ein:

```
    zeile.Cells.Add(leere_zelle)

  Next

Next

End Sub
```

5.1.10 tagesdatum_eintragen

Im Unterprogramm

```
Private Sub tagesdatum_eintragen()
```

wollen wir gegebenenfalls das tatsächliche Datum eines jede Wochentages in der Kopfzeile des Veranstaltungsrahmens eintragen, wenn dort nur die Veranstaltungen einer einzigen Woche dargestellt werden. Dazu lesen wir auch in diesem Unterprogramm die Ganzzahlen des ersten und des letzten Wochentages

```
Dim start_tag As Integer =
  davinci_xml.<Settings>.<WeekStart>.Value

Dim end_tag As Integer =
  davinci_xml.<Settings>.<Weekend>.Value
```

und berechnen daraus, aus wie vielen Tagen die Veranstaltungswoche besteht:

```
Dim n_tage = end_tag - start_tag + 1
```

Egal, mit welchem Wochentag die Woche beginnt, der erste Eintrag beginnt immer in der ersten Spalte:

```
Dim i_spalte As Integer = 1
```

Wenn alle Veranstaltungen aller Wochen dargestellt werden sollen, was beispielsweise außerhalb des Planungszeitraums der Fall ist, wurde die globale Variable `aktuelle_kalenderwoche` in `Button_wochen_Click` auf `''Alle Wochen''` gesetzt. In diesem Fall

```
If aktuelle_kalenderwoche = "Alle Wochen" Then
```

wollen wir nur die Abkürzungen der Wochentage in der Kopfzeile darstellen. Dazu verwenden wir in einer Schleife über alle Wochentage

```
  For i_tag As Integer = start_tag To end_tag
```

die Hilfsfunktion wochentag_uebersetzen, um die passenden Wochentagsabkürzungen in die Kopfzeile einzutragen:

```
    Table_veranstaltungsplan.Rows(0).Cells(i_spalte).Text =
      wochentag_uebersetzen(i_tag)

    i_spalte += 1

  Next
```

Wenn die Nutzerin hingegen eine bestimmte Woche ausgewählt hat

```
Else
```

steht in der Variablen aktuelle_kalenderwoche eine das Anfangs- und Enddatum der Woche beinhaltende Zeichenkette der Form 07.11.2016 - 11.11.2016. Die ersten zehn Zeichen dieser Zeichenkette stellen dann immer das Anfangsdatum der Woche dar:

```
  Dim erster_wochentag_datum As DateTime =
    aktuelles_wochendatum.Substring(0, 10)
```

In der Schleife über alle darzustellenden Wochentage

```
  For i_tag As Integer = start_tag To end_tag
```

berechnen wir das aktuelle Tagesdatum, indem wir zum Anfangsdatum der Woche die entsprechende Anzahl von Tagen hinzuzählen:

```
    Dim aktuelles_tagesdatum =
      erster_wochentag_datum.AddDays(i_tag - 1)
```

Dieses Datum tragen wir dann zusammen mit der Wochentagsabkürzung in die entsprechende Kopfzelle ein

```
    Table_veranstaltungsplan.Rows(0).Cells(i_spalte).Text =
      wochentag_uebersetzen(i_tag) &
      ", " &
      aktuelles_tagesdatum
```

Als letztes müssen wir innerhalb der Schleife nur noch den internen Spaltenzeiger auf die nächste Spalte setzen:

```
    i_spalte += 1

  Next

End If

End Sub
```

5.1.11 sperrungen_eintragen

Im Unterprogramm

```
Private Sub sperrungen_eintragen()
```

tragen wir die periodischen Sperrungen erster Ordnung in den Veranstaltungsplan ein, mit denen die Planerin festlegen kann, dass die Dozentin in diesem Zeitraum keine Veranstaltung durchführen kann.

Für den unwahrscheinlichen Fall, dass die anzuzeigende Woche nicht mit einem Montag beginnt, beschaffen wir uns schon einmal den ersten Wochentag:

```
Dim start_tag As Integer =
  davinci_xml.<Settings>.<WeekStart>.Value
```

Als nächstes lesen wir alle Sperrungen der Kategorie eins der aktuellen Dozentin

```
Dim sperrungen =
  From sperrung
  In davinci_xml.<Teacher>.<Items>.<Item>.
  <TimePreferences>.<Items>.<Item>
  Where sperrung.Parent.Parent.Parent.@ID = dozent_id _
  And sperrung.<Type>.Value = "1"
  Select sperrung
```

und deklarieren den Anfangs- und Endzeitpunkt der Sperrung:

```
Dim start_punkt As DateTime

Dim end_punkt As DateTime
```

In der nun folgenden Schleife über alle Sperrungen

```
For Each sperrung In sperrungen
```

definieren wir den Anfangs- und Endzeitpunkt der aktuellen Sperrung

```
  start_punkt = sperrung.<Start>.Value

  end_punkt = sperrung.<End>.Value
```

und starten eine weitere Schleife, in der wir den Zeilenindex vom Anfang bis zum Ende der Sperrung laufen lassen, indem wir die beiden Zeitpunkte von der Hilfsfunktion `zeit_uebersetzen` in die entsprechenden Ganzzahlen umwandeln lassen.

```
  For plan_zeile =
    zeit_uebersetzen
    (start_punkt.ToUniversalTime.ToString("HH:mm")) To _
    zeit_uebersetzen
    (end_punkt.ToUniversalTime.ToString("HH:mm"))
```

Da Sperrungen gegebenenfalls auch über mehrere Tage gehen können, lassen wir eine weitere Schleife über alle Tabellenspalten laufen, die wir aus dem ersten und letzten zu sperrenden Wochentags und dem ersten tatsächlich anzuzeigenden Wochentag berechnen:

```
    For plan_spalte =
      Convert.ToInt32
      (sperrung.<WeekStart>.Value) - start_tag + 1 To _
      Convert.ToInt32
      (sperrung.<Weekend>.Value) - start_tag + 1
```

Jetzt müssen wir noch abfangen, dass die Planerin – warum auch immer – zwar beispielsweise für Freitag Sperrungen eingetragen hat, die Woche aber nur bis Donnerstag laufen lässt:

```
      If plan_zeile > 0 And
        plan_zeile < Table_veranstaltungsplan.Rows.Count And
        plan_spalte > 0 And
        plan_spalte <
        Table_veranstaltungsplan.Rows(0).Cells.Count Then
```

und können dann endlich die Sperrung mit roter Farbe eintragen:

```
        Table_veranstaltungsplan.Rows(plan_zeile).
          Cells(plan_spalte).BackColor =
          Color.FromArgb(240, 192, 202)
```

Um sicherzustellen, dass Texte überlappender Sperrungen auch alle korrekt voneinander getrennt eingetragen werden, untersuchen wir, ob in der aktuellen Zelle schon ein Text eingetragen ist

```
        If Not Table_veranstaltungsplan.Rows(plan_zeile).
          Cells(plan_spalte).Text = Nothing Then
```

Wenn dies der Fall ist, tragen wir eine horizontale Trennungslinie in den Block ein, bevor wir den nächsten Text darunter schreiben:

```
          Table_veranstaltungsplan.Rows(plan_zeile).
            Cells(plan_spalte).Text &= "<hr>"

        End If
```

Wenn die Nutzerin angesagt hat, dass sie die Anfangs- und Endzeiten der Sperrung sehen möchte

```
        If CheckBox_zeiten.Checked Then
```

tragen wir diese in die aktuelle Zelle ein:

```
            Table_veranstaltungsplan.Rows(plan_zeile).
              Cells(plan_spalte).Text &=
              start_punkt.ToUniversalTime.ToString
              ("HH:mm") & " - " &
              end_punkt.ToUniversalTime.ToString
              ("HH:mm") & "<br>"

          End If
```

Und wenn die Nutzerin außerdem noch die Begründung der Sperrung darstellen lassen möchte

```
          If CheckBox_bemerkungen.Checked Then
```

erfüllen wir ihr natürlich auch diesen Wunsch:

```
            Table_veranstaltungsplan.Rows(plan_zeile).
              Cells(plan_spalte).Text &= sperrung.<Name>.Value

          End If

        End If

      Next

    Next

  Next

End Sub
```

5.1.12 veranstaltungsplan_fuellen

Im Unterprogramm

```
Private Sub veranstaltungsplan_fuellen()
```

kopieren wir die im Einsatzplan dargestellten Informationen in den Veranstaltungsplan. Dazu beschaffen wir uns prophylaktisch schon mal den Wochentag, mit dem die Woche beginnt:

```
Dim start_tag As Integer =
  davinci_xml.<Settings>.<WeekStart>.Value
```

In der jetzt folgenden Schleife gehen wir durch alle Einträge des Einsatzplanes und übertragen die relevanten Informationen in den Veranstaltungsplan:

```
For Each zeile As DataRow In DataTable_einsatzplan.Rows
```

In der Schleife untersuchen wir als erstes, ob die Planerin die Veranstaltung überhaupt schon verplant, also auf einen bestimmten Termin gelegt hat:

```
  If Not IsDBNull(zeile("Beginn")) Then
```

Wenn dies der Fall ist, tricksen wir ein bisschen: Um den Fall abzufangen, dass eine Veranstaltung (bei fehlender Pause) genau auf einem Blockanfang endet[17] und damit auch noch in diesem Block eingetragen werden würde, verkürzen wir – in der Hoffnung, dass dies nirgends sonst eine signifikante Auswirkung hat – intern jede Veranstaltung künstlich um eine Minute. Dazu beschaffen wir uns den eigentlichen Endzeitpunkt der Veranstaltung

```
    Dim eigentliches_ende As DateTime = zeile("Ende")
```

und ziehen davon eine Minute ab:

```
    Dim vorgezogenes_ende = eigentliches_ende.AddMinutes(-1)
```

Als nächstes lassen wir eine Schleife über alle Zeilen im Veranstaltungsplan laufen, in die die aktuelle Veranstaltung eingetragen werden soll:

```
    For plan_zeile =
      zeit_uebersetzen(zeile("Beginn")) To _
      zeit_uebersetzen(vorgezogenes_ende.ToString("HH:mm"))
```

Für die Berechnung der korrekten Spalte müssen wir noch berücksichtigen, dass die Woche möglicherweise nicht mit einem Montag beginnt:

```
      Dim plan_spalte = zeile("tag_integer") - start_tag + 1
```

Jetzt müssen wir noch die (unwahrscheinlichen) Fälle abfangen, in denen die Planerin Veranstaltungen für Wochentage eingetragen hat, die aber überhaupt nicht dargestellt werden sollen:

```
      If plan_zeile > 0 And
        plan_zeile < Table_veranstaltungsplan.Rows.Count And
        plan_spalte > 0 And
        plan_spalte <
        Table_veranstaltungsplan.Rows(0).Cells.Count Then
```

Eine Veranstaltung soll jetzt in drei Fällen in den Veranstaltungsplan eingetragen werden: Wenn[18] die Nutzerin das Auswahlfeld Alle Wochen angeklickt hat, wenn die Veranstaltung sowieso jede Woche stattfindet oder wenn die aktuell ausgewählte Kalenderwoche in der Liste der Wochen enthalten ist, in denen die Veranstaltung stattfindet:

[17] Die philosophische Frage hinter diesem Problem lautet: Kann ein Zeit**punkt** überhaupt zu mehreren nicht überlappenden Zeit**intervallen** gehören? Ist es also streng genommen überhaupt erlaubt, einen Block von 08:00 bis genau 09:30 gehen zu lassen und den nächsten Block dann um genau 09:30 beginnen zu lassen. Mathematisch gesehen, ist die Blockzugehörigkeit um 09:30 keine Funktion mehr, da es zum gleichen Zeitpunkt mehrere Blockwerte gibt.

[18] Wir könnten diese If-Abfrage natürlich auch mit der vorherigen kombinieren, halten eine optische Trennung hier aber für hilfreich für das Verständnis des Codes, da in dieser zweiten Abfrage ein komplett anderes Thema abgearbeitet wird.

```
        If aktuelle_kalenderwoche = "Alle Wochen" Or
          zeile("Kalenderwochen") = "Alle Wochen" Or
          zeile("Kalenderwochen").
          contains(aktuelle_kalenderwoche)
          Then
```

Wenn jetzt schon eine andere Veranstaltung in der aktuellen Veranstaltungszelle eingetragen wurde

```
          If Not Table_veranstaltungsplan.Rows(plan_zeile).
            Cells(plan_spalte).Text = Nothing Then
```

dann fügen wir eine horizontale Trennlinie vor dem aktuellen Eintrag ein:

```
            Table_veranstaltungsplan.Rows(plan_zeile).
              Cells(plan_spalte).Text &= "<hr>"

          End If
```

Wenn die Nutzerin gerne die Anfangs- und Endzeiten der Veranstaltung sehen möchte

```
          If CheckBox_zeiten.Checked Then
```

tragen wir diese als erstes ein:

```
            Table_veranstaltungsplan.Rows(plan_zeile).
              Cells(plan_spalte).Text &=
              zeile("Beginn") & " - " & zeile("Ende") & "<br>"

          End If
```

Als nächstes stellen wir den Modulnamen[19]

```
          Table_veranstaltungsplan.Rows(plan_zeile).
            Cells(plan_spalte).Text &= zeile("Modul")
```

den Semesterverband dar:

```
          Table_veranstaltungsplan.Rows(plan_zeile).
            Cells(plan_spalte).Text &= zeile("Verband")
```

Wenn die Veranstaltung nicht jede Woche stattfindet

```
          If Not zeile("Kalenderwochen") = "Alle Wochen" And
              aktuelle_kalenderwoche = "Alle Wochen" Then
```

tragen wir die Liste der Wochen, an denen die Veranstaltung stattfindet, ebenfalls ein:

[19] Zwischen dem Modulnamen und dem Semesterverband brauchen wir keinen Zeilenumbruch einzufügen, da wir den Modulnamen schon im Einsatzplan in ein gekapselt haben, um den Tooltip eintragen zu können.

```
            Table_veranstaltungsplan.Rows(plan_zeile).
              Cells(plan_spalte).Text &=
              "<br>" & "KW " & zeile("Kalenderwochen")

          End If
```

Anschließend geben wir den Raum aus:

```
          Table_veranstaltungsplan.Rows(plan_zeile).
            Cells(plan_spalte).Text &= "<br>" & zeile("Raum")
```

Wenn die Nutzerin zusätzlich noch die Bemerkung sehen möchte und diese tatsächlich von der Planerin eingetragen wurde

```
          If CheckBox_bemerkungen.Checked And
            Not IsDBNull(zeile("Bemerkung")) Then
```

stellen wir auch noch die Bemerkung in der aktuellen Zelle in einer neuen Zeile dar:

```
            Table_veranstaltungsplan.Rows(plan_zeile).
              Cells(plan_spalte).Text &=
              "<br>" & zeile("Bemerkung")

          End If
```

Jetzt müssen wir noch gegebenenfalls die Zellenfarben an die Wünsche der Planerin anpassen. Wenn die Planerin keine Farbe vorgegeben hat

```
          If IsDBNull(zeile("farbe")) Then
```

verwenden wir die Standardveranstaltungsfarbe, um die Veranstaltungen im Plan etwas hervorzuheben:

```
            Table_veranstaltungsplan.Rows(plan_zeile).
              Cells(plan_spalte).BackColor =
              Color.FromArgb(191, 227, 214)
```

Wenn die Planerin aber selbst eine Farbe vorgegeben hat

```
          Else
```

wandeln wir diese Farbe – wie in `einsatzplan_einfaerben` – in das in ASP.NET übliche Farbformat um

```
            Dim farbe_in_argb = abgr2argb(zeile("farbe"))
```

und verwenden sie als Hintergrundfarbe der aktuellen Veranstaltung:

```
            Table_veranstaltungsplan.Rows(plan_zeile).
              Cells(plan_spalte).BackColor = farbe_in_argb
```

Bei dunklen Hintergrundfarben

```
            If grauwert(farbe_in_argb) < 186 Then
```

benutzen wir außerdem weiß als Schriftfarbe; sowohl für die „normalen" Zelleinträge

```
              Table_veranstaltungsplan.Rows(plan_zeile).
                Cells(plan_spalte).ForeColor = Color.White
```

als auch für die Hyperlinks:

```
              Table_veranstaltungsplan.Rows(plan_zeile).
                Cells(plan_spalte).Text =
                Table_veranstaltungsplan.Rows(plan_zeile).
                Cells(plan_spalte).Text.Replace _
                ("href", "style=""color:white"" href")

            End If

          End If

        End If

      End If

    Next

  End If

Next

End Sub
```

5.1.13 veranstaltungsplan_bereinigen

Im Unterprogramm

```
Private Sub veranstaltungsplan_bereinigen()
```

wollen wir Veranstaltungen, die sich über mehrere Blöcke erstrecken und die daher in veranstaltungsplan_fuellen in mehrere Einzelzellen der Veranstaltungstabelle eingetragen wurden, zu einem größeren Block zusammenfassen, der sich über mehrere Tabellenzellen sprich Tabellenzeilen ausdehnt.

Als erstes beschaffen wir uns die Anzahl der Zeilen

```
Dim n_zeilen = Table_veranstaltungsplan.Rows.Count
```

und der Spalten der Veranstaltungstabelle:

```
Dim n_spalten = Table_veranstaltungsplan.Rows(0).Cells.Count
```

Dann deklarieren wir eine Liste, in die wir gleich die zusammenzufassenden Tabellenzellen eintragen[20] werden:

```
Dim zu_loeschen As New List(Of Integer())
```

Die äußere Schleife läuft über alle zu untersuchenden Spalten

```
For i_spalte = 1 To n_spalten - 1
```

während die innere Schleife eine Zeile weniger betrachten muss, da in der Schleife gleich zwei Zeilen miteinander verglichen werden:

```
  For i_zeile = 1 To n_zeilen - 2
```

In der Schleife untersuchen wir als Bedingung für das Zusammenfassen, ob in zwei übereinander liegenden Zellen der gleiche Text steht. Zusätzlich müssen wir noch sicher stellen, dass in den Zellen überhaupt etwas steht (sonst würden auch leere Zellen zusammengefasst) oder dass es sich um gesperrte Blöcke handelt (diese wollen wir auch dann zusammenfassen, wenn sie keinen Text beinhalten):

```
    If (Not Table_veranstaltungsplan.Rows(i_zeile).
      Cells(i_spalte).Text = Nothing Or
      Table_veranstaltungsplan.Rows(i_zeile).
      Cells(i_spalte).BackColor =
      Color.FromArgb(240, 192, 202) And
      Table_veranstaltungsplan.Rows(i_zeile + 1).
      Cells(i_spalte).BackColor =
      Color.FromArgb(240, 192, 202) And
      Table_veranstaltungsplan.Rows(i_zeile).
      Cells(i_spalte).Text =
      Table_veranstaltungsplan.Rows(i_zeile + 1).
      Cells(i_spalte).Text Then
```

Wenn die aktuelle Zelle also mit der unter ihr liegenden Zelle zusammengefasst werden soll, fügen wir ihren Zeilen- und Spaltenindex als Zeilenvektor zur Löschliste hinzu:

```
      zu_loeschen.Add({i_zeile, i_spalte})

    End If

  Next

Next
```

[20] Der zweistufige Ansatz, zuerst die zu löschenden Tabellenzellen zu identifizieren und im zweiten Schritt erst alle „in einem Rutsch" zu löschen, ist wesentlich übersichtlicher, debugfreundlicher und sicherer als zu versuchen, während des Durchsuchens der Tabelle schon mal einzelne Zellen zu löschen, da man dabei dann sehr gewissenhaft auf die Zeilenzeigerintegrität achten müsste: Durch das vertikale Zusammenfassen zweier Zellen bekommt die darunter liegende Zelle ja einen kleineren Zeilenindex.

In einer zweiten Schleife laufen wir rückwärts[21] durch die Liste der zu löschenden Zellen:

```
For i_loeschen = zu_loeschen.Count - 1 To 0 Step -1
```

Wir beschaffen uns den Zeilen-

```
  Dim i_zeile = zu_loeschen(i_loeschen)(0)
```

und Spaltenindex der aktuellen Zelle

```
  Dim i_spalte = zu_loeschen(i_loeschen)(1)
```

und untersuchen, ob es sich bei der nachfolgenden Zelle schon[22] um eine „zusammengefasste" Zelle handelt:

```
  If Table_veranstaltungsplan.Rows(i_zeile + 1).
    Cells(i_spalte).RowSpan > 1 Then
```

Wenn dies der Fall ist, setzen wir den „Zeilenausdehnungskoeffizienten" der aktuellen auf den um eins erhöhten Koeffizienten der nachfolgenden Zelle:

```
    Table_veranstaltungsplan.Rows(i_zeile).
      Cells(i_spalte).RowSpan =
      Table_veranstaltungsplan.Rows(i_zeile + 1).
      Cells(i_spalte).RowSpan + 1
```

Wenn die nachfolgende Zelle hingegen noch keine zusammengefasste Zelle ist

```
  Else
```

setzen wir den Ausdehnungskoeffizienten der aktuellen Zelle auf zwei:

```
    Table_veranstaltungsplan.Rows(i_zeile).
      Cells(i_spalte).RowSpan = 2

  End If
```

In beiden Fällen löschen wir abschließend die nachfolgende Zelle:

```
  Table_veranstaltungsplan.Rows(i_zeile + 1).
    Cells.RemoveAt(i_spalte)

Next

End Sub
```

[21] Durch das Abarbeiten der Zeilen von unten nach oben umgehen wir das Problem, dass sich der Zeilenindex der nachfolgenden Zeilen beim Zusammenfassen ändert.

[22] Dieser Fall tritt immer dann auf, wenn eine Veranstaltung über mehr als zwei Blöcke verläuft.

5.1.14 woche_des_jahres

In der Hilfsfunktion

```
Function woche_des_jahres(ByVal datum As DateTime) As String
```

ermitteln wir die Kalenderwoche eines bestimmten Datums. Dazu rufen wir als erstes Informationen über das lokal gültige aktuelle Zeitformat ab

```
Dim zeitformatinfo As DateTimeFormatInfo =
  DateTimeFormatInfo.CurrentInfo
```

und erzeugen auf dessen Basis einen lokal gültigen Kalender:

```
Dim kalender As Calendar = zeitformatinfo.Calendar
```

In diesem Kalender lesen wir nun die Kalenderwoche des übergebenen Datums ab:

```
Return kalender.GetWeekOfYear(
  datum,
  zeitformatinfo.CalendarWeekRule,
  zeitformatinfo.FirstDayOfWeek).ToString("00")

End Function
```

Dabei geben wir die Kalenderwoche im einstelligen Fall grundsätzlich mit führender Null aus, damit beispielsweise die zweite Kalenderwoche nicht auch in der Zeichenkette „41, 42, 43" gefunden wird.

5.1.15 wochen_uebersetzen

daVinci stellt die Wochen, in denen eine Veranstaltung stattfindet, als eine Zeichenkette (`101001000`) bestehend aus Nullen und Einsen dar. Eine Eins bedeutet dabei, dass in der entsprechenden Woche eine Veranstaltung stattfindet. In der Hilfsfunktion

```
Function wochen_uebersetzen(ByVal wochen As String) As String
```

wandeln wir dies Zeichenkette in eine leichter interpretierbare Form (`42, 44, 47`) um. Dazu beschaffen wir uns als erstes das Datum, ab dem der Veranstaltungsplan gilt

```
Dim startdatum As DateTime =
  davinci_xml.<Settings>.<TimetablePeriodFrom>.Value
```

und ermitteln die Gesamtanzahl der Wochen aus der Anzahl der Ziffern der übergebenen Zeichenkette:

```
Dim n_wochen = wochen.Count
```

Wir initialisieren die Zeichenkette, die abschließend die Liste der Veranstaltungskalenderwochen enthält

```
Dim kalenderwochen As String = Nothing
```

und beginnen eine nullbasierte Schleife, die durch alle Zeichen der übergebenen Zeichenkette läuft:

```
For i_woche As Integer = 0 To n_wochen - 1
```

Wenn die Zeichenkette an der aktuellen Stelle eine Eins besitzt

```
  If wochen.Chars(i_woche) = "1" Then
```

dann addieren wir zum Datum, ab dem der Veranstaltungsplan gilt, die entsprechenden Wochenzahlen

```
    Dim aktuelles_datum = startdatum.AddDays(7 * i_woche)
```

und ermitteln mit der Hilfsfunktion `woche_des_jahres` die Kalenderwoche dieses neuen Datums:

```
    Dim aktuelle_woche = woche_des_jahres(aktuelles_datum)
```

Wenn es sich nicht gerade um den ersten Eintrag einer Kalenderwoche handelt

```
    If Not IsNothing(kalenderwochen) Then
```

dann fügen wir in die Liste der Veranstaltungskalenderwochen als erstes ein Komma und ein Leerzeichen ein:

```
      kalenderwochen &= ", "

    End If
```

Abschließend hängen wir die gefundene Kalenderwoche an die Liste an:

```
    kalenderwochen &= aktuelle_woche

  End If

Next
```

Wenn die Veranstaltung in allen Semesterwochen stattfinden soll, übergibt daVinci eine Zeichenkette, die aus lauter Nullen (statt Einsen) besteht. In diesem Fall wäre die gerade erstellte Liste der Kalenderwochen leer. Wenn dies der Fall ist

```
If kalenderwochen = Nothing Then
```

tragen wir „Alle Wochen” in die Kalenderwochenliste ein:

```
  kalenderwochen = "Alle Wochen"

End If
```

In beiden Fällen geben wir abschließend die Liste der Kalenderwochen zurück:

```
Return kalenderwochen

End Function
```

5.1.16 faktor_uebersetzen

Die Hilfsfunktion

```
Function faktor_uebersetzen(ByVal faktor As String) As Double
```

fängt zwei Probleme bei der Umwandlung des Anrechnungsfaktors von einer Zeichenkette in eine Fließkommazahl ab. Erstens trägt daVinci gar keinen Faktor ein, wenn dieser den Wert eins besitzt. Wir fragen daher als erstes ab, ob die Zeichenkette leer ist

```
If IsNothing(faktor) Then
```

und setzen den Wert in diesem Fall explizit auf 1:

```
  Return 1
```

Zweitens

```
Else
```

verwendet daVinci für Fließkommazahlen den Dezimalpunkt. Visual Basic möchte in der deutschen Version Fließkommazahlen aber gerne mit einem Dezimalkomma sehen. Wir ersetzen daher den Punkt im Faktor durch ein Komma:

```
  faktor = faktor.Replace(".", ",")
```

und geben den bereinigten Faktor zurück:

```
  Return faktor

End If

End Function
```

5.1.17 periode_uebersetzen

In der Hilfsfunktion

```
Function periode_uebersetzen(ByVal periode As String) As String
```

ermitteln wir die zur übergebenen Perioden-ID gehörenden Wochen

```
Dim wochen =
  From wwochen In davinci_xml.<Periods>.<Items>.<Item>.<Weeks>
  Where wwochen.Parent.@ID = periode
```

und geben die entsprechende Zeichenkette der Form 101001000 zurück:

```
Return wochen.FirstOrDefault

End Function
```

5.1.18 wochentag_uebersetzen

In der Hilfsfunktion

```
Function wochentag_uebersetzen(ByVal wochentag As String)
  As String
```

ersetzen wir die den Wochentag repräsentierende von daVinci gelieferte Ganzzahl durch die Abkürzung des Wochentages. Dazu unterscheiden wir sieben verschiedenen Fälle

```
Select Case wochentag
```

Da die Wochentage üblicherweise nur durch die Zahlen von 1 bis 7 repräsentiert werden, genügen normalerweise genau diese Fälle:

```
  Case "1"

    Return "      Mo"

  Case "2"

    Return "     Di"

  Case "3"

    Return "    Mi"

  Case "4"

    Return "   Do"

  Case "5"

    Return "  Fr"

  Case "6"

    Return " Sa"

  Case "7"

    Return "So"
```

Die unterschiedlich vielen Leerzeichen vor den Abkürzungen bewirken, dass beim Sortieren des Einsatzplans nach den Wochentagen „Mo" tatsächlich vor „Di" erscheint, was ja nach der alphabetischen Ordnung nicht der Fall wäre. Sechs Leerzeichen werden aber alphabetisch vor fünf Leerzeichen einsortiert. Freundlicherweise werden die führenden Leerzeichen im GridView zwar beim Sortieren berücksichtigt, auf der Webseite aber nicht angezeigt.

Wenn – was normalerweise nicht geschehen sollte – der Funktion eine Zahl außerhalb des erlaubten Ganzzahlintervalls von 1 bis 7 übergeben wird, gibt sie die Zeichenkette unverändert wieder zurück:

```
  Case Else

    Return wochentag

End Select

End Function
```

5.1.19 zeit_uebersetzen

Die Hilfsfunktion

```
Function zeit_uebersetzen(ByVal zeit_punkt As String) As Integer
```

dient dazu, herauszufinden, in welchem Block des Veranstaltungsplans ein bestimmter Zeitpunkt liegt.

Wir beschaffen uns dazu als erstes die im Veranstaltungsplan verwendeten Blockgrenzen des Standardzeitrahmens

```
Dim zeiten =
  From zeit
  In davinci_xml.<Settings>.<TimeFrame>.<Rows>.<Items>.<Item>
  Where zeit.Parent.Parent.Parent.<Code>.Value = "Standard"
  Select
  start = zeit.<Start>.Value,
  ende = zeit.<End>.Value
```

starten dann eine Schleife über „alle" Blöcke eines Tages, beginnend mit dem zweiten Block:

```
For i_block = 1 To zeiten.Count - 1
```

Wir lesen den Zeitpunkt des Beginn des Blockes

```
  Dim block_anfang As DateTime = zeiten(i_block).start
```

und untersuchen, ob der fragliche Zeitpunkt schon kleiner als der aktuelle Blockbeginn ist:

```
  If zeit_punkt <
    block_anfang.ToUniversalTime.ToString("HH:mm") Then
```

Wenn dies der Fall ist, haben wir den Block hinter dem gesuchten Block gefunden und können den (nullbasierten) Blockindex (des Folgeblockes) als (einsbasierten) Index des gesuchten Blockes zurückgeben:

```
    Return i_block

  End If

Next
```

Wenn wir aber alle Blöcke untersucht haben und keinen Blockanfang gefunden haben, der größer als der gesuchte Zeitpunkt ist, muss der Zeitpunkt wohl[23] im letzten Block liegen, dessen Index wir dann folgerichtig zurückgeben:

```
Return zeiten.Count

End Function
```

5.1.20 grauwert

Die Hilfsfunktion

```
    Function grauwert(ByVal farbe As Color) As Integer
```

berechnet den Grauwert einer Farbe. Da wir Menschen evolutionsbedingt grün heller wahrnehmen als rot und blau, ergibt sich der Grauwert aus empirisch[24] ermittelten Faktoren, die mit den einzelnen Farbwerten multipliziert werden:

```
Return farbe.R * 0.299 + farbe.G * 0.587 + farbe.B * 0.114

End Function
```

5.1.21 abgr2argb

In der Hilfsfunktion

```
Function abgr2argb(ByVal farbe_in_abgr As Integer) As Color
```

[23] Zeitpunkte, die später hinter dem letzten Blockende liegen – was eigentlich ja nie vorkommen sollte – werden mit dieser Methode automatisch in den letzten Block gelegt. Ebenso landen Zeitpunkte vor dem Beginn des ersten Blockes automatisch im ersten Block.

[24] Wie bei Vektornormen so üblich, gibt es noch mehrere andere Möglichkeit, die „Helligkeit" einer Farbe zu berechnen. Und wie immer ist die Entscheidung darüber, welche Norm denn im jeweiligen Anwendungsfall nun „besser" geeignet ist, ziemlich willkürlich.

wandeln wir eine Farbe aus der in daVinci verwendeten Farbkodierung (Reihenfolge Alpha-Blau-Grün-Rot) in die in ASP.NET verwendeten Kodierung (Reihenfolge Alpha-Rot-Grün-Blau) um. Dazu extrahieren wir aus der ABGR-Farbe ihren an der letzten Stelle[25] stehenden Rotanteil

```
Dim rot = Color.FromArgb(farbe_in_abgr).B
```

den in der dritten Stelle (in beiden Kodierungen) stehenden Grünanteil

```
Dim gruen = Color.FromArgb(farbe_in_abgr).G
```

und den an zweiter Stelle stehenden Blauanteil:

```
Dim blau = Color.FromArgb(farbe_in_abgr).R
```

Jetzt können wir die Farbe in ARGB-Kodierung aus ihren Grundfarben wieder zusammensetzen:

```
Return Color.FromArgb(rot, gruen, blau)

End Function
```

5.1.22 Button_wochen_Click

Das Unterprogramm

```
Private Sub Button_wochen_Click(
  sender As Object,
  e As EventArgs)
  Handles Button_alle_wochen.Click
```

wird vom Laufzeitsystem immer dann aufgerufen, wenn die Nutzerin auf die Schaltfläche Alle Wochen oder auf eine der anderen Wochenauswahlschaltflächen klickt. In diesem Fall laufen wir durch alle Elemente des Wochenpanels

```
For Each panel_control In Panel_wochen.Controls
```

überprüfen, ob es sich bei dem Element tatsächlich um eine Schaltfläche handelt

```
  If TypeOf panel_control Is Button Then
```

und entfernen die Hintergrundfarbe der jeweiligen Schaltfläche

```
    panel_control.BackColor = Color.Empty

  End If

Next
```

[25] Das ist jetzt etwas kompliziert: In der in ASP.NET verwendeten ARGB-Kodierung steht an der letzten Stelle der Blauanteil. Deshalb erhalten wir bei der Extraktion des Blauanteils in ARGB-Interpretation den Rotanteil einer in ABGR-Kodierung angegebenen Farbe.

Wir füllen die aktuelle Kalenderwoche (die wir aus dem Text der angeklickten Schaltfläche entnehmen) in die eine

```
aktuelle_kalenderwoche = sender.text
```

und den Datumsbereich der aktuellen Kalenderwoche (aus dem Tooltip der angeklickten Schaltfläche) in die andere globale Variable:

```
aktuelles_wochendatum = sender.Tooltip
```

Die angeklickte Schaltfläche hinterlegen wir zur Kenntlichmachung der aktuellen Kalenderwoche mit blau:

```
sender.BackColor = Color.FromArgb(194, 212, 226)
```

Abschließend müssen wir natürlich den Veranstaltungsplan (einschließlich der Kopfzeile, die sich ja möglicherweise auch geändert hat) neu aufbauen:

```
veranstaltungsplan_erstellen()

End Sub
```

5.1.23 Button_zurueck_Click

Um nach mehrfachen Plandarstellungsänderungen schnell wieder zur Auswahlseite zurückkehren zu können, gibt es die Schaltfläche Zurück zur Auswahlseite. Wenn die Nutzerin diese anklickt, wird das Unterprogramm

```
Protected Sub Button_zurueck_Click(
  sender As Object,
  e As EventArgs)
  Handles Button_zurueck.Click
```

aufgerufen. Dort führen wir den Rücksprung zur Auswahlseite unter Verwendung des aktuellen Semesters und Teams durch:

```
  Response.Redirect("https://m-server.fk5.hs-bremen.de/plan/
    auswahl.aspx?semester=" &
    Request.QueryString("semester") &
    "&team=" &
    Request.QueryString("team"))

End Sub
```

5.1.24 GridView_einsatzplan_RowDataBound

Das Unterprogramm

```
Private Sub GridView_einsatzplan_RowDataBound(
  sender As Object,
  e As GridViewRowEventArgs)
  Handles GridView_einsatzplan.RowDataBound
```

gibt uns die Möglichkeit, einzustellen, dass in den vier Spalten Modul, Verband, Raum und Bemerkung verwendeter HTML-Code auch tatsächlich als solcher interpretiert wird. Üblicherweise ist das in einem GridView nicht der Fall.

Als erstes müssen wir sicherstellen, dass wir nur den Datenbereich (und nicht beispielsweise die Kopfzeile) des GridViews bearbeiten

```
If e.Row.RowType = DataControlRowType.DataRow Then
```

Wenn dies der Fall ist, starten wir eine Schleife über die ersten vier sichtbaren Spalten

```
  For i_spalte = 2 To 5
```

dekodieren den in der aktuellen Zelle angegebenen Text

```
    Dim dekodierter_text =
      HttpUtility.HtmlDecode(e.Row.Cells(i_spalte).Text)
```

und speichern ihn als ausführbares HTML wieder in die Zelle zurück:

```
    e.Row.Cells(i_spalte).Text = dekodierter_text

  Next

End If

End Sub
```

5.1.25 GridView_einsatzplan_RowCreated

Das Unterprogramm

```
Protected Sub GridView_einsatzplan_RowCreated(
  sender As Object,
  e As GridViewRowEventArgs)
  Handles GridView_einsatzplan.RowCreated
```

gibt uns die Möglichkeit, die Hilfsspalten für die Farbe (Spalte 0) und den Ganzzahlwert des Tages (Spalte 1) im Einsatzplan zu verstecken. Dies ist tatsächlich nur in diesem vom Laufzeitsystem beim Erstellen der einzelnen Zeilen des GridViews aufgerufenen Unterprogramm möglich. Nachdem das GridView „fertig" ist, kann man die einzelnen Spalten nicht mehr unsichtbar machen. Innerhalb dieses Unterprogrammes können wir aber direkt die ersten beiden Spalte verstecken:

```
e.Row.Cells(0).Visible = False

e.Row.Cells(1).Visible = False

End Sub
```

5.1.26 LinkButton_einsatzplan_Click

Das Unterprogramm

```
Protected Sub LinkButton_einsatzplan_Click(
  ByVal sender As Object,
  ByVal e As System.EventArgs)
  Handles LinkButton_einsatzplan.Click
```

wird immer dann vom Laufzeitsystem aufgerufen, wenn die Nutzerin die Schaltfläche Einsatzplan ausblenden, um Veranstaltungsplan zu drucken anklickt. Wie der Text der Schaltfläche vermuten lässt, untersuchen wir, ob der Einsatzplan momentan sichtbar ist:

```
If Panel_einsatzplan.Visible = True Then
```

Wenn dies der Fall ist, verstecken wir das Panel, in dem der Einsatzplan liegt, um der Nutzerin die Möglichkeit zu geben, nur den Veranstaltungsplan auszudrucken

```
  Panel_einsatzplan.Visible = False
```

und passen den Aufdruck der Schaltfläche an:

```
  LinkButton_einsatzplan.Text = "Einsatzplan darstellen"
```

Wenn das Einsatzplanpanel hingegen gerade nicht sichtbar ist

```
Else
```

holen wir es wieder hervor und setzen den Schaltflächenaufdruck zurück:

```
  Panel_einsatzplan.Visible = True

  LinkButton_einsatzplan.Text =
    "Einsatzplan ausblenden, um Veranstaltungsplan zu drucken"

End If

End Sub
```

5.1.27 GridView_einsatzplan_Sorting

Das Unterprogramm

```
Protected Sub GridView_einsatzplan_Sorting(
  ByVal sender As Object,
  ByVal e As GridViewSortEventArgs)
  Handles GridView_einsatzplan.Sorting
```

wird immer dann vom Laufzeitsystem aufgerufen, wenn die Nutzerin im Einsatzplan auf einen der Einträge in seiner Kopfzeile geklickt hat, um ein Sortieren nach der jeweiligen Spalte anzustoßen. In diesem Fall verwenden wir die übergebene Spalteninformation, um die hinter dem Einsatzplan liegende Datentabelle entsprechend sortieren zu lassen:

```
DataTable_einsatzplan.DefaultView.Sort = e.SortExpression
```

Die neu sortierte Tabelle müssen wir jetzt nochmals an den Einsatzplan binden, um sie in ihrer neuen Form auch tatsächlich darzustellen:

```
GridView_einsatzplan.DataBind()
```

Außerdem müssen wir die Farben der einzelnen Zeilen des Einsatzplan neu zuweisen, da diese beim Sortieren bzw. Binden verloren gehen:

```
einsatzplan_einfaerben()

End Sub
```

6 verband.aspx und raum.aspx

Die Seiten `verband.aspx` und `raum.aspx`, die die Pläne eines Semesterverbandes bzw. eines Raumes anzeigen, sind praktisch identisch mit der Seite `dozent.aspx` (Kapitel 5), nur dass natürlich statt des auf der Dozentinnenseite angezeigten Semesterverbandes und des Raumes auf der Semesterverbandsseite die Dozentin und der Raum und auf der Raumseite der Semesterverband und die Dozentin der jeweiligen Veranstaltung angezeigt werden.

Außerdem entfällt auf der Semesterverbands- und der Raumseite die Anzeige des Anrechnungsfaktors im Einsatzplan, da die Lehrdeputatsabrechnung ja für Semesterverbände und Räume nicht durchgeführt werden muss.

7 plan.xml

plan liest fakultätsspezifische Daten aus der XML-Datei `plan.xml` (Abbildung 7.1).

```
<?xml version="1.0" encoding="utf-8" ?>
<plan>
  <team>
    <name>Fakultät 1</name>
    <code>1</code>
    <faktor_ausblenden>false</faktor_ausblenden>
    <semester>
      <name>Wintersemester 2016/2017</name>
      <code>ws16</code>
      <datei>DAV-E2603989 ... AB421089226F</datei>
    </semester>
    <semester>
      <name>Sommersemester 2016</name>
      <code>ss16</code>
      <datei>DAV-D6034CC3 ... 1E542D17C9AF</datei>
    </semester>
    <!--<semester>
      <name>Wintersemester 2015/2016</name>
      <code>ws15</code>
      <datei>DAV-391CEAA9 ... E16C8029CF5E</datei>
    </semester>-->
  </team>
  <team>
    <name>Fakultät 2 Architektur</name>
    <code>2A</code>
:
:
```

Abbildung 7.1: `plan.xml`

Darin hat jede Fakultät (bzw. Abteilung) einen eigenen Knoten <`team`>, unter dem der Langname <`name`> der Fakultät, ihre eindeutige Abkürzung <`code`> und die Information, ob der Anrechnungsfaktor angezeigt werden soll <`faktor_ausblenden`> aufgelistet sind.

Des Weiteren sind für jede Fakultät die Semester <`semester`> aufgeführt, die auf der in Abbildung 2.2 dargestellten Seite zur Auswahl angezeigt werden. Jedes Semester besitzt einen Namen <`name`>, der beispielsweise in den Unterüberschriften in Abbildung 2.4 verwendet wird, eine eindeutige Abkürzung <`code`>, die wir zur Identifizierung des Semesters – beispielsweise im QueryString – benutzen und natürlich den Namen <`datei`> der XML-Datei, in der der Plan der jeweiligen Fakultäts-Semester-Kombination abgelegt ist.

Wenn ein Plan (noch) nicht angezeigt werden soll, können wir den entsprechenden <`semester`>-Knoten einfach auskommentieren.

8 plan.css

In einer CSS-Datei beschreiben wir die Formatierung von HTML-Elementen; wir bestimmen hier also, wie unsere Seite aussieht.

Die im Folgenden definierten Farben entsprechen dem Corporate Design der Hochschule Bremen:

```
.rot {
    background-color: #F0C0CA;
}

.gruen {
    background-color: #BFE3D6;
}

.blau {
    background-color: #C2D4E2;
}
```

Grundsätzlich verwenden wir die Google-Schrift Roboto:

```
body {
  font-family: Roboto, Calibri, Arial, sans-serif;
  margin-left: 2em;
  margin-top: -1em;
}
```

Überschriften setzen wir in einer dunkelblauen Farbe des Hochschuldesigns mit ein bisschen mehr Abstand nach oben als nach unten:

```
h1, h2, h3, h4, h5 {
  color: #0A558C;
  margin-top: 1.5em;
  margin-bottom: 0.5em;
}
```

Den in plan verwendeten Überschriften dritter

```
h3 {
  font-size: 1.58em;
  font-weight: 500;
}
```

und vierter Ordnung weisen wir spezifische Schrifteigenschaften zu:

```
h4 {
  font-size: 1.33em;
  font-weight: 500;
}
```

Außerdem polstern wir den Tabellenzelleninhalt etwas:

```
td {
  padding: 0.2em 0.3em;

}
```

Hyperlinks wollen wir nicht so stark wie üblich hervorheben. Wir setzen sie also wie den Fließtext in schwarz:

```
a {
  color: black;
}
```

Außerdem verhindern wir bei allen Hyperlinks das übliche Unterstreichen:

```
  a:link {
    text-decoration: none;
  }
```

Nur wenn sich der Mauszeiger über dem Hyperlink befindet, unterstreichen wir diesen:

```
  a:hover {
    text-decoration: underline;
  }
```

Schließlich weisen wir kleiner Schrift eine bestimmte Größe zu

```
small {
  font-size: .7em;
}
```

und definieren die Klasse `rechts_oben` für die Positionierung des Logos:

```
.rechts_oben {
  position: absolute;
  top: 0px;
  right: 0px;
}
```

Literaturverzeichnis

[1] STÜBER SYSTEMS GmbH. (2016, November) da Vinci, Professionelle Stunden-, Vertretungs- und Kursplanung. [Online]. Available: http://davinci.stueber.de/

[2] Wikipedia. (2016, November) Hypertext Markup Language. [Online]. Available: https://de.wikipedia.org/wiki/Hypertext_Markup_Language

[3] ——. (2016, November) ASP.NET. [Online]. Available: https://de.wikipedia.org/wiki/ASP.NET

[4] Google. (2016, November) Roboto - Google Fonts. [Online]. Available: https://fonts.google.com/specimen/Roboto