

# Hochschule Darmstadt

## Fachbereich Informatik

### Entwicklung webbasierter Anwendungen

Praktikumsaufgaben



## Semesterthema Webbasierter Pizzaservice bzw. Onlineshop

- Im Lauf des Semesters soll eine integrierte webbasierte Anwendung zur Unterstützung eines Pizzaservices oder eines Webshops eigener Wahl entwickelt werden
  - ⇒ nicht die Funktionalität steht im Vordergrund, sondern die Integration der verschiedenen Techniken und die Methodik der Vorgehensweise
    - Clientseitig
      - Responsive Webseiten mit HTML und CSS, Formularen und JavaScript
    - Serverseitig
      - Apache Webserver
      - dynamische Seiten mit PHP, Anbindung einer MySQL-Datenbank
- Schwerpunkt auf professionelle Webentwicklung
  - ⇒ mit Standardkonformität, Barrierefreiheit, Dokumentation, Test etc.
  - ⇒ keine Homepage-Bastelei, keine Verwendung von "Fertigteilen" !



Reine Funktionalität reicht NICHT für die Abnahme !



# Semesterthema Webbasierter Pizzaservice bzw. Onlineshop

## ■ Anforderungen des Auftraggebers

- ⇒ Der Pizzaservice/Onlineshop soll folgende Webseiten enthalten (gilt entsprechend auch für einen eigenem Webshop):
  - Bestellung              (→ OK)
  - Bestellstatus            (→ OK)
  - Pizzabäcker            (→ Rechnungs- /Versandabteilung etc.)
  - Fahrer                  (→ OK)
- ⇒ Der Pizzaservice soll mit HTML5 und CSS Level 3 dargestellt werden
- ⇒ Als Webserver wird Apache 2 verwendet und als Datenbank MySQL
- ⇒ Die Abnahme erfolgt auf Ihrem Laptops mit den gängigen Browsern
  - installieren Sie dazu ggf. entspr. Web Development Add-ons (bspw. die "Web Developer Toolbar" in Firefox)
- ⇒ Für die PHP-Entwicklung müssen **Seiten-Templates** verwendet werden, die vorgegeben sind
  - Abrufbar über EWA-Moodlekurs

## Praktikum zu "Entwicklung webbasierter Anwendungen"

# Designskizzen "Webbasierter Pizzaservice"

**Kunde (Bestellung)**

	Margherita	4,00 €	<b>Warenkorb</b> Margherita Tonno Prosciutto  <b>14,50 €</b> <b>Meier, Hauptstr. 5</b>
	Salami	4,50 €	
	Prosciutto	5,50 €	
	Tonno	5,00 €	

**Kunde (Lieferstand)**

	bestellt	im Ofen	fertig	unterwegs
Margherita	●	○	○	○
Tonno	○	●	○	○
Prosciutto	○	●	○	○

	bestellt	im Ofen	fertig
Margherita	●	○	○
Tonno	○	●	○
Prosciutto	○	●	○
Salami	○	○	●
Prosciutto	○	○	●

<b>Fahrer (fertige Pizzen)</b>			
<b>Schulz, Kasinostr. 5</b>	<b>13,50 €</b>		
<b>Margherita, Salami, Tonno</b>			
fertig	unterwegs	geliefert	
○	●	○	
<b>Müller, Rheinstr. 11</b>	<b>10,00 €</b>		
<b>Salami, Prosciutto</b>			
fertig	unterwegs	geliefert	
●	○	○	

# Anforderungen des Auftraggebers "Webbasierter Pizzaservice"

## ■ Pizzabestellung

- Hier kann der Kunde seine Pizzen aus der Speisekarte auswählen und in einen Warenkorb übernehmen. Hier wird der Preis der Bestellung angezeigt und es muss eine Lieferadresse angegeben werden.

## ■ Bestellstatus

- Hier kann ein Kunde sehen, in welchem Zustand seine Pizzen sind (bestellt, im Ofen, fertig, unterwegs). Er sieht nur seine Bestellung – und keine Aufträge von anderen Kunden.

## ■ Pizzabäcker

- Hier werden die bestellten Pizzen angezeigt. Der Pizzabäcker kann den Status für jede Pizza von "bestellt" auf "im Ofen" bzw. "fertig" setzen. Übernimmt der Fahrer eine Pizza, so verschwindet sie aus der Liste.

## ■ Fahrer

- Hier werden Bestellungen mit den einzelnen Pizzen, Preis und Adresse angezeigt. Der Fahrer kann den Status der Lieferungen verändern. Eine Lieferung ist entweder "fertig", "unterwegs" oder "geliefert". Lieferungen tauchen erst beim Fahrer auf, wenn alle zugehörigen Pizzen fertig sind. Ausgelieferte Bestellungen verschwinden aus der Liste.

# Anforderungen des Auftraggebers "Webbasierter Pizzaservice"

## ■ Sonstiges

- ⇒ Die Speisekarte kann um weitere Pizzen erweitert werden. Die Preisberechnungen ändern sich dann (ohne Neuprogrammierung)
- ⇒ es werden nur gültige Bestellungen akzeptiert
- ⇒ die Auswahl einer Pizza soll über einen Mausklick auf ein Pizzasymbol erfolgen
- ⇒ Der Warenkorb bietet die allgemein üblichen Funktionen
- ⇒ Die Bestellseite passt ihre Darstellung für schmale Handybildschirme an und verwendet ein responsives Layout
- ⇒ Versuchen Sie das Layout so umzusetzen, wie es in der Designskizze dargestellt ist. Es sollen 4 getrennte Seiten entwickelt werden
- ⇒ Die Seiten "Pizzabäcker" und "Fahrer" sollen sich automatisch aktualisieren

## ■ Interne Anforderungen

- ⇒ Es gibt zu Testzwecken eine weitere Webseite "Übersicht", welche (für einfache Tests) Links zu den 4 Webseiten beinhaltet
- ⇒ Tests sollen so klar formuliert sein, dass sie automatisiert laufen könnten

## Vorbereitung: Zielsetzung

- Stellen Sie sicher, dass Sie die Aufgabe "Pizzaservice" genau verstanden haben und klären Sie offene Punkte frühzeitig
- Aufgabe: Anforderungsanalyse
  - ⇒ Analysieren Sie sämtliche Vorgaben (sowohl vom Auftraggeber als auch interne Vorgaben)
  - ⇒ Identifizieren Sie Inkonsistenzen oder offene Punkte und klären Sie die Fragen mit Ihrem Betreuer

# 1. Übung: Zielsetzung

## Bäcker

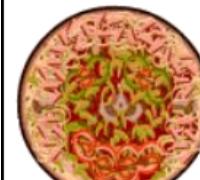
bestellt im Ofen fertig

- Margherita
- Margherita
- Hawai

- [Bestellung](#)
- [Kunde](#)
- [Bäcker](#)
- [Fahrer](#)

Übersichtsseite

## Bestellung



Margherita 4,00 €



Salami 4,50 €



Hawaii 5,50 €

test

15,70 €

[Alle Löschen](#)

[Auswahl Löschen](#)

[Bestellen](#)

## Fahrer

Müller, Freßgasse 11, 65000 Frankfurt

Tonno, Calzone, Margherita, Hawaii, Tonno

Preis: 13,00 €

gebacken unterwegs ausgeliefert

- 
- 
- 

Meier, Hauptstr. 5

Tonno, Tonno, Margherita

Preis: 10,50 €

gebacken unterwegs ausgeliefert

- 
- 
- 

## Kunde

bestellt im Ofen fertig unterwegs

- |            |                       |                                  |                                  |                       |
|------------|-----------------------|----------------------------------|----------------------------------|-----------------------|
| Margherita | <input type="radio"/> | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/> |
| Salami     | <input type="radio"/> | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/> |
| Tonno      | <input type="radio"/> | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/> |
| Hawaii     | <input type="radio"/> | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/> |

- [Neue Bestellung](#)

# 1. Übung: Rohform mit statischem HTML

es geht um  
Struktur und Inhalte;  
Layout und Formate  
sind  
Thema der 2. Übung

- Realisieren Sie alle Seiten mit HTML5
  - ⇒ "Rohform": keine physische Formatierung, kein CSS
  - ⇒ verwenden Sie strukturierende Tags wie <section> statt <div> für die globale Seitenstruktur
  - ⇒ schreiben Sie Umlaute und das €-Zeichen direkt in Ihre Dateien  
(also nicht als "named character entities / benannte Zeichen" wie &auml; etc.)
  - ⇒ achten Sie auf ordentliche Formatierung des HTML-Quelltextes!
  - ⇒ Schicken Sie die Formulardaten an das echo-Skript (siehe Tipps)
  - ⇒ **Verzichten Sie auf den Einsatz von HTML-Tabellen!**
- Werkzeug
  - ⇒ verwenden Sie einen Editor ihrer Wahl
  - ⇒ verwenden Sie als Codierung UTF-8 und schreiben Sie alle Dateien in UTF-8 ohne BOM  
(wegen PHP)
- Realisieren Sie zunächst auch diejenigen Seiten statisch, die später dynamisch aus der Datenbank generiert werden sollen
  - ⇒ tragen Sie Beispieldaten ein – so dass klar ist, was später generiert werden muss
  - ⇒ trennen Sie statische und generierte Daten voneinander
- Testen Sie Ihre Seiten mit unterschiedlichen Browsern und validieren Sie die Standardkonformität mittels gängigen Web-basierten HTML5 Validatoren

## 1. Übung: Tipps für die Entwicklung

### ■ Umgebung

- ⇒ Schalten Sie in der Entwicklungsphase im Firefox-Addon "Web Developer" den Browser Cache ab
- ⇒ Schauen Sie bei Problemen auch einmal in die Fehlerkonsole

### ■ Formulare

- ⇒ die Formulare schicken Sie zur Überprüfung an  
<https://www.fbi.h-da.de/cgi-bin/Echo.pl>  
(dieses CGI-Skript zeigt die übermittelten Formulardaten an)
- ⇒ Wenn ein Radiobutton Daten abschicken soll, verwenden Sie Befehle dieser Art (den Unterschied diskutieren wir noch)  
`<input type="radio" name=...  
      onclick="document.forms['formid'].submit();" />` oder  
`onclick="window.location.href='Baecker.html?pid=27&stat=f';" />`
- ⇒ Beachten Sie, dass die Elemente in einer <select>-Box ausgewählt sein müssen, damit sie übertragen werden

# Praktikum zu "Entwicklung webbasierter Anwendungen"

## 2. Übung: Zielsetzung

The screenshot shows the customer view of the Pipes application. At the top, there is a navigation bar with the logo 'PIPS' and links for 'Übersicht', 'Bestellung', 'Kunde', 'Bäcker', and 'Fahrer'. Below the navigation, there is a section titled 'Kunde' containing three separate boxes for different pizzas:

- Pizza Salami:
  - Bestellt
  - Im Ofen
  - Fertig
  - Unterwegs
  - Geliefert
- Pizza Margherita:
  - Bestellt
  - Im Ofen
  - Fertig
  - Unterwegs
  - Geliefert
- Pizza Margherita:
  - Bestellt
  - Im Ofen
  - Fertig
  - Unterwegs
  - Geliefert

On the right side of the screen, there is a large image of a pizza.

The screenshot shows the order view of the Pipes application. At the top, there is a navigation bar with the logo 'PIPS' and links for 'Übersicht', 'Bestellung', 'Kunde', 'Bäcker', and 'Fahrer'. Below the navigation, there is a section titled 'Bestellung' containing a list of pizzas and their details:

Pizza	Preis
Pizza Margherita - 4€	4€
Pizza Salami - 4,5€	4,5€
Pizza Hawaii - 4,5€	4,5€
Pizza Casanova - 5,5€	5,5€

Below the table, there is a summary box:

Ihr Warenkorb  
1x Pizza Salami  
2x Pizza Margherita

Buttons: Eine Pizza entfernen | Warenkorb leeren

Preis: 12,5 €

Section 'Ihre Daten':  
Johannes  
Link  
Tenilo Park  
1  
64283  
Münster  
Senden | Eingaben Löschen

On the right side of the screen, there is a large image of a pizza.

The screenshot shows the baker view of the Pipes application. At the top, there is a navigation bar with the logo 'PIPS' and links for 'Übersicht', 'Bestellung', 'Kunde', 'Bäcker', and 'Fahrer'. Below the navigation, there are three sections for different orders:

- Bestellung 10: Pizza Margherita
  - Bestellt
  - Im Ofen
  - Fertig
- Bestellung 11: Pizza Margherita
  - Bestellt
  - Im Ofen
  - Fertig
- Bestellung 11: Pizza Margherita
  - Bestellt
  - Im Ofen
  - Fertig

On the right side of the screen, there is a large image of a pizza.

The screenshot shows the driver view of the Pipes application. At the top, there is a navigation bar with the logo 'PIPS' and links for 'Übersicht', 'Bestellung', 'Kunde', 'Bäcker', and 'Fahrer'. Below the navigation, there is a section titled 'Fahrer' containing a box with order details:

Bestellung 11:  
Johannes Link, Tenilo Park 1, 64283 Münster  
Status: Gebacken

On the right side of the screen, there is a large image of a pizza.

# Fortsetzung...

The image is a collage of food-related photographs. On the left, there's a close-up of a dark blue and white striped cloth. Below it is a cluster of fresh green spinach leaves. To the right is a large, round pizza with various toppings like olives and cheese. Further right is a bunch of ripe red tomatoes still attached to their vine.

**PIPS**

Übersicht Bestellung Kunde Bäcker Fahrer

## Bestellung

Bitte wählen Sie Ihre Bestellung:

- Pizza Margherita - 4€
- Pizza Salami - 4.5€
- Pizza Hawaii - 4.5€
- Pizza Casanova - 5.5€

Ihr Warenkorb:

1x Pizza Salami  
2x Pizza Margherita

Eine Pizza entfernen Warenkorb leeren

Preis: 12.5 €

Ihre Daten:

Johannes  
Link  
Tenilo Park  
1  
64283  
Münster

Senden Eingaben Löschen

## 2. Übung: Verfeinerung mit CSS und JavaScript (1)

Testen Sie Ihre Seiten mit den gängigen Browsern und validieren Sie HTML und CSS (per Upload an den W3C-Validator)

- Entwickeln Sie ein gemeinsames Style Sheet für Ihre Website und binden Sie es in alle HTML-Seiten ein
  - ⇒ überlegen Sie sich zu diesem Zweck ein Design-Schema
  - ⇒ Farben, Schriftarten, Schriftgrößen, Abstände, Ränder, Ausrichtung, ...
  - ⇒ verwenden Sie keine physische Formatierung in HTML
  - ⇒ verwenden Sie kein style-Attribut
  - ⇒ setzen Sie die verschiedenen Maßeinheiten sinnvoll ein. Das Layout soll hinsichtlich Fenstergröße und Schriftgröße dynamisch sein!
- Realisieren Sie die Seite „Kunde (Lieferstand)“ nach dem **“Mobile First“-Ansatz**
  - ⇒ Entwickeln Sie ein geeignetes Stylesheet inkl. entspr. Mediaqueries für den Einsatz auf Mobiltelefonen
  - ⇒ Passen Sie die Komposition der Elemente dynamisch der Viewportgröße und –orientierung an
  - ⇒ Testen Sie die Darstellung mit den Web Developer Tools

## 2. Übung: Verfeinerung mit CSS und JavaScript (2)

Tipp: `xxx.toFixed(2)`  
macht aus einer Zahl  
einen String mit 2  
Nachkommastellen und  
vermeidet  
Rundungsfehler

- Einsatzbereiche für ECMAScript im Pizzaservice:
  - ⇒ Klick auf ein Pizzabild trägt diese Pizza in den Warenkorb (Liste) ein
    - 3 Pizzen bestellen ⇒ 3-mal klicken
  - ⇒ Möglichkeit zum Löschen der Einträge im Warenkorb
    - sowohl "Alle löschen" als auch "(Mehrfach-)Auswahl löschen"
  - ⇒ Bestellung wird nur abgeschickt, wenn eine Lieferadresse angegeben ist
  - ⇒ Deaktivierung des Bestellknopfes bei unvollständigen Daten Wie lösen Sie das?
  - ⇒ Berechnung des Preises
  - ⇒ Klick auf Radio-Button in der Statusliste sendet Statusänderung (bei Bäcker und Fahrer)
- Verwenden Sie nur DOM-konforme Attribute und Funktionen
  - ⇒ keine browserspezifischen Spezialitäten  
(Vorsicht mit Vorlagen aus dem Netz !)
- Verwenden Sie "`use strict`"; in allen Funktionen
- Testen Sie die Skripte mit unterschiedlichen Browsern und Lint-Tools

## 2. Übung: Tipps für die Entwicklung

### ■ Umgebung

- ⇒ Aktivieren Sie den im Browser eingebauten Debugger
- ⇒ Bei Problemen mit DOM oder ECMAScript
  - Prüfen Sie die Browser-Konsole auf Fehlermeldungen (Aufruf mit F12)
  - Nutzen Sie den DOM-Inspector und den Befehl "Inspect" im rechten Mausmenü um das DOM zu einem Tag anzuschauen
  - Achten Sie auf Groß-Klein-Schreibung bei DOM-Aufrufen
  - Prüfen Sie Ihr ECMAScript mit JSLint: <http://www.jslint.com> oder ESLint für ECMAScript 6: <http://eslint.org/demo/>

### ■ Formulare

- ⇒ die Formulare schicken Sie zur Überprüfung an  
<https://www.fbi.h-da.de/cgi-bin/Echo.pl>  
(dieses Skript zeigt die übermittelten Formulardaten an)
- ⇒ Achten Sie darauf, dass alle Eingabedaten übermittelt werden !

## 3. & 4. Übung: Zielsetzung und allgemeine Hinweise

### ■ Zielsetzung

- ⇒ Sie sollen die Datenbankanbindung mit PHP und MySQLi verstehen
- ⇒ Sie sollen Objektorientierung mit PHP verstehen und üben
- ⇒ Dazu implementieren Sie den Pizzaservice unter Verwendung von Klassen und verwalten die anfallenden Daten in einer MySQL Datenbank

### ■ Hinweise

- ⇒ Alle Seiten müssen objektorientiert unter Verwendung der gegebenen Templates (Download von der EWA-Moodleseite) implementiert werden
- ⇒ Alle 4 HTML-Seiten werden unter Verwendung der Daten aus der Datenbank mit PHP erzeugt
- ⇒ Alle Zugriffe auf die Datenbank erfolgen über MySQLi
- ⇒ Der Übergang zwischen 3. und 4. Übung ist fließend und bleibt Ihnen überlassen – aber für die Abnahme muss alles fertig sein !

Achtung! Die reine Umsetzung der Funktionalität des Pizzaservice reicht nicht für die Abnahme! Die objektorientierte Umsetzung mit Klassen ist ebenfalls ein Pflichtbestandteil!

## 3. & 4. Übung: Umsetzung mit AJAX (für Fortgeschrittene)

### ■ Hinweise

- ⇒ Sie können die Kommunikation zwischen den Client- und Server-Seiten auch mittels AJAX realisieren
- ⇒ Hierzu senden und empfangen Client und Server idealerweise JSON
- ⇒ D.h., Sie müssen die Seitenklassen derart umschreiben, dass diese statt den HTML-Seiten JSON-Daten generieren, die Client-seitig verarbeitet werden.
- ⇒ Hierbei findet eine Verlagerung der Applikations- bzw. Verarbeitungslogik hin zum Client statt, d.h., der Client ist fortan für die korrekte Verarbeitung und Anzeige der empfangenen Daten verantwortlich; implementieren Sie dies mit JavaScript
- ⇒ Aktualisieren Sie nur die Bereiche, die sich ändern; das komplette Neuladen der Seite ist mit AJAX nicht notwendig
  - Verwenden Sie ggf. einen Timer für die Aktualisierung
- ⇒ Der Datenbankzugriff erfolgt unverändert mit MySQLi und PHP

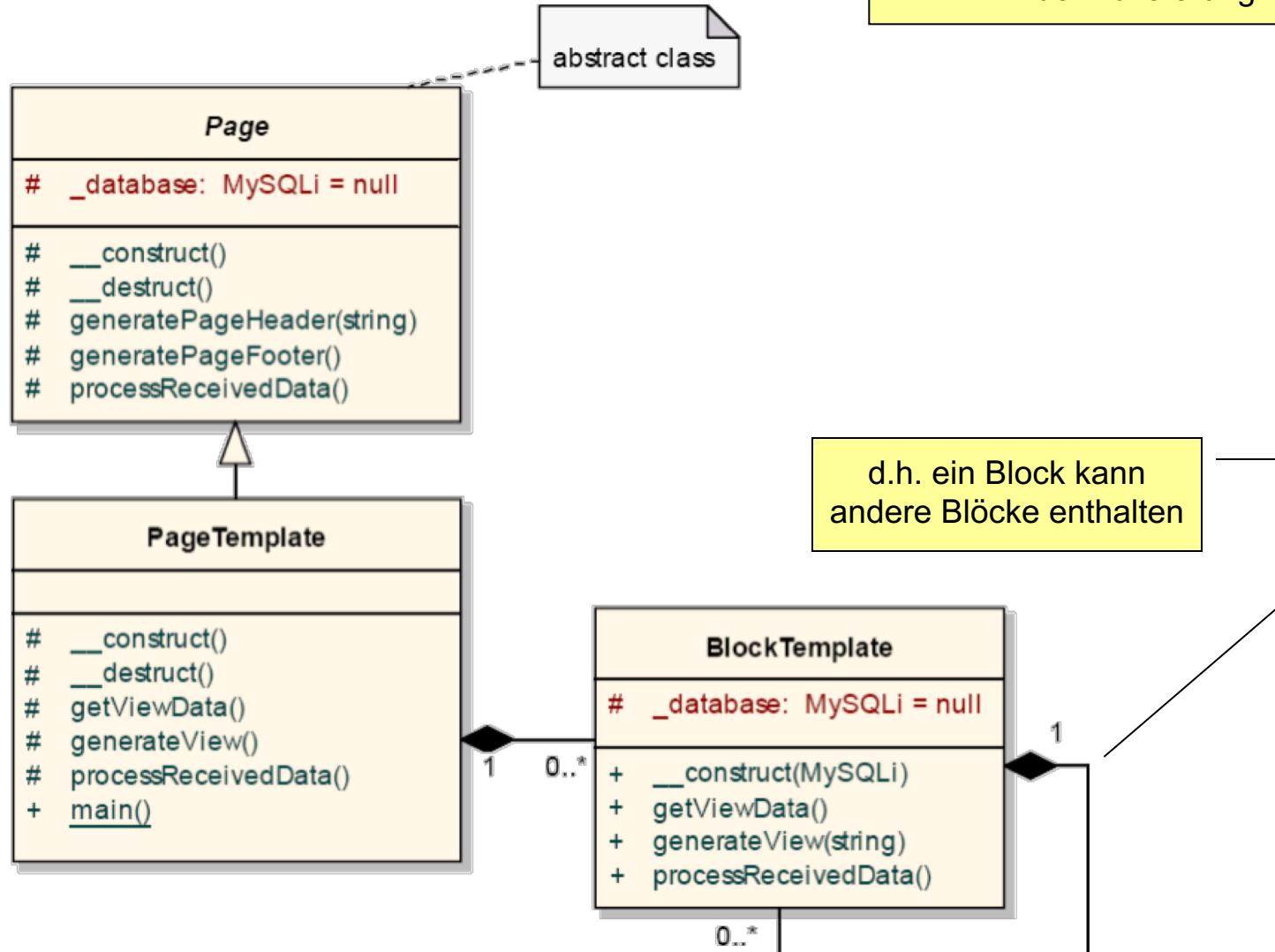
## 3. Übung: Vorbereitung

### ■ Vorbereitung

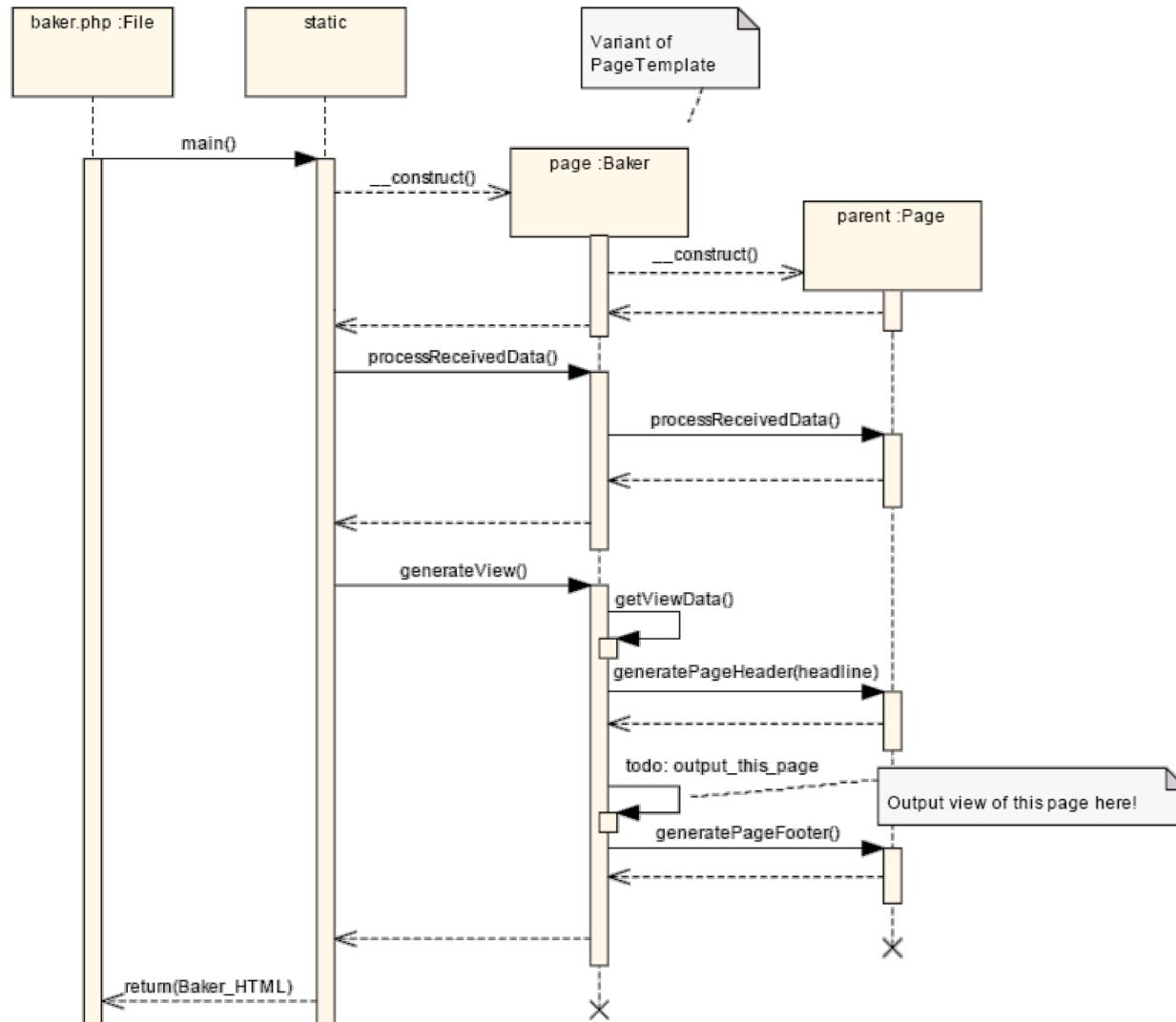
- ⇒ Installieren Sie Apache / PHP / MYSQL auf Ihrem Entwicklungsrechner
- ⇒ Laden Sie die Zulieferung für das Praktikum von der Webseite herunter:
  - PageTemplate.php dient als Vorlage für die 4 Seiten Bestellung.php, Kunde.php, Baecker.php und Fahrer.php
  - Page.php ist die gemeinsame Basisklasse dieser 4 Seiten-Klassen und soll die Datenbank öffnen und schließen und den HTML-Rahmen erzeugen
  - BlockTemplate.php dient als Vorlage für einzelne Blöcke innerhalb der Seiten
  - Pizzaservice\_Documentation.pdf enthält eine Dokumentation der Klassen mit Klassendiagramm und Sequenzdiagramm
- ⇒ Versuchen Sie mit der Dokumentation und dem Quellcode das Zusammenspiel der verschiedenen Klassen zu verstehen. Klären Sie folgende Fragen:
  - Wo erfolgt der eigentliche Aufruf zur Erstellung einer HTML-Seite?
  - Was tun die Methoden getViewData(), generateView() und processReceivedData()?
  - Wo wird der HTML-Rahmen erzeugt? Wo wird er ausgegeben?
- ⇒ Legen Sie 4 Kopien der Klasse PageTemplate.php an und nennen Sie die Dateien Bestellung, Status, Baecker und Fahrer. Ändern Sie auch die Klassennamen und Verweise auf die Klasse innerhalb der Dateien.

## 3. Übung: Klassendiagramm der Templates

siehe auch  
Pizzaservice\_Documentation.pdf  
in der Zulieferung



## 3. Übung: Sequenzdiagramm der Templates



## 3. Übung: HTML in Klassen übertragen

### ■ Vorgehen

- ⇒ verteilen Sie Ihren HTML-Code aus der 3. Übung in die zuständigen Methoden der 5 Klassen Page, Bestellung, Status, Baecker und Fahrer
  - die Kommentare in den Dateien helfen dabei
  - HTML-Ausgaben erfolgen nur in generateView() !
  - die Struktur wird besser, wenn Sie Speisekarte, Warenkorb-Formular und Statustabellen unter Verwendung von BlockTemplate.php realisieren (das ist aber optional)
  - die Ausgabe größerer HTML-Abschnitte ist mit der "Heredoc-Notation" besonders einfach. Achten Sie darauf, dass die Endmarke in der ersten Spalte beginnen muss und höchstens noch ein ; folgen darf
- ⇒ schreiben Sie Hilfs-Methoden oder zusätzliche Klassen nach Bedarf
  - die Methoden der gegebenen Klassen dürfen auch zusätzliche Parameter bekommen

### ■ Test

- ⇒ Prüfen Sie, ob die neuen PHP-Seiten als Ausgabe die ursprünglichen statischen HTML-Seiten erzeugen
- ⇒ Wird der erzeugte HTML-Code immer noch vom Validator akzeptiert?
- ⇒ Wenn die Tests erfolgreich sind, können Sie mit der Implementierung der DB-Zugriffe beginnen

## 4. Übung: Implementierung der Datenbank

Bis hierher sollten Sie in der 3. Übung mindestens kommen!

- Entwerfen Sie das Datenmodell für den Pizzaservice, z.B.
  - ⇒ Angebot: PizzaName, Bilddatei, Preis
  - ⇒ BestelltePizza: PizzaID, fBestellungID, fPizzaName, Status
  - ⇒ Bestellung: BestellungID, Adresse, Bestellzeitpunkt
- Implementieren Sie das Datenmodell mit phpMyAdmin
  - ⇒ verwenden Sie die Kollation utf8\_unicode\_ci (im Vorgabewert utf8\_general\_ci gilt nicht ß=ss)
  - ⇒ PizzaName, PizzaID, BestellungID sind Primärschlüssel; IDs mit Autoincrement
  - ⇒ realisieren Sie die Verknüpfungen zwischen den Primärschlüsseln und den Fremdschlüsseln fBestellungID, fPizzaName in der Datenbank  
Tipp: Mit dem "Designer" in phpMyAdmin können Sie die Beziehungen grafisch eintragen
  - ⇒ füllen Sie die Tabelle "Angebot" manuell mit phpMyAdmin
- Tipp zum Bestellzeitpunkt
  - ⇒ MySQL-Funktion CURRENT\_TIMESTAMP als Standardwert des Feldes

## 4. Übung: Datenbankanbindung mit PHP / MySQLi

### ■ Vorgehensweise

- ⇒ implementieren Sie die Datenbankzugriffe (Select, Insert Into, Update) in den zuständigen Methoden der Klassen
  - der Zugriff auf die Datenbank erfolgt objektorientiert über die Klasse MySQLi
  - Zugriff auf die Datenbank erfolgt nur in getViewData() und processReceivedData()
- ⇒ ersetzen Sie die bisherigen statischen Tabellen durch PHP-Code, der die Zeilen aus den abgefragten Daten generiert
  - bilden Sie den bisherigen statischen HTML-Code exakt nach !
  - schreiben Sie Hilfs-Methoden oder zusätzliche Klassen nach Bedarf
- ⇒ testen und debuggen ... error\_reporting(E\_ALL) hilft dabei

### ■ Tipps zur Umsetzung

- ⇒ var\_dump(\$variable) für die schnelle Testausgabe zwischendurch
- ⇒ number\_format(\$zahl, \$nachkommastellen) formatiert \$zahl
- ⇒ \$mysqli->insert\_id liefert die Autoincrement-ID nach INSERT INTO
- ⇒ Tabellen- und Feldnamen in MySQL ggf. in ` (Gravis / accent grave) einklammern
- ⇒ prüfen Sie mit phpMyAdmin ob die Datenbankeinträge korrekt erstellt werden

Alle Seiten müssen objektorientiert unter Verwendung der gegebenen Templates implementiert werden

## 4. Übung: Sessionverwaltung und Sicherheit

- Der Kunde soll auf seiner Statusseite nur diejenigen Pizzen sehen, die er selbst zuletzt bestellt hat
  - ⇒ Implementieren Sie dieses Feature mittels Sessionverwaltung: speichern Sie die letzte AuftragsNr in der Session und filtern Sie damit die Pizzaliste
- Verhindern Sie SQL-Injection mit Hilfe von `real_escape_string`
  - ⇒ Test: geben Sie `/'"\'` als Lieferadresse ein; diese Zeichen müssen auf der Fahrerseite genau so erscheinen
- Verhindern Sie Cross Site Scripting mit Hilfe von `htmlspecialchars`
  - ⇒ Test: geben Sie `<b>xxx</b>` als Lieferadresse ein; dies muss genau so in der Datenbank und in der Ausgabe auf der Fahrerseite erscheinen
- abschließend testen und generierte Seiten validieren

## 5. Übung: Testfälle bzw. WebSockets und Abnahme

- Allgemein: Auswahl aus 2 Optionen
  - ⇒ a) Automatisiertes Testen mittels Selenium
  - ⇒ b) Umsetzung eines Bereichs Ihrer Wahl mittels WebSockets
  
- a) Funktionstests mittels Selenium IDE bzw. Selenium WebDriver
  - ⇒ Entwickeln Sie 2 sinnvolle(!) Acceptance Tests und implementieren Sie diese mittels Selenium
  - ⇒ Achten Sie auf eine vollständige Umsetzung der Acceptance Tests

### Hinweis:

- Je nach Testfall können Sie entweder Selenium IDE verwenden und den Testfall aufzeichnen oder diesen mittels dem Selenium WebDriver in einer Sprache Ihrer Wahl (bspw. Java) programmieren.
- Überlegen Sie sich auch geeignete Validierungskriterien anhand derer sich eine korrekte Abnahme bzw. Erfüllung festmachen lässt

## 5. Übung: Testfälle bzw. WebSockets und Abnahme

### ■ b) WebSockets

- ⇒ Suchen Sie sich einen Bereich Ihrer Wahl aus dem Pizzaservice und implementieren Sie diesen mittels WebSockets
  - bspw. die Statusupdates auf der Kundenseite
  - die Aktualisierung auf Fahrerseite bzw.
  - die Aktualisierung der Pizzabäckeransicht
- ⇒ Hinweis:
  - Primär steht die Auseinandersetzung mit einer neuen Technologie im Vordergrund und weniger die Ausimplementierung bis ins kleinste Detail
  - Verwenden Sie Server-seitig eine WebSocket-Technologie Ihrer Wahl (bspw. WebSockets für PHP, Node.js oder ein Framework wie Play)
  - Arbeiten Sie mit Testdaten, falls die Anbindung an den Pizzaservice nicht funktioniert

## 5. Übung: Abnahme

- Allgemein: Die 5. Übungseinheit dient zur Abnahme Ihrer Applikation
  - ⇒ **Keine Implementierungsarbeiten mehr!**
  - ⇒ Stellen Sie **vor Beginn(!)** der Einheit sicher, dass Ihre Applikation inkl. der Testfälle vollständig und funktionstüchtig ist
  - ⇒ Die Abnahme findet an Ihren Entwicklungsrechnern statt
  - ⇒ Mock-ups werden nicht akzeptiert
- Bei der Abnahme
  - ⇒ wird die Funktionstüchtigkeit der Applikation überprüft
  - ⇒ sollten Sie Ihren Code verstehen und erklären können
  - ⇒ sollten Sie die Testfälle automatisiert ablaufen lassen bzw. die WebSocket-Umsetzung demonstrieren
  - ⇒ sollten Sie insgesamt gut vorbereitet sein.

## Praktikum zu "Entwicklung webbasierter Anwendungen"

### Abnahme: Checkliste

- Hinweis: Für eine erfolgreiche Abnahme müssen min. 9 Punkte erfüllt sein:

1.	Keine Tabellen im Quellcode enthalten	
2.	Verwendung von PHP-Seiten- oder Blockklassen	
3.	Gesamter Bestellprozess funktioniert fehlerfrei	
4.	Seiten werden vom W3C Validator als fehlerfrei erkannt	
5.	Responsives Layout + Verwendung von Media Queries	
6.	Automatische Seitenaktualisierung (bspw. für Statusänderungen)	
7.	Absicherung der Seite gegen Cross-Site Scripting (XSS)	
8.	Absicherung gegen SQL-Injection	
9.	Daten werden korrekt in die DB geschrieben	
10.	Benutzer sieht nur die eigene Bestellung (Test mit 2 versch. Browsern)	