

DB-Zugriff mit PHP auf eine MySQL-Datenbank

Allgemeine Betrachtung

PHP ist eine serverseitige Programmiersprache, die beispielhaft steht für andere, wie Perl (.cgi, .pl), Java Server Pages (.jsp) oder Python. Diese Sprachen sind unabhängig vom Betriebssystem und Webserver, allerdings müssen entsprechende Interpreter in Betriebssystem und Webserver eingebunden werden können.

Mit dem Zugriff auf ein DBMS (Datenbank-Management-System) wird die erstellte Webanwendung zum Teil abhängig vom angebundenen DBMS. Um Anwendungen mit DB-Zugriff trotzdem weitgehend unabhängig vom verwendeten DBMS zu machen, werden alle Vorgänge, die sich auf ein bestimmtes DBMS beziehen in verschiedene Dateien ausgelagert, mit PHP 5 können Klassen erstellt werden, die dann bei Migration auf ein anderes DBMS leicht angepasst oder ausgetauscht werden können. Genauso soll die Programmierung der Oberfläche von der Gestaltung und der Programmlogik getrennt werden, so dass HTML-Code, css-Style und php-Code weitestgehend getrennt sind.

Aufgabenstellung

Die C#-Anwendung PC-Verwaltung soll im Folgenden mit einer Schritt für Schritt-Anleitung auf eine Web-Anwendung übertragen werden (s. Abb. 1 und PC_Verwaltung.exe).

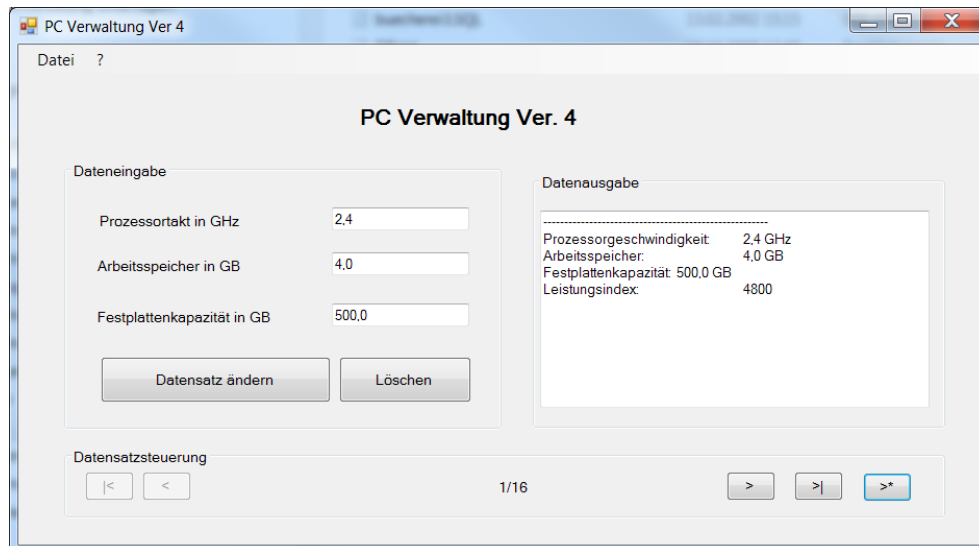


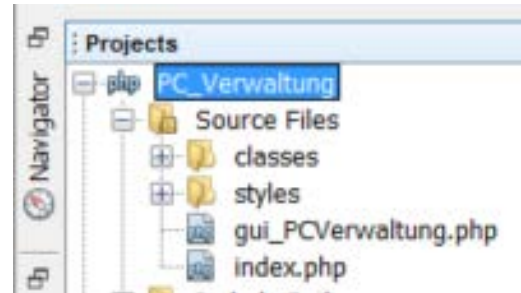
Abb. 1

Entwurf der Oberfläche mit NetBeans

Erstellen Sie mit der folgenden Anleitung eine Oberfläche, die das Layout der obenstehenden Windows-Forms-GUI übernimmt.

- Legen Sie dazu ein neues PHP-Projekt PCVerwaltung an.
- Die Datei ‚index.php‘ bleibt vorerst unbearbeitet.

- Erstellen Sie die Datei ,gui_PCVerwaltung.php', die den HTML-Code der Seite enthalten wird.
- Erstellen Sie dann im Ordner ,Source Files' einen Unterordner (,Folder') ,styles' und darin die .css-Datei ,guiPC_Verwaltung.css', die die Styles für die HTML-Seite enthalten wird.
- Erstellen Sie den Unterordner ,classes', der die PHP-Klassen enthalten wird.
- **Verschieben** Sie den gesamten **HTML**-Code aus der Datei ,index.php' in die ,gui_PCVerwaltung.php' .
- Binden Sie in der Datei ,index.php' die Datei ,gui_PCVerwaltung.php' ein mit:



```
<?php
    require './gui_PCVerwaltung.php';
?>
```

- Ergänzen Sie den Titel der Webseite und binden Sie die Stylesheet-Datei in die HTML-Code-Datei ,gui_PCVerwaltung.php' ein mit:

```
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" type="text/css" href="styles/guiPC_Verwaltung.css">
    <title>.: PC-Verwaltung :.</title>
</head>
```

Gestaltung der Oberfläche

Abb. 2 zeigt die Oberfläche, wie sie aussehen wird, nachdem Sie die folgende Anleitung durchgearbeitet haben.

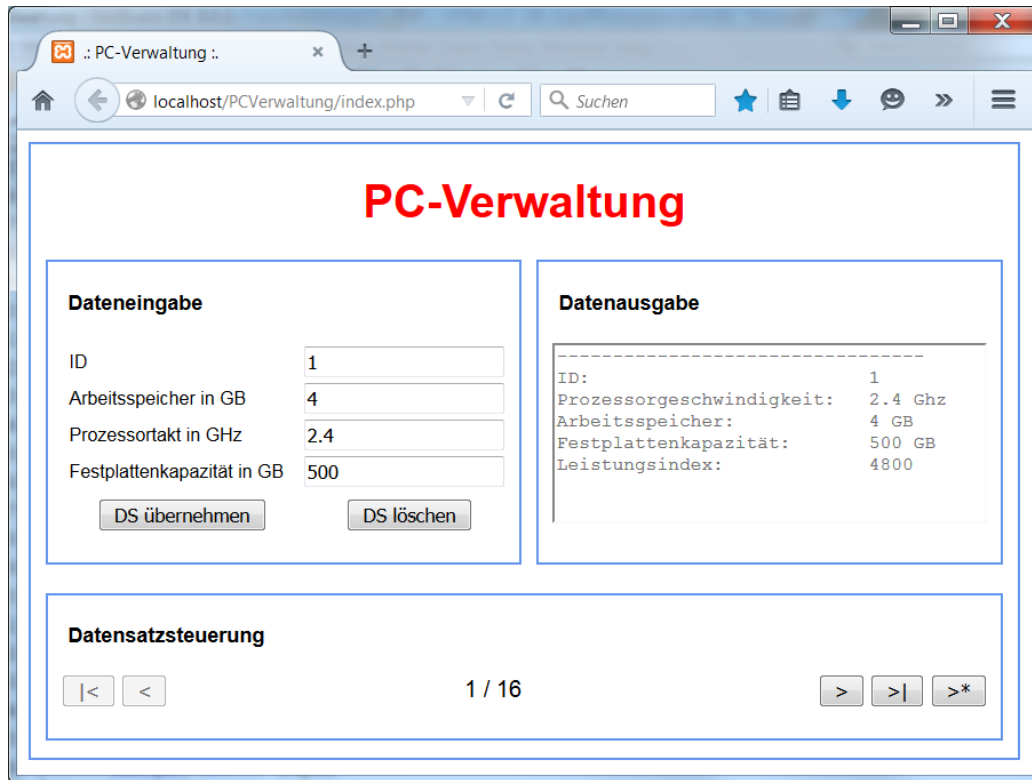


Abb.2

Die GroupBoxes von C# können mit div-Containern in HTML nachgebildet werden (s. hellblaue Rechtecke in Abb. 2).

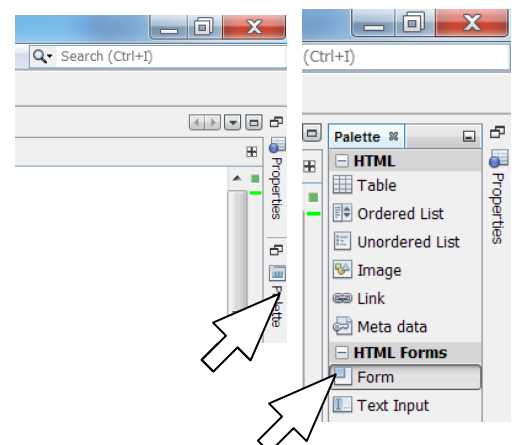
Damit die Webanwendung mit dem PHP-Skript interagieren kann, muss die ganze Webseite in der ‚gui_PCVerwaltung.php‘ mit einem <form>-tag umschlossen werden. Blenden Sie dazu die ‚Palette‘ ein. Sollte diese nicht angezeigt werden, können Sie diese erreichen über **Window/IDE Tools/Palette** oder Strg+Umschalt+8.

Ziehen Sie den Form-Tag an die Cursor-Position.

```

</head>
<body>
|
</body>
</html>

```



Ziel des Formulars („action“) ist die ‚index.php‘ und als Übertragungsmethode wählen Sie ‚get‘.

Um die Boxen herzustellen verwenden Sie ,div'-Container. Als erstes legen Sie den äußeren Rahmen an mit:

```
<form action="index.php">
  <div id="main">
    </div>
  </form>
```

Erläuterung:

id="main" wird für die Cascading Style Sheets verwendet. Infos zu Cascading Style Sheet folgen etwas weiter unten.

Fügen Sie nun im div-Container die H1-Überschrift ,PC-Verwaltung' hinzu. Die Webseite sieht nun – völlig unspektakulär – so aus:



Nun zu den Style Sheets

Infoabschnitt zu Cascading Style Sheets (css)

Cascading Style Sheets sind Dateien, die sich auf HTML-Dateien beziehen und das Layout einer Seite angeben. HTML gibt den Inhalt der Seite an, der Style (css) gibt das Aussehen einer Seite an. Welche Gestaltungsmöglichkeiten man mit Style Sheets besitzt, zeigt die Seite ,[CSS ZEN GARDEN](#)', die den gleichen Inhalt mit sehr vielen unterschiedlichen Layouts zeigt.

Styles können auf HTML-Tags angewendet werden. Zum Beispiel sollen alle Texte der Seite in die Schrift ,Arial' gesetzt werden. Fügen Sie dazu in der Datei ,gui_PCVerwaltung.css' ein Layout für den <body>-Tag ein:

```
body{
  font-family: Arial, Helvetica;
}
```

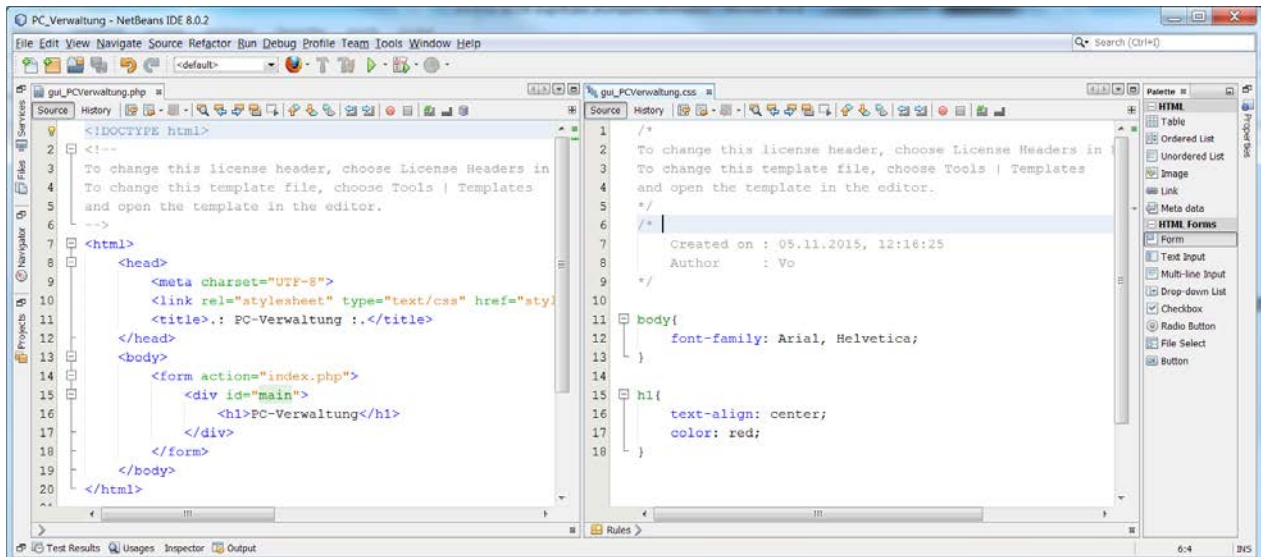
Außerdem sollte die Überschrift zentriert und rot sein:

```
h1{
  text-align: center;
  color: red;
}
```

Damit sollte das Ganze so aussehen:



Da Sie nun häufig mit der Datei ‚gui_PCVerwaltung.php‘ und der Datei ‚gui_PCVerwaltung.css‘ parallel arbeiten ist es günstig die Fenster in Netbeans nebeneinander anzuordnen:



Dazu ziehen Sie am Reiter der einen Datei, bis die gewünschte Anordnung sichtbar wird.

Warum nun die ‚id‘ im div-Container? Der Style den man einem Tag gibt – bei uns dem H1-Tag – gilt für alle Elemente eines Dokuments, d.h. alle H1-Überschriften wären zentriert und rot. Man kann auch mehrere Elemente mit dem gleichen Style versehen, dann verwendet man das Schlüsselwort ‚class‘. Will man nur ein Element mit einem ganz bestimmten Style versehen, verwendet man das Schlüsselwort ‚id‘, wie beim div-Container. Der soll jetzt gestaltet werden:

```
div#main{
    margin: auto;
    width: 700px;
    height: 405px;
    border: 2px solid cornflowerblue ;
    padding: 10px;
}
```

Erläuterungen:

div#main verweist auf den div-Container mit der id="main"

margin legt den Abstand zum nächsten Element fest

 auto zentriert in der Mitte

width legt die Breite in Pixeln fest

height legt die Höhe in Pixeln fest

border legt den Rahmen fest

padding legt einen Abstand aller inneren Elemente zum Rand in Pixeln fest

Nun sollte die Anwendung so aussehen:



Fügen Sie nun in der HTML-Datei drei div-Container hinzu, die sich im vorhandenen Container befinden. In allen Containern erstellen Sie eine H4-Überschrift.

Vorgaben für die Container:

id "dateneingabe"
Text für die Überschrift: Dateneingabe
Breite: 320px
Höhe: 210px
Rahmen: wie der umschließende Container
alle innenliegenden Elemente sollen 10px Abstand zum linken Rand halten.

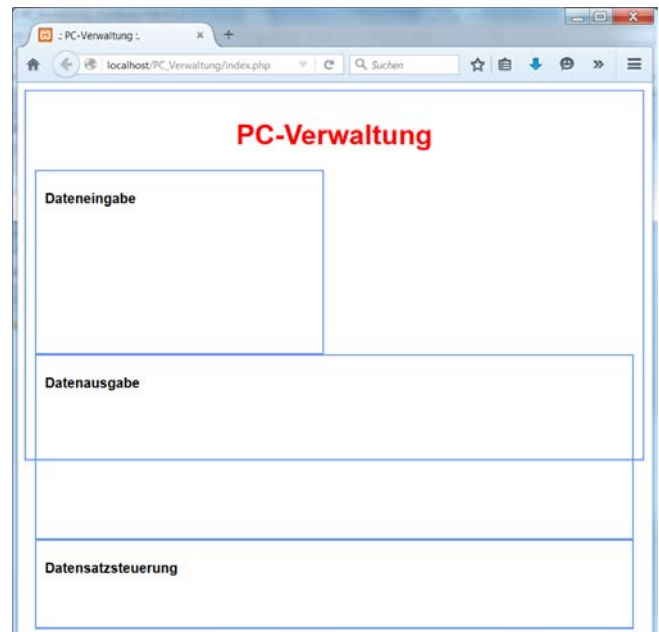
id "ein_ds_anzeige"
Text für die Überschrift: Datenausgabe
Höhe: 210px
Rahmen: wie der umschließende Container
alle innenliegenden Elemente sollen 10px Abstand zum linken und zum rechten Rand halten.

id "datensatzsteuerung"
Text für die Überschrift: Datensatzsteuerung
Höhe: 100px
Rahmen: wie der umschließende Container
alle innenliegenden Elemente sollen 10px Abstand zum linken und zum rechten Rand halten.

Nun sieht die Oberfläche so aus:

Das entspricht noch nicht dem gewünschten Aussehen. Versuchen Sie den Container ‚Datenausgabe‘ in der Breite zu begrenzen, auch damit ordnet er sich nicht neben den Container Dateneingabe ein. Um mehrere Container nebeneinander zu setzen müssen diese ‚floaten‘, d.h. ein fließendes Layout annehmen.

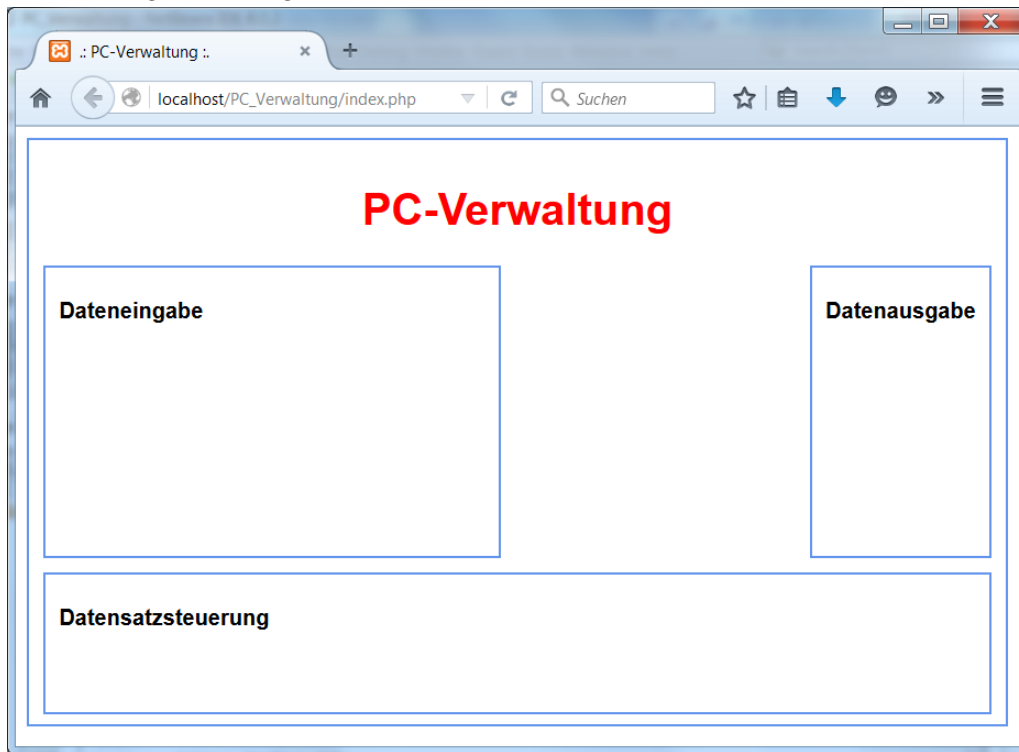
Deshalb fügen Sie dem Container ‚Dateneingabe‘ den Style `float: left;` hinzu. Testen Sie das Aussehen!



Damit der Container ‚Datenausgabe‘ eigenständig bleibt, benötigt er ein `float: right;`

Die Datensatzsteuerung soll unten in der Mitte bleiben und darf daher nicht mehr floaten. Dies erreicht man mit `clear: both;`. Den Abstand zwischen Dateneingabe und Datensatzsteuerung erreichen Sie mit `margin-bottom: 10px;` im Style für die Dateneingabe.

Nun hat die Anwendung fast das gewünschte Aussehen:



Die Breite der Datenausgabe wird mit dem innenliegenden Steuerelement geregelt.

Eine Einführung zu css und weiterführende Informationen finden Sie unter

<http://www.peterkropff.de>

NetBeans unterstützt Sie bei der Verwendung der Selektoren und Eigenschaften.

Wir fügen nun die Steuerelemente im Container Dateneingabe und Datenausgabe hinzu. Zur Anordnung der Elemente in der Dateneingabe kann eine Tabelle verwendet werden. Verwenden Sie die Tabelle aus der Palette von NetBeans. Da die Schrift eine bestimmte Größe haben sollte, muss dieses über das Stylesheet geregelt werden. Dazu weisen Sie den Zellen für die Beschreibung der Eingabefelder eine Style-Klasse zu, z.B.:

```
<td class="label">
    ID
</td>
```

und definieren den Style für diese Style-Klasse mit:

```
td.label{
    font-size: 10.5pt;
}
```

Nun sollten die Beschreibungen der Eingabefelder einzellig dargestellt sein.

Die Textfelder entnehmen Sie auch der Palette von NetBeans und vergeben die Namen ,txtid‘, ,txtram‘, ,txthdd‘, ,txttakt‘. Das Textfeld für die ID soll ,readonly‘ gesetzt werden.

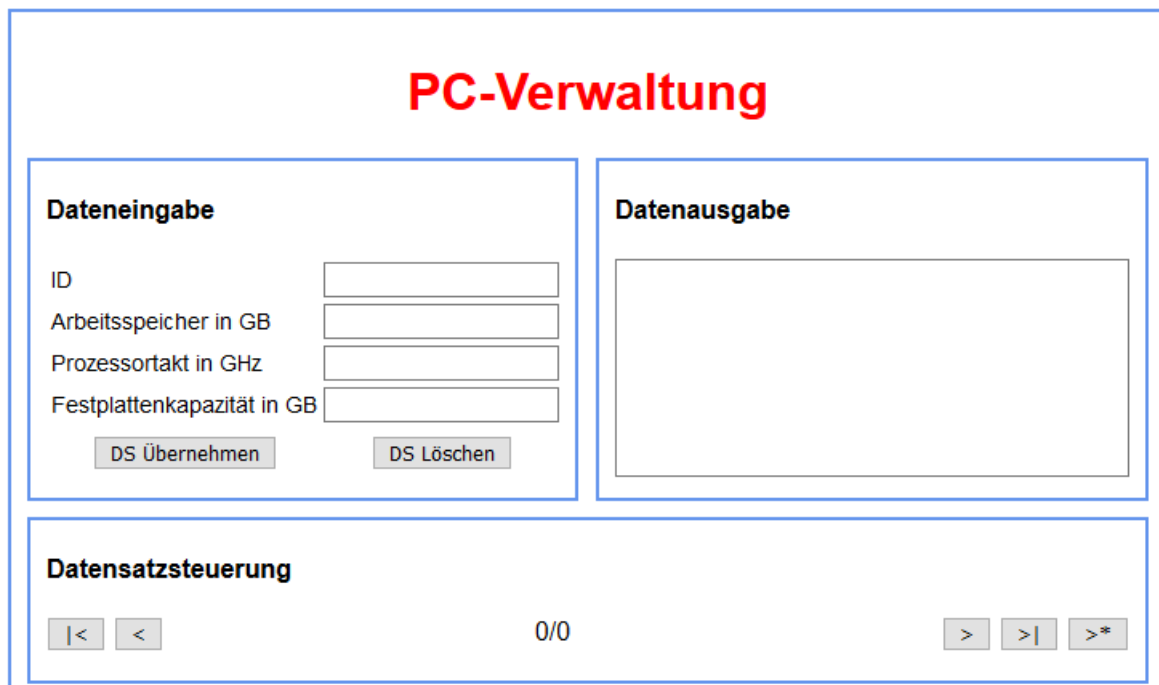
Entnehmen Sie die Buttons ,DS Übernehmen‘ und ,DS Löschen‘ der Palette als Submit-Buttons und vergeben Sie die Namen ,btn_uebernehmen‘ und ,btn_loeschen‘. Damit die Buttons zentriert in den Tabellenzellen und etwas abgesetzt erscheinen, vergeben Sie eine Klassenbezeichnung für die zwei Tabellenzellen, weisen dieser Klasse den Style `text-align: center;` zu und stellen einen oberen Abstand zum Inhalt der Zelle von 5px her.

In der TextArea (aus NetBeans Palette ,MultiLineInput‘) wählen Sie ,readonly‘ und löschen die Angaben für die Zeilen und die Spalten. Als Style definieren Sie ,width: 315px;‘, ,height: 130px;‘ und, damit die Größe nicht verändert werden kann, ,resize: none;‘.

Für die Buttons der Datensatzsteuerung wählen Sie immer den Submit-Type und die Namen ,btn_first‘, ,btn_prev‘, ,btn_next‘, ,btn_last‘ und ,btn_new‘. Tooltips können Sie mit dem title-Attribut hinzufügen, z.B.: `title="Erster Datensatz"` für den btn_first. Diese Buttons müssen im Container ,floaten‘ und sollten einen Abstand von 5px zueinander haben. Der Text, der den aktuellen Index und die Anzahl der Datensätze anzeigt, muss in einem eigenen div-Container zentriert definiert werden.

Achtung, die Reihenfolge im Quelltext entspricht nicht der Reihenfolge der Buttons auf der Oberfläche.

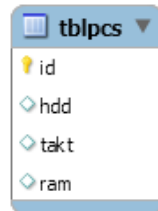
Am Ende sollte die Oberfläche so aussehen:



Nachdem die Oberfläche steht, kann nun der Datenbankzugriff programmiert werden.

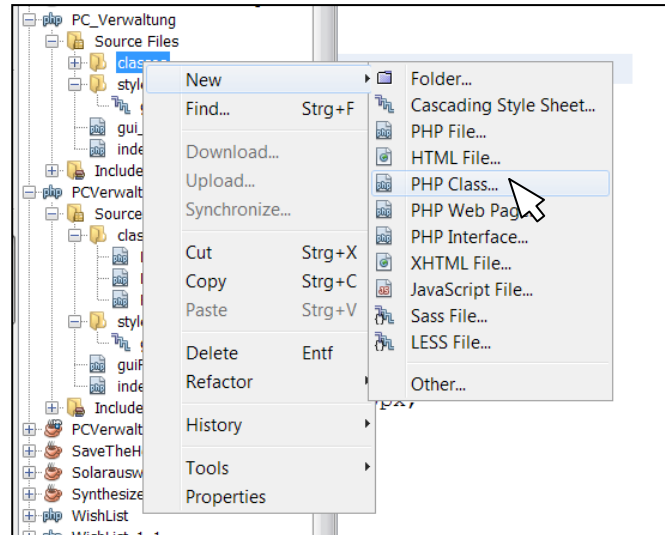
DB-Zugriff auf eine MySql-Datenbank über PHP (mysqli-Bibliothek)

Als Vorbereitung importieren Sie die Datenbank ‚pcVerwaltung‘ aus der Datei ‚pcVerwaltung.sql‘. Die Datenbank enthält nur eine Tabelle, die ‚tblpcs‘.



Klasse PC programmieren

Für das objektrelationale Mapping – d.h. für jede Tabelle wird eine passende Klasse hergestellt – muss eine Klasse PC programmiert werden, mit den Attributen, die den Tabellenattributen entsprechen.



Erstellen Sie dazu im Ordner ‚classes‘ eine neue PHP-Klasse ‚PC‘ mit den zur Datenbanktabelle passenden Attributen und get- und set-Methoden, wie im folgenden Beispiel.

Attributdeklaration mit Get- und Set-Methoden:

```
private $id=0;

public function setId($id) {
    $this->id=$id;
}

public function getId() {
    return $this->id;
}
```

Erläuterungen:

Datentypen von Attributen, lokalen Variablen und Parametern, sowie Rückgabetypen von Methoden werden in PHP nicht angegeben.

\$this ist die Referenz auf das eigene Objekt und in PHP zwingend, wenn auf Attribute der Klasse zugegriffen wird.

-> Der Pfeiloperator ist das gleiche, wie der Punktoperator in C# und stellt einen Zugriff auf ein Attribut oder eine Methode eines Objekts dar.

Als Vereinfachung der Darstellung legen Sie noch eine Methode ‚getAusgabeString():string‘ an, die die Ausgabe für die TextArea erzeugt (s. Abb). Der Leistungsindex wird dabei von einer Methode ‚getLeistungsindex():double‘ erzeugt, in der alle drei Attribute der Klasse multipliziert werden und das Ergebnis zurückgegeben wird.

Datenausgabe

```
-----
ID: 4
Prozessorgeschwindigkeit: 3.4 Ghz
Arbeitsspeicher: 2 GB
Festplattenkapazität: 800 GB
Leistungsindex: 5440
```

Jetzt können Sie den ersten Test mit der Oberfläche durchführen:

Verknüpfung der Fachklasse ‚PC‘ mit der Oberfläche

Binden Sie die Klasse PC in den Quellcode der ‚index.php‘ ein, erstellen Sie einen neuen PC und geben Sie Daten für diesen PC an:

```
require './classes/PC.php';

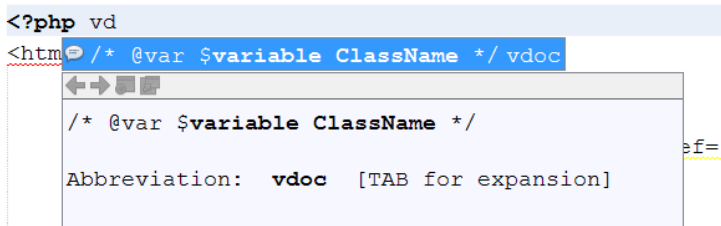
$index=0;
$anzahl=1;

$pc=new PC();
$pc->setHdd(1000);
$pc->setId(1);
$pc->setRam(16);
$pc->setTakt(4.6);

// Datensatzzeiger erzeugen
$ds_anzeiger=($index+1)." / ".$anzahl;

require './gui_PCVerwaltung.php';
```

Nun muss der PC auf der Oberfläche, also in ‚gui_PCVerwaltung.php‘ angezeigt werden. Dazu benötigen Sie Zugriff auf die Methoden des PC. Um sich die Arbeit zu erleichtern, legen Sie zu Beginn des HTML-Dokuments einen Hinweis für NetBeans an, dass die Variable \$pc vom Typ PC ist. Schreiben Sie ‚<?php vd‘ und drücken Sie Strg+Leertaste:



```
<?php vd
<html /* @var $variable ClassName */ vdoc
/* @var $variable ClassName */
Abbreviation: vdoc [TAB for expansion]
```

Bestätigen Sie mit der Enter-Taste, und geben Sie als VariablenNamen ‚\$pc‘ und als Klassennamen ‚PC‘ an. Dann sieht der Quellcode so aus:

```
<?php /* @var $pc PC */ ?>
<html>
<head>
```

Zur Anzeige der Daten des PC müssen in den Attributen ,value' der Textboxen die entsprechenden get-Methoden des PC aufgerufen werden:

Beispiel ,ID':

```
<input type="text"
      name="txtid"
      value="<?php echo $pc->getId(); ?>"
      readonly="readonly" />
```

In der TextArea muss die Methode ,getAusgabeString():string' aufgerufen werden:

```
<textarea id="datenausgabe"
          name=""
          readonly="readonly"
          ><?php echo $pc->getAusgabeString(); ?></textarea>
```

Achtung: Fügen Sie keine Leerzeichen vor dem php-Abschnitt ein!

Die Angabe ,0/0' in der Datensatzsteuerung muss ersetzt werden durch:

```
<?php echo $ds_anzeiger; ?>
```

Damit sollte die Oberfläche so aussehen:

PC-Verwaltung

Dateneingabe

ID	<input style="width: 90%;" type="text" value="1"/>
Arbeitsspeicher in GB	<input style="width: 90%;" type="text" value="16"/>
Prozessortakt in GHz	<input style="width: 90%;" type="text" value="4.6"/>
Festplattenkapazität in GB	<input style="width: 90%;" type="text" value="1000"/>

Datenausgabe

```

-----
ID: 1
Prozessorgeschwindigkeit: 4.6 Ghz
Arbeitsspeicher: 16 GB
Festplattenkapazität: 1000 GB
Leistungsindex: 73600
          
```

Datensatzsteuerung

1 / 1

Datenbankzugriff

Nachdem jetzt alle Vorbereitungen getroffen sind, kann nun der DB-Zugriff programmiert werden.

Grundsätzlich gilt folgender Ablauf für einen Datenbankzugriff

1. Herstellen der DB-Verbindung (Abholen eines Connection Objekts) über eine DB-Connector-Klasse. Diese Klasse kennt nur das DBMS und die Datenbank und kann in anderen Anwendungen, die auf das gleiche DBMS zugreifen wiederverwendet werden.
2. Erstellen eines (SQL-)Statements in einer DBAdapter-Klasse. Diese Klasse kennt die Fachklassen der Anwendung und hält ein Objekt der DB-Connector-Klasse um auf die Datenbank zugreifen zu können.
3. Senden des Statements in der DBAdapter-Klasse.
4. Entgegennehmen des Ergebnisobjekts (ResultSet, Recordset, Dataset o.ä.) in der DBAdapter-Klasse.
5. Auswerten des Ergebnisses (Auslesen der Ergebnisdatensätze aus dem ResultSet.) in der DBAdapter-Klasse.
6. Freigeben des ResultSets in der DBAdapter-Klasse.

DBConnector-Klasse anlegen

1. Legen Sie die Klasse ‚DBConnector‘ im Ordner ‚classes‘ an Diese Klasse soll alle Informationen verwalten, die ganz speziell auf MySQL zugeschnitten sind. Daher muss diese Klasse von der Datenbankzugriffsklasse ‚mysqli‘ erben

```
class DBConnector extends mysqli {
```

2. DBMS-Zugriffsparemeter bereitstellen

```
// Attribute mit den Parametern für die DB-Verbindung
private $user = "root";
private $pass = "";
private $dbHost = "localhost";
```

3. Objekt-Referenz für die Verbindung bereitstellen.

```
// eine statische Variable für die DB-Verbindung bereitstellen
private static $instance = null;
```

Erläuterung:

static es gibt nur einen Speicherplatz im RAM für alle Objekte dieser Klasse, bzw. dieser Speicherplatz existiert auch ohne Objekte.

4. **getConnector-Methode** (DB-Verbindung öffnen):

Die getConnector-Methode öffnet die Datenbankverbindung und gibt eine Referenz auf das DB-Verbindungsobjekt zurück, mit dem dann Abfragen an die Datenbank gesendet werden können.

Erstellen Sie auch den Kommentar mit **/** + Entertaste**. Ergänzen Sie auf jeden Fall den Datentyp, da NetBeans daran seine Quellcodeergänzung orientiert.

```
/**
 * Diese Methode gibt die Referenz auf ein Objekt von sich selbst
 * zurück. Falls das Objekt existiert, wird die Referenz auf das
 * existierende Objekt zurückgegeben, sonst wird ein neues Objekt erzeugt.
 * @return DBConnector
 */
public static function getConnector() {
    if (!self::$instance instanceof self) {
        self::$instance = new self;
    }
    return self::$instance;
}
```

Erläuterung:

self:: Aufruf von Elementen der eigenen Klasse, in diesem Fall der Klassenvariable **\$instance**.

Diese Methode funktioniert allerdings noch nicht alleine, sondern benötigt einen Konstruktor im DBConnector, dieser baut eigentlich die Verbindung zur Datenbank auf.

5. Privater Konstruktor

```
/**
 * Privater Konstruktor.
 * Ein privater Konstruktor ist eigentlich sinnlos, da von außerhalb kein
 * Objekt erstellt werden kann. Sinnvoll ist er nur durch die
 * 'Factory'-Methode 'getConnection()', die ein Objekt der eigenen Klasse
 * erstellt und den privaten Konstruktor verwenden kann.
 *
 * Im Konstruktor werden die DBMS und DB-Parameter an die Datenbank
 * gesendet und dadurch eine Verbindung aufgebaut. Wenn dies nicht möglich
 * ist, wird eine Fehlermeldung ausgegeben.
 */
private function __construct() {
    parent::__construct($this->dbHost,
                       $this->user,
                       $this->pass);

    if (mysqli_connect_error()) {
        exit('Verbindungsfehler(' . mysqli_connect_errno() . ') '
            . mysqli_connect_error());
    }
    parent::set_charset('utf-8');
}
```

Erläuterungen:

- `__construct` Definition eines Konstruktors in PHP (Achtung, vorne **zwei** Unterstriche)
- `parent::` Adressierung der Oberklasse, in diesem Fall des parametrisierten Konstruktors der Klasse `mysqli`, die in PHP die DB-Verbindung herstellt.
- `mysqli_connect_error()` Angabe, ob ein Fehler beim Verbindungsaufbau aufgetreten ist und welcher.
- `mysqli_connect_errno()` Ausgabe der Fehlernummer (**error numero**) des aufgetretenen Verbindungsfehlers.

DB-Zugriffsklasse ,DBAdapterPCs' für die Fachklasse PC

- Erstellen Sie eine neue Klasse ,**DBAdapterPCs**' im Ordner ,classes'. In dieser DB-Zugriffsklasse, die auf die Datenbank und die Anwendung zugeschnitten ist, werden die SQL-Statements erzeugt, abgesetzt und die Ergebnisse (ResultSets) ausgewertet. Das bedeutet, dass für jede Abfrage eine eigene Methode hergestellt wird, die die Funktionalität im Namen trägt, falls nötig Parameter entgegennimmt und das Ergebnis als Objekt einer Klasse zurückgibt, die in der Anwendung weiterverwendet werden kann.
- Attribute für die Verbindung zur Datenbank, den Konstruktor und einen Hinweis auf die Klasse ,PC' erstellen:

```
private $connector=null;
private $dbName = "pcverwaltung";
/* @var $pc PC */

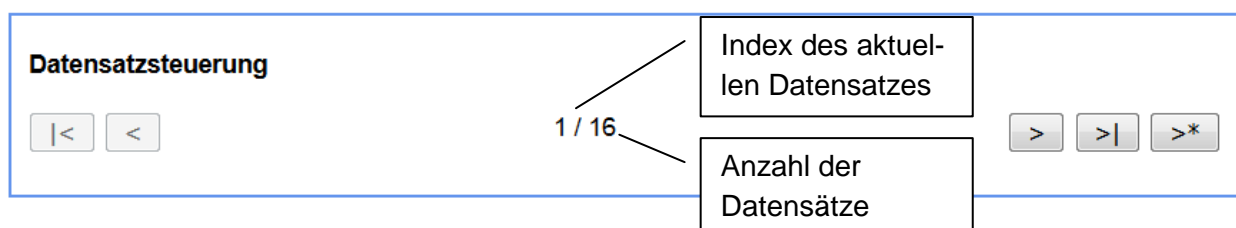
public function __construct() {
    $this->connector= DBConnector::getConnector();
    $this->connector->select_db($this->dbName);
}
```

Erläuterungen:

Im Konstruktor der Klasse muss ein Objekt der DBConnector-Klasse von der ,getConnector'-Methode abgeholt werden und die Datenbank für dieses Objekt ausgewählt werden.

Nun können Methoden programmiert werden, die Datenbankaktionen ausführen.

Die Datensatzsteuerung in der Web-Anwendung basiert auf der Anzeige von jeweils einem Datensatz an einem bestimmten Index und der Anzeige der absoluten Menge der Datensätze.



Dazu werden im DBAdapterPCs die zwei Methoden ,get_PC_by_index(index:int):PC‘und ,getAnzahlPC():int‘ benötigt.

3. Methode ,get_PC_by_index(index:int):PC‘

Diese Methode wird beispielhaft für den DB-Zugriff gezeigt.

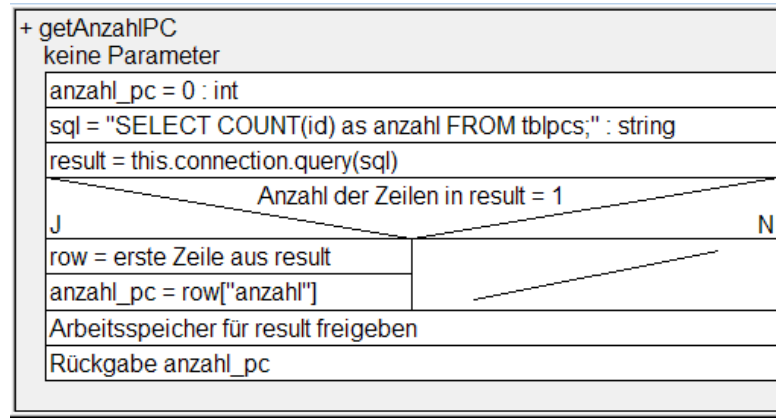
```
/**
 * Entnimmt den PC mit dem angegebenen Index aus der Datenbank.
 * @param integer $index
 * @return PC
 */
public function get_PC_by_index($index) {
    $pc=new PC();
    $anzahl_pc= $this->getAnzahlPC();
    if ($index >=0 && $index <$anzahl_pc) {
        $sql="select id, hdd, takt,ram from tblpcs "
            . "limit ".$index.",1;";
        $result= $this->connector->query($sql);
        if($result->num_rows==1){
            $row = mysqli_fetch_array($result);
            $pc->setId($row["id"]);
            $pc->setHdd($row["hdd"]);
            $pc->setTakt($row["takt"]);
            $pc->setRam($row["ram"]);
        }
        mysqli_free_result($result);
    }
    return $pc;
}
```

Erläuterungen:

- .. limit “.\$index.“,1;“ begrenzt die Abfrage auf einen Datensatz ab dem angegebenen Index.
 - ..->query(\$sql) schickt die Abfrage an die Datenbank, gibt das Ergebnis zurück und legt es als Tabelle mit Spaltenüberschriften in \$result ab.
 - mysqli_fetch_array(..) entnimmt dem Ergebnis (\$result) den ersten Datensatz (eine Zeile / Row) – bei jedem weiteren Aufruf den nächsten Datensatz – und gibt ihn als assoziatives Array zurück. Assoziativ bedeutet, die Werte des Arrays werden nicht mit ihrem Index angesprochen, sondern mit den Bezeichnungen der Spalten der Ergebnistabelle.
 - row[“id“] Enthält den Wert, den die Abfrage für die ID des PCs ermittelt hat.
 - mysqli_free_result(...) Gibt den Speicher, der für die Ergebnistabelle reserviert wurde wieder frei.
- Fügen Sie auch zu dieser Methode den Kommentar mit ,/** + Entertaste‘ hinzu, um sich die Arbeit später zu erleichtern.

Damit diese Methode funktioniert ist in der Klasse DBAdapterPCs auch die Methode ,getAnzahlPC():int‘ zu implementieren, die vom Aufbau (siehe folgendes Struktogramm) ähnlich ist.

4. Methode getAnzahlPC()



Nach so viel Vorarbeit, sollte endlich etwas auf der Oberfläche zu sehen sein, zeigen Sie den ersten Datensatz an!

Jetzt muss in der Datei ‚index.php‘ gearbeitet werden.

Binden Sie als erstes alle Klassendateien ein:

```
require './classes/PC.php';
require './classes/DBConnector.php';
require './classes/DBAdapterPCs.php';
```

Der Aufruf der **GUI bleibt am Ende stehen**

Dann erzeugen Sie eine Variable für den DB-Zugriff d.h. eine Variable von DBAdapterPCs:

```
// DB-Zugriff initialisieren
$dbAdapter=new DBAdapterPCs();
```

Ersetzen Sie die Anzahl der PCs (\$anzahl) und den neu erstellten PC (\$pc) durch Werte aus der Datenbank mit folgendem Quellcode:

```
$dbAdapter = new DBAdapterPCs();

// Arbeitsvariablen in einen vordefinierten Zustand versetzen
$index=0;
$anzahl=$dbAdapter->getAnzahlPC();

$pc=$dbAdapter->get_PC_by_index($index);
$pc->setHdd(1000);
$pc->setId(1);
$pc->setRam(16);
$pc->setTakt(4.6);

// Datensatzzeiger erzeugen
$ds_anzeiger=($index+1)." / ".$anzahl;

require './gui_PCVerwaltung.php';
```


Datensatzsteuerung über den Index und die Namen der Buttons

Wenn ein Button gedrückt wird, muss jedes Mal der Index des aktuellen Datensatzes an die index.php übertragen werden. Um diesen für den User unsichtbar auf der HTML-Seite vorzuhalten, wird im Formular ein verstecktes Steuerelement, ein hidden Input vorgehalten, das den aktuellen Index enthält.

```
<form action="index.php" method="get">
    <input type="hidden" name="hdIndex" value="<?php echo $index; ?>" />
```

Beim ersten Aufruf der Webseite, also der index.php ist diese Information nicht vorhanden, daher muss zuerst abgefragt werden, ob das Element gesetzt ist. Dies geschieht, nachdem die Arbeitsvariablen gesetzt sind und bevor der Datensatzzeiger erzeugt wird.

```
// Arbeitsvariablen in einen vordefinierten Zustand versetzen
$index=0;
$anzahl=$dbAdapter->getAnzahlPC();
$pc=$dbAdapter->get_PC_by_index($index);

if (isset($_GET['hdIndex']) ) {
    // Übernahme der Daten von der sendenden Seite
    $index=$_GET['hdIndex'];
}

// Datensatzzeiger erzeugen
$ds_anzeiger=($index+1)." / ".$anzahl;
```

Nun kann ausgewertet werden, ob einer der Knöpfe gedrückt wurde:

Dazu legen Sie zu Beginn der **index.php** eine Variable \$message an. Diese Variable wird am Anfang der **gui_PCVerwaltung.php**, ausgegeben.

```
<body>
    <?php echo $message; ?>
```

Funktional stark unterschiedlich sind die Knöpfe ‚Übernehmen‘, ‚Löschen‘ und ‚Neuer Datensatz‘, wohingegen die Knöpfe für den ersten, den letzten, den vorherigen und den nachfolgenden Datensatz fast gleich sind und deshalb zusammengefasst werden können.

Der Quellcode in der index.php sieht damit so aus:

```
if (isset($_GET['hdIndex'])) {
    // Übernahme der Daten von der sendenden Seite
    $index=$_GET['hdIndex'];

    if (array_key_exists("btn_uebernehmen", $_GET)) {
        $message="einfügen oder ändern";
    } else if (array_key_exists("btn_loeschen", $_GET)) {
        $message="löschen";
    } else if (array_key_exists("btn_new", $_GET)) {
        $message="neuer Datensatz";
    } else{
        if (array_key_exists("btn_first", $_GET)) {
            $message="erster Datensatz";
        } else if (array_key_exists("btn_prev", $_GET)) {
            $message="Vorheriger Datensatz";
        } else if (array_key_exists("btn_last", $_GET)) {
            $message="Letzter Datensatz";
        } else if (array_key_exists("btn_next", $_GET)) {
            $message="Nächster Datensatz";
        }
    }
}
```

Beim Testen erscheint nun am Anfang der Seite die jeweilige Nachricht.

Beim Durchblättern muss der Index der Situation angepasst werden und der PC für den entsprechenden Index aus der Datenbank ausgelesen werden:

```
if (array_key_exists("btn_first", $_GET)) {
    //$message="erster Datensatz";
    $index=0;
} else if (array_key_exists("btn_prev", $_GET)) {
    //$message="Vorheriger Datensatz";
    $index--;
} else if (array_key_exists("btn_last", $_GET)) {
    //$message="Letzter Datensatz";
    $index=$anzahl-1;
} else if (array_key_exists("btn_next", $_GET)) {
    //$message="Nächster Datensatz";
    $index++;
}
$pc=$dbAdapter->get_PC_by_index($index);
```

Nun ist es allerdings möglich, dass man mit den Buttons ‚vor‘ und ‚zurück‘ die Grenzen ‚1‘ und ‚Anzahl‘ übertreten kann. Verhindern kann man dieses entweder mit Verzweigungen, die verhindern, dass der Index kleiner 0 werden kann oder größer als (anzahl -1), oder man hebt die Funktion der Steuerelemente nach Bedarf auf. Diese Variante ist im Folgenden beschrieben.

Buttons ausgrauen (disabled) und wieder funktionstüchtig machen

Im Verlauf der Anwendung sind alle Buttons, bis auf den Button ‚DS übernehmen‘ und den Button neu manchmal nicht funktionstüchtig. Daher ist es sinnvoll, alle Buttons zu Beginn des Skripts auszuschalten und je nach Situation nur bestimmte Buttons einzuschalten.

Das Ausschalten der Buttons wird durch das Attribut `disabled="disabled"` bewirkt. Da das Ein- und Ausschalten von der ‚index.php‘ dynamisch gesteuert werden muss, legen Sie dazu direkt nach den ‚require‘-Anweisungen mehrere Variablen an.

```
// Literale (String-Konstanten) für enabling/disabling buttons
$disabled="disabled=\"disabled\"";
$enabled="";

/* Alle Buttons, die ihren Zustand ändern, werden standardmäßig
 * disabled.
 */
$btnNextIsEnabled=$disabled;
$btnFirstIsEnabled=$disabled;
$btnLastIsEnabled=$disabled;
$btnPrevIsEnabled=$disabled;
$btnLoeschenIsEnabled=$disabled;
```

Alle veränderlichen Buttons müssen dann einen kleinen PHP-Abschnitt erhalten, der den Inhalt der Variablen ausgibt, hier gezeigt am Beispiel des ‚btnLoeschen‘:

```
<input type="submit"
      name="btn_loeschen"
      value="DS löschen"
      <?php echo $btnLoeschenIsEnabled; ?>
/>
```

Damit sollte die Oberfläche jetzt so aussehen:

PC-Verwaltung

Dateneingabe

ID	<input type="text" value="1"/>
Arbeitsspeicher in GB	<input type="text" value="4"/>
Prozessortakt in GHz	<input type="text" value="2.4"/>
Festplattenkapazität in GB	<input type="text" value="500"/>

Datenausgabe

```

ID: 1
Prozessorgeschwindigkeit: 2.4 Ghz
Arbeitsspeicher: 4 GB
Festplattenkapazität: 500 GB
Leistungsindex: 4800
          
```

Datensatzsteuerung

1 / 16

Jetzt müssen die notwendigen Buttons der Datensatzsteuerung wieder eingeschaltet werden, bevor die GUI in die ‚index.php‘ eingebunden wird.

```
// Datensatzzeiger erzeugen
$ds_anzeiger=($index+1)." / ".$anzahl;

if ($index>0) {
    $btnFirstIsEnabled=$enabled;
    $btnPrevIsEnabled=$enabled;
}

if ($index<$anzahl-1) {
    $btnLastIsEnabled=$enabled;
    $btnNextIsEnabled=$enabled;
}

require './gui_PCVerwaltung.php';
```

Der Button ‚DS Löschen‘ soll funktionieren, wenn ein vorhandener Datensatz angezeigt wird:

```
if ($index>=0 && $index<$anzahl) {
    $btnLoeschenIsEnabled=$enabled;
}
```

Dieser Quellcode muss auch noch vor der Einbindung der GUI erstellt werden.

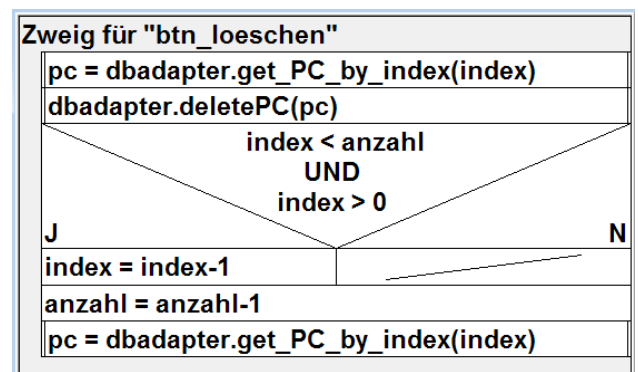
Die Funktionen ‚Neuer Datensatz‘, ‚DS Löschen‘ und ‚DS übernehmen‘

Im Zweig für ‚Neuer Datensatz‘ muss der Index auf die Anzahl der Datensätze gesetzt und ein neuer leerer PC erzeugt werden.

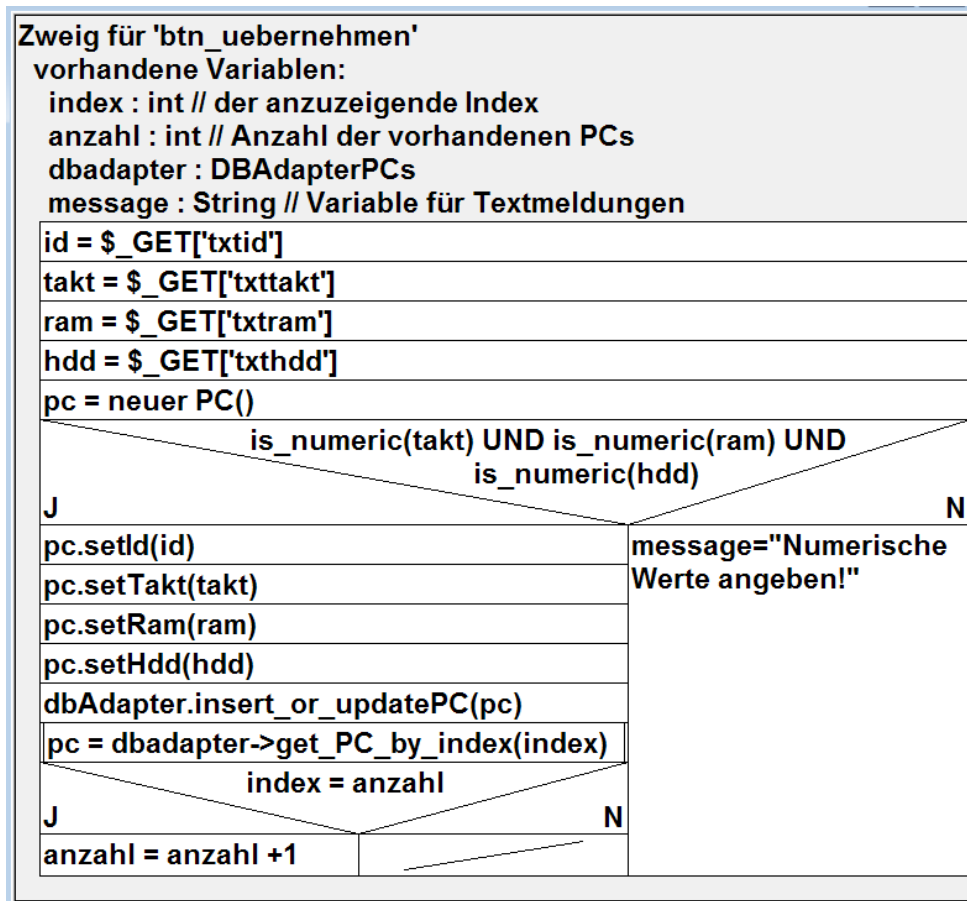
Für den Zweig ‚DS Löschen‘ muss erst die Methode ‚+ deletePC(pc:PC)‘ in der Klasse DBAdapterPCs programmiert werden (Erstellen Sie erst den Methodenmantel, dann den Kommentar und dann den Quelltext, der die Funktionalität herstellt).

```
/**
 * Löscht einen PC aus der Datenbank.
 * @param PC $pc
 */
public function deletePC($pc) {
    $sql="delete from tblpcs where id=".$pc->getId().";";
    $this->connector->query($sql);
}
```

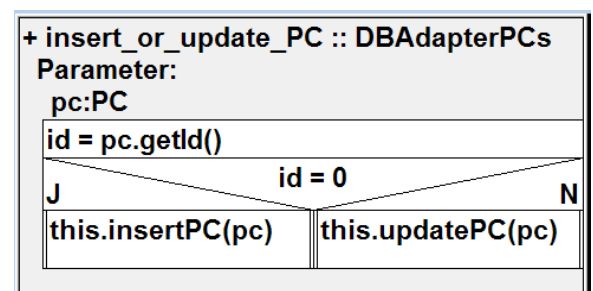
Dann kann nebenstehendes Struktogramm implementiert werden:



Der if-Zweig für btn_übernehmen in der ‚index.php‘ kann folgendermaßen implementiert werden.



Die Methode ‚+ insert_or_updatePC(pc:PC)‘ im **DBAdapterPCs** kann nach nebenstehendem Struktogramm implementiert werden.



Für den if-Zweig ‚DS übernehmen‘ in der ‚index.php‘ müssen die Methoden ‚- insertPC(pc:PC)‘ und ‚- updatePC(pc:PC)‘ in der Klasse **DBAdapterPCs** (s.u.) programmiert werden.

```
/**
 * Fügt der Tabelle tblpcs den übergebenen PC hinzu.
 * @param PC $pc
 */
private function insertPC($pc) {
    $sql="insert into tblpcs (hdd,takt,ram) "
        . "values (". $pc->getHdd() .", "
        . $pc->getTakt() .", ". $pc->getRam() .")";
    $this->connector->query($sql);
}

/**
 * Ändert den übergebenen PC in der Datenbank.
 * @param PC $pc
 */
private function updatePC($pc) {
    $sql="update tblpcs "
        . "set hdd=". $pc->getHdd() .", "
        . "    takt=". $pc->getTakt() .", "
        . "    ram=". $pc->getRam() . " "
        . "where id=". $pc->getId() .";";
    $this->connector->query($sql);
}
```

Nun sollte die Anwendung voll funktionstüchtig sein.