# Datenbank-basierte Webserver

## Dateneingabe über Web-Formulare

Karl Riedling
Institut für Sensor- und Aktuatorsysteme

## Dateneingabe über Web-Formulare

- Übergabe von Parametern an den Webserver
- Elemente von Formularen
- Übernahme der Parameter in ein PHP-Programm
- Spezielle Aufgabenstellungen



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Dateneingabe über Web-Formulare

- **Übergabe von Parametern an den Webserver**
- Elemente von Formularen
- Übernahme der Parameter in ein PHP-Programm
- Spezielle Aufgabenstellungen



## Übergabe von Parametern an den Webserver

- Direkte Übergabe von Parametern in einer URL
- Übergabe aus Web-Formularen
- Parameter-Übergabe aus (PHP-) Programmen



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Übergabe von Parametern an den Webserver

- **Direkte Übergabe von Parametern in einer URL**
- Übergabe aus Web-Formularen
- Parameter-Übergabe aus (PHP-) Programmen



## Direkte Übergabe von Parametern

- „URL“ – *Uniform Resource Locator* = Unterbegriff von „URI“ (*Uniform* (fälschlich manchmal „*Universal*”) *Resource Identifier*); gibt die Adresse einer Webseite an
- In vielen Fällen sind URIs und URLs faktisch das Gleiche (HTML-Standard: „URI“, allgemeiner Sprachgebrauch: „URL“)
- Je nach Quelle werden URIs und URLs im Deutschen männlich oder weiblich gebraucht. (In den aktuellen Wikipedia-Artikeln sind sie weiblich – „*die* Adresse“).



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Direkte Übergabe von Parametern

- Parameter einer URL (*Query String*): Der Teil der Adresse, der hinter einem „?“ steht:
  `http://www2.isas.tuwien.ac.at/lva.php?lva=366.026`
- Eine beliebige Anzahl von Parametern kann durch „&“ getrennt übergeben werden:
  `phone.php?name=Meier&vorname=Johann&ort=Wien`
- Der Webserver trennt die Parameter in Name-Wert-Paare und übergibt diese an das (PHP-) Programm



## Direkte Übergabe von Parametern

- URLs wie die auf der vorigen Seite dargestellten können auch (in `<a href=…>`-*Tags*) auf einer Webseite vorkommen
- Damit können dynamische Webseiten über Links (auch von statischen HTML-Seiten aus) aufgerufen werden



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Direkte Übergabe von Parametern

- Vorsicht: „&“ ist das Einleitungs-Zeichen für HTML-*Character Entities*; Parameter-Namen in URLs auf einer Webseite (z.B. in einem `<a href=…>`-*Tag*) *könnten* vom Browser als *Character Entities* interpretiert werden. Abhilfe: statt „&“ *Character Entity* „`&`“ verwenden:
  `<a href="phone.php?name=Meier&vorname=Johann">`
- In PHP kann die Umwandlung von „&“ in „`&`“ durch die Standard-Funktion `htmlentities()` erfolgen
- Das alternativ vorgeschlagene Parameter-Trennzeichen „`;`“ funktioniert mit Apache in der Standard-Konfiguration *nicht*!



## Direkte Übergabe von Parametern

- Zulässige Zeichen in URLs:
  - Druckbare US-ASCII-Zeichen mit Ausnahme von reservierten und „unsicheren“ Zeichen
  - Alle anderen sowie reservierte und „unsichere “ Zeichen durch „%<nn>“ (<nn> = hexadezimaler Zeichen-Code) ersetzen, z.B. Leerzeichen = „`%20`“
  - In PHP ist dies unter Verwendung der Standard-PHP-Funktion `urlencode()` möglich



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Direkte Übergabe von Parametern

- Reservierte Zeichen in URLs:
  - Haben spezielle Bedeutung in einer URL oder sind dort verboten

| | | | |
|---|---|---|---|
| Leerzeichen | %20 | : | %3A |
| ; | %3B | @ | %40 |
| / | %2F | = | %3D |
| ? | %3F | & | %26 |



## Direkte Übergabe von Parametern

- „Unsichere“ Zeichen:
  - Können je nach Kontext Fehlinterpretationen verursachen

| | | | | | |
|---|---|---|---|---|---|
| < | %3C | % | %25 | ~ | %7E |
| > | %3E | { | %7B | [ | %5B |
| " | %22 | } | %7D | ] | %5D |
| # | %23 | \| | %7C | ` | %60 |
| ^ | %5E | \ | %5C | | |



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Übergabe von Parametern an den Webserver

- Direkte Übergabe von Parametern in einer URL
- **Übergabe aus Web-Formularen**
- Parameter-Übergabe aus (PHP-) Programmen



## Parameter-Übergabe aus Web-Formularen

- Zwei grundsätzliche Methoden
  - *Get*-Methode: Der Browser übergibt die Daten aus dem Web-Formular in einer URL mit Parametern an den Webserver:
    `phone.php?name=Meier&vorname=Johann&ort=Wien`
  - *Post*-Methode: Zuerst Aufruf der (aktiven) Ziel-Seite, dann Parameter-Übergabe in einem zweiten Übertragungsschritt



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Parameter-Übergabe aus Web-Formularen

- *Get*-Methode:
  - + Einfach und schnell (nur *ein* Übertragungsschritt)
  - + Leicht testbar (die Parameter erscheinen in der URL-Zeile des Browsers für die Zielseite)
  - + Bei Verwendung anderer Server-seitiger Software als PHP (z.B. bei Aufruf von CGI-Programmen) leichter implementierbar



## Parameter-Übergabe aus Web-Formularen

- *Get*-Methode:
  - - Nur für einfache Formulare mit wenigen kurzen Parametern geeignet (Browser- und Server-seitige Begrenzungen – z.B.: Apache: 8 KBytes) (Vorsicht: Diese Limits sind oft nicht explizit spezifiziert!)
  - - Weniger sicher – Parameter scheinen im Klartext auf, nicht nur in der URL-Zeile, auch in Server-Logfiles! (Vorsicht bei Passwort-Übergabe!)



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Parameter-Übergabe aus Web-Formularen

- *Post*-Methode:
  - Für beliebig komplexe und umfangreiche Daten (einschließlich Datei-*Upload*) geeignet
  - Sicherer – Parameter sind nicht Teil der URL und scheinen daher nirgendwo auf



## Parameter-Übergabe aus Web-Formularen

- *Post*-Methode:
  - Aufwändiger (zwei Übertragungsschritte)
  - Bei anderer Server-seitiger Software als PHP (z.B. bei Aufruf von CGI-Programmen) schwieriger zu implementieren
  - Nur bei Datenübergabe aus Web-Formularen (nicht bei einfachen Links) realisierbar
  - Schwieriger zu testen: Parameter eines Seitenaufrufs sind nachträglich nicht mehr eruierbar



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Parameter-Übergabe aus Web-Formularen

- *Get*- oder *Post*-Methode:
  - Wenn nicht speziell angegeben, wählt der Browser die ihm am geeignetsten erscheinende Methode
  - Unter PHP sind beide Methoden (unter bestimmten Voraussetzungen sogar mit gleichem Programmcode) anwendbar
  - Die gleiche PHP-Seite kann in diesem Fall gleichermaßen mit *Get* und mit *Post* aufgerufen werden



## Übergabe von Parametern an den Webserver

- Direkte Übergabe von Parametern in einer URL
- Übergabe aus Web-Formularen
- **Parameter-Übergabe aus (PHP-) Programmen**



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Parameter-Übergabe aus PHP-Programmen

- Server-seitige Scripts (z.B. PHP-Programme) können je nach Konfiguration des Servers auch Seiten auf fremden Servern mit Parametern aufrufen:
  - Aufruf einer URL mit *Get*-Parametern: Unter PHP bei entsprechender Server-Konfiguration mit Dateisystem-Funktionen möglich
  - Aufruf einer URL mit *Get*- oder *Post*-Parametern (einschließlich erweiterter Funktionalität wie Datei-Upload): mit cURL



## Parameter-Übergabe aus PHP-Programmen

- cURL („Client for URLs“ oder „Curl URL Request Library“): Programmbibliothek, die es einem Server-seitigen Programm ermöglicht, wie ein Webbrowser zu agieren.
- Unter PHP dafür *curl-Extension* erforderlich
- Anwendungen:
  - Einbinden fremder Seiteninhalte in die eigene Website
  - Kommunikation zwischen Servern
  - Typisch: Zahlungs-Funktionen (z.B. PayPal)



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Parameter-Übergabe aus PHP-Programmen

- Ziel-URL, Aufruf-Parameter und Aufruf-Modus (z.B. HTTPS) werden durch Funktionsaufrufe registriert und der Aufruf abgesetzt
- Die Antwort des Ziel-Servers steht als Ergebnis einer Funktion zur Verfügung und kann in den eigenen Programmcode eingebaut werden
- Vorteil von cURL: Auch Aufrufe mit *Post*-Parametern und Datei-Uploads sind möglich



## Dateneingabe über Web-Formulare

- Übergabe von Parametern an den Webserver
- **Elemente von Formularen**
- Übernahme der Parameter in ein PHP-Programm
- Spezielle Aufgabenstellungen



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Elemente von Formularen

- `<form>` – Definition eines Formulars
- `<input>` – Textfelder, *Radio Buttons*, *Checkboxes* und Standard-Schaltflächen
- `<select>` – Klapplisten und Listenfelder
- `<textarea>` – mehrzeilige Textfelder
- `<button>` – Benutzerdefinierte Schaltflächen



## Elemente von Formularen

- **`<form>` – Definition eines Formulars**
- `<input>` – Textfelder, *Radio Buttons*, *Checkboxes* und Standard-Schaltflächen
- `<select>` – Klapplisten und Listenfelder
- `<textarea>` – mehrzeilige Textfelder
- `<button>` – Benutzerdefinierte Schaltflächen



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <form>-*Tag*

- Formulare sind auf einer HTML-Seite zwischen <form> und </form>-*Tags* „eingeklammert“
- Alle Formularelemente zwischen diesen *Tag*s werden als zusammengehörig betrachtet und gemeinsam (beim Betätigen eines *Submit-Buttons*) an den Server geschickt



## Der <form>-*Tag*

- Außer Formularelementen darf beliebiger HTML-Inhalt zwischen den <form>- und </form>-*Tag*s stehen
- Auf einer HTML-Seite können beliebig viele <form> – </form>-Paare vorkommen
- <form> – </form>-Paare dürfen einander aber nicht überlappen



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <form>-*Tag*

- Beispiel:

```
<form name="test" id="test" action="formtest.php"
method="post">

<!-- Beliebiger HTML-Inhalt, einschließlich
Formular-Elemente -->

</form>
```



## Der <form>-*Tag*

- Attribute in einem <form>-*Tag*:
  - name, id: Name des Formulars; optional. Ist erforderlich, um in benutzerseitigem JavaScript-Code das Formular identifizieren zu können. name und id sollten, wenn beide verwendet, den gleichen Wert haben.
  - action: Gibt Namen der aktiven Webseite an, die beim Absenden des Formulars aufgerufen wird; unbedingt erforderlich!
  - method: „get“ oder „post“; optional



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <form>-*Tag*

- Optionale Attribute in einem <form>-*Tag*:
  - enctype (*encoding type*): Gibt an, wie Formularinhalte für die Übertragung zum Server codiert werden sollen ➔ Abschnitt „Spezielle Aufgabenstellungen“
  - target: Erlaubt Angabe des Ziel-*Frames*, in dem die unter „action“ angeführte Seite angezeigt werden soll
  - JavaScript-Event Handler ➔ Abschnitt „Client-seitige Operationen unter Verwendung von JavaScript“



## Der <form>-*Tag*

- Weitere optionale Attribute in einem <form>-*Tag*: ➔ Literatur über HTML



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Elemente von Formularen

- `<form>` – Definition eines Formulars
- **`<input>` – Textfelder, *Radio Buttons*, *Checkboxes* und Standard-Schaltflächen**
- `<select>` – Klapplisten und Listenfelder
- `<textarea>` – mehrzeilige Textfelder
- `<button>` – Benutzerdefinierte Schaltflächen



## Der `<input>`-*Tag*

- Struktur eines `<input>`-*Tag*s:
  `<input type="..." name="..." ...>`
- Bei allen `<input>`-*Tag*s ist das Attribut „`type="..."`“ unbedingt erforderlich; bei allen außer Bildern und *Submit*- sowie *Reset*-Schaltflächen auch das Attribut „`name="..."`“
- Der Wert von `name="..."` gibt den Namen an, unter dem der Inhalt des Formularelements für die Zielseite zugänglich ist



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- Mit <input>-*Tag*s definiert:
  - (Einzeilige) Textfelder (type="text")
  - Felder für Passwort-Eingabe (type="password")
  - *Radio Buttons* (type="radio")
  - *Checkboxes* (type="checkbox")
  - Datei-Upload (type="file")
  - Verborgene Felder (type="hidden")
  - Bilder (type="image")



## Der <input>-*Tag*

- Mit <input>-*Tag*s definiert:
  - Submit-Schaltflächen (type="submit")
  - Reset-Schaltflächen (type="reset")
  - Benutzerdefinierte Schaltflächen (type="button")



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- Mit <input>-*Tag*s definiert:
  - **(Einzeilige) Textfelder (type="text")**
  - Felder für Passwort-Eingabe (type="password")
  - *Radio Buttons* (type="radio")
  - *Checkboxes* (type="checkbox")
  - Datei-*Upload* (type="file")
  - Verborgene Felder (type="hidden")
  - Bilder (type="image")



## Der <input>-*Tag*

- (Einzeilige) Textfelder (type="text"):
  - Eingabe von beliebigem Text oder Zahlenwerten
  - Beispiel:

```
Textfeld:<br>
<input type="text" name="textfeld"
  value="Textfeld" size="20"
  maxlength="32">
```

© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- (Einzeilige) Textfelder (`type="text"`):
  - Optionale Attribute (➔ HTML-Dokumentation):
    - `value="..."`: Voreingestellter Inhalt des Textfelds
    - `size="..."`: Breite des Textfelds in Zeichen (alle neueren Browser verwenden für Textfelder einen proportionalen SansSerif-Font → nicht alle Zeichen gleich breit und daher kein gutes Maß; statt `size` besser *Styles* verwenden!)



## Der <input>-*Tag*

- (Einzeilige) Textfelder (`type="text"`):
  - Optionale Attribute (➔ HTML-Dokumentation):
    - `maxlength="..."`: Maximale Anzahl von Zeichen, die in das Textfeld eingegeben werden können (Anpassen an Feldgröße in der Datenbank)



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- (Einzeilige) Textfelder (type="text"):
  - An die aufgerufene Seite übergeben wird der Inhalt des Textfeldes zum Zeitpunkt des Absendens des Formulars
  - Ein Parameter mit dem Namen des Textfeldes (name="...") wird unabhängig vom Inhalt *immer* übergeben
  - Bei einem leeren Textfeld wird ein leerer String ("") an die aufgerufene Seite übergeben



## Der <input>-*Tag*

- Mit <input>-*Tag*s definiert:
  - (Einzeilige) Textfelder (type="text")
  - **Felder für Passwort-Eingabe (type="password")**
  - *Radio Buttons* (type="radio")
  - *Checkboxes* (type="checkbox")
  - Datei-*Upload* (type="file")
  - Verborgene Felder (type="hidden")
  - Bilder (type="image")



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- Passwort-Eingabe (`type="password"`):
  - Inhalt wird durch einen „*“ pro Eingabezeichen angezeigt, sonst wie Textfeld
  - Beispiel:

```
Password:<br>
<input type="password" name="passwort"
  value="topsecret" size="20"
  maxlength="16">
```

## Der <input>-*Tag*

- Passwort-Eingabe (`type="password"`):
  - Optionale Attribute (➔ HTML-Dokumentation):
    - `value="..."`: Voreingestelltes Passwort – sollte in der Praxis nicht gesetzt sein!
    - `size="..."`: Breite des Passwort-Feldes in Zeichen (wie bei `type="text"`)
    - `maxlength="..."`: Maximale Anzahl von Zeichen, die in das Passwort-Feld eingegeben werden können (Anpassen an Feldgröße in der Datenbank)



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- Passwort-Eingabe (`type="password"`):
  - An die aufgerufene Seite übergeben wird der Inhalt des Passwort-Feldes zum Zeitpunkt des Absendens des Formulars *im Klartext*! *Secure HTTP* (HTTPS) verwenden!
  - Ein Parameter mit dem Namen des Passwort-Feldes (`name="..."`) wird unabhängig vom Inhalt *immer* übergeben
  - Bei einem leeren Passwort-Feld wird ein leerer String (`""`) an die aufgerufene Seite übergeben



## Der <input>-*Tag*

- Mit <input>-*Tag*s definiert:
  - (Einzeilige) Textfelder (`type="text"`)
  - Felder für Passwort-Eingabe (`type="password"`)
  - ***Radio Buttons*** **(`type="radio"`)**
  - *Checkboxes* (`type="checkbox"`)
  - Datei-*Upload* (`type="file"`)
  - Verborgene Felder (`type="hidden"`)
  - Bilder (`type="image"`)



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- *Radio Buttons* (type="radio"):
  - Erlauben die Auswahl von (genau) einer aus mehreren Optionen
  - Beispiel:

```
<i>Radio Buttons</i>:<br>
<input type="radio" name="radiobutton"
  value="eins" checked> Option 1 
<input type="radio" name="radiobutton"
  value="zwei"> Option 2 
<input type="radio" name="radiobutton"
  value="drei"> Option 3
```



## Der <input>-*Tag*

- *Radio Buttons* (type="radio"):
  - Beispiel:

*Radio Buttons*:
Option 1 Option 2 Option 3

  - Hinweis: *Radio Buttons* mit gleichem name-Wert werden gruppiert, d.h., sie lösen sich gegenseitig aus



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- *Radio Buttons* (`type="radio"`):
  - Optionale Attribute (➔ HTML-Dokumentation):
    - `value="..."`: Beliebiger *eindeutiger* numerischer oder String-Wert; wird bei Wahl des jeweiligen *Radio Buttons* beim *Submit* übergeben; braucht nicht mit der Beschriftung des Elements korreliert zu sein
    - `checked`: Bei jenem *Radio Button* angeben, der bei der Anzeige des Formulars als aktiv angezeigt werden soll – nur *ein* Radio Button pro Gruppe sollte dieses Attribut haben!



## Der <input>-*Tag*

- *Radio Buttons* (`type="radio"`):
  - An die aufgerufene Seite übergeben wird der `value`-Parameter des selektierten *Radio Buttons* zum Zeitpunkt des Absendens des Formulars
  - Ein Parameter mit dem Namen der *Radio Button*-Gruppe (`name="..."`) wird *nur dann* übergeben, wenn ein *Radio Button* selektiert ist!



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- *Radio Buttons* (type="radio"):
  - Betätigung des *Radio Buttons* kann mit einem <label for="...">-*Tag* erleichtert werden:
    ```
    <input type="radio" name="radiobutton"
    id="opt_1" value="eins"><label for="opt_1">
    Option 1</label>
    ```
  - *Radio Button* für Option 1 wird auch dann gesetzt, wenn eine Benutzerin / ein Benutzer auf den Text „Option 1“ klickt.
  - Funktioniert bei allen gängigen Browsern

## Der <input>-*Tag*

- Mit <input>-*Tag*s definiert:
  - (Einzeilige) Textfelder (type="text")
  - Felder für Passwort-Eingabe (type="password")
  - *Radio Buttons* (type="radio")
  - ***Checkboxes* (type="checkbox")**
  - Datei-*Upload* (type="file")
  - Verborgene Felder (type="hidden")
  - Bilder (type="image")

© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- *Checkboxes* (type="checkbox"):
  - Erlauben das Setzen je eines Boole'schen Parameters
  - Beispiel:

```
<i>Checkbox</i>:<br>
<input type="checkbox" name="box1"
  value="gesetzt"> Checkbox mit
  &quot;value&quot;-Attribut<br>
<input type="checkbox" name="box2"
  checked> Checkbox ohne
  &quot;value&quot;-Attribut
```



## Der <input>-*Tag*

- *Checkboxes* (type="checkbox"):
  - Beispiel:

*Checkbox*:
☐ Checkbox mit "value"-Attribut
☑ Checkbox ohne "value"-Attribut



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- *Checkboxes* (`type="checkbox"`):
  - Optionale Attribute (➔ HTML-Dokumentation):
    - `value="..."`: Beliebiger *eindeutiger* numerischer oder String-Wert (Default: "on"); wird bei Aktivierung der jeweiligen *Checkbox* übergeben
    - `checked`: Dieses Attribut bei jenen *Checkboxes* angeben, die beim Laden des Formulars als aktiv angezeigt werden sollen



## Der <input>-*Tag*

- *Checkboxes* (`type="checkbox"`):
  - An die aufgerufene Seite übergeben wird dann und nur dann, wenn eine Checkbox zum Zeitpunkt des Absendens des Formulars aktiviert ist, das `value`-Attribut der Checkbox
  - Wurde kein `value="..."`-Attribut angegeben, wird bei aktiver *Checkbox* `"on"` übergeben
  - Ein Parameter mit dem Namen der *Checkbox* (`name="..."`) wird nur dann übergeben, wenn diese aktiviert ist



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- *Checkboxes* (`type="checkbox"`):
  - Hinweis: HTML erlaubt die folgenden Varianten bei Vorhandensein mehrerer *Checkboxes* in einem Formular:
    - Jede *Checkbox* hat einen anderen Namen und ein beliebiges (oder kein) `value`- Attribut
    - Mehrere *Checkboxes* haben das gleiche `name`-, aber individuell verschiedene `value`- Attribute



## Der <input>-*Tag*

- *Checkboxes* (`type="checkbox"`):
  - Im zweiten Fall muss in Verbindung mit PHP als Name der *Checkbox* der Name eines Arrays angegeben werden: `name="box[]"`
  - Diese Methode ist aber nicht mit Client-seitigen JavaScript-Funktionen kompatibel und kann daher nicht verwendet werden, wenn per JavaScript auf die Checkbox zugegriffen werden soll!



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- *Checkboxes* (type="checkbox"):
  - Betätigung der *Checkbox* kann mit einem <label for="...">-*Tag* erleichtert werden:

```
<input type="checkbox" name="box1"
id="chk_box1"><label for="chk_box1"> Ich bin
eine Checkbox</label>
```

  - Zustand der *Checkbox* wird auch dann umgeschaltet, wenn eine Benutzerin / ein Benutzer auf den Text „Ich bin eine Checkbox“ klickt.
  - Funktioniert bei allen gängigen Browsern



## Der <input>-*Tag*

- Mit <input>-*Tag*s definiert:
  - (Einzeilige) Textfelder (type="text")
  - Felder für Passwort-Eingabe (type="password")
  - *Radio Buttons* (type="radio")
  - *Checkboxes* (type="checkbox")
  - **Datei-*Upload* (type="file")**
  - Verborgene Felder (type="hidden")
  - Bilder (type="image")



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- Datei-*Upload* (`type="file"`):
  - ➔ Abschnitt „Spezielle Aufgabenstellungen“



## Der <input>-*Tag*

- Mit <input>-*Tag*s definiert:
  - (Einzeilige) Textfelder (`type="text"`)
  - Felder für Passwort-Eingabe (`type="password"`)
  - *Radio Buttons* (`type="radio"`)
  - *Checkboxes* (`type="checkbox"`)
  - Datei-*Upload* (`type="file"`)
  - **Verborgene Felder (`type="hidden"`)**
  - Bilder (`type="image"`)



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- Verborgene Felder (`type="hidden"`):
  - Für die BenutzerInnen unsichtbare Übergabe von Formulardaten
  - Beispiel:

```
Verborgenes Feld:<br>
<input type="hidden" name="versteckt"
  value="unsichtbar">
```

Verborgenes Feld:



## Der <input>-*Tag*

- Verborgene Felder (`type="hidden"`):
  - Optionale Attribute (➔ HTML-Dokumentation):
    - `value="..."`: Voreingestellter Inhalt des verborgenen Feldes
  - Verborgene Felder kommen auf reinen HTML-Formularseiten selten vor (evtl. zur Identifikation des Formulars), sind aber für aktiv generierte Formularseiten wichtig (z.B. zur Übergabe einer *Session-ID* oder anderer Parameter)



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- Verborgene Felder (`type="hidden"`):
  - An die aufgerufene Seite übergeben wird der Inhalt des verborgenen Feldes zum Zeitpunkt des Absendens des Formulars
  - Ein Parameter mit dem Namen des verborgenen Feldes (`name="..."`) wird unabhängig vom Inhalt von `value="..."` *immer* übergeben (gegebenenfalls ein leerer String)



## Der <input>-*Tag*

- Mit <input>-*Tag*s definiert:
  - (Einzeilige) Textfelder (`type="text"`)
  - Felder für Passwort-Eingabe (`type="password"`)
  - *Radio Buttons* (`type="radio"`)
  - *Checkboxes* (`type="checkbox"`)
  - Datei-*Upload* (`type="file"`)
  - Verborgene Felder (`type="hidden"`)
  - **Bilder (`type="image"`)**



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- Bilder (`type="image"`):
  - Dienen zur Realisierung von serverseitigen *Clickable Maps* (oder von *Submit*-Schaltflächen auf Basis eines beliebigen Bildes)
  - Mit einem Mausklick auf das Bild wird das Formular abgeschickt
  - Beispiel:

```
<i>Clickable Map</i>:<br>
<input type="image" name="map"
  src="img/clickmap.gif">
```

## Der <input>-*Tag*

- Bilder (`type="image"`):
  - Unbedingt erforderliches Attribut:
    - `src="..."`: Pfad und Name der Bild-Datei
  - Optionale Attribute (➔ HTML-Dokumentation):
    - `name="..."`: Name des Formularelements
    - Weitere Attribute ähnlich wie bei `<img="...">`-*Tag*s

© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- Bilder (`type="image"`):
  - An die aufgerufene Seite übergeben wird der horizontale bzw. vertikale Offset der Stelle des Mausklicks (gemessen in Pixel) in einem Parameter mit dem Namen im `name`-Attribut, gefolgt von „`.x`“ bzw „`.y`“ (in unserem Beispiel „`map.x`“ bzw. „`map.y`“ (linke obere Ecke = `(1, 1)`)



## Der <input>-*Tag*

- Mit <input>-*Tag*s definiert:
  - ***Submit*-Schaltflächen (`type="submit"`)**
  - *Reset*-Schaltflächen (`type="reset"`)
  - Benutzerdefinierte Schaltflächen (`type="button"`)



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- *Submit*-Schaltflächen (`type="submit"`):
  - Abschicken der Formulardaten
  - Beispiel:

```
Abschicken:<br>
<input type="submit" name="submitbutton"
value="Abschicken">
```

## Der <input>-*Tag*

- *Submit*-Schaltflächen (`type="submit"`):
  - Optionale Attribute (➔ HTML-Dokumentation):
    - `name="..."`: Bezeichnung der Schaltfläche
    - `value="..."`: Text auf der Schaltfläche; Default (je nach Browser) : „`Fragen senden`“, „`Anfrage senden`“ o.Ä. bzw. „`Submit Query`“, „`Submit`“



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- *Submit*-Schaltflächen (`type="submit"`):
  - In einem Formular kann es beliebig viele *Submit*-Schaltflächen geben, zwischen denen bei Bedarf über das `name`- oder das `value`-Attribut unterschieden werden kann
  - In jedem Fall *muss* dazu aber ein `name`-Attribut definiert sein
  - Eine Unterscheidung zwischen mehreren *Submit*-Schaltflächen über unterschiedliche `name`-Attribute ist zweckmäßiger



## Der <input>-*Tag*

- *Submit*-Schaltflächen (`type="submit"`):
  - An die aufgerufene Seite wird der gesamte Inhalt des Formulars übergeben
  - Über JavaScript-*Event Handler* (➔ Client-seitige Operationen unter Verwendung von JavaScript) können Client-seitige Operationen z.B. zum Überprüfen der Daten vorgenommen werden



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- Mit <input>-*Tag*s definiert:
  - *Submit*-Schaltflächen (type="submit")
  - ***Reset*-Schaltflächen (type="reset")**
  - Benutzerdefinierte Schaltflächen (type="button")



## Der <input>-*Tag*

- *Reset*-Schaltflächen (type="reset"):
  - Zurücksetzen der Formulardaten auf die beim Laden des Formulars angezeigten Werte
  - Beispiel:

```
Zurücksetzen:<br>
<input type="reset" name="resetbutton"
value="Zurück">
```

Zurücksetzen:
Zurück



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- *Reset*-Schaltflächen (`type="reset"`):
  - Optionale Attribute (➔ HTML-Dokumentation):
    - `value="..."`: Text auf der Schaltfläche; Default: „`Zurücksetzen`“ bzw. „`Reset`“
  - *Reset*-Schaltflächen werden vom Browser ausgewertet; es werden keine Daten an den Server übergeben



## Der <input>-*Tag*

- Mit <input>-*Tag*s definiert:
  - *Submit*-Schaltflächen (`type="submit"`)
  - *Reset*-Schaltflächen (`type="reset"`)
  - **Benutzerdefinierte Schaltflächen (`type="button"`)**

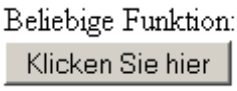


© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Der <input>-*Tag*

- Benutzerdefinierte Schaltflächen (`type="button"`):
  - Ausführung beliebiger Operationen
  - Beispiel:

```
Beliebige Funktion:<br>
<input type="button" name="customerbutton"
value="Klicken Sie hier">
```






## Der <input>-*Tag*

- Benutzerdefinierte Schaltflächen (`type="button"`):
  - Ein `name`- und ein `value`-Attribut werden benötigt (kein Default für Schaltflächen-Text!)
  - Benutzerdefinierte Schaltflächen werden vom Browser ausgewertet; es werden keine Daten an den Server übergeben
  - Über JavaScript-*Event Handler* (➔ Client-seitige Operationen unter Verwendung von JavaScript) können jedoch beliebige Operationen vorgenommen werden



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Elemente von Formularen

- `<form>` – Definition eines Formulars
- `<input>` – Textfelder, *Radio Buttons*, *Checkboxes* und Standard-Schaltflächen
- **`<select>` – Klapplisten und Listenfelder**
- `<textarea>` – mehrzeilige Textfelder
- `<button>` – Benutzerdefinierte Schaltflächen



## `<select>` – Klapplisten und Listenfelder

- Struktur eines `<select>`-Konstrukts:

```
<select name="..." ...>
  <option value="..." ...>Text1</option>
  <option value="..." ...>Text2</option>
  ...
</select>
```

- `name="..."` bestimmt den Namen des an die aufgerufene Seite übergebenen Parameters; sein Wert kommt aus dem `value="..."`-Feld der gewählten Option



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## <select> – Klapplisten und Listenfelder

- Beispiele:

```
Auswahllisten - einfache Auswahl:<br>
<select name="liste1" size="1">
<option value="0">Option Nr. 1</option>
<option value="1" selected>Option Nr. 2</option>
<!-- weitere "option"-Zeilen -->
</select><br><br>

<select name="liste2" size="4">
<option value="0">Option Nr. 1</option>
<option value="1">Option Nr. 2</option>
<option value="2" selected>Option Nr. 3</option>
<!-- weitere "option"-Zeilen -->
</select>
```



## <select> – Klapplisten und Listenfelder

- Beispiele (Fortsetzung):

Auswahllisten - einfache Auswahl:

Option Nr. 2

Option Nr. 1
Option Nr. 2
Option Nr. 3
Option Nr. 4



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## <select> – Klapplisten und Listenfelder

- Beispiele (für `method="get"`):
  - Selektierte Zeilen auf der Formularseite:

```
<select name="liste1" size="1">
<option value="1">Option Nr. 2</option>
</select><br><br>
<select name="liste2" size="4">
<option value="2">Option Nr. 3</option>
</select>
```

  - Übergebene URL mit Formularparametern:
    `formtest.php?liste1=1&liste2=2`



## <select> – Klapplisten und Listenfelder

- Der `<select>`-*Tag*
  - Das `name`-Attribut ist unbedingt erforderlich
  - Optionale Attribute (➔ HTML-Dokumentation):
    - `size="..."`: Anzahl der gleichzeitig angezeigten Zeilen; Default = 1
    - `multiple`: Mehrfachauswahl ist zulässig (sonst immer nur eine Option – wie bei *Radio Buttons*)
  - Im Default-Fall (`size="1"`) wird meist eine Klappliste angezeigt, sonst ein Listenfeld



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## <select> – Klapplisten und Listenfelder

- Der <select>-*Tag*
  - <select multiple>-Listen sind wenig benutzerfreundlich und sollten, wenn irgend möglich, vermieden werden. Alternativen:
    - Reihe von Checkboxes (bei nicht zu großer Anzahl von Optionen)
    - Mehrstufiges Hinzufügen von Optionen zu einer Auswahlliste – realisierbar sowohl in PHP als auch in JavaScript



## <select> – Klapplisten und Listenfelder

- Der <select>-*Tag*
  - Die Breite einer Klappliste oder eines Listenfelds ergibt sich aus ihrem Inhalt und kann nicht durch Attribute des <select>-*Tags* modifiziert werden – Alternative: Styles:
    ```
    <select name="liste" style="width:450px;">
    ```
  - Zeilen, die länger als die angegebene Listenbreite sind, werden je nach Browser beim Aufklappen einer Klappliste aber unterschiedlich behandelt



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## <select> – Klapplisten und Listenfelder

- <select multiple> – Beispiel:

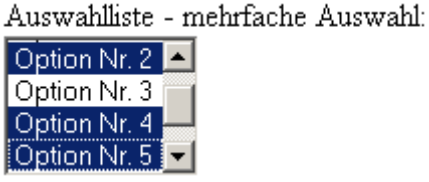
```
Auswahlliste - mehrfache Auswahl:<br>
<select multiple name="liste[]" size="4">
  <option value="0">Option Nr. 1</option>
  <option value="1" selected>Option Nr. 2</option>
  <option value="2">Option Nr. 3</option>
  <option value="3">Option Nr. 4</option>
  <option value="4">Option Nr. 5</option>
  <option value="5">Option Nr. 6</option>
</select>
```

- Bei einer PHP-Zielseite muss der Wert von „name=“ der Name eines Arrays sein!



## <select> – Klapplisten und Listenfelder

- <select multiple>-Beispiel (Fortsetzung):






© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## <select> – Klapplisten und Listenfelder

- <select multiple>-Beispiel (für method="get"):
  - Selektierte Zeilen auf der Formularseite:

```
<select multiple name="liste[]" size="4">
  <option value="1">Option Nr. 2</option>
  <option value="3">Option Nr. 4</option>
  <option value="4">Option Nr. 5</option>
  </select>
```

  - Übergebene Formularparameter:
    formtest.php?**liste[]=1**&**liste[]=3**&**liste[]=4**



## <select> – Klapplisten und Listenfelder

- Der <option>-*Tag*
  - Optionale Attribute (➔ HTML-Dokumentation):
    - value="...": Wert, der an die aufgerufene Seite übergeben wird; Default: Inhalt des <option>-*Tag*s
    - selected: Vorselektierte Auswahl
  - </option> ist unter HTML ≤ 4 (nicht unter XHTML!) optional



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Elemente von Formularen

- `<form>` – Definition eines Formulars
- `<input>` – Textfelder, *Radio Buttons*, *Checkboxes* und Standard-Schaltflächen
- `<select>` – Klapplisten und Listenfelder
- **`<textarea>` – mehrzeilige Textfelder**
- `<button>` – Benutzerdefinierte Schaltflächen



## `<textarea>` – mehrzeilige Textfelder

- Struktur eines `<textarea>`-Konstrukts:

```
<textarea name="..." ...>
  Voreingestellter Text des Textfelds
</textarea >
```

- `name="..."` bestimmt den Namen des an die aufgerufene Seite übergebenen Parameters; sein Wert ist der Inhalt des mehrzeiligen Textfelds zum Zeitpunkt des Absendens des Formulars



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## <textarea> – mehrzeilige Textfelder

- Beispiel:

```
Textbereich:<br>
<textarea name="textbereich" cols="50"
  rows="5">
  Das ist der voreingestellte Text in einem
  Textbereich.¶
  Es kann Text mit unbegrenzter Länge¶
  (Begrenzung nur durch maximale POST-Größe des
  Servers) hier eingefügt werden.
</textarea>
```



## <textarea> – mehrzeilige Textfelder

- Beispiel (Fortsetzung):

Textbereich:

```
Das ist der voreingestellte Text in einem
Textbereich.¶
Es kann Text mit unbegrenzter Länge¶
(Begrenzung nur durch maximale POST-Größe des
Servers) hier eingefügt werden.
```



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## <textarea> – mehrzeilige Textfelder

- Der <textarea>-*Tag*
  - Das name-Attribut ist unbedingt erforderlich
  - Optionale Attribute (ab HTML 4 verpflichtend ➔ HTML-Dokumentation):
    - cols="...", rows="...": Anzahl der Spalten bzw. Zeilen des Textbereichs



## <textarea> – mehrzeilige Textfelder

- Der <textarea>-*Tag*
  - Text zwischen <textarea> und </textarea> wird „wörtlich“ übertragen; vom User eingegebene Zeilenumbrüche werden im Text je nach Plattform durch „%0d%0a“ (Windows), „%0d“ (Mac-OS) oder „%0a“ (UNIX / Linux) ersetzt
  - HTML-*Tag*s werden daher als Teil des Textes ohne spezielle Bedeutung interpretiert!
  - Länge des Texts ist bei allen aktuellen Browsern unbeschränkt (abhängig von Server-Konfiguration)



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Elemente von Formularen

- `<form>` – Definition eines Formulars
- `<input>` – Textfelder, Radio Buttons, Checkboxes und Standard-Schaltflächen
- `<select>` – Klapplisten und Listenfelder
- `<textarea>` – mehrzeilige Textfelder
- **`<button>` – Benutzerdefinierte Schaltflächen**

## `<button>` – Benutzerdefinierte Schaltflächen

- Erweiterte Variante von `<input type="button">`
- Erlaubt beliebige, auch graphische, Schaltflächen-Inhalte
- Kann je nach `<type="...">`-Attribut als *Submit*-, *Reset*- oder generischer Button funktionieren
- Details ➔ HTML-Dokumentation

© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Dateneingabe über Web-Formulare

- Übergabe von Parametern an den Webserver
- Elemente von Formularen
- **Übernahme der Parameter in ein PHP-Programm**
- Spezielle Aufgabenstellungen



## Übernahme der Parameter in ein PHP-Programm

- Die Werte aller HTML-Formularelemente sind unter PHP 3 bis einschließlich PHP 5 als Variable (oder Arrays) bzw. ab PHP 4 als Elemente globaler Arrays mit Namen verfügbar, die dem `name="..."`-Attribut des jeweiligen Formularelements entsprechen



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Übernahme der Parameter in ein PHP-Programm

- Beispiel: HTML-Formularelement:
  `<input type="text" name="tfeld" value="Textfeld">`
  - PHP 3: `$tfeld == "Textfeld"`
  - Funktioniert mit Default-Konfiguration in `php.ini` bis einschließlich PHP 4.1; mit `register_globals == on` bis PHP 5.3
  - PHP 3 – PHP 5.2:
    - `$HTTP_GET_VARS['tfeld'] == "Textfeld"`
    - `$HTTP_POST_VARS['tfeld'] == "Textfeld"`



## Übernahme der Parameter in ein PHP-Programm

- Beispiel (Fortsetzung):
  - Ab PHP 4.1:
    - `$_GET['tfeld'] == "Textfeld"`
    - `$_POST['tfeld'] == "Textfeld"`
    - `$_REQUEST['tfeld'] == "Textfeld"`
  - Default ab PHP 4.2: `register_globals == off`
    → Zugriff mit PHP 3-Methode (`$tfeld == "Textfeld"`) geht defaultmäßig nicht mehr!



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Übernahme der Parameter in ein PHP-Programm

- Workarounds für alte (PHP 3-) Websites:
  - Variable `register_globals` in `php.ini` auf „on“ setzen (ab PHP 5.3 nicht mehr möglich!)
  - Aufrufparameter mit den Funktionen `import_request_variables()` oder `extract()` in das globale Scope importieren (geht auch noch in PHP 5.3, nicht mehr ab PHP 5.4)



## Übernahme der Parameter in ein PHP-Programm

- Workarounds für alte (PHP 3-) Websites:
  - Import der Daten aus den Parameter-Arrays „zu Fuß“:

```
<?php

foreach ($_REQUEST as $name => $value)
   $$name = $value;

?>
```

  - Geht unabhängig von der PHP-Version immer
  - Sollte aber wegen der damit verbundenen Sicherheits-Implikationen wenn irgend möglich vermieden werden



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Übernahme der Parameter in ein PHP-Programm

- PHP 4.1+-„*Superglobals*“: Sind ohne zusätzliches „`global`“ im gesamten Programm, auch in Funktionen, verfügbar:
  - `$_GET`: Parameter, die mit Methode „`GET`“ übergeben wurden
  - `$_POST`: Parameter, die mit Methode „`POST`“ übergeben wurden
  - `$_REQUEST`: `$_GET` + `$_POST` + `$_COOKIE` (Vorsicht! `$_REQUEST` enthält auch Cookie-Daten!)



## Übernahme der Parameter in ein PHP-Programm

- Die Werte einzelner Formularelemente (*Radio Buttons*, *Checkboxes*, mehrzeilige `<select>`-Listen, `<select multiple>`-*Tags*) können `NULL` sein, wenn kein Element selektiert ist



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Übernahme der Parameter in ein PHP-Programm

- `<select multiple>`-*Tags*: Ein Array mit dem Namen des Felds wird übergeben
- Beispiel:
  - HTML-Formularelement:
    ```
    <select multiple name="liste[]">
    ```
  - PHP-Code zur Auswertung:
    ```
    foreach ($_REQUEST['liste'] as $key => $val)
      echo "\$liste[$key] = $val<br>\n";
    ```

## Übernahme der Parameter in ein PHP-Programm

- `<select multiple>`-*Tags*: **Vorsicht:** Wenn keine Option selektiert wurde, wird *kein* Array mit dem Namen des `<select multiple>`-Konstrukts übergeben → Laufzeitfehler!
- Besser:
  ```
  if (isset($_REQUEST['liste']) &&
  isarray($_REQUEST['liste'])))
   foreach ($_REQUEST ['liste'] as $key => $val)
      echo "\$liste[$key] = $val<br>\n";
  ```

© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Übernahme der Parameter in ein PHP-Programm

- `<image>`-*Tags*:
  ```
  <input type="image" name="myimage" src="bild.jpg"
  width="800" height="600" border="0" alt="">
  ```
  - Browser liefert beim Submit z.B.:
    ```
    zielseite.php?myimage.x=342&myimage.y=221
    ```
  - PHP macht daraus
    ```
    $_REQUEST['myimage_x'] == 342
    $_REQUEST['myimage_y'] == 221
    ```

## Dateneingabe über Web-Formulare

- Übergabe von Parametern an den Webserver
- Elemente von Formularen
- Übernahme der Parameter in ein PHP-Programm
- **Spezielle Aufgabenstellungen**

© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Spezielle Aufgabenstellungen

- Hochladen von Dateien
- Mail-Versand
  - Mail-Versand über `mailto:`-action-Parameter im `<form>`-*Tag*
  - Mail-Versand über PHP



## Spezielle Aufgabenstellungen

- **Hochladen von Dateien**
- Mail-Versand
  - Mail-Versand über `mailto:`-action-Parameter im `<form>`-*Tag*
  - Mail-Versand über PHP



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Hochladen von Dateien

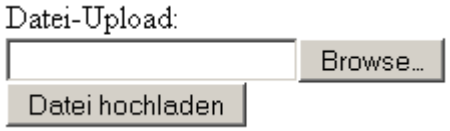
- Beispiel:

```
<form enctype="multipart/form-data"
  action="uploadtest.php" method="post">

<input type="hidden"
  name="MAX_FILE_SIZE" value="1048576">

Datei-Upload:<br>
<input type="file" name="userfile" size="20"><br>

<input type="submit" value="Datei hochladen">

</form>
```



## Hochladen von Dateien

- Beispiel (Fortsetzung):

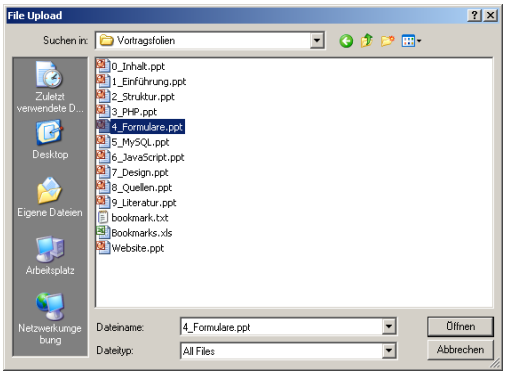





© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Hochladen von Dateien

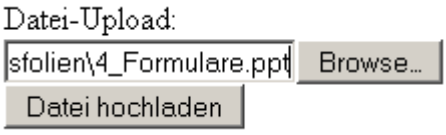
- Schaltfläche „`Durchsuchen...`“ oder „`Browse...`“ öffnet Standard-Windows-Datei-Öffnen-Dialog:






## Hochladen von Dateien

- Beispiel (Fortsetzung):




- Bei modernen Browsern entfällt häufig das Textfeld, in das bei älteren Browsern ein Datei-Pfad direkt eingegeben werden konnte. Es wird nach Auswahl einer Datei nur mehr der Dateiname (ohne Pfad) angezeigt.



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Hochladen von Dateien

- Hochladen von Dateien erfordert spezielle Server-seitige Einstellungen in `php.ini`:
  - `file_uploads = On`
  - `upload_max_filesize` > max. Dateigröße
  - `post_max_size` > max. Dateigröße
  - `memory_limit` evtl. anpassen bei Bearbeitung der hochgeladenen Datei
  - `max_execution_time` evtl. vergrößern für zeitaufwändige Bearbeitung von Uploads



## Hochladen von Dateien

- Es sollte gelten
  - `upload_max_filesize` > `MAX_FILE_SIZE`
  - `post_max_size` > `upload_max_filesize`
  - Dateien größer als `post_max_size` können über PHP nicht mehr detektiert und als Fehler gemeldet werden: Apache 2.x macht Seiten-Reset → vom Browser generierte (in der Regel nicht sehr aussagekräftige) Fehlermeldung!



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Hochladen von Dateien

- `memory_limit`-Parameter in `php.ini`:
  - Bei Weiterbearbeitung hochgeladener Dateien in PHP (*nicht* bei reinem Abspeichern) ist als `memory_limit` (mindestens) die doppelte maximale Dateigröße erforderlich
  - Zu große Werte für `memory_limit` können aber Fehlfunktionen des Webserver-Prozesses bewirken!



## Hochladen von Dateien

- Spezielle Einstellungen im Upload-Formular:

```
<form enctype="multipart/form-data"
  action="formtest.php" method="post">
```

  - `enctype="multipart/form-data"` für die korrekte Übertragung binärer Dateien
  - `method` *muss* gleich `"post"` sein



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Hochladen von Dateien

- Spezielle Einstellungen im Upload-Formular:
```
<input type="hidden"
  name="MAX_FILE_SIZE" value="30000">
```
- Dieser *Tag* muss *vor* dem `<input type="file"...>`-Tag stehen
- `MAX_FILE_SIZE` *sollte* vom Browser ausgewertet werden



## Hochladen von Dateien

- `MAX_FILE_SIZE`-Parameter:
  - Gibt die maximal hochladbare Dateigröße in Bytes an (*ganzzahliger* Wert!)
  - Es werden auf jeden Fall bis zu `MAX_FILE_SIZE` Bytes auf den Server hochgeladen
  - Empfehlung für den Browser, wird aber auch von PHP überprüft (erst nach komplettem Upload!)
  - Sollte aber nicht als einzige Dateigrößen-Beschränkung dienen
  - Wird von den meisten Browsern offenbar ignoriert



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Hochladen von Dateien

- Spezielle Einstellungen im Upload-Formular:
  `<input type="file" name="userfile" size="20">`
- `type="file"`!
- `name` ist verpflichtend, `size` ist optional (und wirkt, wenn überhaupt, wie bei Textfeldern (`<input type="text"...>`)



## Hochladen von Dateien

- Hochgeladene Datei wird in einem *Temp*-Verzeichnis des Servers abgelegt
- Informationen zur hochgeladenen Datei stehen in PHP-Variablen zur Verfügung
- PHP-Seite muss die Datei an einen endgültigen Speicherort kopieren (oder anderweitig bearbeiten) und die *Temp*-Datei löschen
- Beim Beenden des Scripts für die Zielseite (selbst bei PHP-Laufzeitfehlern!) wird die *Temp*-Datei automatisch gelöscht



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Hochladen von Dateien

- PHP-Datei-Upload-Variable:
  - Ab PHP 4.1.0: `$_FILES` („*Superglobal*“-*Array* – ist im gesamten PHP-Programm, auch in Funktionen, direkt sichtbar)
  - PHP 4.0.0 – 5.2: `$HTTP_POST_FILES` (*Array*, *nicht* „*Superglobal*“ – in Funktionen „`global $HTTP_POST_FILES`“ erforderlich!)
  - PHP 3: Eine Reihe von globalen Variablen



## Hochladen von Dateien

- Name der Array-Elemente bzw. der globalen Variablen durch `name="..."` im `<input type="file" ...>`-*Tag* bestimmt
- Folgendes Beispiel für `<input type="file" name="<b>userfile</b>">`
- Statt „`$_FILES`“ könnte im nachfolgenden Beispiel auch „`$HTTP_POST_FILES`“ verwendet werden; „`$HTTP_POST_FILES`“ wird aber ab PHP 5.3 nicht mehr unterstützt!



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Hochladen von Dateien

| | |
|---|---|
| Ursprünglicher Dateiname (am Client-Rechner) | `$_FILES['userfile']['name']`<br>`$userfile_name` |
| MIME-Datentype der Datei (z.B. „`image/gif`“) | `$_FILES['userfile']['type']`<br>`$userfile_type` |
| Dateigröße in Bytes | `$_FILES['userfile']['size']`<br>`$userfile_size` |
| Name der temporären Datei auf dem Webserver | `$_FILES['userfile']['tmp_name']`<br>`$userfile` |
| Fehlercode beim Datei-Upload (ab PHP 4.2.0)<br>➔ PHP-Dokumentation | `$_FILES['userfile']['error']` |



## Hochladen von Dateien

- Upload-Beispiel für PHP 3:

```
<?php
// $BASEPATH enthält den absoluten Pfad des
// Verzeichnisses, in dem die hochgeladenen
// Dateien für einen HTTP-Zugriff abgelegt werden
// sollen (z.B. $BASEPATH = "/var/www/files/"; )

// Datei aus Temp-Verzeichnis kopieren
exec ("cp $userfile ".$BASEPATH.$userfile_name);
// Temp-Datei löschen
if (file_exists ($userfile))
    unlink ($userfile);
?>
```



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Hochladen von Dateien

- Upload-Beispiel für PHP 4+:

```
<?php
// $BASEPATH enthält den absoluten Pfad des
// Verzeichnisses, in dem die hochgeladenen
// Dateien für einen HTTP-Zugriff abgelegt werden
// sollen (z.B. $BASEPATH = "/var/www/files/"; )

// Datei aus Temp-Verzeichnis verschieben
move_uploaded_file($_FILES['userfile']['tmp_name'],
   $BASEPATH.$_FILES['userfile']['name']);
?>
```



## Hochladen von Dateien

- PHP 3-Methode war mit großem Sicherheitsrisiko verbunden:
  - `http://upload.php3?userfile=%2Fetc%2Fpasswd` würde eine PHP-Variable „`userfile`“ mit Inhalt „`/etc/passwd`“ erzeugen; PHP 3-Beispiel würde diese Datei über den Webserver sichtbar machen und das Original zu löschen versuchen!
  - PHP 4+-Methode überprüft, ob die Datei mit dem in „`$_FILES['userfile']['tmp_name']`“ stehenden Namen tatsächlich hochgeladen wurde



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Hochladen von Dateien

- Gleichzeitiges Hochladen mehrerer Dateien ➔ PHP-Dokumentation



## Spezielle Aufgabenstellungen

- Hochladen von Dateien
- **Mail-Versand**
  - **Direkter Mail-Versand aus dem Browser**
  - Mail-Versand über PHP



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Direkter Mail-Versand aus dem Browser

- HTML-Formular mit `mailto:`-URL im „`action=...`“-Attribut des `<form>`-*Tags*:

```
<form method=post
action="mailto:mail@server.domain.com"
enctype="text/plain">
```

- Erforderliche Attribute:
  - `method="post"`
  - `enctype="text/plain"`: Gewährleistet, dass die Formulardaten in lesbarer Form als Mail übermittelt werden



## Direkter Mail-Versand aus dem Browser

- Grundsätzlich möglich, funktioniert aber nicht konsistent
- Je nach verwendetem Browser wird der Default-Mailer für die Erstellung einer Mail aufgerufen.
- Übergabe der mailto:-Adresse funktioniert meist; Übergabe des Formularinhalts in unterschiedlicher Form (als Name-Wert-Paare oder in Form einer GET-URL) oder überhaupt nicht



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Spezielle Aufgabenstellungen

- Hochladen von Dateien
- **Mail-Versand**
  - Direkter Mail-Versand aus dem Browser
  - **Mail-Versand über PHP**



## Mail-Versand über PHP

- Mail-Versand erfolgt über den Webserver
- PHP-Funktion `mail()`
- Die Mail kann Daten aus folgenden Quellen enthalten:
  - HTML-Formular
  - Datenbank-Inhalte
  - Beliebige andere für den Webserver zugängliche Inhalte



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Mail-Versand über PHP

- Vorteile:
  - Inhalt und Header der Mail können beliebige Daten in beliebig wählbarer Formatierung enthalten
  - Formular-Eingaben, die in die Mail übernommen werden sollen, können mit einer sicheren HTTP-Verbindung (HTTPS – *Secure HTTP*) übermittelt werden
  - Funktioniert unabhängig von Type und Version des Browsers



## Mail-Versand über PHP

- Nachteile:
  - Spezielle Konfigurationseinstellungen am Server sind notwendig, daher nicht von allen kommerziellen Sites unterstützt
- Vorsicht: Angaben in PHP-Dokumentation sind zum Teil irreführend oder sogar falsch!



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Mail-Versand über PHP

- Voraussetzungen am Webserver:
  - Ein *Mail Transport Agent* (MTA), z.B. `sendmail` am Server vorhanden (nicht notwendigerweise, aber besser als Prozess aktiv)
  - Eintrag in `php.ini` (z.B. bei Verwendung von `sendmail`):
    `sendmail_path = /usr/lib/sendmail –t`



## Mail-Versand über PHP

- Voraussetzungen am Webserver:
  - Eventuell in `sendmail.cf` ein Eintrag des Apache-Users (z.B. `apache`) in `/etc/mail/trusted-users`:
    `apache`



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Mail-Versand über PHP

- Mail-Versand mit PHP-Funktion mail():

```
<?php
// $to   ... "To:"-Adresse
// $subj ... Subject-Zeile
// $msg  ... Inhalt der Mail; beliebiger Text
// $head ... Zusätzliche Header-Zeilen,
//           durch Line Feeds(\n) getrennt
// $add .... (optional) Zusätzliche Aufruf-
//           parameter für den MTA (z.B.
//           "-fwebmaster@myserver.at")

// Fortsetzung auf nächster Folie
```

## Mail-Versand über PHP

- Mail-Versand mit PHP-Funktion mail():

```
$head = "From: test@www.domain.com";
$head .= "\nReply-to: me@myserver.com";
$head .= "\nX-Mailer: PHP/".phpversion();

$add = "-fwebmaster@myserver.at";

mail ($to, $subj, $msg, $head, $add);
?>
```

© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Mail-Versand über PHP

- In der Dokumentation der PHP-Funktion `mail()` ist angegeben, dass die Zeilen des Header-Bereichs (`$head` in unserem Beispiel) durch Carriage Return – Line Feed-Paare (`\r\n`) zu trennen sind. Das macht in vielen Konfigurationen Probleme (weil `mail()` oder der MTA ein „`\r`“ hinzufügt und das resultierende „`\r\r\n`“ von manchen Mail Clients fehlinterpretiert wird)! Bei einer Reihe unterschiedlicher Linux-basierter Websites waren mit allen Mail Clients kompatible Mails nur mit „`\n`“ in Header und Mailtext zu erhalten.
- Das steht im Widerspruch zur PHP-Dokumentation und den Mail-RFCs!



## Mail-Versand über PHP

- Besser: Statt „`\n`“ oder „`\r\n`“ zur Zeilentrennung die PHP-Core-Konstante `PHP_EOL` (definiert ab PHP 5.0.2 ) verwenden. `PHP_EOL` wird durch die in der jeweiligen Umgebung gültige Zeilenende-Zeichenfolge ersetzt:
  - Linux/Unix: `PHP_EOL` → `\n`
  - Windows: `PHP_EOL` → `\r\n`
- Im Mailtext (`$msg` in unserem Beispiel) sollte aber *immer* „`\n`“ als Zeilenende-Zeichenfolge verwendet werden (speziell, wenn der Mailtext mit `imap_8bit()` in das „Quoted-printable“-Format umgesetzt werden soll)!



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Mail-Versand über PHP

- Insbesondere beim Versand deutschsprachiger Mails (Umlaute!) ist die Angabe der folgenden zusätzlichen Zeilen für den Parameter $head zweckmäßig:

```
$head .= PHP_EOL."MIME-Version: 1.0";
$head .= PHP_EOL."Content-type: text/plain;
charset=ISO-8859-1";
$head .= PHP_EOL."Content-transfer-encoding:
8BIT";
$head .= PHP_EOL."Content-description: Mail
message body";
```

## Mail-Versand über PHP

- Unter PHP 5+ besser:

```
$head .= PHP_EOL."MIME-Version: 1.0";
$head .= PHP_EOL."Content-type: text/plain;
charset=ISO-8859-1";
$head .= PHP_EOL."Content-transfer-encoding:
Quoted-printable";
$head .= PHP_EOL."Content-description: Mail
message body";
```

© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Mail-Versand über PHP

- Unter PHP 5+ besser (Fortsetzung):
  - Subject der Mail mit `iconv_mime_encode()` codieren (wenn Zeichen außerhalb des ASCII-Bereichs 33 … 126 vorhanden);
  - Codierung des Mailtextes mit `imap_8bit()` (erzeugt „Quoted-printable“-Format); Vorsicht: „`imap`“-Funktionen sind nicht in allen PHP-Installationen verfügbar! Die gleiche Funktionalität kann aber problemlos selbst in PHP programmiert werden – siehe z.B. `http://www.freesoft.org/CIE/RFC/1521/6.htm`



## Mail-Versand über PHP

- Unter PHP 5+ besser (Fortsetzung):
  - Zeilenvorschübe in Header und Mailtext müssen in der *abgehenden* Mail als Carriage Return / Line Feed-Paare (`\r\n`) dargestellt werden; `mail()` setzt selbst den System-Zeilenvorschub (`PHP_EOL`) im Header und Mailtext in `\r\n` um!
  - Zeilenlänge: Empfohlen: ≤ 78 Zeichen, maximal zulässig: 998 Zeichen



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Mail-Versand über PHP

- Für Multipart-Mails (z.B. Text + binäres Attachment):

```
$head .= PHP_EOL."Content-type: Multipart/Mixed;
boundary=Message-Boundary-{eindeutiger String}
```

- Für Text-Inhalt:

```
$msg = PHP_EOL."--Message-Boundary-{eindeutiger String}";
$msg .= PHP_EOL."Content-type: text/plain;charset=ISO-
8859-1";
$msg .= PHP_EOL."Content-transfer-encoding: Quoted-
printable";
$msg .= PHP_EOL."Content-description: Mail message body";
$msg .= PHP_EOL.PHP_EOL;
$msg .= imap_8bit({eigentlicher Mailtext});
$msg .= PHP_EOL;
```



## Mail-Versand über PHP

- Für binäres Attachment z.B.:

```
$msg .= PHP_EOL."--Message-Boundary-{eindeutiger String}";
$msg .= PHP_EOL."Content-type: Application/Octet-stream;
        name=\"{Dateiname}\"; type=Unknown";
$msg .= PHP_EOL."Content-disposition: attachment;
        filename=\"{Dateiname}\"";
$msg .= PHP_EOL."Content-transfer-encoding: base64";
$msg .= PHP_EOL.PHP_EOL;

$attach = base64_encode ($attach_contents);
    // liefert das Attachment in einer langen Zeichenfolge
```



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Mail-Versand über PHP

- Für binäres Attachment (Fortsetzung):

```
$start = 0;
do           // zerhacken auf Zeilen mit <= 72 Zeichen
{
    $msg .= ($chunk = substr ($attach, $start, 72));
    $msg .= PHP_EOL;
    $start += 72;
} while ($chunk != "");

// beim letzten Schleifendurchlauf entsteht eine Leerzeile

$msg .= "--Message-Boundary-{eindeutiger String}--";
$msg .= PHP_EOL;
```



## Mail-Versand über PHP

- Bei Mails, die mit `mail()` versandt werden, steht defaultmäßig im *Envelope-From*-Feld (= jene Adresse, an die unzustellbare Mails zurückgehen) etwas wie „`apache@myserver.at`“. Um solche Mails sinnvoll zu handhaben, sollte mit dem optionalen fünften Parameter von `mail()` eine *gültige* und *existierende* Adresse im Format „`-f<Rücksende-Adresse>`“ angegeben werden (z.B. „`-fwebmaster@myserver.at`“). (Funktioniert nicht in allen Webhosting-Umgebungen!)



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme

## Mail-Versand über PHP

- `mail()` ist nicht gut für den Versand von Massen-Mails geeignet (schlechte Performance!). Für gleichzeitigen Versand einer großen Anzahl von Mails besser PEAR-Pakete verwenden!



© Karl Riedling, TU Wien, Institut für Sensor- und Aktuatorsysteme