合肥學院
HEFEI UNIVERSITY

Distributed Computing

Lesson 11: Web Forms

Thomas Weise · 汤卫思

tweise@hfuu.edu.cn · http://www.it-weise.de

Hefei University, South Campus 2
Faculty of Computer Science and Technology
Institute of Applied Optimization
230601 Shushan District, Hefei, Anhui, China
Econ. & Tech. Devel. Zone, Jinxiu Dadao 99

合肥学院 南艳湖校区/南2区
计算机科学与技术系
应用优化研究所
中国 安徽省 合肥市 蜀山区 230601
经济技术开发区 锦绣大道99号

Outline

website

Overview

- Now we can design web pages and know how a browser obtains them.
- In other words, we can do communication in one direction: content from the server is sent to the user.
- Can we also send content from the user back to the web server?

User Interaction: Forms

- We can develop dynamic and re-active application with HTML and HTTP

User Interaction: Forms

- We can develop dynamic and re-active application with HTML and HTTP
- HTML therefore offers the ability to construct `form`s

User Interaction: Forms

- We can develop dynamic and re-active application with HTML and HTTP
- HTML therefore offers the ability to construct `form`s
- `form`s use the possibility of sending data to server via HTTP or encoding it in URLs:
  - Web server can process data or forward it to external applications (e.g., CGI scripts or servlets)

Basic Form Components

- HTML Tag `<form>`

Basic Form Components

- HTML Tag `<form>`
- Components are *types* of the tag `input`

Basic Form Components

- HTML Tag `<form>`
- Components are *types* of the tag `input`:
  - text fields (`text`)

Basic Form Components

- HTML Tag `<form>`
- Components are *types* of the tag `input`:
  - text fields (`text`), text areas (tag `textarea`)

Basic Form Components

- HTML Tag `<form>`
- Components are *types* of the tag `input`:
  - text fields (`text`), text areas (tag `textarea`)
  - buttons (`submit`)

Basic Form Components

- HTML Tag `<form>`
- Components are *types* of the tag `input`:
  - text fields (`text`), text areas (tag `textarea`)
  - buttons (`submit`), radio buttons (`radio`)

Basic Form Components

- HTML Tag `<form>`
- Components are *types* of the tag `input`:
  - text fields (`text`), text areas (tag `textarea`)
  - buttons (`submit`), radio buttons (`radio`), check boxes (`checkbox`)

Basic Form Components

- HTML Tag `<form>`
- Components are *types* of the tag `input`:
  - text fields (`text`), text areas (tag `textarea`)
  - buttons (`submit`), radio buttons (`radio`), check boxes (`checkbox`)
  - drop-down boxes (`select)`

Basic Form Components

- HTML Tag `<form>`
- Components are *types* of the tag `input`:
  - text fields (`text`), text areas (tag `textarea`)
  - buttons (`submit`), radio buttons (`radio`), check boxes (`checkbox`)
  - drop-down boxes (`select`)
- Hidden fields (`hidden`) can be used to send session information that are not displayed but useful for the server

Example Form

Listing: Example form

```
<form action="http://www.test.com/domain1/FormServlet"
    method="get">
  <input type="text"    name="familyName"   size="30">
  <input type="submit"  value="Send">
  <input type="reset"   value="Clear">
  <input type="hidden"  name="session_id"
     value="B92A3CCF247">
  ...
</form>
```

- A click on the "Send" button will send the following HTTP GET request to the server `www.test.com`:

  `GET /domain1/FormServlet?familyName=Weise&session_id=B92A3CCF247 HTTP/1`

Methods for Sending Data

- GET Method

Methods for Sending Data

- GET Method:
  - All form data will be attached to the URL

Methods for Sending Data

- GET Method:
  - All form data will be attached to the URL
  - A "`?`" is attached to the base url, followed by "`fieldName=fieldValue`" pairs

Methods for Sending Data

- GET Method:
  - All form data will be attached to the URL
  - A "`?`" is attached to the base url, followed by "`fieldName=fieldValue`" pairs
  - The "`fieldName=fieldValue`" pairs are separated by "&"

Methods for Sending Data

- GET Method:
  - All form data will be attached to the URL
  - A "`?`" is attached to the base url, followed by "`fieldName=fieldValue`" pairs
  - The "`fieldName=fieldValue`" pairs are separated by "`&`"
  - Receiving server or script needs to analyse parameter string

Methods for Sending Data

- GET Method:
  - All form data will be attached to the URL
  - A "`?`" is attached to the base url, followed by "`fieldName=fieldValue`" pairs
  - The "`fieldName=fieldValue`" pairs are separated by "&"
  - Receiving server or script needs to analyse parameter string
  - And, as HTTP response, send a proper web page

Methods for Sending Data

- GET Method:
  - All form data will be attached to the URL
  - A "`?`" is attached to the base url, followed by "`fieldName=fieldValue`" pairs
  - The "`fieldName=fieldValue`" pairs are separated by "&"
  - Receiving server or script needs to analyse parameter string
  - And, as HTTP response, send a proper web page
- POST Method

Methods for Sending Data

- GET Method:
  - All form data will be attached to the URL
  - A "`?`" is attached to the base url, followed by "`fieldName=fieldValue`" pairs
  - The "`fieldName=fieldValue`" pairs are separated by "`&`"
  - Receiving server or script needs to analyse parameter string
  - And, as HTTP response, send a proper web page
- POST Method:
  - Field data is part of the payload (response body)

Methods for Sending Data

- GET Method:
  - All form data will be attached to the URL
  - A "`?`" is attached to the base url, followed by "`fieldName=fieldValue`" pairs
  - The "`fieldName=fieldValue`" pairs are separated by "`&`"
  - Receiving server or script needs to analyse parameter string
  - And, as HTTP response, send a proper web page
- POST Method:
  - Field data is part of the payload (response body)
  - Receiver usually gets data as stream (e.g., via stdin)

Processing Data

. . . Receiving server or script needs to analyse parameter string

And, as HTTP response, send a proper web page. . .

Processing Data

. . . Receiving server or script needs to analyse parameter string
And, as HTTP response, send a proper web page. . .

- This cannot be done in a static way

Processing Data

. . . Receiving server or script needs to analyse parameter string
And, as HTTP response, send a proper web page. . .

- This cannot be done in a static way:
  - HTML is not a programming language, we cannot define actions to process data on the server side.

Processing Data

. . . Receiving server or script needs to analyse parameter string
And, as HTTP response, send a proper web page. . .

- This cannot be done in a static way:
  - HTML is not a programming language, we cannot define actions to process data on the server side. HTML documents are just "dead" junks of text.

Processing Data

. . . Receiving server or script needs to analyse parameter string
And, as HTTP response, send a proper web page. . .

- This cannot be done in a static way:
  - HTML is not a programming language, we cannot define actions to process data on the server side. HTML documents are just "dead" junks of text.
  - JavaScripts runs on the client side (the JavaScript calculator example ran in the browser!), so it does not help here

Processing Data

. . . Receiving server or script needs to analyse parameter string
And, as HTTP response, send a proper web page. . .

- This cannot be done in a static way:
  - HTML is not a programming language, we cannot define actions to process data on the server side. HTML documents are just "dead" junks of text.
  - JavaScripts runs on the client side (the JavaScript calculator example ran in the browser!), so it does not help here
  - CSS has nothing to do with all of that

Processing Data

. . . Receiving server or script needs to analyse parameter string
And, as HTTP response, send a proper web page. . .

- This cannot be done in a static way:
  - HTML is not a programming language, we cannot define actions to process data on the server side. HTML documents are just "dead" junks of text.
  - JavaScripts runs on the client side (the JavaScript calculator example ran in the browser!), so it does not help here
  - CSS has nothing to do with all of that
- But the server is a process, written in a programming language

Processing Data

. . . Receiving server or script needs to analyse parameter string
And, as HTTP response, send a proper web page. . .

- This cannot be done in a static way:
  - HTML is not a programming language, we cannot define actions to process data on the server side. HTML documents are just "dead" junks of text.
  - JavaScripts runs on the client side (the JavaScript calculator example ran in the browser!), so it does not help here
  - CSS has nothing to do with all of that
- But the server is a process, written in a programming language
- The server can do more than just sending file contents to the user

Processing Data

. . . Receiving server or script needs to analyse parameter string
And, as HTTP response, send a proper web page. . .

- This cannot be done in a static way:
  - HTML is not a programming language, we cannot define actions to process data on the server side. HTML documents are just "dead" junks of text.
  - JavaScripts runs on the client side (the JavaScript calculator example ran in the browser!), so it does not help here
  - CSS has nothing to do with all of that
- But the server is a process, written in a programming language
- The server can do more than just sending file contents to the user
- It can actually process data and generate new documents (HTML pages) in memory and send them. . .

Sessions

- HTTP is state-less (no information preserved between different HTTP requests)

Sessions

- HTTP is state-less (no information preserved between different HTTP requests)
- How to implement *Sessions*?

Sessions

- HTTP is state-less (no information preserved between different HTTP requests)
- How to implement *Sessions*?
  - login/logout

Sessions

- HTTP is state-less (no information preserved between different HTTP requests)
- How to implement *Sessions*?
  - login/logout
  - "shopping basket"-like stuff

Sessions

- HTTP is state-less (no information preserved between different HTTP requests)
- How to implement *Sessions*?
- Create unique session id remembered by server

Sessions

- HTTP is state-less (no information preserved between different HTTP requests)
- How to implement *Sessions*?
- Create unique session id remembered by server and then
  - put session id as hidden form field

Sessions

- HTTP is state-less (no information preserved between different HTTP requests)
- How to implement *Sessions*?
- Create unique session id remembered by server and then
  - put session id as hidden form field or
  - encode session ID as part of URLs (e.g., in the `<a href="...`-tags)

Sessions

- HTTP is state-less (no information preserved between different HTTP requests)
- How to implement *Sessions*?
- Create unique session id remembered by server and then
  - put session id as hidden form field or
  - encode session ID as part of URLs (e.g., in the `<a href="...`-tags) or
  - store session ID in cookie

Sessions

- HTTP is state-less (no information preserved between different HTTP requests)
- How to implement *Sessions*?
- Create unique session id remembered by server and then
  - put session id as hidden form field or
  - encode session ID as part of URLs (e.g., in the `<a href="...`-tags) or
  - store session ID in cookie:
    - small piece of data send from web server to web browser as part of HTTP response

Sessions

- HTTP is state-less (no information preserved between different HTTP requests)
- How to implement *Sessions*?
- Create unique session id remembered by server and then
  - put session id as hidden form field or
  - encode session ID as part of URLs (e.g., in the `<a href="...`-tags) or
  - store session ID in cookie:
    - small piece of data send from web server to web browser as part of HTTP response
    - always sent by browser back to web server in subsequent requests

Summary

- So via HTML and `form`s, we can build an application that:
  - presents the user a form to fill in information
  - receives the input information once the user clicks a "Submit" button
  - can do something on the server side and present another web page
- Server-side things and program code can obviously not specified via HTML
- So how can we process data that is send to the server?

谢谢

Thank you

Thomas Weise [汤卫思]
tweise@hfuu.edu.cn
http://www.it-weise.de

Hefei University, South Campus 2
Institute of Applied Optimization
Shushan District, Hefei, Anhui,
China

Caspar David Friedrich, “Der Wanderer über dem Nebelmeer”, 1818
http://en.wikipedia.org/wiki/Wanderer_above_the_Sea_of_Fog