Verteilte Web-basierte Systeme – SS 2006

Verteilte Web-basierte Systeme

Dr.-Ing. Martin Gaedke
Universität Karlsruhe (TH)

Verteilte Web-basierte Systeme – SS 2006

Part VI

Development

Part 6 – Overview

1. Build and Test – "Creating the solution"
2. Data Technologies
   1. Standard XML Schemas
   2. Other Media Types
   3. Structural Linking
3. UIX Technologies
   1. Presentation
   2. Navigation
   3. Dialogue
4. Systems Technologies
   1. Web Server
   2. URI Handler
   3. Communication
   4. Security
   5. Federation

Verteilte Web-basierte Systeme – SS 2006

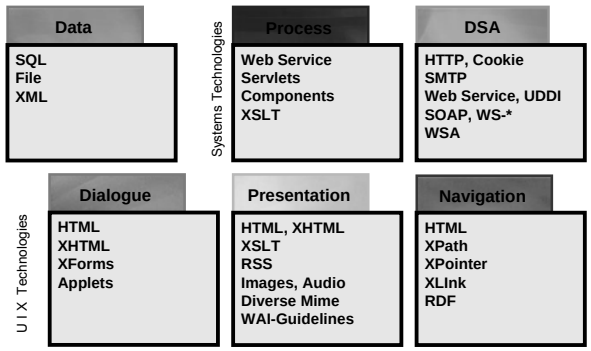
Chapter://1

Build and Test – "Creating the solution"

Introduction

- Goal
  - Transform FuncSpec to real code
  - Develop final documentation
  - Implementation of the solution
- Challenges
  - Mapping from design to code
  - (Possibly) development in parallel
  - Tests

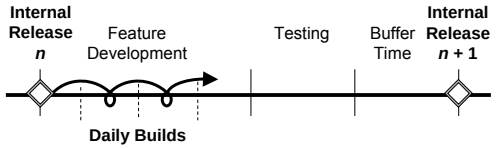
Non-Exhaustive Tec-Map



Further help: Cf. supporting standards and guidelines, like IEEE Std 2001-2002

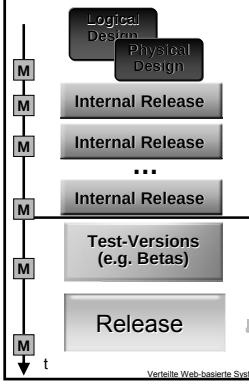
Building Releases…

- Daily builds or Continuous Integration
  - A way to make the product and its progress visible
  - The *heartbeat* of the development process



And a lot of other core tools and methods like in software development should be used – not to forget Version Control …

Testing & Evolution in Context



- Development Cycles "behave" like in common software production
- Testing… a continuous process…
- Evolution: Further developments towards the vision (ongoing process)

Verteilte Web-basierte Systeme – SS 2006

Chapter://2

Data Technologies

Goal

- Transforming designs into real code
  - In many cases we are already done due to physical design
  - Sometimes "fine-tuning" necessary
- Aspects
  - XML schemas
  - Other media types
  - Structural linking of data

Verteilte Web-basierte Systeme – SS 2006

Section://1

Standard XML Schemas

There is no THE STANDARD

- Apply well-known schemas if available
  - Check W3C, OASIS, WfMC etc.
  - Apply Standards accepted in many domains, e.g. RDF, OWL, eXtensible Rights Markup Language, RSS
- Focus on Domain-specific standards
  - Microsoft Office XML Schemas http://www.microsoft.com/office/xml/
  - Workflow XML (WfXML) http://www.wfmc.org/
  - Task done by Domain Experts

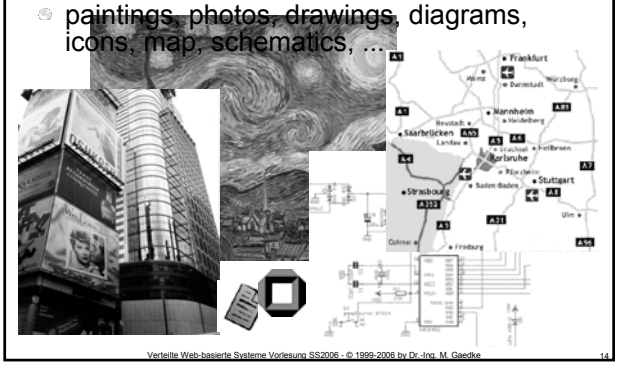
Verteilte Web-basierte Systeme – SS 2006

Section://2

Other Media Types

Media Type Image I

- paintings, photos, drawings, diagrams, icons, map, schematics, ...



Media Type Image II

- Issues to think about:
  - Color Depth: black&white ... full color (1, 2, 4, 8, 16, 24, 32 Bit)
  - Size: Poster, ..., Icon
  - Format: Depending on subject
  - Many more...
- In most cases these parameter influence the file size!

Example - Color Depth: 4 Bit

Original (24 Bit, 16 Mio. colors)

4 Bit, 16 colors

Content-Type: image/gif

- Graphics Interchange Format, GIF [Compuserve 1987]
  - GIF87a, GIF89a
  - Up to 256 colors, 8Bit
  - Loseless compression
    - algorithm used for compression: LZW (Lempel-Zev-Welch)
    - groups of pixels are run length coded
    - patented by Unisys
  - Animation
  - Transparency (1-bit)
  - Interlacing
  - Supported nearly everywhere and always ..
- File size is dependent on:
  - Image Size
  - Color Depth
  - Image Content (e.g. large areas of same color compress well)
  - Dithering → usually results in poor compression

GIF – Interlacing

- Image Interlacing
  - One dimensional coded in four steps
  - 12,5% of data → first coarse image (like a mosaic)
  - 25%, 50% improved quality
  - 100% complete image
- Advantage
  - users can recognize the image faster
- Disadvantage
  - slightly larger files
  - not useful for icons

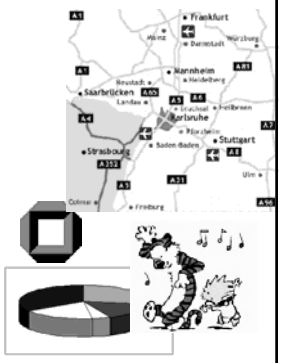
GIF – Transparency

- Image Transparency
  - One color defined as transparent
- Advantage
  - The image "supports" different backgrounds
- Problem
  - Artifacts on the border of images, due to anti-aliased functions

Text-Text-Text

When to use GIF?

- Typical requirements:
  - Few colors
  - Small images
  - Lines and edges
  - Transparency
  - Simple animations
  - Visible on any platform
- Sample applications
  - Diagrams
  - Forms
  - Icons, banners
  - Rendered text (avoid!)
  - Comics
  - Line drawings



Content-Type: image/jpg

- Joint Photographic Experts Group, [ISO 1993]
  - 16,7 Mio. colors, 24 Bit
  - JPG compression
    - Based on frequency
    - 8x8 Pixel base blocks
    - 64 values as a discrete signal, transformation into spectrum
    - Indexing, Discrete Cosine Transform (DCT), Quantization, Arithmetic/Huffman
- File Size is dependent on
  - Image size
  - Image structure, e.g. details, edges
  - Compression factor
  - Preprocessing: e.g. gauß filter, unsharpened

Example – JPG Compression

original (24 Bit, 16 Mio. colors, Q=0, 521 KB raw, 165 KB jpg)

compressed, Q=10, 56 KB

Example – JPG Compression

original (24 Bit, 16 Mio. colors, Q=0, 521 KB raw, 165 KB jpg)

compressed, Q=90, 10 KB

Example – JPG Compression

original (24 Bit, 16 Mio. colors, Q=0, 521 KB raw, 165 KB jpg)

compressed, Q=97, 6 KB

When to use JPG?

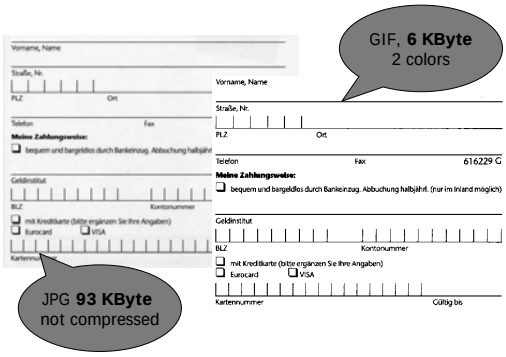
- Typical requirements :
  - A lot of colors
  - Large images
  - Smooth images
  - Small files
  - (bandwidth requirements)
- Sample applications, e.g.
  - Photos
  - Paintings

Comparison JPG – GIF

JPG, compressed, Q=30, 31 KB

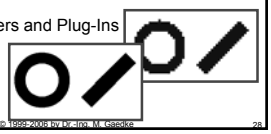
GIF, 256 colors, 170 KB

Comparison JPG – GIF



Many Other Formats Exist

- PNG (say “Ping”) [RFC 2083]
  - Loseless compression with free (non-patented) algorithm
  - Mechanisms to check data integrity
  - Support for embedding text (Keywords, Copyright)
  - No animation (see MNG)
  - Transparent true-color Images
  - http://www.w3.org/Graphics/PNG
- SVG (Scalable Vector Graphics)
  - Vector format based on XML
  - Supports CSS, transparency, animations, event-handling, scripting, meta data
  - Supported by graphics tools, viewers and Plug-Ins
  - http://www.w3.org/Graphics/SVG/
- …



Verteilte Web-basierte Systeme – SS 2006

Section://3

Structural Linking – XLink

Motivation

- Motivation http://www.w3.org/TR/NOTE-xlink-principles
- It is a requirement to allow for "open systems" of linking where not all resources are under the control of a single person or organization (along with easier "closed systems"). For example, broken links must be tolerated.
- Both unidirectional links (common on the Web today) and multidirectional links (commonly used in commercial hypermedia systems) must be supported.

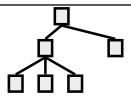
XML Linking Language

- XML Linking Language (XLink) Version 1.0
  - W3C Recommendation 27 June 2001
  - http://www.w3.org/TR/xlink/
- Describes how elements should be inserted into XML documents in order to create and describe links between resources.
- It uses XML syntax to create structures that can describe links similar to the simple unidirectional hyperlinks of today's HTML, as well as more sophisticated links.

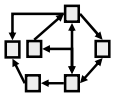
What is a Link? (Concepts)

- **Link (XLink)** – A link is an explicit relationship between resources or portions of resources.
- **Participation** – When a link associates a set of resources, those resources are said to **participate** in the link.
- **Resource** – Any addressable unit of information or service (Cf. IETF RFC 2396).
- **Hyperlink** – A link that is intended primarily for presentation to a human user.
  - Simple Example: HTML-Link

XLink Design Goals



- XLinks shall be human-readable
- XLinks may reside outside the documents in which the participating resources reside
- XLink shall represent the abstract structure and significance of links



- For further linking concepts, cf. Hypermedia literature like Dexter, OHS, MicroCosm

XLink - Example

- XLink-Namespace: http://www.w3.org/1999/xlink
- Example:
  - **<MyLink**
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xlink:type="simple"
    xlink:href="http://hotel.dom/rooms.xml">
    Current List of Rooms**</MyLink>**
- Cf. HTML Link:
  - <A href="http://hotel.dom/rooms.html">Current List of Rooms</A>

Linking-Element

- **Linking-Element** – An element that contains the XLink attribute **xlink:type**. The XLink attribute specifies the type of the link that the Linking-Element represents.
- Two types of links
  - **Simple Link**

  and
  - **Extended Link**

Simple Links

- **Simple Link** – A link that associates exactly two resources, one local and one remote, with an arc going from the former to the latter
  - xlink:type="simple"
  - Means: An **outbound link** from the local to a remote resource
- Examples

```
<MyLink xlink:href="remoteURI">…
</MyLink>

<MyLink
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xlink:type="simple"
    xlink:href="remoteURI">…</MyLink>
```

Linking-Element and DTDs

- DTD for Linking-Element “MyLink”

```
<!ELEMENT MyLink (#PCDATA)>
<!ATTLIST MyLink
   xmlns:xlink CDATA #FIXED  “http://www.w3.org/1999/xlink”
   xlink:type CDATA #FIXED “simple”
   xlink:href CDATA #REQUIRED
>
```

- Example:
  <MyLink xlink:href=“http://hotel/rooms.xml“>
  Current List of Rooms</MyLink>

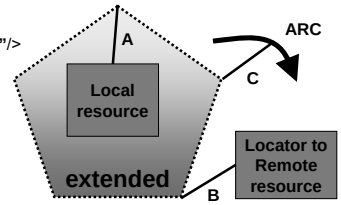
Extended Links

- **Extended Link** – A link that associates an arbitrary number of resources. The participating resources may be any combination of remote and local
  - xlink:type = "extended"
  - Allow for inbound, outbound, third-party arcs, and allow for arbitrary numbers of participating resources
  - Additional XLinks (specified by xlink:type): Resource, Locator, Arc, Title

Extending A Simple-Type…

- Extended-Version
- <MyLink **xlink:type=“extended”**>
  <A **xlink:type=“resource” xlink:label=“local”**>…</A>
  <B **xlink:type=“locator” xlink:label=“remote” xlink:href=“remoteURI”**/>
  <C **xlink:type=“arc” xlink:from=“local” xlink:to=“remote”**/>
- </MyLink>

- Simple-Version
- <MyLink **xlink:type=“simple” xlink:href=“remoteURI”**/>



Local Resource

- **Local Resource** – Link element that defines the participating local resources that appear inside the extended link.
  - xlink:type="resource“
  - Example
    <A **xlink:type=“resource” xlink:label=“local”**>…</A>
  - Useful for Arcs (Cf. xlink:from attribute)

Remote Resource

- **Remote Resource** – Link element that defines the participating remote resources of an extended link by being addressed with a URI reference.
  - **xlink:type="locator“**
  - Example
    <B **xlink:type=“locator” xlink:label=“remote” xlink:href=“remoteURI”**/>
  - Useful for Arcs (Cf. xlink:from, xlink:to attribute)

Traversal Rules

- **Traversal** – Term for using or following a link for any purpose
  - Traversal always involves a pair of resources: From **starting resource** to **ending resource**
- **Arc** – Information about how to traverse a pair of resources, including the direction of traversal and possibly application behavior
- **Multidirectional Link** – If two arcs in a link specify the same pair of resources as A→B, and B→A
  - This is not the same as "going back" after traversing a link

Traversal Rules

- **Outbound** – A relationship defined by an arc that has a local starting resource and a remote ending resource.
- **Inbound** – A relationship defined by an arc that has a local ending resource and a remote starting resource.
- **Third-Party Arc** – A relationship defined by an arc that has neither a local starting resource nor a local ending point.outbound not inbound.

Traversal Rules

- **Traversal Rules** – Link element that defines rules among its participating resources (of an extended link).
  - xlink:type="arc“
  - Example
    <C
    xlink:type=“arc”
    xlink:from=“local”
    xlink:to=“remote”/>
  - From and To point to labels (Cf. locator or resource)

Example Extended Link

- **Link database / Linkbase** – Documents that contain collections of inbound and third-party (arcs).
- <ERPLinkBase **xlink:type=“extended”**>
  - <Item
    xlink:type=“locator” xlink:label=“TVSet” xlink:href=“URI1”/>
  - <Item
    xlink:type=“locator” xlink:label=“Radio” xlink:href=“URI2”/>
  - <Room
    xlink:type=“locator” xlink:label=“Room42” xlink:href=“URI2”/>
  - <Room
    xlink:type=“locator” xlink:label=“Room48” xlink:href=“URI2”/>
  - <Contains
    xlink:type=“arc” xlink:from=“Room42” xlink:to=“TVSet”/>
  - < Contains
    xlink:type=“arc” xlink:from=“Room42” xlink:to=“Radio”/>
  - < Contains
    xlink:type=“arc” xlink:from=“Room48” xlink:to=“Radio”/>
- </ ERPLinkBase>

Global XLink Attributes

- **Behavior** – For use with simple and arc types
  - xlink:show= *new*, *replace*, *embed*, other, none
  - xlink:actuate= onLoad, onRequest, other, none
  - Provides data for UIX processing
- **Semantic** – Describe the meaning of resources within the context of a link
  - xlink:title, xlink:role, xlink:arcrole

Verteilte Web-basierte Systeme – SS 2006

Section://4

Structural Linking – XPath

XML Path Language (XPath)

- XML Path Language (XPath)
  - W3C Recommendation 16 November 1999
  - http://www.w3.org/TR/1999/REC-xpath-19991116
- XPath is a language for addressing parts of an XML Document, designed to be used by both XSLT and XPointer
- Motivation
  - How to address node(s) in an xml-document
  - **Notice: A MUST learn language!!!**

XPath - Examples

- Location path are the most important constructs
  - Location path is absolute or relative
- Some example constructs:

  /order/price
  - Describes the way from the root to the node price

  //book
  - Select all book elements in the order document

  //book[@isbn="777-842"]
  - Select all book elements with an isbn attribute of value 777-842 in the order document

  ./@isbn
  - Select the isbn attribute of the context node

Verteilte Web-basierte Systeme – SS 2006

Section://5

Structural Linking – XPointer

Motivation

- Motivation http://www.w3.org/TR/NOTE-xlink-principles
- Need for Anchors (cf. HTML Anchor) in XML
- XPointers address into XML documents
  - XPointers shall be straightforwardly usable in URI's
  - The XPointer syntax shall be reasonably compact and human-readable

XPointer

- XML Pointer Language (XPointer)
  - W3C Last Call Working Draft 8 January 2001
  - W3C Candidate Recommendation 11 September 2001
  - Language to be used as the basis for a fragment identifier for an XML resource
  - W3C Working Draft 16 August 2002 → XPointer document has been superceded!
- Now: Xpointer Framework

XPointer Framework

- XPointer Framework
  - W3C Recommendation 25 March 2003
  - http://www.w3.org/TR/xptr-framework/
  - Supports: Shorthand and Scheme-Based Pointer, Namespace Binding
- Example for http//server/resource.xml append:
  - #xpointer(/order/price)
  - #xpointer(id('orderId')/price[1])element(//price)

Verteilte Web-basierte Systeme – SS 2006

Chapter://3

UIX Technologies – Presentation

Verteilte Web-basierte Systeme – SS 2006

Section://1

CSS

Cascading Style Sheets

- Formats and layouts for HTML documents
  - CSS level 1 (CSS1) [Lie et al. 1996]
  - CSS2, positioning
- Rule based
  - Values are assigned to properties of HTML elements
  - E.g.:

```
P { text-align: right; color: green }
```

  - Selectors can be context dependent
  - very expressive: color, font, layout, position ...
- Inheritance of attribute values along the HTML document hierarchy

CSS Rule Syntax

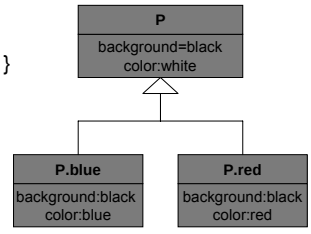
- Style Sheets have one or more rules that describe the format of elements
- Rules consist of selector and declaration
- Declaration consists of one or more pairs with property and value
- Syntax:
  - `<selector> "{" <declaration> "}"`
  - `<declaration>::= <property> ":" <value-list> [";" <property> ":" <value-list>]*`
  - `<value-list>::= <value> ["," <value>]*`

CSS: Applying styles

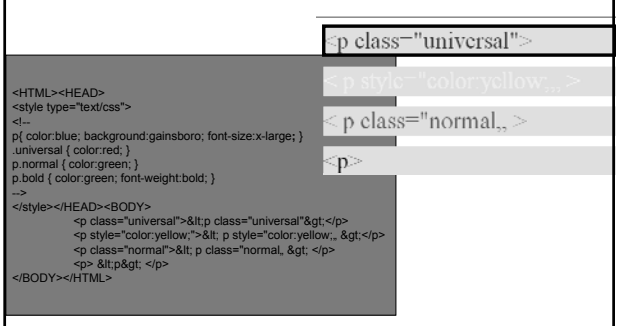
- HTML tags can be styled using
  - **class attribute**
    <p class=„classname“>
  - **style attribute**
    <p style=„text-align:center“>
  - **overwriting default style**
    <p>
  - **subclassing: extending default style**
    <p class=„bold“>

CSS: Inheritance

- Subclassing tags
  - **Defining a default style**
    p { background:black; }
  - **extending default style**
    p.blue { color:blue; }
    p.red { color:red; }
    .....



CSS Example I




```
<HTML><HEAD>
<style type="text/css">
<!--
p{ color:blue; background:gainsboro; font-size:x-large; }
.universal { color:red; }
p.normal { color:green; }
p.bold { color:green; font-weight:bold; }
-->
</style></HEAD><BODY>
        <p class="universal">&lt;p class="universal"&gt;</p>
        <p style="color:yellow;">&lt; p style="color:yellow;„ &gt;</p>
        <p class="normal">&lt; p class="normal„ &gt; </p>
        <p> &lt;p&gt; </p>
</BODY></HTML>
```

CSS Example II

```
<HTML><HEAD>
<style type="text/css">
<!--
p{ color:blue; background:gainsboro; font-size:x-large; }
.universal { color:red; }
p.normal { color:green; }
p.bold { color:green; font-weight:bold; }
-->
</style></HEAD><BODY>
        <p class="universal">&lt;p class="universal"&gt;</p>
        <p style="color:yellow;">&lt; p style="color:yellow;„ &gt;</p>
        <p class="normal">&lt; p class="normal„ &gt; </p>
        <p> &lt;p&gt; </p>
</BODY></HTML>
```


CSS Example III

```
<HTML><HEAD>
<style type="text/css">
<!--
p{ color:blue; background:gainsboro; font-size:x-large; }
.universal { color:red; }
p.normal { color:green; }
p.bold { color:green; font-weight:bold; }
-->
</style></HEAD><BODY>
        <p class="universal">&lt;p class="universal"&gt;</p>
        <p style="color:yellow;">&lt; p style="color:yellow;„ &gt;</p>
        <p class="normal">&lt; p class="normal„ &gt; </p>
        <p> &lt;p&gt; </p>
</BODY></HTML>
```

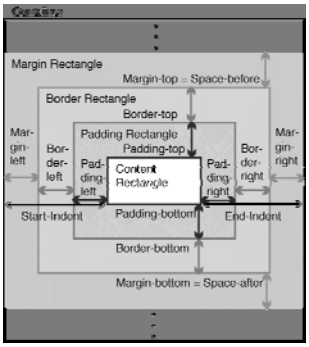

CSS Example IV

```
<HTML><HEAD>
<style type="text/css">
<!--
p{ color:blue; background:gainsboro; font-size:x-large; }
.universal { color:red; }
p.normal { color:green; }
p.bold { color:green; font-weight:bold; }
-->
</style></HEAD><BODY>
        <p class="universal">&lt;p class="universal"&gt;</p>
        <p style="color:yellow;">&lt; p style="color:yellow;„ &gt;</p>
        <p class="normal">&lt; p class="normal„ &gt; </p>
        <p> &lt;p&gt; </p>
</BODY></HTML>
```


Positioning with CSS

- Position, Layer
- Example

```
#p {
text-align: justify;
text-indent: 4;
word-spacing: 7;
z-index: 2;
border: 3px double #000000;
margin-left: 2;
margin-right: 3;
margin-top: 5;
margin-bottom: 6;
padding-left: -4;
padding-right: 6;
padding-top: 10;
padding-bottom: 11 }
```



“Cascading”

- p i { color:yellow; }
  - apply on all <i> nested in
  - *depth=infinity*

<i/> <i/>

- p > i { color:yellow; }
  - apply on <i> nested in a
  - *depth=1*

<i/> <i/>

- p * i { color:yellow; }
  - apply on all <i> nested in
  - *depth>1*

<i/> <i/>

Verteilte Web-basierte Systeme – SS 2006

Section://2

XHTML

XHTML Introduction

- The Extensible HyperText Markup Language (XHTML™)
  - W3C Recommendation 26 January 2000
  - http://www.w3.org/TR/2000/REC-xhtml1-20000126
- Specification defines XHTML 1.0, a reformulation of HTML 4 as an XML 1.0 application
- Three DTDs corresponding to the ones defined by HTML 4
- Semantics of the elements and their attributes are defined in the W3C Recommendation for HTML 4

Example XHTML Document

- Document Root element **html**
- Referencing xhtml namespace
- Elements and attributes must be conform to XML notation rules

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "DTD/xhtml1-
    strict.dtd">
<html
    xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
    <head><title>XHTML Example</title></head>
    <body>
        <p>XHTML is great. </p> <hr/>
        <p>A    <a href="http://webengineering.org/">
                WebE-Link</a>.</p>
    </body>
</html>
```


Differences To HTML

- Documents must be well-formed
  - Incorrect: Overlapping Elements <a>
  - Correct: <a> or <a>
- Element and attribute names must be in lower case
- For non-empty elements, end tags are required
  - Incorrect: A new paragraph starts here
  - Correct: A new paragraph starts here
- Attribute values must always be quoted

Differences To HTML II

- Attribute minimization
  - Attribute names like compact or checked must be written in full
  - Incorrect: <dl compact>
  - Correct: <dl compact="compact">
- Using ampersands in attribute values
  - '&' must be expressed as a character entity reference
  - Incorrect: http://example/cgi/script?a=guest**&**name=martin
  - Correct: http://example/cgi/script?a=guest**&**name=martin

Differences To HTML III

- Empty Elements
  - Must be XML conform:
 →
  <hr>→<hr/>
- Whitespace handling in attribute values
  - User Agents will strip leading and trailing Whitespace from Attribute Values
- Script and Style elements
  - <script> <![CDATA[... unescaped script content ...]]> </script>
- SGML exclusions
  - SGML gives the Writer of a DTD the Ability to exclude specific Elements from being contained within an Element. Such Prohibitions (called "exclusions") are not possible in XML.
  - For example, the HTML 4 Strict DTD forbids the nesting of an '**a**' element within another '**a**' element to any descendant depth

Differences To HTML IV

- The Elements with 'id' and 'name' Attributes
  - HTML 4 defined the **name** attribute for the elements **a**, **applet**, **form**, **frame**, **iframe**, **img**, and **map**. HTML 4 also introduced the **id** attribute.
  - name and id are attributes designed to be used as fragment identifiers (are of type ID therefore unique).
  - XHTML 1.0 Documents MUST use the **id** Attribute when defining fragment identifiers, even on elements that had a **name** attribute
  - Check compatibility – if necessary provide both: id="foo" name="foo"

Verteilte Web-basierte Systeme – SS 2006

Section://3

XSL(T)

Introduction XSL

- Extensible Stylesheet Language (XSL)
- Description of a transformation necessary
- "XSL is a language for expressing stylesheets. Given a class of structured documents or data files in XML, designers use an XSL stylesheet to express their intentions about how that structured content should be presented; that is, how the source content should be styled, laid out and paginated onto some presentation medium such as a window in a Web browser or a set of physical pages in a book, report, pamphlet, or memo." (http://www.w3.org/TR/WD-xsl/)

XSL Tranformations – XSLT

- This specification defines the syntax and semantics of XSLT, which is a language for transforming XML documents into other XML documents
- XSL specifies the styling of an XML document by using XSLT to describe how the document is transformed into another XML document that uses the formatting vocabulary
- A transformation expressed in XSLT describes rules for transforming a *Source Tree* into a *Result Tree*
- The transformation is achieved by associating patterns with templates. A pattern is matched against elements in the source tree. A template is instantiated to create a part of the result tree
- http://www.w3.org/TR/xslt

XSLT Architecture

- Source tree as input
- Result tree as output
- XSLT processor takes two inputs
  - XSL style sheet
  - XML Document as source tree

XSLT Processing Model

- Input in Form of a Tree
  - Recursive process
  - Checks for template when a new item is encountered
  - Transform source nodes into result nodes
  - Rearranges the items based on style sheet

XSLT Example

- XML to XML
- Takes one XML document as source tree
- Apply templates using XSLT stylesheet
- Transforms it into another XML document as a result tree (here the result tree element are conform to HTML element names;-)

Demo

- Demo with files
  - XPath-XSLT.xml
  - XPath-XSLT.xsl

Source Tree

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="sampleB-IE5.xsl" ?>

<fruit_salad_ingredients>
  <fruit>
    <name>oranges</name>
  </fruit>
  <fruit>
    <name>pineapples</name>
  </fruit>
  <fruit>
    <name>starfruit</name>
  </fruit>
  <fruit>
    <name>watermelon</name>
  </fruit>
</fruit_salad_ingredients>
```

This PI-Entity is supported by Microsoft Internet Explorer 5.5 and newer.

Note: Allows you to easily test your XSL-skills

XSL Style Sheet

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/" >
    <HTML>
      <BODY>
        <table border="1">
          <th>Fruit Salad Ingredients</th>
          <!-- Display the name of each fruit-->
          <xsl:for-each select="/fruit_salad_ingredients/fruit">
            <tr>
              <td><xsl:value-of select="name"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </BODY>
    </HTML>
  </xsl:template>
</xsl:stylesheet>
```

(sampleB-IE5.xsl)

Result As Shown By IE

Homework!!

- Create an XML document
- Create an XSLT document
  - Link the XSLT file to the XML document
  - Apply XPath to find some nodes in the XML file
- Use latest version of Microsoft IE or Netscape to check the result
  - An example and learning tool are provided at the lecture site

Verteilte Web-basierte Systeme – SS 2006

Chapter://4

UIX Technologies – Navigation

Verteilte Web-basierte Systeme – SS 2006

Section://1

Links in HTML

Navigation in Documents

RESOURCE: DRINKS.HTML

```
<html>
    <body>
        The following drinks are served:
        <a href="#water">Water</a>,
        <a href="#beer">Beer</a>,
        <a href="#wine">Wine</a>.
        <hr>
        <a name="water">Water 1 EUR.</a><br>
        <a name="beer">Beer 1.5 EUR.</a><br>
        <a name="wine">Wine 1.5 EUR.</a><br>
    </body>
</html>
```

Links

- **HTML-Link (A-Tag)** – A markup element defining a link that describes a oneway relationship from the incorporating document to a target (may be a remote resource) defined by a URI.
  - **Explaining Text**
- Examples
  - WebE
  - Show Picture
  - Send Email

Anchor

- **HTML-Anchor** – A markup element defining a dedicated area in a document. Anchors may be used as document internal targets for links.
  - Content
- Targeting HTML-Anchors: <a href="URI"...
  - URI: http://example.com/path/doc.html#anchorname
  - URI: path/doc.html#anchorname (relative)
  - URI: #anchorname (relative same document)

Verteilte Web-basierte Systeme – SS 2006

Chapter://5

UIX Technologies – Dialogue

Verteilte Web-basierte Systeme – SS 2006

Section://1

HTML-Forms

Introduction to HTML Forms

- **Form-Element** – A Markup to describe Interaction Units as part of an HTML document.
  - Form Action-Attribute: Binds Instance-Data to Processing Unit
- Syntax:
  <**form**
  method="GET|POST"
  action="URI" *(E.g.:mailto:..., http:…)*
  name="form-id"
  enctype="multipart/form-data|..."
  target="name of frame – If used">
  Form Controls and HTML
  </**form**>

Form Controls

- **Data Binding**: name-value pairs of form controls
- **Form Controls** defined by Input or Select elements:
- <input type="**typename**" attributes…>
  - **checkbox**-Attributes: name, value, checked
  - **radio**-Attributes: name, value, checked
  - **hidden**-Attributes: name, value
  - **text**-Attributes: name, value, size, maxlength
  - **password**-Attribute: name, value
  - **textarea**-Attributes: id, value, rows, cols, wrap
- <select size="number" name="id">
  - E.g. Pull-Down Menu or Window with selectable Items
  - Attributes: size, name
  - <OPTION>A selectable Value</OPTION>

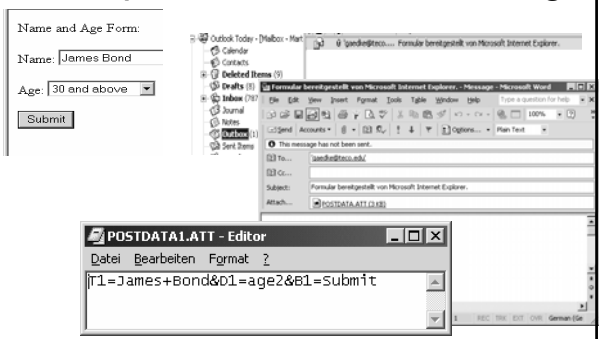
Example

```
...
<p>Name and Age Form:</p>
<form method="POST" action= "mailto:gaedke@example.com">
  <p>Name: <input type="text" name="T1" size="20"></p>
  <p>Age:
     <select size="1" name="D1">
          <option value="15-30">29 and younger</option>
          <option value="age2">30 and above</option>
     </select>
  </p>
  <input type="submit" value="Submit" name="B1">
</form>
```


Demo

- Check file:
  - Forms-example.htm

Example Presentation Binding



HTML Action Model

- **BUTTON – Elements**:
  - <input type="typename" attributes>
  - **Submit, button, reset**-Attributes: name, value
  - **image**-Attributes: name, src
    - src: Url for Image to be displayed
    - Note: Server receives name.x and name.y!
  - **file**-Attributes: name
- For further details check HTML literature at lecture site.

Advanced Action Technologies

- Client-Side Scripting
  - Script code part of HTML and interpreted by the Browser
  - Different scripting languages exist
    - ECMAScript standardized language (similar to JavaScript/JScript)
    - JavaScript, JScript (Netscape, Internet Explorer)
    - VBScript (Internet Explorer)
  - Example: Input validation at client, DHTML (e.g. menus)
- Example:

```
<script language="JavaScript">
<!--   function clickMethod() {…}   //-->
</script>
```


HTML Event Processing

- Form Controls have a set of common behaviors
  - Event Handler are executed when an event occurs
  - Event Handler are assigned to HTML objects
- Event Handler Examples:
  - onChange, onClick, onMouseOver
- Example:
  - <input type="button" value="StartAction" **onClick="clickMethod()"**>

Pros and Cons of Scripting…

- JavaScript is supported by the main Browsers
  - Note: Even if Browser supports JavaScript – support may be disabled
  - Document Object Model is different between Netscape Communicator und MS Internet Explorer
- Example

```
If (navigator.appName == `Microsoft Internet
Explorer´) {
   myHead = head1;
 } else {
    myHead = document.head1;
}
 myHead.style.color = `green´;
```


Verteilte Web-basierte Systeme – SS 2006

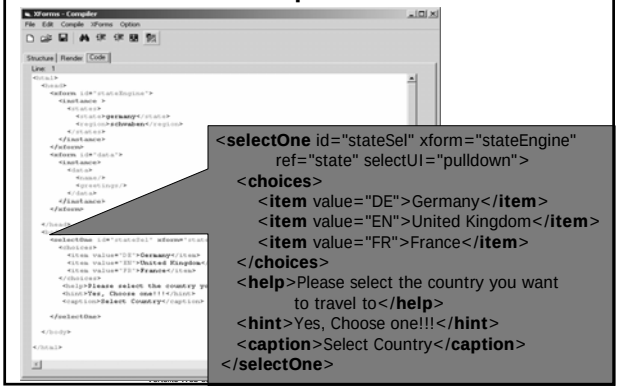
Section://2

XForms

XForms

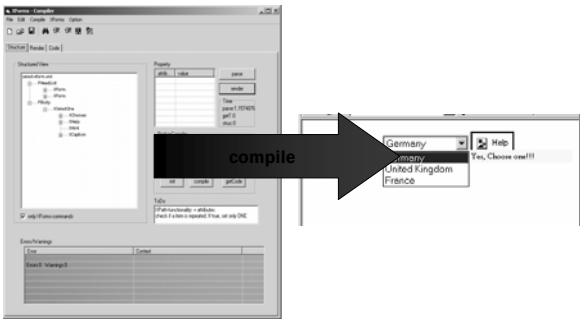
- XForms 1.0
  - W3C Candidate Recommendation 11 November 2002
  - http://www.w3.org/MarkUp/Forms/
  - W3C's approach for future Web Forms
- Very interesting Approach
  - Competition from big software vendors
- Cf. Interaction Design Model

XForms Example



```
<selectOne id="stateSel" xform="stateEngine"
        ref="state" selectUI="pulldown">
  <choices>
    <item value="DE">Germany</item>
    <item value="EN">United Kingdom</item>
    <item value="FR">France</item>
  </choices>
  <help>Please select the country you want
        to travel to</help>
  <hint>Yes, Choose one!!!</hint>
  <caption>Select Country</caption>
</selectOne>
```

XForms Example II



Further Information

- Testing the new technology possible!
- **XForms 1.0 Public Test Suite**
  - w3.org/MarkUp/Forms/2002/Examples/Test/
  - Available on W3C site since 20 January 2003
- First PlugIns and ActiveX controls available

Verteilte Web-basierte Systeme – SS 2006

Chapter://6

UIX Technologies – Browser

Client-Side Code

- What is Client-Side Code?
  - Software that is downloaded from web server to browser and then executes on the client-side
- Why Client-Side Code?
  - Scalability: less work done on server
  - Performance/User experience
  - Create UI constructs not inherent in HTML
    - Drop-down and pull-out menus
    - Tabbed dialogs
  - Special effects, e.g. animation, sound
  - Data validation

Examples / Technologies

- DHTML/JavaScript
- ActiveX / COM
- Java Applets
- AJAX
- Dedicated technologies
  - Depends on browser, e.g. Netscape
  - Examples:
    Plug-ins or Helpers used for dedicated MIME content-types

Dynamic HTML (DHTML)

- Script that is embedded within an HTML page
- Usually written in JavaScript (ECMAScript, JScript) for portability
  - Internet Explorer also supports VBScript and other scripting languages
- Each HTML element becomes an object that has associated events (e.g. onClick)
- Script provides code to respond to browser events

Malicious (D)HTML

```
<html>
        <head><title>eBay Angebot</title></head>
        <body>
                <p id="bewertung">positive bewertung 0 %</p>
                .....

                 <p id=„angebot">
                        <script> p.bewertung=„100 %"</script>
                 </p>

                 ......
         </body>
</html>
```


XSS (Cross Site Scripting)

- Find exploits on web pages and insert malicious code
  - Used for session hijacking
- A simple <img> can do some interaction
  - <img src="javascript:txt='UghhOghh.!!! My Screen Just Ran Away!!!'; txt2='Now come on you have to admit that was funny *S*';x=0;y=80; function niceguy(){nice=confirm(txt2); if(nice==1){window.setTimeout('parent.window.moveTo(0,0)',2100)};} function ha(x){parent.window.moveTo(x,y);if(x==1800)alert('Hehehe...me went Bye-bye ;)'); window.setTimeout('if(x!=1800){ha(x+=30);};else{niceguy()}',25)}; alert('*Yawn*..me tired');ha();">

  Check: IMG-Malicious.html
- **CHECK ALL INPUT OF YOUR FORMS!!!!**

ActiveX

- Based on COM
- "Good when you know your users (e.g. Intranet) and need system access"
  - Native only to Internet Explorer
  - Supported in Netscape with a Plug-in
  - Small, efficient code
- Don't use if you can

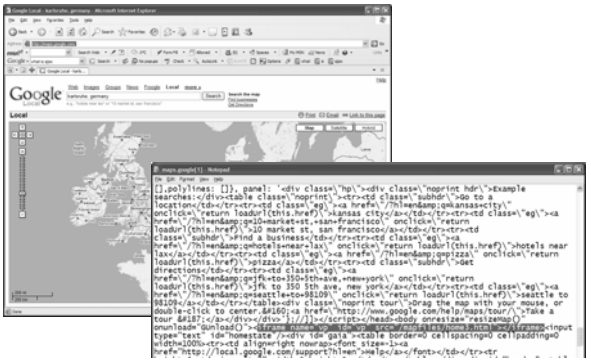
Java Applets

- Based on Java Bytecode
- Held great promise as a portable, pain-free way to download client-side code:
  - "Write once, run anywhere"
- Compatibility and performance issues have prevented common usage
- Don't use if you can

AJAX

- Asynchronous JavaScript And XML
- Intent: Combine different technologies for creating web pages that "behave" like desktop applications
  - "Pages are not reloaded" - Script code controls behavior
  - Script transforms user interactions and controls retrieving and rendering data
- AJAX frameworks combine
  - XHTML/HTML
  - CSS
  - Client-Side Scripting Language, e.g. JavaScript/JScript
  - Hidden IFrame and/or HTTPRequests
  - Data transfer formats, e.g. XML, HTML, plain text, JavaScript Object Notation (JSON) etc.
- Many Pros & Cons
  - Pro: includes behavior well-known, rich interactivity
  - Cons: includes usability issues, JavaScript required

AJAX – maps.google.com



Verteilte Web-basierte Systeme – SS 2006

Chapter://7

Systems Technologies

Verteilte Web-basierte Systeme – SS 2006

Section://1

Web Server

Processing

Process Data

Client-Side:
- Prepare Request
- Request Resource

Server-Side:
- **Handle Request**
  - **(1) Wait for Request**
  - **(2) Prepare Processing**
  - **(3) Process/Compute**
- **Send Resource**
  - **(4) Server Response**

- Handle Response
- Process Data of Resource

1 – Wait For Request

Process Data

- Waiting for an HTTP request
  - Listen on Server Port (Usually Port 80)
  - Accept Connection
    - Iterative server – One Request after another (no Concurrency)
    - Concurrent Server – One Process/Thread per Request
- Example:
  - Server-Loop:
    - wait for connection request
    - accept connection
    - create Thread/Process
    - Goto Server-Loop
  - Thread/Process:
    - Cf. next Step: Prepare Processing

2 – Prepare Processing

Process Data

- Analyze HTTP Request
  - Extract URI
  - Extract Host-Header
  - Extract Port
- Example:
  - GET / HTTP/1.1
  - Host: server.com
- Cf. Protocol HTTP

2 – Prepare Processing

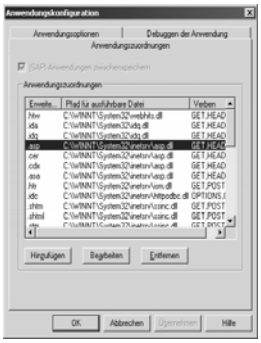
Process Data

- Process Client-Request
  - Method, Header, URI supported?
  - E.g. GET may be supported, but not POST, PUT, DELETE allowed
- Find URI – Handler responsible for (Host, Port)
  - Query Server internal scheme-specific-part Database for Handler responsible for URI
    - http://example.com/images → File-Handler for Server (example.com, 80)
    - http://127.0.0.1/cgi/login.exe → CGI-Handler for Server (127.0.0.1, 80)
    - http://localhost/cgi/login.exe → Error-Handler for Server (localhost, 80)
    - http://localhost/WebService1/service1.asmx/HelloWorld → WebService-Handler for Server (localhost, 80)
  - **Note: server.com, localhost, and 127.0.0.1 may be the same physical Machine, but the Web Server may support virtual Web Servers!**
- Create/Call URI Handler
  - Not necessarily supported by Web Server, but should be...
- Or respond with Error code

Calling URI Handler

Process Data

- URI Handler inside Web Server process
  - Iterative server
  - Concurrent server: thread runs URI Handler
- URI Handler in separate Process
  - CGI, FastCGI, NSAPI/ISAPI
- New approaches focus on separating
  - HTTP-Processing → OS-Kernel
  - URI Handler → Pooled processes



3 – Process/Compute

Process Data

- URI Handler receives request
  - **NOTE: Implements a Process Model**
  - **Allows for different Process Models on one Web Server!**
  - Advanced URI Handler prepares/provides mechanisms for Session Handling
- Process / Compute
  - Wired-Components are executed
  - Non-wired Components are called using communication
- Compute Response

4 – Server Response

Process Data

- URI Handler
  - Creates header for HTTP-Response
  - Adds response resource
  - Return complete Response to answering Process
- Answering Process (E.g. Web Server)
  - May create additional Header elements for HTTP-Response
  - Sends Response to Client

A Web Server In Java I

```
import java.net.*;
import java.io.*;
import java.util.*;

public class jhttp extends Thread {
  Socket theConnection;
  static File docroot;
  static String indexfile = "index.html";

  public jhttp(Socket s) {
    theConnection = s;
  }

  public static void main(String[] args) {
    int thePort;
    ServerSocket ss;
    // get the Document root
    try {
      docroot = new File(args[0]);
    }
    catch (Exception e) {
      docroot = new File(".");
    }
```


A Web Server In Java II

```
// set the port to listen on
    try {
      thePort = Integer.parseInt(args[1]);
      if (thePort < 0 || thePort > 65535) thePort = 80;
    }
    catch (Exception e) {
      thePort = 80;
    }

    try {
      ss = new ServerSocket(thePort);
      System.out.println("Accepting connections on port "
        + ss.getLocalPort());
      System.out.println("Document Root:" + docroot);
      while (true) {
        jhttp j = new jhttp(ss.accept());
        j.start();
      }
    }
    catch (IOException e) {
      System.err.println("Server aborted prematurely");
    }
  }
```


A Web Server In Java III

```
public void run() {
    String method;
    String ct;
    String version = "";
    File theFile;
try {
      PrintStream os = new PrintStream(theConnection.getOutputStream());
      DataInputStream is = new DataInputStream(theConnection.getInputStream());
      String get = is.readLine();
      StringTokenizer st = new StringTokenizer(get);
      method = st.nextToken();
      if (method.equals("GET")) {
        String file = st.nextToken();
        if (file.endsWith("/")) file += indexfile;
        ct = guessContentTypeFromName(file);
        if (st.hasMoreTokens()) {
          version = st.nextToken();
        }
        // loop through the rest of the input lines

        while ((get = is.readLine()) != null) {
          if (get.trim().equals("")) break;
}
```

A Web Server In Java IV

```
try {
  theFile = new File(docroot, file.substring(1,file.length()));
  FileInputStream fis = new FileInputStream(theFile);
  byte[] theData = new byte[(int) theFile.length()];
  // need to check the number of bytes read here
  fis.read(theData);
  fis.close();

  if (version.startsWith("HTTP/")) {  // send a MIME header
    os.print("HTTP/1.0 200 OK\r\n");
    Date now = new Date();
    os.print("Date: " + now + "\r\n");
    os.print("Server: jhttp 1.0\r\n");
    os.print("Content-length: " + theData.length + "\r\n");
    os.print("Content-type: " + ct + "\r\n\r\n");
  }  // end if

  // send the file
  os.write(theData);
  os.close();
}  // end try
```


A Web Server In Java V

```
    catch (IOException e) {                    // can't find the file
      if (version.startsWith("HTTP/")) {  // send a MIME header
        os.print("HTTP/1.0 404 File Not Found\r\n");
        Date now = new Date();
        os.print("Date: " + now + "\r\n");
        os.print("Server: jhttp 1.0\r\n");
        os.print("Content-type: text/html" + "\r\n\r\n");
      }
      os.println("<HTML><HEAD><TITLE>File Not
                                    Found</TITLE></HEAD>");
      os.println("<BODY><H1>HTTP Error 404: File Not
                                    Found</H1></BODY></HTML>");
      os.close();
    }
  }
```


A Web Server In Java VI

```
  else {  // method does not equal "GET"
    if (version.startsWith("HTTP/")) {  // send a MIME header
      os.print("HTTP/1.0 501 Not Implemented\r\n");
      Date now = new Date();
      os.print("Date: " + now + "\r\n");
      os.print("Server: jhttp 1.0\r\n");
      os.print("Content-type: text/html" + "\r\n\r\n");
    }
    os.println("<HTML><HEAD><TITLE>Not
                  Implemented</TITLE></HEAD>");
    os.println("<BODY><H1>HTTP Error 501: Not
                  Implemented</H1></BODY></HTML>");
    os.close();
  }
 }
```


A Web Server In Java VII

```
        catch (IOException e) {
    }
    try {
      theConnection.close();
    }
    catch (IOException e) {
    }
  }

  public String guessContentTypeFromName(String name) {
    if (name.endsWith(".html") || name.endsWith(".htm"))
        return "text/html";
    else if (name.endsWith(".txt") || name.endsWith(".java"))
        return "text/plain";
    else if (name.endsWith(".gif") )
        return "image/gif";
    else if (name.endsWith(".class") )
        return "application/octet-stream";
    else if (name.endsWith(".jpg") || name.endsWith(".jpeg"))
        return "image/jpeg";
    else return "text/plain";
  }
}
```


Verteilte Web-basierte Systeme – SS 2006

Section://2

URI Handler - SSI

Server-Side Include (SSI)

- HTML page is parsed by the SSI-Handler for command expressions
  - Commands return strings that are dynamically inserted at the location of the command
  - Components – HTML & SSI-Commands
  - Wiring done by position of commands and SSI-Handler
- Transparent for the Client
  - Only HTML is provided by the Server
- Used for
  - navigational elements, header, footer
- Server Configuration
  - e.g. Apache: AddType text/x-server-parsed-html .shtml
    Registry-Entry for Microsoft IIS .shtml

SSI - Directives (Excerpt)

- echo *var*
  - Puts in the value of environment variables
- include *file*
  - Includes the content from a File
  - *Allows for simple component approaches*
- exec *cmd*
  - Executes a command and includes the output
- flastmod *file,* fsize *file*
  - Puts in the date of the last change, size of a file

SSI - Examples I

- echo var

```
<HTML><HEAD><TITLE>SSI Echo Sample</TITLE></HEAD>
<BODY>
Welcome to my Server: <!--#echo var="SERVER_NAME"--> <BR>
Local Time here is: <!--#echo var="DATE_LOCAL"--> <BR>
You are connected from: <!--#echo var="REMOTE_HOST"--> <BR>
And you requested the following file:
<!--#echo var="DOCUMENT_NAME"-->
</BODY> </HTML>
```

- include file
  - virtual: relative to server root directory
  - file: from the current directory

```
<HTML><HEAD><TITLE>SSI Include Sample</TITLE></HEAD>
<BODY>
Here is our Navigation Header<BR>
<!--#include virtual="/header.html"-->
Here is the Text of the page ... <BR>
Here is our Footer<BR> <!--#include file="footer.html"-->
</BODY> </HTML>
```


SSI - Examples II

- Exec cmd (attention - security!)

```
<HTML><HEAD><TITLE>SSI Exec Sample</TITLE></HEAD>
<BODY>
call a cgi file:
<!--#exec cgi="/scripts/add.pl?FirstName+LastName" --><BR>
call a programm file:
<!--#exec cmd="/bin/finger $REMOTE_USER@REMOTE_HOST" --><BR>
</BODY> </HTML>
```

- flastmod file, fsize file

```
<HTML><HEAD><TITLE>SSI flastmod Sample</TITLE></HEAD>
<BODY>
The file test.pdf was last modified at
<!--#flastmod file="test.pdf"-->
the file size is
<!--#fsize file="test.pdf"-->
</BODY> </HTML>
```


Verteilte Web-basierte Systeme – SS 2006

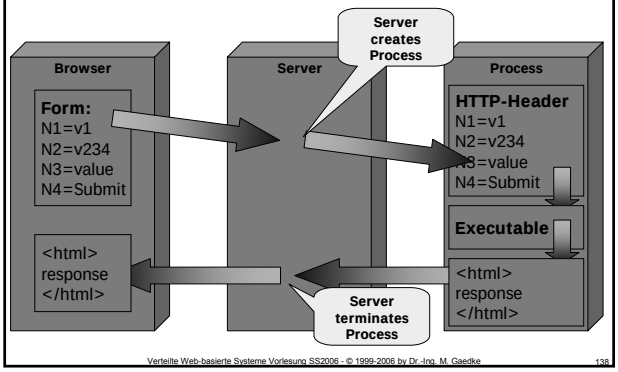
Section://3

URI Handler - CGI

Common Gateway Interface

- **Common Gateway Interface (CGI)** – Protocol that specifies how information can be passed from a Web page via a Web Server to an executable and back to the browser.
  - Environment & Language of CGI-executable define its Component and Wire Model
  - Each CGI-executable is started in a separate Process
  - Example: Compiled C++ program

CGI In Use



CGI and Server Configuration

- The Server needs to know that a certain File should be executed
- Example Apache-Server
  - Directory (e.g. everything in /cgi/ will be executed):
  - `ScriptAlias /cgi /usr/http/cgi-bin/`
  - All files of a certain Type will be executed:
  - `AddType application/x-http-cgi .pl.cgi`

CGI Parameter Using GET

- Variables are coded in the Request-URL
- <URL>?value1
  - CGI-Program called with command-line Argument)
- <URL>?var1=val1&var2=val2
  - The Environment Variable QUERY_STRING is set with the string after the '?' and can be used by the CGI-Program
- Problem: On some systems the Length of a URL is restricted

CGI Parameter using POST

- Variables are coded in the Body of the Request
- The Body of the Request is provided to the Application as Standard Input (e.g. same as myProg.pl < file)

Verteilte Web-basierte Systeme – SS 2006

Section://4

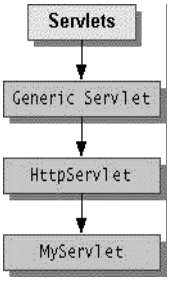
URI Handler - Servlets

Servlets

- URI Handler is a Java Virtual Machine
  - Servlet (java byte-code) is executed on request
  - Classes GenericServlet and HttpServlet
  - Methods that are called in a Servlet (usually provided by the programmer): init(), service(), destroy()
- Further information available:
  - http://java.sun.com/docs/books/tutorial/servlets/TOC.html
  - http://java.apache.org/jserv/papers/techniques.pdf
  - http://java.apache.org/jserv/howto.load-balancing.html

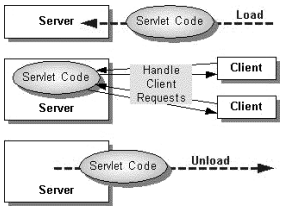
Servlets

- Interaction with the client
- If the call to the servlet is accepted the following objects are available in the servlet:
  - **ServletRequest** – supports communication from the client to the servlet
  - **ServletResponse** – supports communication from the servlet to the client



Life Cycle of Servlets

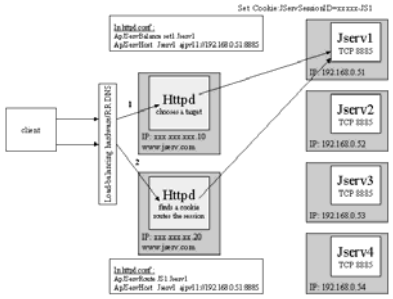
- Servlet Engine (URI Handler) loads and initializes the Servlet
- The Servlet serves 0 or more requests from clients
- Servlet Engine removes the Servlet (e.g. when the server is shut down)



Servlet – Example

```
public class SimpleServlet extends HttpServlet
{
    /** Handle the HTTP GET method by building a simple web page. */
    public void doGet (HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        PrintWriter out;
        String title = "Simple Servlet Output";
    // set content-type and other response header fields first
        response.setContentType("text/html");
    // then write the data of the response
        out = response.getWriter();
        out.println("<HTML><HEAD><TITLE>" + title + "</TITLE></HEAD>");
        out.println("<BODY><H1>" + title + "</H1>");
        out.println("</BODY></HTML>");
        out.close();
    }
}
```


Scalability – Fault Tolerance



http://java.apache.org/jserv/howto.load-balancing.html

Verteilte Web-basierte Systeme – SS 2006

Section://5

URI Handler - Scripting

Server-Side Script vs. CGI

- Server-Side Script: Program Code included in HTML
  - Similar Approach used by PHP, JSP, etc.
- Example:
  - <HTML><BODY>
  - Hello <%=getName(SID)%>!
  - </BODY></HTML>
- Compare to CGI Program: HTML is the output of the Program
  - Println (“<HTML><BODY>Hello ”);
  - Println (toString(getName(SID)));
  - Println (“</BODY></HTML>”);

Active Server Pages

- Microsoft Internet Information Server (Version 3.0 and up)
  - Include Scripts in HTML-Pages on IIS
  - Transaction Handling
  - Session Handling
- Process Model
  - Component Model
    - Support for COM/DCOM/COM+ components
  - Wire Model
    - Different scripting languages wire components
    - Example: VBScript, Jscript, third-party languages (e.g. Python)

ASP - Example

- **<% … %> Script-Tag**
  - <SCRIPT LANGUAGE="VBSCRIPT" RUNAT=Server>
- Example
  - <%
    Set obj = Server.CreateObject("MyComponent")
    str = obj.DoSomeAction("Hello World")
    Response.Write (str)
    %>

ASP - Objects

- Application
  - Data in one Application, shared data space for multiple users
  - Changing data requires locking
- Request
  - Access to variables from forms, cookies, and query-string
- Response
  - Create the output document
  - Support for cookies and redirects
- Server
  - Server specific data
  - Parameter for scripts
- Session
  - Manage user data within time context (by Cookie or Url-Encoded)

ASP – Input and Output

- Using the Objects Request and Response
- Access to Environment Variables, e.g.
  - Request.ServerVariables("HTTP_USER_AGENT")
- Access to Values of Variables from Forms, e.g.
  - Request.QueryString("FirstName")
- Write Data in the Response Document, e.g.
  - Response.Write("Hello World!")
- State-Handling, e.g.
  - Response.Cookies("WebE")="A Cookie"
  - Response.Cookies("WebE").Expires …

Verteilte Web-basierte Systeme – SS 2006

Section://6

Communication - HTTP

Hypertext Transport Protocol

- The Hypertext Transport Protocol (HTTP) is used to request and return resources (Web pages, images, Microsoft Word documents, Adobe PDF documents, other MIME typed resources etc.)
- Protocol
  - Transmission of messages based on TCP/IP
  - Communication state less
  - Two message types: Request or Response
  - Messages are ASCII coded
  - Methods: GET, POST, HEAD, etc.
- HTTP Allows for Extensions
- HTTP 1.1 defined by [RFC 2068]
  - More possibilities supported, like Proxy, Multi-homed Servers etc.

Generic Message Structure

- Generic message structure allows for extensions!
- Important: Underlying Concept for all Protocols on top of HTTP!

- Generic-Message = Start-Line
  *Header
  CRLF
  [Message-Body]

- Start-Line = Request-Line | Response-Line
- Header = field-name ":" [field-value] CRLF
  - field-name = token
  - field-value = *(field-content | LWS)
  - LWS = Linear White Space
- Message-Body
  - If exists MUST be encoded
  - Presence signaled by header field Content-Length or Transfer-Encoding

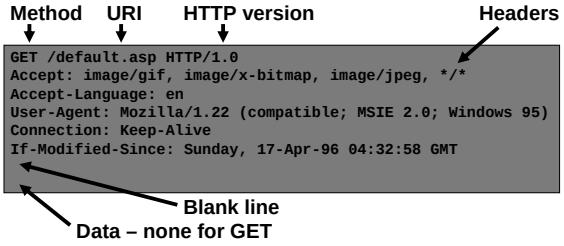
Request Message Structure

- Message structure
  <Method>“ ”<URI>“ ”<Protocol>
  <Headers>
  CRLF
  [<Data>]
- Method ::= “GET” | “POST” | “HEAD” | …
- Protocol ::= “HTTP/1.0” | “HTTP/1.1” | …
- Headers ::= <hName>“:”<hValue>
  - hName – Specific header name h
  - hValue – Value of the value space of header h
- Data ::= <TEXT>

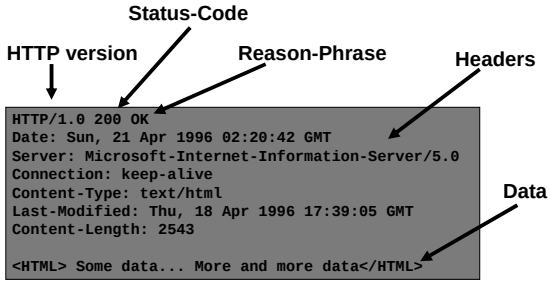
Response Message Structure

- Message Structure
  <Protocol>“ ”<Status-Code>“ ”<Reason-Phrase>
  <Headers>
  CRLF
  [<Data>]
- Protocol ::= “HTTP/1.0” | “HTTP/1.1”
- Status-Code ::= DIGIT+ ; for use by automata
- Reason-Phrase ::= <TEXT> ; for use by human user
- Headers ::= <hName>“:”<hValue>
  - hName – Specific Header Name h
  - hValue – Value of the value space of header h
- Data ::= <TEXT>

HTTP Request



HTTP Response



Common HTTP Methods I

- For detailed description of the methods, cf. [RFC 2068]
- OPTIONS
  - Request for information about available communication options
- GET
  - Retrieve whatever information is identified by the Request-URI
- HEAD
  - Identical to GET except that the server MUST NOT return a message-body in its response
- POST
  - Request that the destination server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI
  - This allows for:
    - Annotation of existing resources
    - Posting a message to an application, e.g. Black-Board, Front-end for Email, etc.
    - Providing a block of data (Submitting a form) to a data-handling process
    - Extending a database through an append operation.

Common HTTP Methods II

- PUT
  - The enclosed entity should be stored under supplied Request-URI
- DELETE
  - The origin server should delete the resource identified by Request-URI
- TRACE
  - Method to invoke remote, application-layer request loop-back
  - Allows the client to see what is being received at the other end

Common HTTP Headers I

- Content-Type
  - Indicates the media type
- Expires
  - Date/Time after which the response should be considered stale
  - Useful for caching!
- Host
  - Specifies internet host and port number of the resource being requested
  - Needed for multi-homed servers
- Last-Modified
  - Date and time the variant was last modified
  - Useful for caching!

Common HTTP Headers II

- Location
  - Used to redirect the recipient to a location other than the Request-URI
  - Very useful for concatenating application logic by URL
- Referer
  - Allows the client to specify, for the server's benefit, the address (URI) of the resource from which the Request-URI was obtained
  - Very useful for maintenance, because allows for
    - Lists of back-links (where does the client come from), e.g which search-engine
    - Logging, optimized caching, etc.
    - Find obsolete or mistyped links
  - Don't use for securing your application
- User-Agent
  - Information about the User Agent originating the request
- Many other exists, e.g. to handle caching, authorization, content encoding etc.

Common HTTP Status Codes

Code	Description
200	OK
201	Created
301	Moved Permanently
302	Moved Temporarily
400	Bad Request – not understood
401	Unauthorized
403	Forbidden – not authorized
404	Not Found
500	Internal Server Error

Verteilte Web-basierte Systeme – SS 2006

Section://7

Communication - Cookie

Introduction

- A mechanism to store a small amount of data (up to 4KB) at the client
  - http://home.netscape.com/newsref/std/cookie_spec.html
  - http://www.huecker.com/msw/cookie_spec.shtml (mirror)
- A cookie is associated with a specific web site
- Cookie is sent in HTTP header
- Cookie is sent with each HTTP request
- Can last for only one session (until browser is closed) or can persist across sessions
- Can expire some time in the future

Overview

- Protocol Primitives
  - **Set-Cookie** – Request from server asking client to store a cookie, included in the response header
  - **Cookie** – If a Cookie is stored for the current domain and dedicated path of the request then the stored data this is sent to the server (as part of request header)
- Used to implement Sessions
  - E.g. supported by ASP using Session Object
  - Libraries, e.g. http://www.worldwidemart.com/scripts/readme/cookielib.shtml

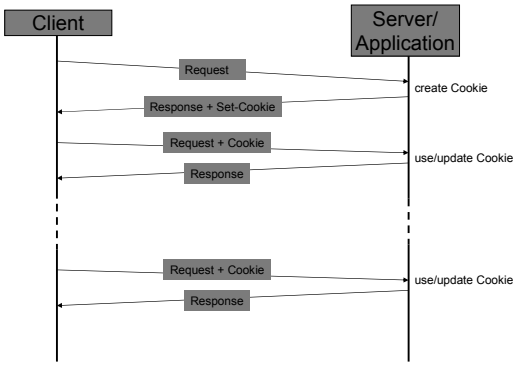
Set-Cookie (RFC 2109)

- set-cookie = "Set-Cookie:" cookie
- cookie = NAME "=" VALUE (";" cookie-av)*
- cookie-av = "Comment" "=" value
  |"domain" "=" value
  |"Max-Age" "=" value
  |"path" "=" value
  |"Secure"
  |"Version" "=" 1*DIGIT

Cookie – Syntax (RFC 2109)

- cookie = "Cookie:" cookie-version
  1*((";" | ",") cookie-value)
- cookie-value = NAME "=" VALUE [";" path]
  [";" domain]
- cookie-version = "$Version" "=" value

Preserving State



Cookies – Example I

```
GET / HTTP/1.1
Accept: */*
Host: www.example.com
```

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Wed, 03 Nov 1999 02:57:09 GMT
Set-Cookie: p_uniqid=48BpFe5tJJwdL+7QaB;
  expires=Fri, 21-Dec-2012 08:00:00 GMT;
  domain=.example.com; path=/
Connection: Keep-Alive
Content-Type: text/html
Content-Length: 15982

<html>
<head><title>...
```


Cookies – Example II

```
GET / HTTP/1.1
Accept: */*
Host: www.example.com
Cookie: p_uniqid=48BpWdpn5FAl3jq1oB
```

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Wed, 03 Nov 1999 02:57:09 GMT
Connection: Keep-Alive
Content-Type: text/html
Content-Length: 15982

<html>
<head><title>...
```


Verteilte Web-basierte Systeme – SS 2006

Section://8

Communication - WebDAV

Introduction

- **Distributed Authoring and Versioning Protocol for the World Wide Web (WebDAV)** – An extension to the HTTP/1.1 protocol that allows clients to perform remote Web content authoring operations.
- IETF Standard: RFC 2518, February 1999

Terminology

- **Collection** – A resource that contains a set of termed member URIs, which identify member resources
- **Member URI** – A URI which is a member of the set of URIs contained by a collection
- **Property** – A name-value pair that contains descriptive information about a resource
  - **Live Property** – Semantics and syntax enforced by the server: E.g. "getcontentlength" live property: length of the entity returned calculated by the server
  - **Dead Property** – Semantics and syntax are not enforced by the server:
    Server only records the value - client is responsible for maintaining the value

E.g. Creating Collections

- **Request:**
  MKCOL /martin/contacts/ HTTP/1.1
  Host: www.example.com
- **Response:**
  HTTP/1.1 201 Created

<<DEMO>>
Try this at home!

Distributed Authoring Methods

New and extension of HTTP-Methods

- **PROPFIND – retrieve Properties for a Resource (URI)**
- **PROPPATCH – set and/or remove Properties on a URI**
- **MKCOL – create a new collection**
- **GET, HEAD for Collections – as defined in RFC 2068**
- **POST for Collections – semantics as defined**
- **DELETE – removes URI (all or none sematics)**
- **PUT – replaces Get Response Entity**
  - **Properties defined on the URI may be reomputed**
  - **Put without a parent collection must fail**
- **COPY – create a Duplicate of Source-URI in the Destination-URI**
- **MOVE – move Resource to Destination-URI**
- **LOCK – take out a Lock of any Access Type on a given Resource (URI)**
  - **Method describes only those Semantics that are specific to the LOCK**
  - **But independent of the Access Type of the Lock being requested**
  - **Shared or Exclusive Lock, e.g. <D:locktype><D:write/></D:locktype><D:lockscope><D:exclusive/></D:lockscope>**
- **UNLOCK – remove the Lock identified by the Lock Token for a URI**

E.g. – Property Retrieval I

- **Request:**
  PROPFIND /martin/contacts HTTP/1.1
  Host: www.example.com
  Content-type: text/xml; charset="utf-8"
  Content-Length: …

```
<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
<D:prop
xmlns:R="http://www.example.com/contactschema/">
    <D:allprop/>
</D:propfind>
```


E.g. – Property Retrieval II

- **Response:**

```
HTTP/1.1 207 Multi-Status
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:">
<D:response>
      <D:href>http:// www.example.com/martin/contacts</D:href>
      <D:propstat>
            <D:prop xmlns:R="http://www.example.com/contactschema/">
                  <R:description>Contacts of Martin Gaedke</R:description>
                  <D:creationdate>1997-12-01T17:42:21-08:00</D:creationdate>
            </D:prop>
            <D:status>HTTP/1.1 200 OK</D:status>
            <D:resourcetype><D:collection/></D:resourcetype>
      </D:propstat>
.....
</D:multistatus>
```

Verteilte Web-basierte Systeme – SS 2006

Section://9

Communication - SOAP

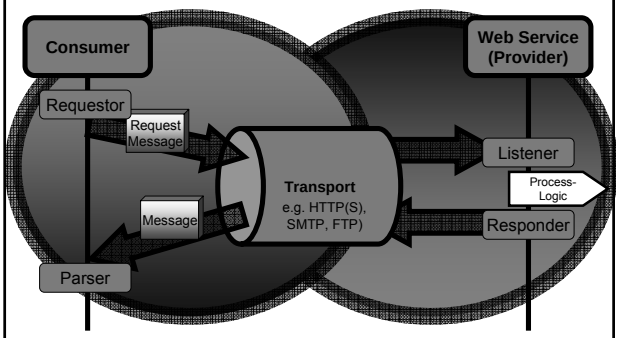
Accessing Objects

- SOAP Version 1.2
  W3C Recommendation 24 June 2003
  - Part 0- Tutorial: http://www.w3.org/TR/soap12-part0/
  - Part1: Defines Messaging Framework
  - Part2: Adjuncts (may be used in messages)
- **SOAP** provides a simple and lightweight Mechanism for exchanging structured and typed Information between Peers in a decentralized, distributed Environment
  - Formerly known as Simple Object Access Protocol
  - Does not itself define any Application Semantics, e.g. Programming Model

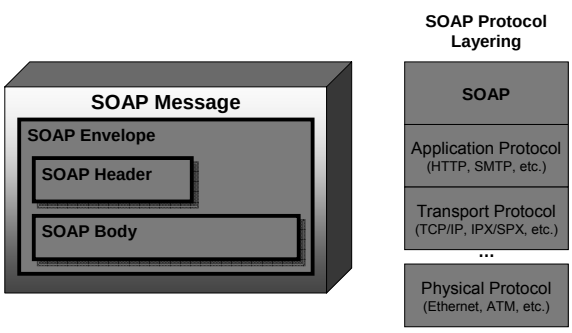
SOAP

- SOAP consists of three Parts:
  - **SOAP envelope** - Defines what is in a message; who should deal with it, and whether it is optional or mandatory
  - **SOAP encoding rules** - Define a serialization mechanism for application-defined data types.
  - **SOAP RPC representation** - Define a convention that can be used to represent remote procedure calls and responses.

General Web Service Model



Message as SOAP Message



SOAP and Client/Server…

- In order for SOAP to work, the client must have code running that is responsible for building the SOAP request.
- In response, a server must also be responsible for understanding the SOAP request, invoke the specified method, build the response message, and return it to the client.
- These details are up to you:
  your **Web application**

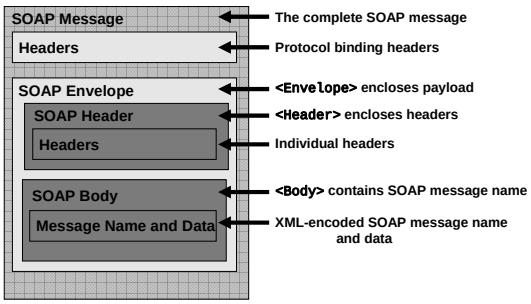
The HTTP Aspect

- A SOAP request via HTTP POST requests

```
POST /WebCalculator/Calculator.asmx HTTP/1.1
Content-Type: text/xml
...
SOAPAction: "http://tempuri.org/Add"
Content-Length: 386

<?xml version="1.0"?>
<soap:Envelope ...>
  ...
</soap:Envelope>
```


Message Structure



SOAP Message Example

- An XML document using the SOAP schema:

```
<?xml version="1.0"?>
<soap:Envelope ...>
  <soap:Header ...>
    ...
  </soap:Header>
  <soap:Body>
    <MyQuery xmlns="http://tempuri.org/">
      <n1>12</n1>
      <n2>10</n2>
    </MyQuery >
  </soap:Body>
</soap:Envelope>
```


Encoding Complex Data

- Data structures are serialized as XML:

```
<soap:Envelope ...>
 <soap:Body>
  <MyQueryResult xmlns="http://tempuri.org/">
   <result>
    <Description>Plastic Novelties Ltd</Description>
    <Price>129</Price>
    <Ticker>PLAS</Ticker>
   </result>
  </MyQueryResult>
 </soap:Body>
</soap:Envelope>
```


Example of a SOAP Request

- SOAP message over HTTP-POST:

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml;
charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<soap:Envelope
       xmlns:soap="http://www.w3.org/2001/09/soap-envelope">
   <soap:Body>
      <m:GetLastTradePrice xmlns:m="Some-URI">
         <symbol>DIS</symbol>
      </m:GetLastTradePrice>
   </soap:Body>
</soap:Envelope>
```


A SOAP Response

- SOAP response over HTTP

```
HTTP/1.1 200 OK
Content-Type: text/xml;
charset="utf-8"
Content-Length: nnnn

<soap:Envelope
       xmlns:soap="http://www.w3.org/2001/09/soap-envelope">
   <soap:Body>
      <m:GetLastTradePriceResponse xmlns:m="Some-URI">
         <Price>34.5</Price>
      </m:GetLastTradePriceResponse>
   </soap:Body>
</soap:Envelope>
```

Example of a SOAP Error

- SOAP response over HTTP

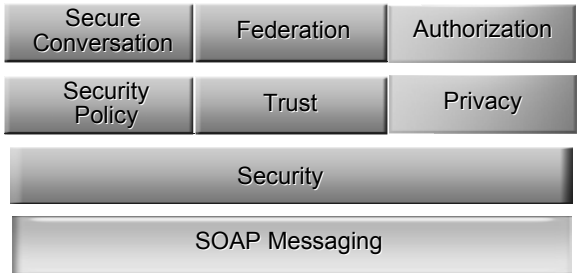
```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<soap:Envelope
       xmlns:soap="http://www.w3.org/2001/09/soap-envelope">
  <soap:Body>
     <soap:Fault>
       <faultcode>SOAP: MustUnderstand</faultcode>
       <faultstring>SOAP Must Under Error</faultstring>
     </soap:Fault>
   </soap:Body>
</soap:Envelope>
```


Security and Features

- In context of HTTP – builds on existing security
  - HTTPS
  - X.509 certificates
- Developers explicitly choose which methods to expose
- Extensibility - **the major strength** of SOAP
  - E.g. check the WS-* specifications http://msdn.microsoft.com/webservices
  - Cf. WS-Security Roadmap

WS-Security Stack



SOAP – Industry Support

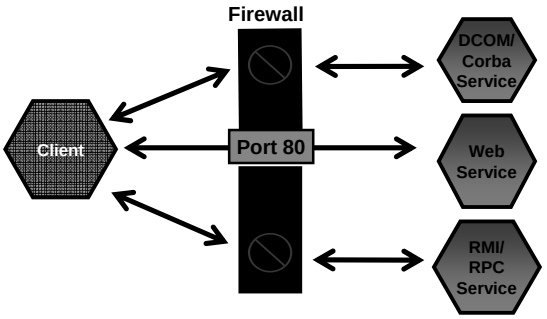
- DevelopMentor Inc.
- Digital Creations
- IONA Technologies PLC
- Jetform
- ObjectSpace Inc.
- Rockwell Software Inc.
- SAP
- Compaq
- Intel
- Microsoft
- Rogue Wave Software Inc.
- Scriptics Corp.
- Secret Labs AB
- UserLand Software Inc.
- Zveno Pty. Ltd.
- IBM
- Hewlett Packard
- Many more…

Verteilte Web-basierte Systeme – SS 2006

Section://10

Security Concerns

Security And Web



Core Considerations

- **Code Access Security**
  - Protection against malicious mobile code
- **Security Model is based on Permissions**
  - **Permission** – A rule the runtime must follow to check
  - Typical rules that may influence behavior
    - Code access permissions
    - Identity permissions
    - Role-based permissions

Role-based Security

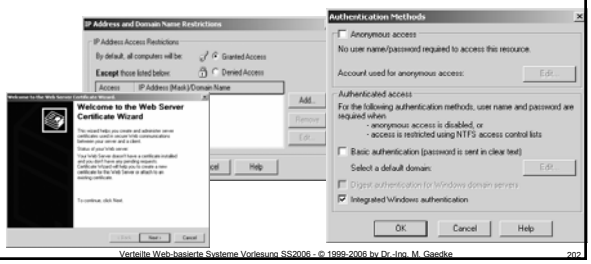
- **Principal** – a user or an agent that acts on the user's behalf
- **Role** – Metadata assigned to user
  - Usually related to actors of a business process
  - E.g. roles in Hotel: Guest, Concierge, Director, etc.
- Roles are a part of the Analysis Document
  - Note: Define all Roles in a dedicated Role-Model

Authorization & Authentication

- **Authorization** – Determine whether an identity should be granted the requested type of access to a given resource.
- **Authentication**
  - Is the process to determine the identity of a person or program by checking characteristics (e.g. asking for userid and password)
  - Different approaches and mechanisms exist – try to focus on standards
  - Note: **Apply the strongest applicable standard**

Example: Server Security

- Many security mechanisms are supported by standard Web Servers – there is no excuse if these are not used



Authentication Mechanisms

- Mechanisms – use carefully
  - **Internet Protocol Security (IPSec)**
    - Client addresses must be known a priori
  - **Basic**
    - Username and password are sent in plain text
    - Don't use!!!
  - **Digest**
    - Uses secure (Crypto) hash algorithm
    - Not supported on all platforms
  - **Integrated Kerberos or NTLM**
    - Good candidate for intranets – but does not support Proxy-Server or Firewalls
    - Not supported on all platforms

Authentication Mechanisms

- Mechanisms – you should use
  - **Basic over SSL**
    - Similar to Basic, but encrypted channel (Secure Socket Layer)
    - User know that this is safe
    - Drawback: SSL slow
  - **Client certificates**
    - Good viable option to SSL
    - Requires certification authority
    - Issuing certificates is secure (Public Key Mechanism)
    - Automation by Browser
    - Approach for Web Services?

Authentication Mechanisms

- Mechanisms – with focus on authenticating real people
  - **Passport and related identity approaches:**
    - Could also be used to authenticate machines or applications
    - Single-Sign On (SSO) approach
    - Enhances B2C applications
- Many future scenarios will be based on WS-Federation or Liberty Alliance

Verteilte Web-basierte Systeme – SS 2006

Section://11

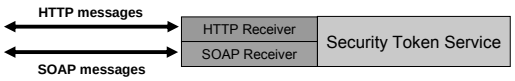
Federation Technologies

Federations Specifications

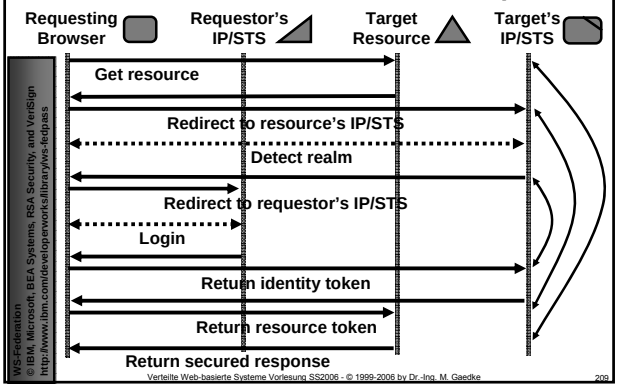
- Mechanisms that allow different security realms to federate
  - This includes brokering trust of identities, attributes, and authentication between participating partners
  - Examples: Single Sign On (SSO), pseudonyms etc.
  - Realized by protocols and rules
- Protocols and rules for realization of federations
  - WS-Federation
    - http://www-106.ibm.com/developerworks/webservices/library/ws-fed
  - Liberty Alliance
    - http://www.projectliberty.org/resources/specifications.php
  - SAML
    - http://www.oasis-open.org/specs
  - Shibboleth
    - http://shibboleth.internet2.edu/

Example WS-Federation

- Two profiles (scenarios and protocol specifications)
  - Active Requestor Profile (ARP) – Smart/Active clients using SOAP
  - Passive Requestor Profile (PRP) – Passive clients (Browser) using HTTP and JavaScript
  - Realized by using HTTP and WS*-protocols

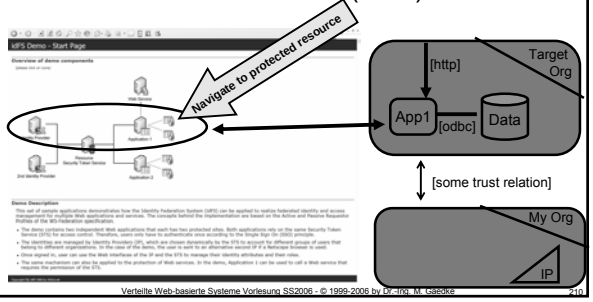


WS-Federation PRP Sample

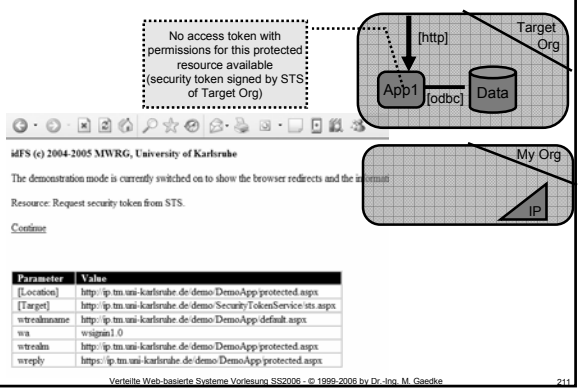


Example – WAM Applied

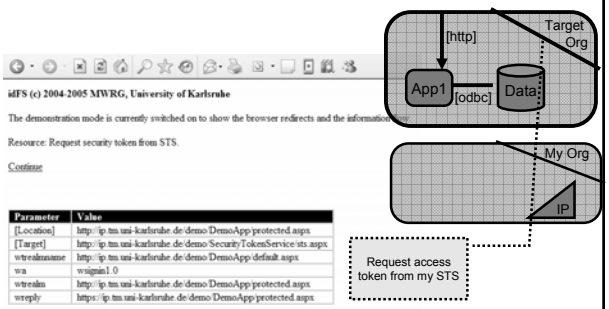
- http://ip.tm.uni-karlsruhe.de/demo/demo/
- SSO with WS-Federation (PRP)



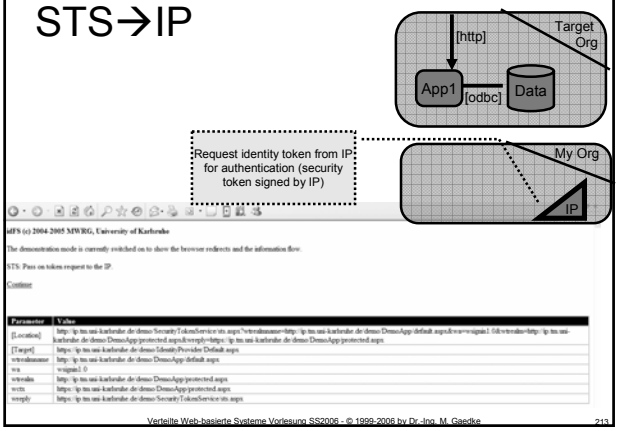
Call Protected Resource



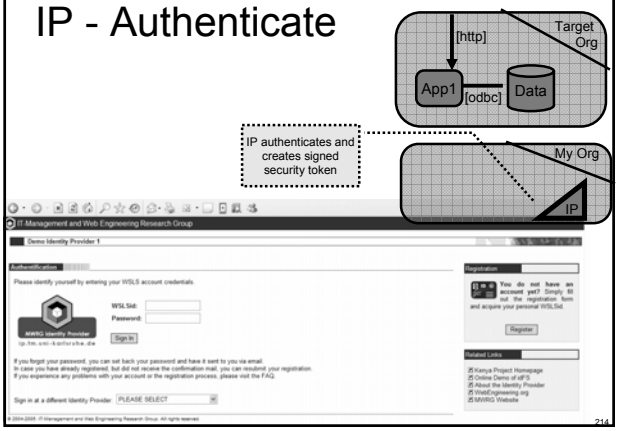
Protected Resource → STS



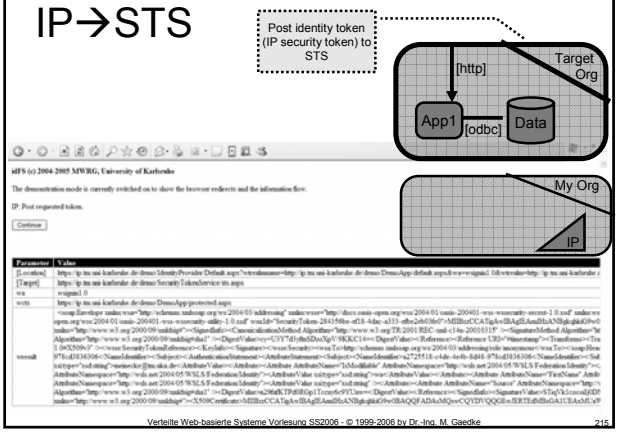
STS→IP



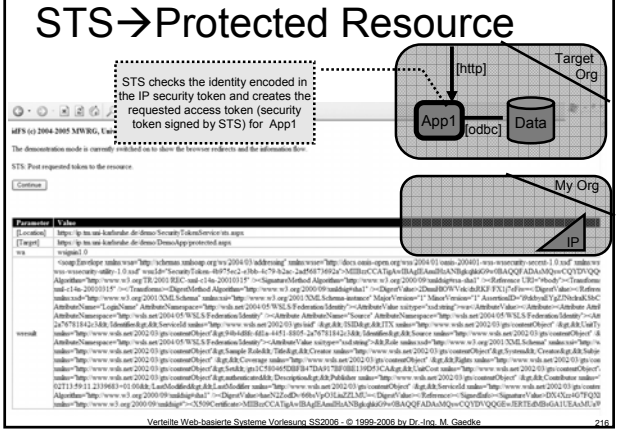
IP - Authenticate



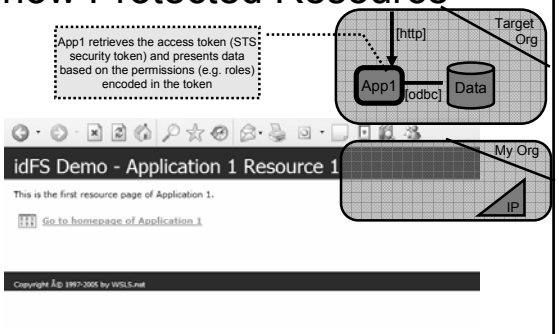
IP→STS



STS→Protected Resource



Show Protected Resource



Verteilte Web-basierte Systeme – SS 2006

Chapter://8

Further Readings

Literature

- M. Gaedke, M. Nussbaumer, O. Jung, M. Diekmann, Kapitel 6: Implementierungstechnologien für Web-Anwendungen, in Web Engineering – Systematische Entwicklung von Web-Anwendungen, G. Kappel, B. Pröll, S. Reich, W. Retschitzegger (Hrsg.), dpunkt.verlag
- Chapter 3, 7: T. A. Powell, Web Site Engineering, Prentice Hall PTR
- Chapter 15, 16, 28, 29: Ian Sommerville, Software Engineering, Addison-Wesley
- Chapter 5-7: Scott Seely, SOAP – Cross Platform Web Service Development Using XML, Prentice Hall
- Chapter 3, 4, 6, 7, 8, 9: William L. Oellermann, Jr., Architecting Web Services, apress
- Chapter 10: I. Jacobson, G. Booch, J. Rumbaugh, The Unified Software Development Process, Addison-Wesley, 1999
- Chapter 9, 10: S. W. Ambler, Process Patterns – Building Large-Scale Systems Using Object Technology, Cambridge University Press
- R. Tolksdorf, XML und darauf basierende Standards: Die neuen Auszeichnungssprachen des Web, in Informatik Spektrum, vol. 22(6), S. 407-421, 1999.
- V. Turau, Techniken zur Realisierung Web-basierter Anwendungen, in Informatik Spektrum, vol. 22(1), S. 3-12, 1999.
- U. Bettag, Web-Services, Informatik Spektrum, vol. 24(5), S. 302-304, 2001.

==========

Further information available at Lecture Web Site

==========

Literature

- T. Berners-Lee, Information Management: A Proposal, CERN, Web Page 1990. http://www.w3.org/Proposal.html
- T. Bray, D. Hollander und A. Layman, Namespaces in XML, World Wide Web Consortium, Recommendation 1999.
- T. Bray, J. Paoli und C. M. Sperberg-McQueen, Extensible Markup Language (XML) 1.0, World Wide Web Consortium (W3C), Recommendation REC-xml-1998021, 02.10.1998 1998. http://www.w3.org/TR/1999/REC-xml-1998021
- J. Clark und S. DeRose, XML Path Language (XPath) Version 1.0, World Wide Web Consortium, Recommendation 1999.
- M. L, Creech, Author-oriented Link Management, The 5th International World Wide Web Conference, Paris, France, 1996.
- J. December und N. Randall, The World Wide Web unleashed, 1st ed. Indianapolis, IN: Sams Pub., 1994.
- A. Denning, ActiveX controls inside out. Redmond, Wash.: Microsoft Press, 1997.
- S. DeRose, E. Maler und D. Orchard, XML Linking Language (XLink) Version 1.0, World Wide Web Consortium Proposed Recommendation, 2000.
- M. Dubinko, J. Dietl, R. Merrick, D. Raggett, T. V. Raman und L. Bucsay Welsh, XForms 1.0, World Wide Web Consortium (W3C), Working Draft WD-xforms-20010608, 08.06.2001 2001. http://www.w3.org/TR/2001/WD-xforms-20010608

==========

Further information available at Lecture Web Site

==========

Literature

- R. Fielding, J. Gettys, J. Mogul, H. Frystyk Nielsen, L. Masinter, P. Leach und T. Berners-Lee, *Hypertext Transfer Protocol -- HTTP/1.1*, World Wide Web Consortium, RFC 2616, 01.07.1999 1999. http://www.w3.org/Protocols/rfc2616/rfc2616.html
- M. Gaedke, *ActiveX: Componentware im Internet*, in HMD, Theorie und Praxis der Wirtschaftsinformatik, vol. 34. Jahrg(Heft 197), 1997.
- T. Pixley, *Document Object Model (DOM) Level 2 Events Specification*, World Wide Web Consortium 2000. http://www.w3.org/TR/DOM-Level-2-Events/
- T. Ritchey, *Java!* Indianapolis, Ind.: New Riders Pub., 1995.
- J. Robie, J. Lapp und D. Schach, *XML Query Language (XQL)*, World Wide Web Consortium, Homepage 07/07/1999 1998. http://www.w3.org/TandS/QL/QL98/pp/xql.html
- World Wide Web Consortium
  - *Extensible Markup Language (XML) Specification: W3C Working Draft 21 Apr 1996*. http://www.w3.org/, 1996.
  - *CSS2 Specification*, http://www.w3.org/, W3C Recommendation REC-CSS2-19980512, 01.07.1999 1998a.
  - *Schema for Object-oriented XML*, World Wide Web Consortium, Technical Report NOTE-SOX-19980930, 30.09.1998 1998b. http://www.w3.org/
  - *Extensible Stylesheet Language (XSL) Specification: W3C Working Draft 21 Apr 1999*. http://www.w3.org/TR/WD-xsl/, 1999a.

==========

Further information available at **Lecture Web Site**

==========

Web References

- URLs
  - http://msdn.microsoft.com/
    - http://msdn.microsoft.com/webservices/
  - http://w3.org/
  - http://www.oasis-open.org/