

# Dynamic webpages

**Database connectivity, Blog/Wiki/RSS, CMS,  
Personalization**

Institute for Information Processing and  
Microprocessor Technology (FIM)  
Johannes Kepler University Linz, Austria

E-Mail: [sonntag@fim.uni-linz.ac.at](mailto:sonntag@fim.uni-linz.ac.at)  
<http://www.fim.uni-linz.ac.at/staff/sonntag.htm>



# Questions?

**Please ask immediately!**



- From static to dynamic webpages
- Database connectivity
  - Separation, type of integration
- Various techniques:
  - Blogs: More than personal diaries?
  - RSS: News the fast way?
  - WIKI: Free-for-all webpages?
- CMS: Content Management Systems
  - Reasons
  - Functionality
- Personalization / Adaptivity
- Technological, organization and legal issues



# From static to dynamic webpages

- Static webpages are rather "easy"
  - Hard- & Software requirements very low
  - Build (or let build) once; update seldom
- But they cause several problems, e.g.
  - Frequently changing content is a huge burden (archives?!?)
  - The same page for all (but different) target groups
- Therefore "dynamic" webpages are required
- They come in several forms:
  - Dynamic presentation: "Animated" content (Flash, JScript, ...)
    - » No real difference to static webpages; not handled here
  - Dynamic layout: Assembled per request from individual parts
    - » Solves some problems; not handled here (prerequisite for next)
  - Dynamic content: Content is changed frequently; manually or from a database; interactive modifications



# Advantages of dynamic content

- Content can be exchanged easily
  - Important e.g. for webshops (products, prices, ...)
- Integration with other software possible
  - Retrieving availability info directly from business software
- Personalisation for individual users
  - From visual over navigation to content adaptation; see later
- Interaction with visitors
  - Visitors can add content themselves
  - Specific requests can be answered, e.g. search results
- Maintenance costs reduction
  - Elements can be changed more easily and consistently
- Immediate delivery of electronic products



# Problems of dynamic content

- Links: Depending on technology linking might be impossible
  - » Links from other sites to a specific local place
  - Or linking to certain pages (e.g. POST responses)
- Changing layout/content can be distracting to visitors
  - Especially dangerous if usage patterns change
  - » "It used to work like this..., but no longer!"
- Changed/new content must be produced, approved, ...
  - Might be similar to editing a newspaper!
- Hard- & software requirements increase
- Customization almost always required
  - Assembling/programming the features desired/required
- Supervision of interactive content
  - Verifying content of visitors, deleting unwanted one, ...
- Search engines might not be able to index all pages



# Database-driven webpages

- "Publishing" the database to the web
  - Technically relatively simple (e.g. ODBC, ADO, JDBC)
    - » Practically a lot of work if not **ONLY** presentation but interaction!
- But there are also some issues to think about:
  - Will it be "the" database or a copy? How often synchronized?
  - Access restrictions required or which subset is published?
  - Can the information be misused (e.g. automatic extraction)?
  - Legal issues (privacy, third party rights on content, ...)
  - Transactions needed/supported by your database?
  - Performance of database and connection sufficient?
  - Will it be a mixed (static pages/parts from files and dynamic data from the DB) or DB-only (everything is in DB) solution?
  - Browser-dependent pages needed/desired?
  - Who may see / change what parts?



- Ever more communication takes place using XML
  - Therefore it must also be stored!
- Dynamic webpages: Many formats are already XML
  - Separating the content from the presentation usually ends with the data being in XML (→ storage needed)
    - Content here ≠ DB content; rather: page layout, static parts etc.
    - » Presentation through XSL requires XML as base data
  - Frameworks often define a website through XML
- Storing "directly" in XML avoids the need for constant conversions native ↔ XML for communication
  - Other option: Store it as text
    - » But then no database features like selection possible!
- Integrating data from several sources is easier in XML





- XML mappings: Based on a relational DB
  - Each element is a separate row
  - Good for weakly nested and rigid formats
    - » When XML is just a wrapper for data; transport formats
  - Available for most databases; high speed; referential integrity
- Native XML DB: Data is stored directly as XML
  - » Usually based on relational or OO databases at the lowest level
  - Good for highly nested and variable documents or fragments
    - » XML is the "main" format; documents (e.g. XHTML)
      - Usually from external sources to be stored and found, but not created from the database
  - Rather rare; performance varying (bad for finding specific elements, very good for e.g. XQuery)

The decision is very important, but depends largely on the kind of data to store and the queries!



# Various communication methods

- New and "alternative" communication channels exist:
  - They stem from private/non-commercial use
  - Use in special areas for E-Business still possible!
- Blogs, Wiki: Nothing fundamentally new
  - These are just ordinary webpages
    - » But: By everyone and from everywhere accessible & editable
  - Difference: Method of creation and usage pattern
- RSS is "new": Content interchange between websites
  - Also allows aggregation of content similar to a newsreader
    - » Subscription to certain "channels" (reality: content filtering)
  - Brings some "active" part to the web
    - » With additional software notifications on new additions possible
- Common advantage: Technologically simple and cheap
  - » Drawbacks: "Yet another system", suitable for special tasks only



- Blog = Weblog
  - Periodic posts on a webpage; usually in reverse chron. order
- Usually handled completely through HTML/browsers
  - Administration, editing, reading, searching, ...
    - » No additional client software; accessible from everywhere; integrates well with firewalls/access schemes, easy integration,...
  - Less intrusive than E-Mail: For non-time-critical information
- A "simple" tool: Avoids complicated decisions/interfaces/...
  - Mostly one topic and one main thread (=Consecutive posts)
    - » Posts might have comments, which could have again comments
      - Then similar to discussion groups/forums!
- "Frequent" changes expected
  - Several times/day up to every few days
- Many blogs are rather informal and personal



- PR: Providing customers with information on updates, new functionality, development state etc.
  - Also possibility to informally look for customer's state of mind
  - "Personal face" of the company
    - » E.g. "private" blog of executives
      - Danger: Changes in management
- Knowledge base: Ideas, insights, problem solutions are collected and easily searchable
  - » But low structure and annotation can be problematic!
  - Also a good kind of "introduction" for new employees
  - Can serve as a handbook to look up details
    - » Through comments related areas can be included, unlike a typical "knowledge base", where links must be added explicitly (and are therefore often not added!)



# Application fields

- Collaboration tool: Project progress, design issues etc.
  - Asynchronous and location-independent
  - Project = smaller group → Low structure less of a problem!
    - » Informality within a group matches the informality of a blog
- Blackboard: Improves personal relations between employees
  - Anniversary, births/weddings, lost&found, this weeks menu, ...
- FAQs: Pre-version for later extraction and/or consolidation
  - Everyone can easily add to it



# Blogs vs. forums

- Blogs are suitable for few posters (1:N), forums also support huge numbers of active participants (M:N)
- Blog = One topic; Forum = Many top-level items
  - Blog = One thread; Forum = Many threads
  - Blog has a much simpler structure (trunk + small branches)
    - » Forum: Full-blown and complete tree of arbitrary depth
- Forums have no inherent notion of time
  - You can reply to any post at any time
  - Blogs are temporally ordered (comments not necessarily)
- Blogs usually support cross-links to other entries; even on other blogs ("trackback capability", "permalink")
- Blogs are more of a one-way message (speakers corner in hyde park); forums are group communication (club meeting)



- RSS = Really Simple Syndication
  - Or: Rich Site Summary, RDF Site Summary, ...
- Lightweight format for sharing headlines and web content
  - Based on XML; separate specification
  - Provides the content in short (or long) form
- Typically blogs provide their data as RSS too
  - This enables other websites to copy or link to the content
    - » Can be presented in own format/presentation (XSLT!)
- Aggregators: Combine several RSS feeds into a single one
  - Monitoring many blogs and notifying the user of new entries
    - » Can also filter them according to various methods
  - Still based on polling (monitoring the RSS file for changes)
  - Can be offline (=reader) or online (=dynamic webpages updating themselves when other sites change)





## RSS (V 2.0) example

- Take care: Several version of RSS exist  
→ Some are widely differing (e.g. XML vs. RDF)!

```
<rss version="2.0">
  <channel>
    <title>RSS Example</title>
    <link>http://www.fim.uni-linz.ac.at/</link>
    <description>An RSS example feed.</description>
    <language>en-uk</language>
    <item>
      <title>RSS for E-Business</title>
      <link>http://www.fim.uni-linz.ac.at/LVA/E-Business/</link>
      <description>A sample feed.</description>
    </item>
  </channel>
</rss>
```





- A collaboratively edited website ("virtual whiteboard")
  - Each visitor can change the webpage
  - Editing takes place online (simple editor)
  - Editing: Usually not plain HTML, but a much more simple form of markup (differing for each Wiki system!)
  - Sometimes plugins/JavaScript for a (nearly) full HTML editor
- Links are "automatic"
  - Creating a link produces a "broken link", which can then be easily filled by a new page (tries to avoid orphans)
- Version control is an integral feature
  - Going back to a previous version (before any edit) is easy
    - » Difficulties: Branching
- Basic idea: No control over users
  - Everybody can change everything in every way
  - If it is wrong or vandalism occurs → version control!



# Application areas

- Similar to Blogs, but can be more structured
  - More similar to: "Conventional webpage + Online-Frontpage"
  - Better suited for a set of related areas
    - » Blog: Small, individually useful and self-contained elements
  - Examples: Knowledge base, collaboration tool, ...
- Collaboratively authoring specifications/documents
  - Asynchronous and distributed brainstorming
  - Problem: Tracking changes, history, keeping up-to-date, ...
    - » Possible for "small" Wikis, but difficult for large ones
- FAQs: Every new question or improved answer can be inserted immediately
  - Editing probably restricted to a group; not for general use

"Commercial" problems: Liability, reputation, hierarchy, ...



## Inside vs. outside use

- Inside: Using Blogs, Wiki, RSS for employees/departments
  - Less problems: Sanctions on misuse available
  - Employees can be "motivated" to use them
  - Can reduce internal mass- & fun-mailing
  - Small version of knowledge management tool
- Outside: Communication with customers or the public
  - Can be difficult: Customers **expect** regular updates!
    - » And visitors might actually try to implement "interactivity"!
  - Leaking of company secrets or legal problems
  - Might conflict with public image
  - Pure marketing won't work: Visitors expect useful content
  - Confusion: What is in forums, FAQ, Blog, or Wiki?
    - » These are similar concepts; clear division of content difficult
  - What is "the company view"? Is there really a single one?



# Content Management Systems (CMS)

- Content: Any unit of information
  - Everything to be published: text, image, video, documents, ...
    - » Document: Several pieces of information together
    - » Information: A single useful and self-contained element
- Managing: How to get the content in and out of the web
  - Consists of rules, processes and workflows
    - Therefore different from a WIKI!
      - » Rules: Not everyone is allowed to do everything
      - » Processes: Automating tasks
      - » Workflows: Approving content for publications
  - Centralized system and decentralized content "creation"
    - » If the webmaster creates all pages → no real need for a CMS!
    - » Use an offline editor and when everything is fine, publish it
- System: Tools and software to support and automate this
  - Ensures common visual, technical and handling metaphors



# Content Management Systems (CMS)

- CMS are intended for:
  - Centralized delivery (=webserver) and rules
    - » Everything will look similar, act similar and can be administrated by single instance
  - Decentralized content creation by "non-technical" staff
    - » Potentially everyone can add content without special software, HTML markup knowledge, design experience, ...
  - Fully automated procedures: Archiving, search engine integration, navigation creation etc.
    - » Get all "for free" and at least partially automated what you would have to do manually otherwise



# Why a CMS?

- Providing a large and dynamic website is like authoring a regularly appearing newspaper
  - This is usually not what companies are good at!
    - » They produce something or provide various services
  - One single technician is no longer appropriate for this
- Ensuring a common visual appearance, though many persons create/change content
  - All actions also happen live (no stopping the server)
  - Upon publishing it must be "perfect" immediately
  - Avoiding broken links even though everything changes often
- Integration of "online tools": Blogs, Wiki, forums, search engines, contact forms etc.
  - Alternative: Separate & different tools; manual integration



# Advantages of CMS

- Fresh, consistent information and flexible website
  - Process of publication is sped up
- Separating tasks and assigning them to specialists
  - Serving pages separated from layout and content creation
- Asserting rules and procedures
  - Content for some areas must be approved by 1..n persons
  - Depending on classification, content is archived after x days
- Audit trail and versioning
  - Complete archive of all content in all versions
  - Enables integration of statistics with individual content
    - » Also for individual creators or areas
- Incentive for visitors to come back
  - Continually new content improves customer experience and public relations, supports sales and after-sales care





# Content Management Systems (CMS)

- Tasks a CMS must support:
  - Creation: Usually only for the "online text", i.e. HTML
    - » But in simple form: Technical aspects (=tags, ...) hidden!
    - » Otherwise: Import of all other content types/elements
  - Editing: Different persons work on the same information
    - » Modifying, enhancing and arranging content of others
    - » Directly online, i.e. no media breaks
  - Managing: Process of creation, approval, publishing, archiving
    - » Enforcing and automating the process
    - » Notifying the correct persons and preparatory steps
    - » Version control for all content and actions
  - Publishing: Deciding when, where, and whether to go live
    - » Workflow from beginning to end of content lifetime
  - Presentation: Delivering the content in one or more formats



# CMS process elements (1)

- Creation and editing:
  - Requires a custom and easy-to-use presentation "format"
    - » "Bold", may be shown as bold, different colour, higher volume, ...
  - Single source: Everything derived from single complete and authoritative source (typically a database)
  - Requires locking: Only one person may change at any time
  - Links: Must work also after restructuring, moving etc.
    - » This usually requires a custom "link" format which is translated into hyperlinks only on actual rendering!
- Management:
  - Version control for accountability, backup etc.
    - » Some parts should also be available to authors (see Wiki!)
  - Security: Not everyone is allowed to do anything
  - Integration of other software + reporting (active and passive)



# CMS process elements (2)

- Publishing: The flow of the content
  - Customization area: Must be tailored to the company
    - » Take care of staff changes, organizational restructuring, ...
    - » Workflow might require a lot of modelling
  - High initial investment; if done properly → real savings!
- Presentation: What the user sees
  - Stylesheets: Not only CSS, but also "arrangement" templates
  - Extensibility: Plugins for new functionality
  - Multiple formats: HTML + Printed, PDF, WAP, ... (needed?)
    - » What about exporting/importing the content?
  - Usability, accessibility, browser support, speed, HTML validity
  - Client side functionality necessary?
    - » JavaScript (AJAX), JRE, Flash, RealPlayer etc. required?
      - Fallback if not present?
  - Metadata: In webpages or external; which standard?



- The basic framework is usually comparable
  - What often is different are the additional "applications"
  - These should be a major part of a CMS selection!
    - » Including the possibility/effort for adding custom parts
- Examples:
  - Interaction: Blog, chat, forum, groupware, guest book, polls, surveys, tests
  - E-Mail integration: Online E-Mail, mail sending, newsletters
  - Management: Contacts, calendar, expenses, project tracking
  - Distribution: Files, open positions, classifieds, image galleries
  - Other: FAQ, help desk, bug reporting, site map, web services
  - Personalization: "My page", search engine, RSS
  - Internal: Link management, usage statistics, online editors
  - E-Commerce: Shopping cart, payment solution (-interface)



# CMS shortcomings

- CMS alone is no solution: The content must still be created!
  - The most important element of a CMS is **NOT** the software, but the **authors of the content!**
  - The complete company culture might need to be changed!
    - » So part of everyone's job is adding/reviewing/approving content
- Publications require editorial expertise
  - This requires specific skills (and perhaps staff)
- A complete workflow must be defined and followed
  - Can be expensive!
- Full-featured CMS are quite complicated
  - Expect training requirements for IT, editors, ...
  - CMS often are frameworks: Extensive customization needed
- No real interoperability standards between CMS
  - Content in one system is easily "lost" (vendor lock-in)



# Introducing a CMS to a company ...

- Most CMS require **lots** of work for realizing your own idea
  - What you usually get are specific modules
    - » Others must be programmed; all must be integrated
    - » Advantageous: Start with few modules and add more later
- Separation of technology from content
  - IT manages the CMS, marketing (...) produces the content, and management defines rules and approves content
- Software is only a fraction of the total cost:
  - Author work time, management by IT, resources etc.
- Surrounding issues
  - Skills & resources required: Available or training needed?
  - Scalability, external constraints: Current IT infrastructure, ...
  - Training, documentation, warranty, maintenance, costs

**First issue: What business function should be fulfilled?**





- Also called "adaptable", "adaptive", ...
- Main idea: Create a model of the user and present that part of all content available he is actually interested in and do it in a way suitable for him
- Consists of three major elements:
  - **User model: What the system thinks the user finds interesting**
    - » Created explicitly: Asking the user (e.g. forms)
    - » Created implicitly: Monitoring the users actions (e.g. cookies)
  - **Content model: What the computer thinks a resource is about**
    - » Created explicitly: Provided in metadata (internal or external)
    - » Created implicitly: Derived from content, user responses etc.
  - **Adaptation engine: "Matching" both models**
    - » Observing the user and deriving the user model
    - » Comparing the user model to the content model and deciding upon the actual content so serve

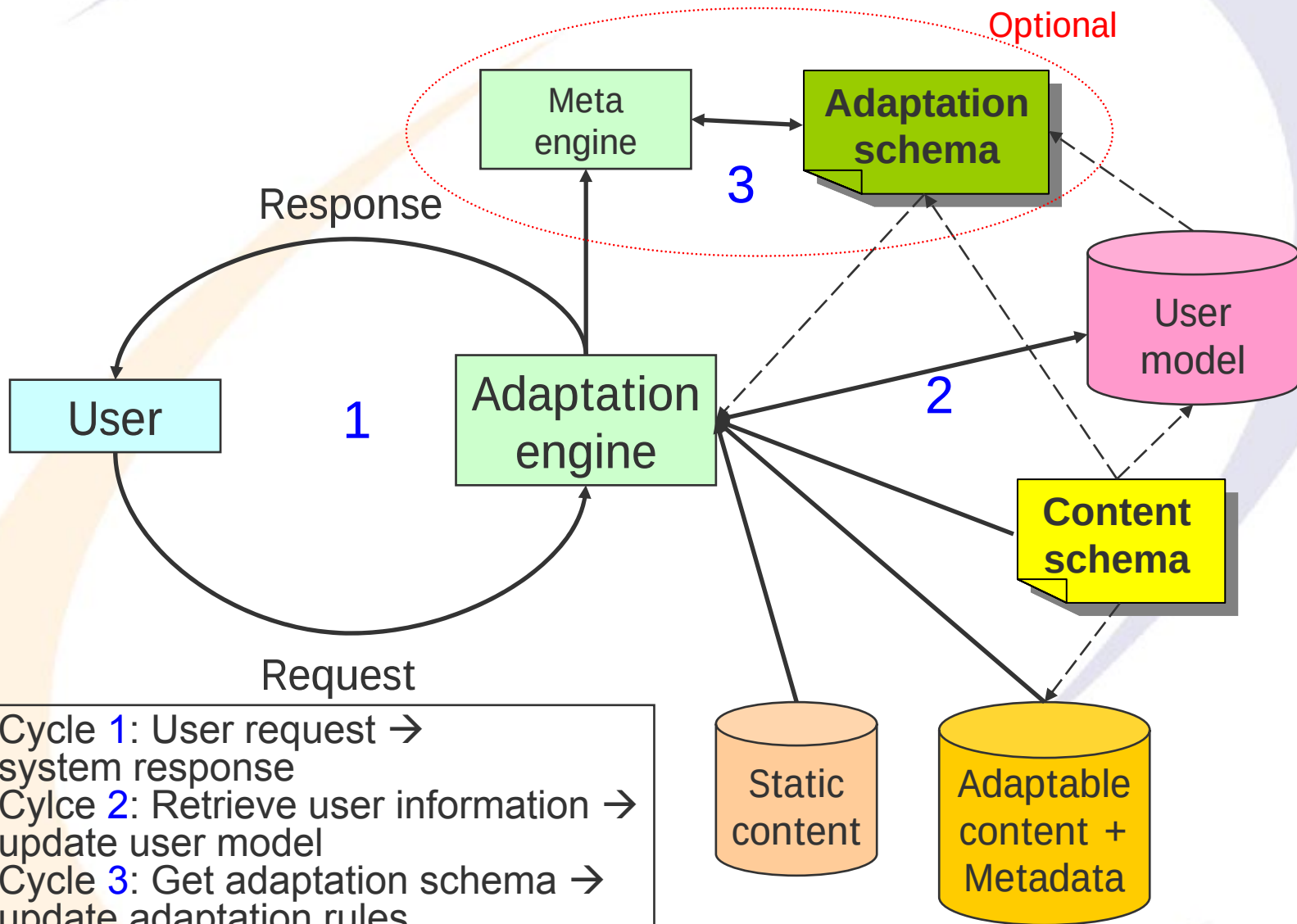




# Kinds of adaptation

- Adaptable: Explicit user profile (form)
  - Advantages: The user really knows what she is interested in; works right from the start; simple; content classification is usually identical; rules possible
  - Disadvantages: Unless the user changes her profile, changes will go unnoticed; (large?) up-front effort required; classification must match exactly
- Adaptive: Implicit user profile (observation)
  - Advantages: Learns from actions and incorporates changes of interests; immediate begin; continuous improvement
  - Disadvantages: Slow start; no rules; complex; computation intensive; depends completely on algorithm; user actions ambiguous
  - Classification can be defined independently

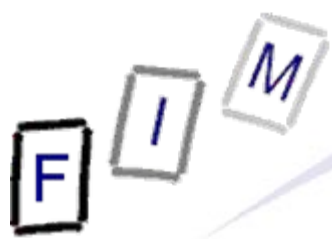
# The adaptation cycle



- Cycle **1**: User request → system response
- Cycle **2**: Retrieve user information → update user model
- Cycle **3**: Get adaptation schema → update adaptation rules



- Some adaptation methods require annotation of materials
  - This is "classical" metadata: Data about other data
- Often accompanied by an ontology
  - Describes the concepts used and their relation to each other
  - Orthogonal to: Explicitly provided or automatically derived!
  - Drawback of an ontology: It is content specific and must be created anew for different content and maybe expanded for new one (could mean re-annotating all existing content!)
  - Advantage: Classification is consistent and adaptation rules are easier to determine
  - User model is usually directly based on ontology, so some kind of it is always needed (if not explicitly, it gets created implicitly through the evolving user models)
- Adding metadata important for good personalization!



# Drawbacks of adaptation

- Identifying the user's interests works only slowly and results are not always correct or complete
- No standardisation: Content suitable for one adaptation engine might not work with another one
  - In E-Learning MD standardization exists
    - » See later; however, these are not (yet?) widely/actively used
- Creating the adaptation is difficult: When to do what?
  - Automatic systems: Depend on other users actions or heuristics; all of which might (and will sometimes) also fail
    - » Automatic systems are sometimes very difficult to understand
- Adaptation means often removing uninteresting things
  - This requires a lot more content to be there to still be useful!
- Pages visited might suddenly look different
  - Breaks expectations, but e.g. also bookmarks!



# Personalization in E-Business

- Usage of web personalization very low
  - Different to e.g. E-Learning!
- But used very successfully for certain small parts:
  - Recommending things based on ratings (e.g. films, books)
    - » Or cross-/up-selling
  - User analysis (no immediate influence; for statistics, advertisements, special offers and recommendations later on)
  - Explicit configuration of layout, elements, content
    - » E.g. newspapers or various portal sites
  - Targeted advertisements (but real success doubtful!)
  - Technical changes (not content!): Fonts, colours, language, user name, account details, ...
  - Reminders and notifications
- Common: Simple, explicit configuration, predefined rules
  - Exception: Collaborative filtering and variants!



# The technological view

- Dynamic webpages require more investment
  - Most software already supports them
  - More hardware power is needed: pages must be created on demand and each time anew
- Caching much more difficult
  - A page is created for a single user and cannot be reused
  - Caching therefore integrated into the server: knows parts valid for more users, static pages, ...
  - Sometimes special languages (e.g. Oracle) for distributing dynamic webpage creation and assembly with static parts
- Database required
  - High speed and transaction support required
  - Same or different (fast and dedicated network) computer?



# Base systems/languages

- ASP: Active Server Pages (Microsoft)
  - Third-party products support them on other platforms
    - » But note, that not all components (COM, .NET, ...) are available!
  - Supports various programming languages
    - » Positive: Integrating modules from various sources
    - » Negative: Maintenance can be difficult
- PHP: Scripting language
  - Very popular, but webpages scripting is approx. the only area
  - Many libraries available for free
  - Itself a programming language
    - » "Immature": Object orientation just recently added
    - » Insecure: E.g. no type safety
  - Very easy to learn and use





# Base systems/languages

- JSP: Java Server Pages (Sun)
  - Java: Platform independent, large developer pool
  - Many extensions, frameworks, libraries, ... available for free
  - Secure programming language
  - Will take some time to learn, as no "complete set", but rather a collection of various specifications
    - » Java Servlets, Java Server Faces, STL, JMS; Struts, ...
- Python, Perl: (Interpreted) scripting/programming languages
  - Used also for applications etc.
  - Online library repositories
- ColdFusion (Macromedia): Tag-based scripting language
  - Based on J2EE: Portable
  - Multi-platform; for (very) large projects



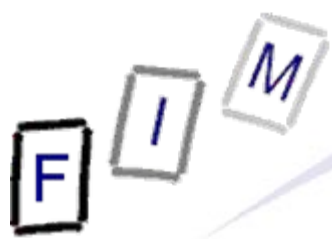
# The organizational view

- Introducing dynamic webpages is not (only) an IT task
  - » Or rather: only in rare and usually bad instances!
  - Content designer: Identifying the topics/issues to publish
  - Content creators: Creating the actual content
  - Approving content: Verifying prospective content against legal issues, secret information and overall policy/image
- IT: Setup of system and administration
  - NOT: Identification, creation, approval of content!
- This requires top management commitment
  - New jobs created/existing job descriptions modified
  - Long-term commitment: "Static" dynamic webpages are worse than just static webpages!

Most important: What should it do/improve?



- Static webpages need only be verified once, dynamic webpages much more often!
  - Every new bit of content should be reviewed
  - Every possible combination should be reviewed
    - » Or at least whether individual parts or the combination method could lead to problems
- When third parties can provide content, liability can occur
  - "Hosting provider": No liability unless known, no immediate action upon notification or own personnel
    - » Notification and take-down procedure required
    - » No monitoring required, however!
- Third parties: Liability for links
  - Depends on kind and context of link to the dynamic content
  - See special lecture in next semester!



- Leigh Dodds: XML and databases? Follow your nose  
<http://www.xml.com/pub/a/2001/10/24/follow-yr-nose.html>
- James Robertson: How to evaluate a CMS  
[http://www.steptwo.com.au/papers/kmc\\_evaluate/](http://www.steptwo.com.au/papers/kmc_evaluate/)