Network Services, VU 2.0

Dynamic Web Technologies

Dipl.-Ing. Johann Oberleiter
Institute for Informationsystems, Distributed Systems Group

Overview

- Generic Mechanisms
- ISAPI/Apache Modules
- CGI (Common Gateway Interface)
- PHP
- ASP
- Java related
  - Java Servlets, Java Server Pages, Java Server Faces
  - Java Web Applications
- ASP.NET
- Cocoon, Struts

Dynamic Web – Why?

- Web Servers return only with static files
- Interactive Content
  - Created based on user interaction
- Dynamic Content
  - Created on the fly
  - Database access

Generic principles

- Separation of layout, content and program logic
  - Good design principle (not only in Web)
  - Allows parallel tasks of
    - Developer
    - Web Designer
- Layout
  - HTML
- Content
  - Which Text when dynamically generated
- Program Logic
  - What overall structure
  - What navigational structures

ISAPI

- Internet Server API (Microsoft IIS)
  - IIS only
  - Extension mechanism for IIS
  - C-based (like Windows API)
- Implements number of callbacks
  - Functions that are called by server
- Request sent to ISAPI extension via API functions (callbacks)
- Respond via API functions (callbacks)
- Supports use of Multithreaded features
- Features
  - Scalability
  - Supports ISAPI extensions
    - Similar to CGI-scripts
  - Supports ISAPI filter
    - Pre & postprocessing of a request, may be chained
- NSAPI
  - Netscape Server API

Apache Modules

- Extension mechanism for Apache
  - C based
  - Based on Apache Portable Runtime (APR), and C standard library
- Based on Hooks (callbacks)
- Categories
  - Authentication, Authorization, Accounting
  - Cache
  - Filters
    - Modify output from another module
  - Mappers
    - Map requests from URLs to resources on disc
  - Loggers

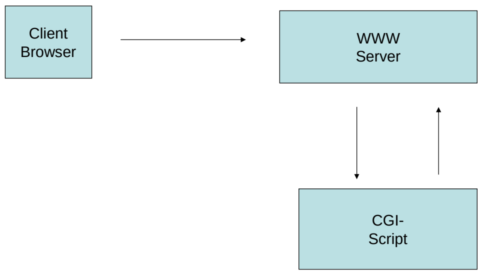
Common Gateway Interface

- RFC 3875
- Running external programs
  - From HTTP servers
  - Platform-independent mechanism
- CGI script & HTTP server together
  - Servicing a client request
  - Creating response
- CGI script addressed with URI

CGI / 2

- Supported by most programming languages
  - Requires access of standard input stream, standard output stream, environment
- Supported by most programming languages
  - requirements
    - Access standard input stream
    - Access standard output stream
    - Access environment variables
  - Web Server
    - Invocation of executables (stand-alone executables)
    - Invocation of interpreter (interpreter languages)
  - Typical
    - C, Perl
- New Process per request

CGI / 3



CGI / 4

```
void main(void)
{
   printf("Content-type: text/html\r\n");
   printf("\r\n");
   printf("Hello world!<br>\r\n");
   exit(0);
}
```

Fast-CGI

- CGI performance problem
  - Many requests require multiple processes
  - Initialization of connections/resources (database)
- FastCGI
  - Script remains in memory (via endless loop)
  - Predefined protocol/API for communication with HTTP server

Fast-CGI / 2

```
void main(void)
{
   int count=0;
   while(FCGI_Accept() >= 0) {
        printf("Content-type: text/html\r\n");
        printf("\r\n");
        printf("Hello world!<br>\r\n");
   }
   exit(0);
}
```

PHP

- Abbreviation for PHP: Hypertext Preprocessor
- Dynamic Web Scripting language
  - Syntax resembles C and Perl
  - Currently most frequently used Web programming language
  - Usually embedded in HTML
  - Supported by most Internet service providers
- Wellknown through LAMP
  - Linux – Apache – MySql – PHP
- Many libraries
  - In particular libraries for database access
- Problem
  - Mix of HTML and script code
  - Language grown over the years
- Performance
  - Quite Good

PHP - sample

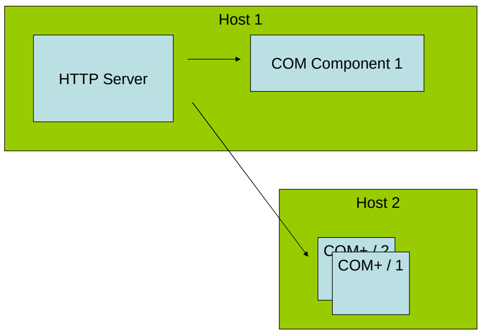
```
<html>
  …
<body>
  <?php echo „<p>Hello World</p>“; ?>
</body>

</html>
```

Active Server Pages

- Server-side scripting ala Microsoft
  - Relies on MS scripting languages
    - Usually VB.NET or JScript (JavaScript)
- Programming model
  - Program Logic via COM components
  - Principally scalable (via COM+)
  - Principally secure (server-side via COM+, ADSI)
- Problems
  - Mixture of layout and content
    - Principally possible to do it in a clean way (via COM comp.)
  - Vendor-lockin (only IIS supported)
    - though special solutions for Apache exist
  - Interpreted

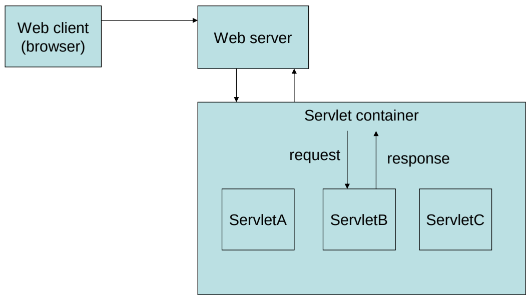
ASP / 2



Java Servlets

- Web component
  - implemented in Java
- Generates dynamic content
- Managed by a servlet engine (container)
  - Web server extensions
- Request/response paradigm
  - Interaction with Web clients

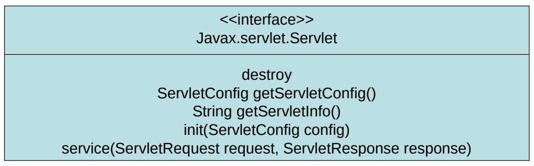
Request/response Interaction



Servlets characteristics

- Much faster than CGI scripts (in general)
  - Because different process model
- Standard API supported by many Web servers
- Supports Java and API's

Servlet interface / 1

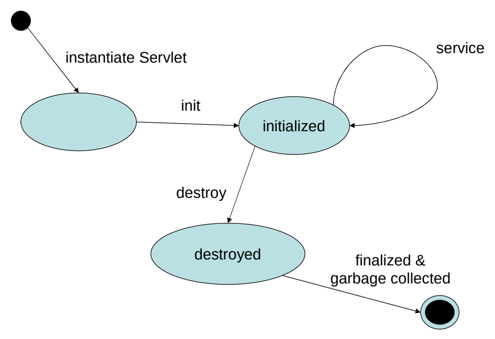

<<interface>>
Javax.servlet.Servlet

destroy
ServletConfig getServletConfig()
String getServletInfo()
init(ServletConfig config)
service(ServletRequest request, ServletResponse response)

Servlet implementation

- Server implements servlet interface
- Typically by inheriting from (predefined) implementation classes
  - GenericServlet
  - HttpServlet

Servlet lifecycle



Request Handling

- Through Service method
- ServletRequest object used
- Concurrent requests to same servlet
  - Concurrent execution of service method on different threads
- HTTP specific Request Handling
  - HttpServlet adds HTTP specific methods
    - primarily doGet & doPost,
    - doPut, doDelete, doHead, doOptions, doTrace
    - getParameterXXX methods provide
      - from URI query string and POST-ed data
    - getHeaderXXX methods

Response Generation

- By using methods of ServletResponse object
- Manual generation of any response
- HttpServletResponse interface
  - sendRedirect
  - sendError

Servlet example

```
public void doGet(HttpServletRequest req,
   HttpServletResponse res) throws ...
{
   res.setContentType("text/html");
   PrintWriter out= res.getWriter();

   out.println("<HTML>");
   out.println("<HEAD>");
...
   out.close();
}
```

Filtering

- Filter
  - Java component
  - Allow on the fly transformation
  - Implements javax.servlet.Filter
- Filter transforms content of
  - HTTP requests
  - Responses
  - Header information
- Modify or adapt
  - Requests for a resource (dynamic&static content)
  - Responses from a resource

Filter examples

- Authentication filters
- Logging and auditing filters
- Image conversion filters
- Data compression filters
- Encryption filters
- Tokenizing filters
- Filters triggering resource access events
- XSL/T filters
- MIME-type chain filters
- Caching filters

Filter implementation

- doFilter(ServletRequest req, ServletResponse res, FilterChain next)
  1. Examine request
  2. May wrap Request object with a custom implementation to filter/modify content or headers for input filtering
  3. May wrap Response object with a custom implementation to filter/modify content or headers for output filtering
  4. May invoke next filter in chain or block further processing
  5. After invocation 4. examine response headers and modify output

– Last element of chain is target servlet

Session Tracking

- Cookies
  - Supported by servlet container
  - Cookie name JSESSIONID
- SSL sessions
  - Only when SSL/TLS is in use
  - Built-in mechanism to distinguish multiple requests
- URL Rewriting
  - Adds session ID to request URL
  - Eg. http://www.xyz.com/index.html;jsessionid=1234
- Supports storage of key-value pairs
  - Keys are object names (strings)
  - Values arbitrary Java objects
- Session timeouts

Other Servlet Issues

- Request forwarding
  - Via Request Dispatchers
  - Support for event listeners
    - For state changes in ServletContext, HttpSession, ServletRequest
      - Lifecycle, changes to attributes, session migration, object binding
- Problem
  - Not often supported by public Web hosters

Java Server Pages (JSP)

- JSP page
  - Textual document how to create a response object from a request object for a given protocol
  - Defines a JSP page implementation class
    - Implements semantics of the JSP page
    - Implements javax.servlet.Servlet interface
  - HTTP default protocol for requests/responses
  - Default ending .jsp
- Traditional usage
  - Generation of HTML
- Generating XML possible
  - More modern JSP XML-like syntax

JSP

- JSP container
  - Life-cycle management
  - Runtime support
- Translation phase
  - Validates syntactic correctness of JSP page
  - Locates/Creates implementation class
- Execution phase
  - Container delivers events to JSP page
- Compilation
  - Into implementation class + deployment info during deployment possible
    - Removal of start-up lag for transition phase
    - Reduction of memory footprint needen to run JSP container – no compiler required

JSP - Syntax

- Elements
  – Element type known to JSP container
- Template Data
  – JSP translator not aware about
- Allows (a little bit) separation
  – Look/Layout
  – Behaviour

JSP Elements

- Directives
  – Global information, independent of specific request
  – Info for translation phase
  – Syntax: <%@ directive ... %>
- Actions
  – Infos for request processing
  – Standardized (by JSP specification)
  – Custom (portable tag extension mechanism)
  – Syntax: <mytag attr="value">xyz</mytag>
- Scripting Elements
  – Glue around template text and actions
  – Manipulation of objects and to perform computation
  – Invocation of methods on Java objects
  – Catching of Java language exceptions
  – Expression language (EL) to access data from different sources

JSP Expression Language

- Simple expressions without Java code
- Enclosed within ${...}
  – ${a+b}
  – <mytag attr1="${mybean.data}"/>
- Access of Java beans
- Operators as in Java
  – Includes arithmetic, relational operators, logical operators
  – Conditional Operator ${expr ? a:b}

JSP Documents

- JSP page that is also a XML document
  – Well-formed, validation
  – Entity resolution may be applied
  – <% style syntax not supported
    - Use <jsp:directive.xyz/> instead
  – Default convention .jspx
  – Specification calls it so-called XML view

JSP Taglibs

- Extension of tags a JSP container interprets
  - Tab library
  - Taglib directive required
    - <%@ taglib=http://www.xyz/mysupertags prefix="mysuper"/>
  - XML view
    - xmlns:prefix on root of JSP document (urn:jsptld:uriValue)
  - <mysuper:MyOwnTag>
    ...
  - </mysuper:MyOwnTag>

JavaServer Faces

- Extension of Servlets/JSPs
- Definition of Web components
  - Custom tag from tag library
  - Event processing (similar to JavaBeans)
    - ActionListener
    - ValueChangedListener
  - Components render themselves as HTML
  - Navigation rules in XML files
    - Store targets of navigation links
    - Automatically resolved
  - Validators

JavaServer Faces - Sample

```
<h:command_button id="submitButton"
  label="OK" commandName="submit">
  <f:action_listener
      type="myCl.MyActionListener">
</h:command_button>
```

Web Applications

- Consists of
  - Servlets
  - JSPs
  - Utility Classes
  - Static documents (HTML, images, ...)
  - Client side Java applets, beans, classes
  - Descriptive meta information above everything above

Web Applications

- Structured Hierarchy of Directories
  - Root of hierarchy is document root for application files
- Special directory WEB-INF
  - All things related to the application not in the document root
  - No file of WEB-INF served directly to client
  - Eg. Configuration
  - Deployment descriptor /WEB-INF/web.xml
  - Servlet and Utility classes in /WEB-INF/classes/
  - Java ARchive Files (JAR) in /WEB-INF/lib/
- Packaged in Web ARchive Format (WAR-File)
  - JAR Format
- Supports references to other J2EE technologies
  - eg EJB,JNDI,WebServices

Web Applications

/index.html
/howto.jsp
/feedback.jsp
/images/banner.gif
/images/jumping.gif
/WEB-INF/web.xml
/WEB-INF/lib/jspbean.jar
/WEB-INF/classes/com/mycorp/servlets/MyServlet.class
/WEB-INF/classes/com/mycorp/servlets/MyUtils.class

Example:
Context-Path: /catalog in Web-Container
Request: /catalog.index.html

Web Applications

- Deployment Descriptor (XML File)
  - ServletContext Init Parameters
  - Session Configuration
  - Servlet/JSP Definitions
  - Servlet/JSP Mappings
    - /foo/bar/* servlet1
    - /catalog servlet2
  - Application Lifecycle Listener classes
  - Filter Definitions and Filter Mappings
  - MIME Type Mappings
  - Welcome File list
    - Default files for unmachted URIs (eg. default.jsp)
  - Error Pages
    - List of error page descriptions
  - Locale and Encoding Mappings
  - Security

Security

- Declarative Security
  - Expressing an application's security structure external to the application
    - Roles, Access Control, Authentication Requirements
  - Described in deployment descriptor
- Programmatic Security
  - HttpServletRequest
    - getRemoteUser (user name client used for auth.)
    - isUserInRole
    - getUserPrincipal (principal name of current user)
- Servlet container
  - Enforces declarative or programmatic security
    - For the principal associated with an incoming request based on security attributes of the principal

Security Role

- Logical Grouping of users
  - Defined by Application Developer
  - Assembler
- On application deployment
  - Roles mapped by developer to principals or groups in the runtime environment

Security Constraint (deployment descriptor)

```
<security-contraint>
    <web-resource-collection>
            <web-resource-name>admin-only</web-resource-name>
            <url-pattern>/mypage/admin/*</url-pattern>
            <http-method>GET</http-method>
            <http-method>POST</http-method>
    </web-resource-collection>

    <auth-constraint>
            <role-name>ADMINISTRATOR</role-name>
            <role-name>BACKUP-ADMIN</role-name>
    </auth-constraint>

    <user-data-constraint>
            <!– NONE, INTEGRAL, CONFIDENTIAL - ->
            <transport-guarantee>CONFIDENTIAL</transport-gurantee>
    </user-data-constraint>
</security-constraint>
```

Jakarta Tomcat

- Most important Servlet engine
  - Became reference implementation
- Usually port 8080
- Hosts JSPs, Servlets
  - Connections to other technologies
- Stand-alone WebServer
  - Supports SSI
  - Supports CGI
  - Not as sophisticated as Apache
    - Performance, no support for non-Java languages, available tools, ...

Apache Integration with Tomcat / 1

- Sharing load using different port numbers
  - Eg. Apache runs on port 80, tomcat on 8080
    - Same or different server
  - Problems
    - User see URLs that contain different ports/servers – interesting for bookmarking
    - 2 WebServers to tune, maintain, secure
    - Apache security does not know about Tomcat security (file access, user authentication)

Apache Integration with Tomcat / 2

- Proxying Apache to Tomcat
  - Apache hands over all requests to specific URIs to Tomcat
    - Using Apache module mod_proxy
  - Problems
    - No load balancing for more than one proxy
    - 2 WebServers to tune, maintain, secure
    - HTTP proxying slower than custom connectors
    - Dual authentication

Apache Integration with Tomcat / 3

- Custom connector protocol
  - Apache module mod_jk2
    - a module for IIS is also available
  - AJP protocol (Apache JServ protocol)
  - Supports load balancing
  - Supports In-Process JVM
    - Tomcat runs inside Apache

ASP.NET

- Microsoft‘s answer to Servlets/JSP
- Requires .NET
- Supported by IIS 6.0
  - Special support for Apache / Mono
- Completely different to ASP
  - More like Java Servlets/JSP/Java ServerFaces
- Supports different programming languages
  - Any .NET capable programming language
    - Microsoft supports C#, C++, VB.NET, J#
- Today frequently supported
  - With Windows 2003 Server Web Edition

ASP.NET / 2

- ASP.NET (aspx) page translated to .NET class
  - Inherits from System.Web.UI.Page
  - Only if source has changed
- Server-side Web Controls
  - .NET classes
  - Server-side representation of HTML elements
    - or more complex elements
  - Implement their own renderering facility
  - May be rendered in any browser
  - May raise events (.NET event/delegate model)
- Programming model
  - Accessing .NET assemblies & components

ASP.NET / 3

- ASP.NET page classes
- In-line code
  - Program code within <script> tag
    - <script runat=„server“ language=„c#“>…</script>
  - Problem: mixture of HTML and program logic
- Code behind
  - Refers to code separated in a different class file
    - <%@ Language=„c#“ Inherits=„MyOther.MyClass“>
  - Inherits from the provided class
  - Addition of new methods,properties allowed
  - .NET Methods/Properties may be accessed with special script elements
- Allows separation of layout and contents

ASP.NET / 4

- Validators
  - Validation of user input
  - Different types
    - RequiredFieldValidator,
    - RangeValidator
    - RegularExpressionValidator
    - CustomValidator
  - Happens after button is pressed
  - Error message may be placed on a Web control "ValidationSummary"

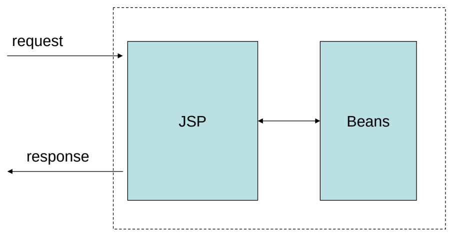
ASP .NET / 5

```
<%@ Page Language="C#"%>
<body>
   <form Runat="server">Name:
        <asp:textbox id="lastname" runat="server"/>
        <asp:RequiredFieldValidator id="reqVal"
        ErrorMessage="Required field!"    runat="server"/>
        <asp:Button ID="ok" Text="OK"
        OnClick="HandleClick" Runat="server"/>
   </form>
</body>
```

Dynamic Web Architectures / 1

- Sun's Model 1 architecture
  - One JSP processes request and generates reply
  - JSP's contain also process intensive task
  - Significant amount of code may be embedded in HTML code
  - Weak concerning separation of content and presentation

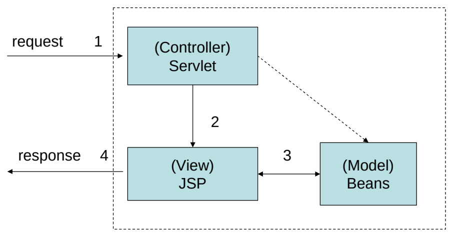
Model 1 architecture



Dynamic Web Architectures / 2

- Sun's Model 2 architecture
  - MVC paradigm
    - Model – underlying Datamodel
    - View – GUI
      - Changes in the model are automatically represented in the view
    - Controller
      - interprets user interaction
      - sends appropriate commands to Model
  - Servlet acts as controller
  - JSP acts as View
  - JavaBeans act as Model (created by the servlet)
    - Contain also connections to databases, EJB servers, ...
  - No processing logic in the JSPs
    - Processing logic
  - Better separation of content and presentation

Model 2 architecture



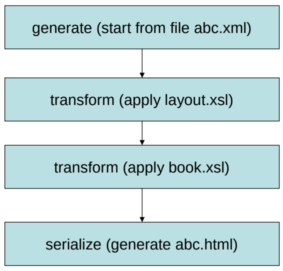
Struts

- Framework for
  - Model 2 architectures
- Provides its own controller
- Supports different technologies for Model
  - JDBC, EJB, Hibernate
- Supports different technologies for View
  - JSP, JSF, XSLT, JSTL, Velocity Templates

Cocoon

- XML based servlet framework
- Based on XSLT pipelines
- Sitemaps controls generation of pages
  - generate starts pipeline
  - transform applies stylesheet
  - Serialize generates final output file
- Continuations
  - Navigation flow written with JavaScript

Cocoon / 2


generate (start from file abc.xml)
transform (apply layout.xsl)
transform (apply book.xsl)
serialize (generate abc.html)