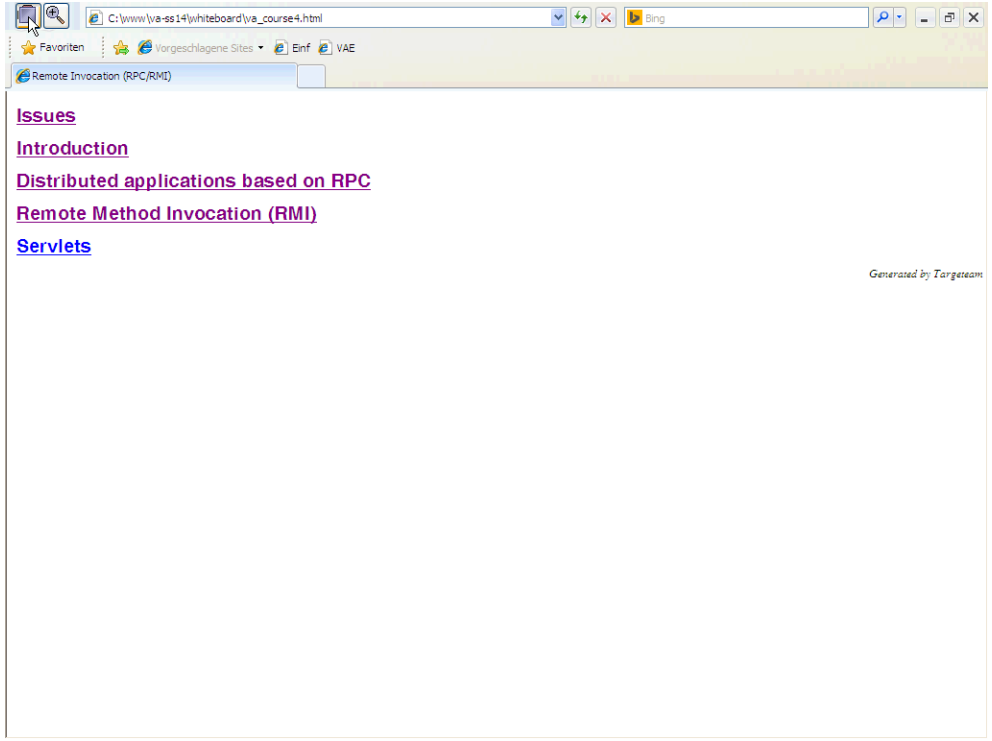
Script **generated by TTT**

Title: Distributed_Applications (19.05.2014)

Date: Mon May 19 09:18:13 CEST 2014

Duration: 45:07 min

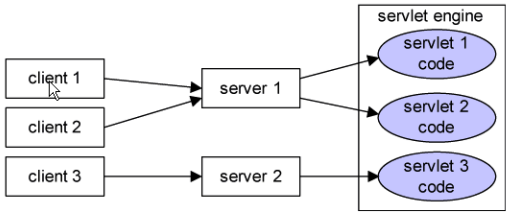
Pages: 15



Servlets

Servlets (Java Servlets) are programs invoked by a client and executed on the server host:

used to extend the functionality of the server.



Servlet Properties
Servlet Lifecycle
HttpServlet Interface
Structure of a Servlet

Servlet Properties

execution of a servlet in the context provided by the servlet engine.

Apache Tomcat : free, open-source implementation of Java servlet technology.

methods specified within each servlet object and invoked by the servlet engine

init: when a servlet is initialized.

shutdown: when a servlet is no longer needed.

service: when a client request is forwarded to the servlet.

servlets are invoked via HTTP requests (get or post method), e.g.

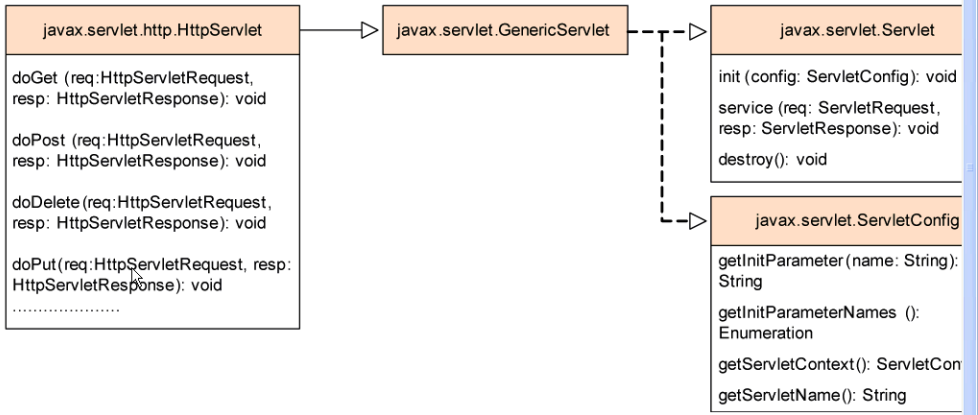
```
<form method="post"
action="http://myhost:8080/servlet/formServlet">
.... arguments of the form .....
```

HttpServlet Interface

HttpServlet inherits abstract class GenericServlet which implements interfaces Servlet and ServletConfig.

GenericServlet defines a generic protocol-independent servlet

HttpServlet defines a servlet for the HTTP protocol



doGet is invoked to respond to a GET request

doPost is invoked to respond to a POST request

doDelete is invoked to respond to a DELETE request; normally used to delete a file on the server

Structure of a Servlet

```
import javax.servlet.*;
import javax.servlet.http.*
import java.io.*;

public class MyServlet extends HttpServlet {
    /** called by the servlet engine to initialize servlet */
    public void init() throws ServletException { .... }
    /** process the HTTP Get request */
    public void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOEXception { .... }
    /** process the HTTP Post request */
    public void doPost(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOEXception { .... }
    /** called by the servlet engine to release the resource */
    public void destroy () { .... }
    // other methods
}
```

Example - CurrentTime

Example - CurrentTime

```
import javax.servlet.*;
import javax.servlet.http.*
import java.io.*;

public class CurrentTime extends HttpServlet {
    /** process the HTTP Get request */
    public void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOEXception {
        response.setContentType('text/html');
        PrintWriter out = response.getWriter();
        out.println("<p>The current time is " + new java.util.Date() );
        out.close(); // close stream
    }
}
```

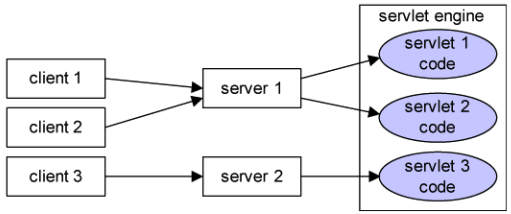
Invocation

http://localhost:8080/.../servlet/CurrentTime

Servlets

Servlets (Java Servlets) are programs invoked by a client and executed on the server host:

used to extend the functionality of the server.



Servlet Properties
Servlet Lifecycle
HttpServlet Interface
Structure of a Servlet

Issues

The following section discusses several important basic issues of distributed applications.

Data representation in heterogeneous environments.

Discussion of an execution model for distributed applications.

What is the appropriate error handling?

What are the characteristics of distributed transactions?

What are the basic aspects of group communication (e.g. algorithms used by ISIS) ?

How are messages propagated and delivered within a process group in order to maintain a consistent state?

External data representation

Time

Distributed execution model

Failure handling in distributed applications

Distributed transactions

Group communication

Distributed Consensus

Authentication service Kerberos

Heterogeneous environment means different data representations

⇒ requirement to enable data transformation.

independence from hardware characteristics while exchanging messages means: use of external data representation.

Marshalling and unmarshalling

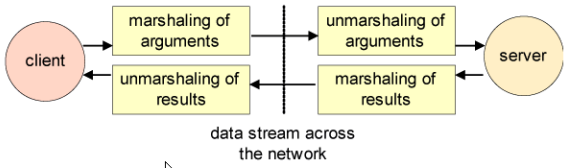
Centralized transformation

Decentralized transformation

Common external data representation

XML as common data representation

Java Object Serialization



marshal : parameter serialization to a data stream.

unmarshal : data stream extraction and reassembly of arguments.

software for argument transformation either provided by RPC system or as plugin by the application programmer.

Heterogeneous environment means different data representations

⇒ requirement to enable data transformation.

independence from hardware characteristics while exchanging messages means: use of external data representation.

Marshalling and unmarshalling

Centralized transformation

Decentralized transformation

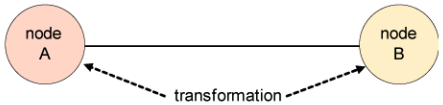
Common external data representation

XML as common data representation

Java Object Serialization

Generated by Targeteam

Decentralized transformation



All nodes execute data transformations.

Variants

A transforms data which are then sent to B; B transforms data which are then sent to A.

A transforms data by B; B transforms data by A.

A and B transform data in a network-wide standard format; the respective recipients retransform the received data into the local format.

If new system components are dynamically added to the distributed system, the new system components simply have to "learn" about the network-wide unique standard representation.

No special hardware is required.

Example: XDR as part of ONC by Sun.

Common external data representation

Two aspects of a common external data representation are of importance:

a machine-independent format for data representation, and

a language for description of complex data structures.

Examples: XDR ("eXternal Data Representation") by Sun and ASN.1 (Abstract Syntax Notation). Other formats are

Corba's common data representation: structured and primitive types can be passed as arguments and results.

Java's object serialization: flattening of single objects or tree of objects.

Representation of numbers

External representation of strings

External representation of arrays

Transfer of pointers

Representation of numbers

For the representation of numbers in main memory, one of the following methods are generally used.

"little endian" representation: the lower part of a number is stored in the lower memory area

"big endian" representation: the higher part of a number is stored in the lower memory area, e.g. the Sun-Sparc architecture

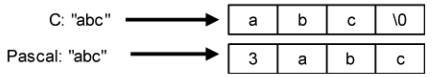
Example representation of the number 1347

Memory Address	1000	1001	1002	1003
Big Endian	00000000	00000000	00000101	01000011
Little Endian	01000011	00000101	00000000	00000000

Convention: for network transfer, numbers which encompass several bytes are structured according to a well-defined representation, such as "big endian".

External representation of strings

There are different internal representations for strings:



Standardized external representation: