Script **generated by TTT**

Title: Distributed_Applications (28.05.2013)

Date: Tue May 28 14:31:52 CEST 2013

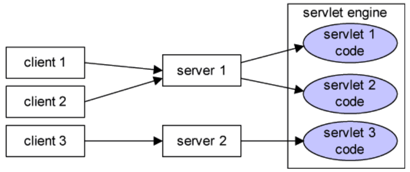
Duration: 88:17 min

Pages: 26

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

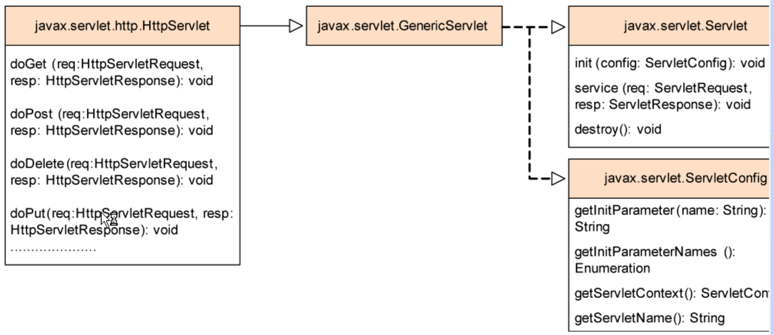
Structure of a Servlet

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 - Registration of Students

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

public class GetParameters extends HttpServlet {
    /** process the HTTP Get request */
    public void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
       response.setContentType('text/html');
       //obtain parameters from the client
       String lastName = request.getParameter("lastName");
       String firstName = request.getParameter("firstName");
       String gender = request.getParameter("gender");
       String major = request.getParameter("major");
       String[] minors = request.getParameterValues("minor");
       String tennis = request.getParameter("tennis");
       String soccer = request.getParameter("soccer");
       String golf = request.getParameter("golf");
       String remarks = request.getParameter("remarks");
       out.println("Last Name: <b>" + lastName + "</b> First Name: <b>"
          + firstName + "</b><br>");
       out.println("Gender: <b>" + gender + "</b><br>");
```



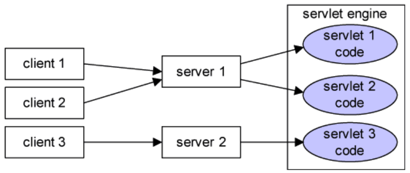
```
response.setContentType('text/html');
//obtain parameters from the client
String lastName = request.getParameter("lastName");
String firstName = request.getParameter("firstName");
String gender = request.getParameter("gender");
String major = request.getParameter("major");
String[] minors = request.getParameterValues("minor");
String tennis = request.getParameter("tennis");
String soccer = request.getParameter("soccer");
String golf = request.getParameter("golf");
String remarks = request.getParameter("remarks");
out.println("Last Name: <b>" + lastName + "</b> First Name: <b>"
   + firstName + "</b><br>");
out.println("Gender: <b>" + gender + "</b><br>");
out.println("Major: <b>" + major + "</b> Minor: <b>");
if (minors != null)
   for (int i = 0; i < minors.length; i++)
      out.println(minors[i] + " ");
out.println("</b><br> Tennis: <b>" + tennis + "</b> Soccer: <b>"
   + soccer + "</b> Golf: <b>" + golf + "</b><br>");
out.println("Remarks: <b>" + remarks + "</b>");
```

get value

to obtain parameters

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

used to extend the functionality of the server.



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

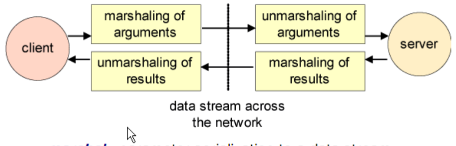
Heterogeneous environment means different data representations

⇒ 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]

Marshalling and unmarshalling



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.

External data representation

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]

Generated by Targeteam

External data representation

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]

Generated by Targeteam

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

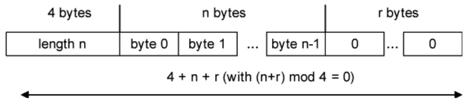
- Convention: for network transfer, numbers which encompass several bytes are structured according to a well-defined representation, such as "big endian".

External representation of strings

There are different internal representations for strings:

C: "abc" →	a	b	c	\0
Pascal: "abc" →	3	a	b	c

Standardized external representation:



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.

Example: primitive datatypes

Request for the invocation of the Java method

```
echoString("cat")
```

SOAP body of request that sends a string.

```
<soap:Body>
   <n:echoString
   xmlns:n='http://tempuri.org/mapping.server.Primitive'>
   <value xsi:type='xsd:string'>cat</value>
   </n:echoString>
</soap:Body>
```

XML as common data representation

Complex data types can be mapped to XML for transmission across the network.

- primitive datatypes → XSD equivalent
  - boolean, byte, unsignedShort (used for char), int, long, float, string, ...

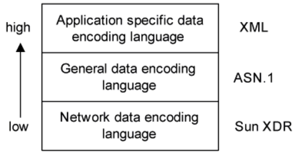
Example: primitive datatypes

SOAP provides built-in support for encoding arrays.

Example: array datatype

complex data types are mapped to XML schema types;

- SOAP platforms provide API for creating custom mapping.
- e.g. writeSchema to specify an XML schema definition



Level of Abstraction

Generated by Targeteam

Example: array datatype

Request for Java method invocation

```
echoInts([1, 2, 3])
```

SOAP body of request that sends an array.

```
<soap:Body>
   <n:echoInts
   xmlns:n='http://tempuri.org/mapping.server.Array'>
   <ints href='#id0'>
   </n:echoInts>
   <id0 id='id0'
      soapenc:root='0'
      xsi:type='soapenc:Array'
      soapenc:ArrayType='xsd:int[3]'>
      <i xsi:type='xsd:int'>1</i>
      <i xsi:type='xsd:int'>2</i>
      <i xsi:type='xsd:int'>3</i>
   </id0>
</soap:Body>
```

Generated by Targeteam

XML as common data representation

Complex data types can be mapped to XML for transmission across the network.

- primitive datatypes → XSD equivalent
  - boolean, byte, unsignedShort (used for char), int, long, float, string, ...

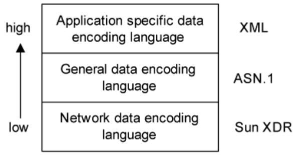
Example: primitive datatypes

SOAP provides built-in support for encoding arrays.

Example: array datatype

complex data types are mapped to XML schema types;

- SOAP platforms provide API for creating custom mapping.
- e.g. writeSchema to specify an XML schema definition



Level of Abstraction

Generated by Targeteam

External data representation

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

Generated by Targeteam

Time

Time is an important and interesting issue in distributed systems

- We need to measure time accurately:
  - to know the time an event occurred at a computer
  - to do this we need to synchronize its clock with an authoritative external clock
- Algorithms for clock synchronization useful for
  - concurrency control based on timestamp ordering
  - authenticity of requests e.g. in Kerberos

Three notions of time:

- time seen by an external observer ⇒ global clock of perfect accuracy.
  - However, there is ***no global clock in a distributed system***
- time seen on clocks of individual processes.
- logical notion of time: event a occurs before event b.

Introduction

Synchronizing physical clocks

Generated by Targeteam

Introduction

Each computer in a distributed system (DS) has its own internal clock

- used by local processes to obtain the value of the current time
- processes on different computers can timestamp their events
- but clocks on different computers may give different times
- computer clocks drift from perfect time and their drift rates differ from one another.
- clock drift rate: the relative amount that a computer clock differs from a perfect clock

⇒ Even if clocks on all computers in a DS are set to the same time, their clocks will eventually vary quite significantly unless corrections are applied.

Timestamp

Skew between clocks

Coordinated Universal Time (UTC)

Generated by Targeteam

Timestamp

To timestamp events, we use the computer's clock

1. At real time t, the operating system reads the time on the computer's hardware clock $H_i(t)$
2. It calculates the time on its software clock
   - $C_i(t) = a\,H_i(t) + b$
   - e.g. a 64 bit number giving nanoseconds since some "base time"
   - in general, the clock is not completely accurate,
   - but if C_i behaves well enough, it can be used to timestamp events at p_i

Generated by Targeteam

Introduction

Each computer in a distributed system (DS) has its own internal clock

- used by local processes to obtain the value of the current time
- processes on different computers can timestamp their events
- but clocks on different computers may give different times
- computer clocks drift from perfect time and their drift rates differ from one another.
- clock drift rate: the relative amount that a computer clock differs from a perfect clock

⇒ Even if clocks on all computers in a DS are set to the same time, their clocks will eventually vary quite significantly unless corrections are applied.

Timestamp

Skew between clocks

Coordinated Universal Time (UTC)

Generated by Targeteam

Time

Time is an important and interesting issue in distributed systems

- We need to measure time accurately:
  - to know the time an event occurred at a computer
  - to do this we need to synchronize its clock with an authoritative external clock
- Algorithms for clock synchronization useful for
  - concurrency control based on timestamp ordering
  - authenticity of requests e.g. in Kerberos

Three notions of time:

- time seen by an external observer ⇒ global clock of perfect accuracy.
  - However, there is ***no global clock in a distributed system***
- time seen on clocks of individual processes.
- logical notion of time: event a occurs before event b.

[Introduction](#)

[Synchronizing physical clocks](#)

External - internal synchronization

External synchronization

- A computer's clock C_i is synchronized with an external authoritative time source S, so that:
  - $|S(t) - C_i(t)| < D$ for $i = 1, 2, \ldots N$ over an interval I of real time t.
  - The clocks C_i are accurate to within the bound D.

Internal synchronization

- The clocks of a pair of computers are synchronized with one another so that:
  - $| C_i(t) - C_j(t)| < D$ for $i, j = 1, 2, \ldots N$ over an interval I of real time t.
  - The clocks C_i and C_j agree within the bound D.

Internally synchronized clocks are not necessarily externally synchronized, as they may drift collectively.

if the set of processes P is synchronized externally within a bound D, it is also internally synchronized within bound 2D.

Synchronizing physical clocks

physical clocks are used to compute the current time in order to timestamp events, such as

- modification date of a file
- time of an e-commerce transaction for auditing purposes

[External - internal synchronization](#)

[Clock correctness](#)

[Synchronization in a synchronous system](#)

[Cristian's method for an asynchronous system](#)

[Network Time Protocol (NTP)](#)

[Precision Time Protocol (PTP)](#)