Java Web Programming
Servlets

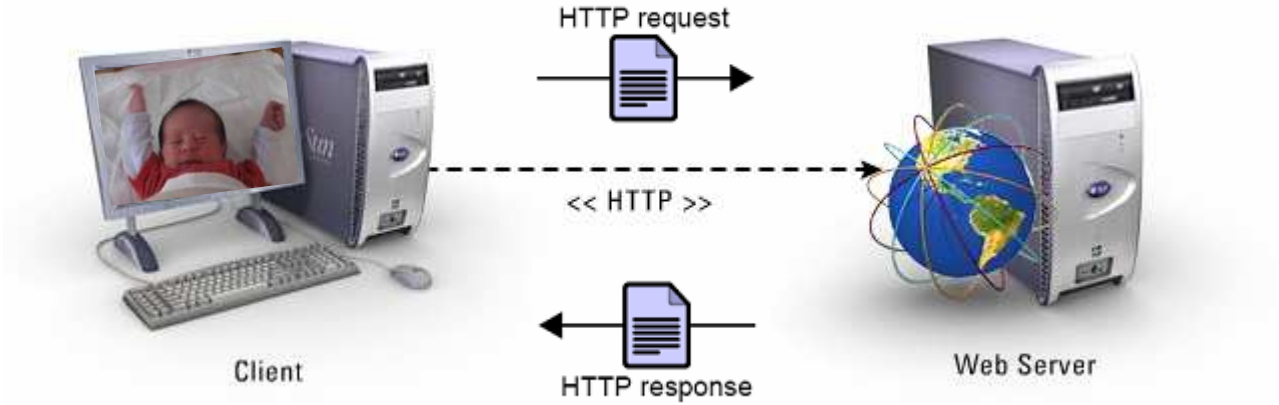
אוהד ברזילי

פיתוח מערכות תוכנה מבוססות Java
בית הספר למדעי המחשב, אוניברסיטת תל אביב

פרוטוקול HTTP והמוטיבציה לטכנולוגיית השרותונים

פרוטוקול HTTP

- גלישה לאתרי אינטרנט מתבצעת בעזרת פרוטוקול HTTP (מעל TCP/IP)
- דפדפן האינטרנט הוא הלקוח (יוזם ההתקשרות)
- הדפדפן ממלא שני תפקידים:
  - מימוש פרוטוקול התקשורת (HTTP)
  - הצגת התשובה על המסך (HTML Parsing)



פרוטוקול HTTP

- פרוטוקול HTTP מגדיר את מבנה ההודעות הבא (קיימות גם אחרות):

```
GET <filename> <version>
```

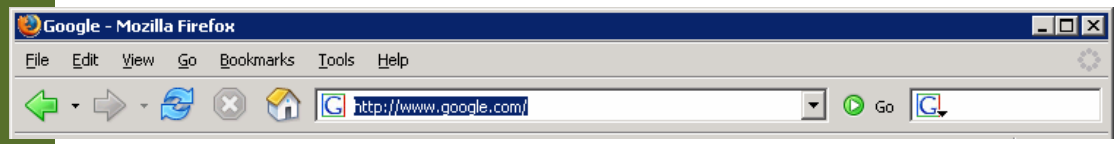
<שורת רווח>

- תוכניות שרת (web servers) משתמשות בשלוחה 80

- כאשר הם מקבלים הודעת GET הם פותחים את הקובץ המתאים ושולחים אותו ללקוח

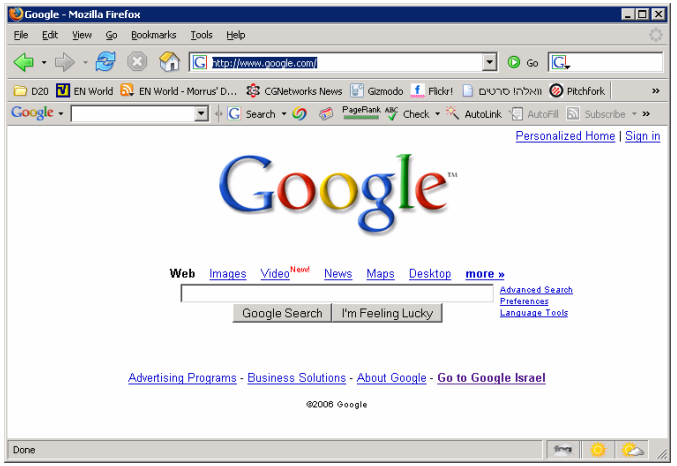
אז מה זה HTTP?

- שרת HTTP מחכה לבקשות על Port 80
- בהינתן בקשה, השרת מחזיר תשובה... בעצם קובץ



משתמש מקליד כתובת אינטרנט

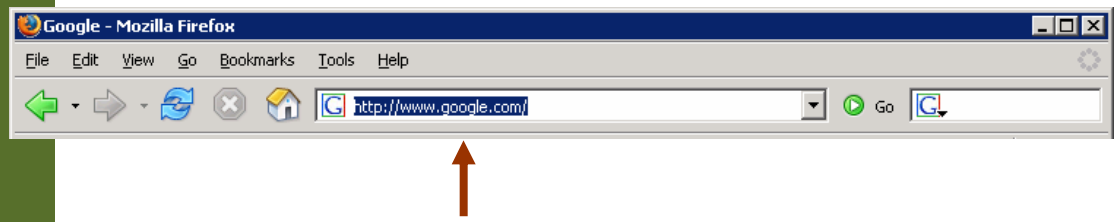
השרת של GOOGLE מטפל בבקשה



מחזיר דף HTML המוצג בדפדפן

אז מה זה HTTP?

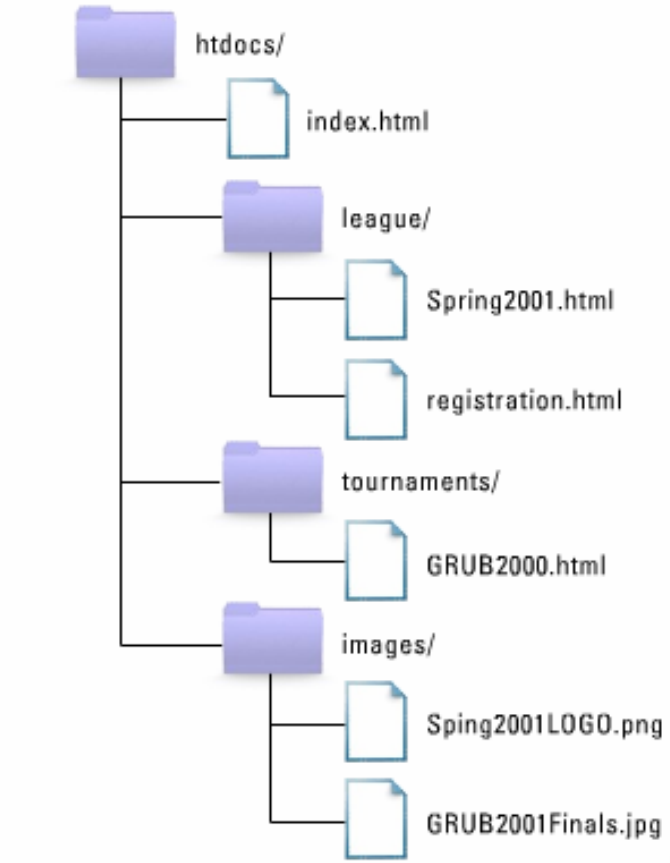
- שרת HTTP מחכה לבקשות על Port 80
- בהינתן בקשה, השרת מחזיר תשובה... בעצם קובץ



URL (Uniform Resource Locator)

מייצג משאב כלשהוא ברשת, דף אינטרנט, קובץ, שירות כלשהוא...

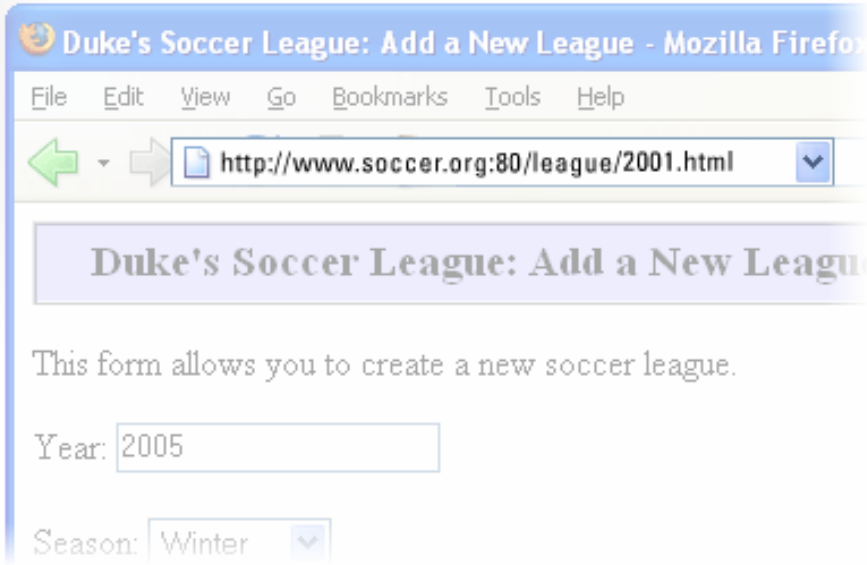
שרת קבצים



URL

A canonical name that locates a specific resource on the Internet

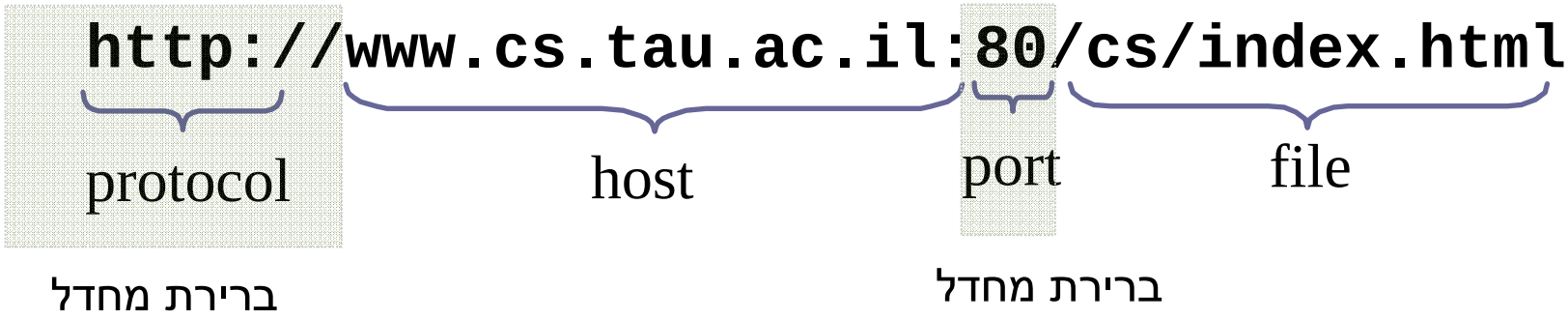
- It consists of:
  protocol://host:port/path/file



עבודה מעל HTTP

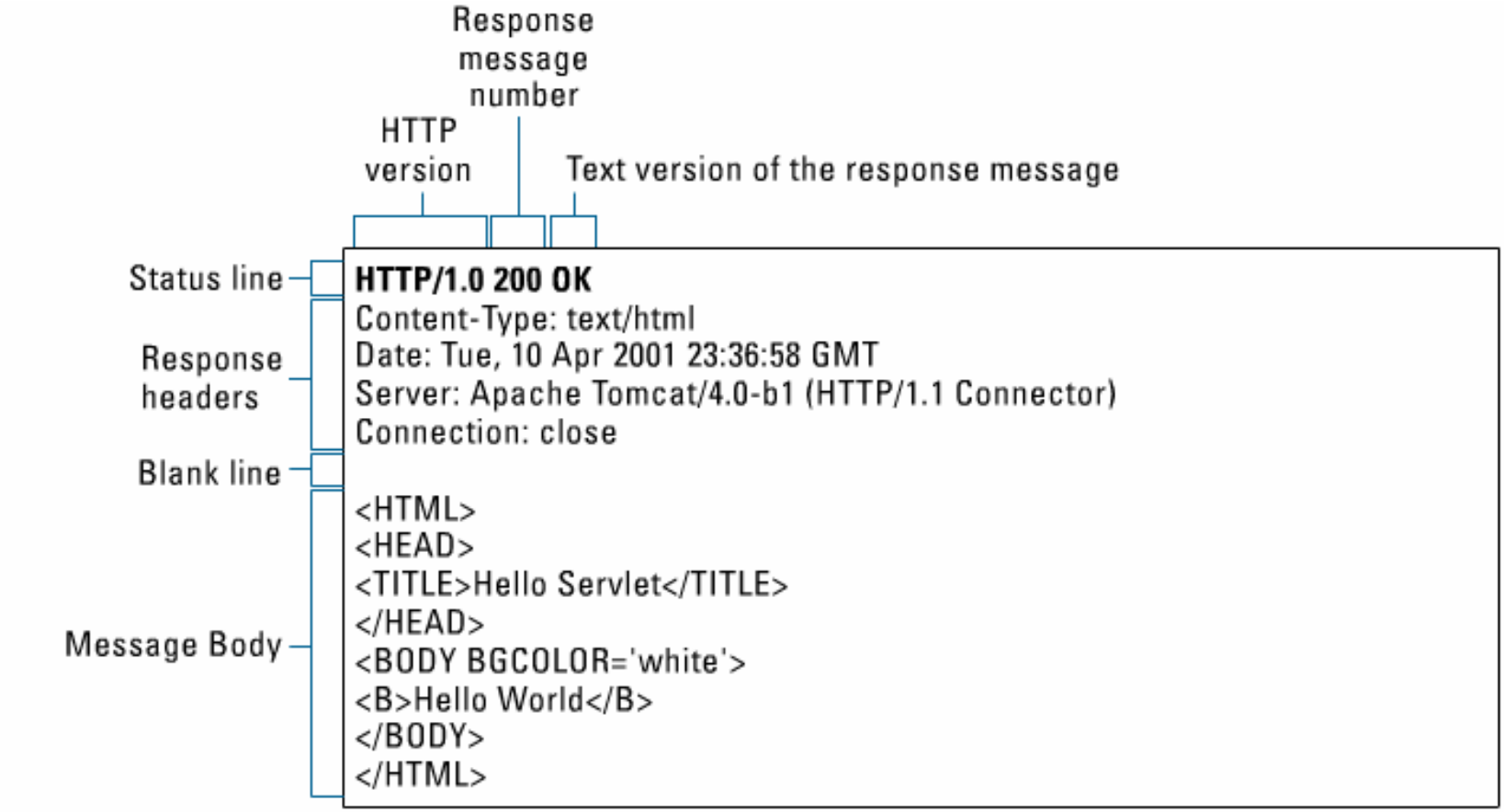
- איך עובד דפדפן?

- דפדפנים מחלקים את הכתובת שהוכנסה בשורת הכתובת (URL) ל-5 חלקים, ובונים את ההודעה המתאימה. לדוגמא:



HTTP GET Request

HTTP GET Response



שרת דינאמי

- עם התפתחות האינטרנט עלה הרעיון **להריץ תוכניות** מעל פרוטוקול HTTP
  - תיבת הדואר שלי ב Gmail למשל נוצרת דינאמית
  - הפלט של התוכניות יהיה קובץ HTML שיוצג בדפדפן כרגיל
  - ציון שם התוכנית והעברת פרמטרים לתוכנית יתבצעו ע"י שורת הפקודה לדוגמא:

http://www.google.co.il/search?hl=iw&q=cow&btn&meta=

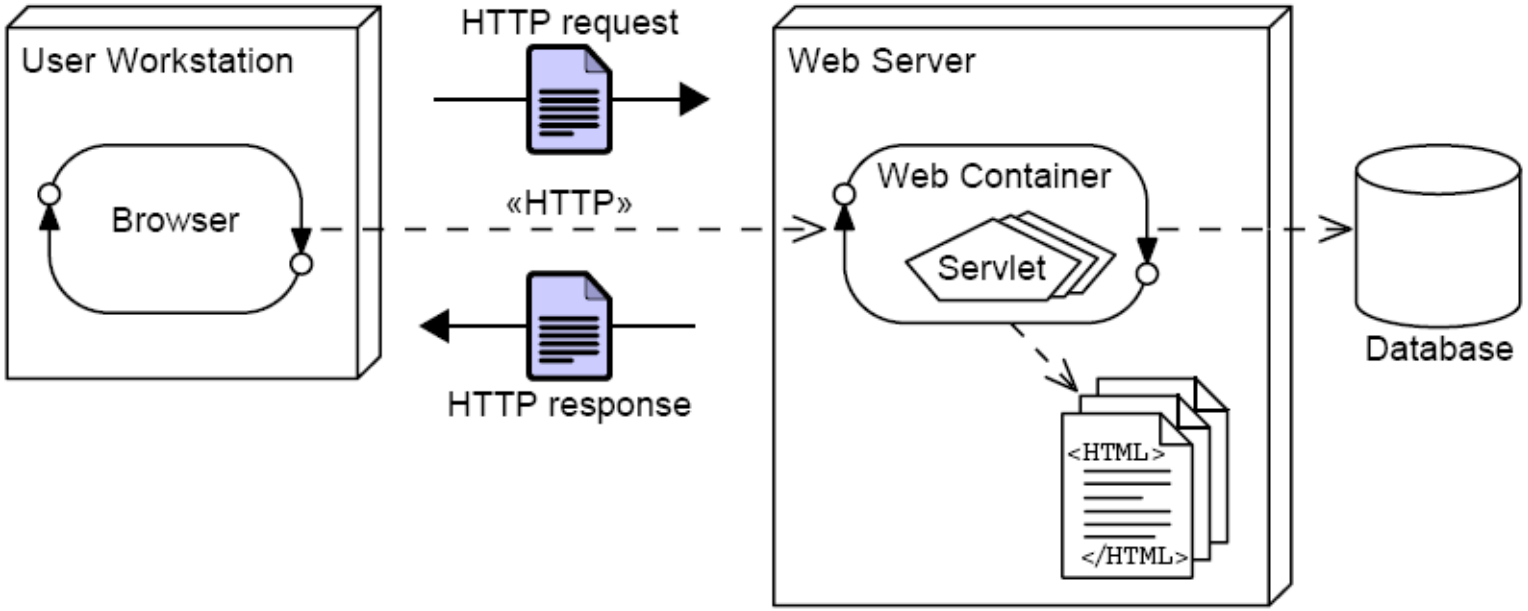
- נרצה להשתמש **בשרת אינטרנט סטנדרטי** כדי לא לכתוב מחדש קוד המיישם את פרוטוקול HTTP
- בעיה: שרת האינטרנט שהצגנו מקודם אינו דינאמי – הוא יכול להציג רק דפים שהוכנו מראש
- היינו רוצים שרת שיענה על 2 הדרישות הבאות:
  - שידע לטפל בהבטי התקשורת (ואולי גם הבטים אחרים)
  - שיאפשר לנו לכתוב לוגיקה נוספת (ולא רק להחזיר דפים שהוכנו מראש)

הרעיון

- הפרדה בין:
  - **המסגרת (framework):** החוזרת על עצמה
    - "ההבטים": טיפול בלקוח חדש, יצירת חוט, יצירת שקע ועוד...
  - **הלוגיקה העסקית**: מה התוכנית עושה
    - קוד Java שיכול לעשות כל דבר

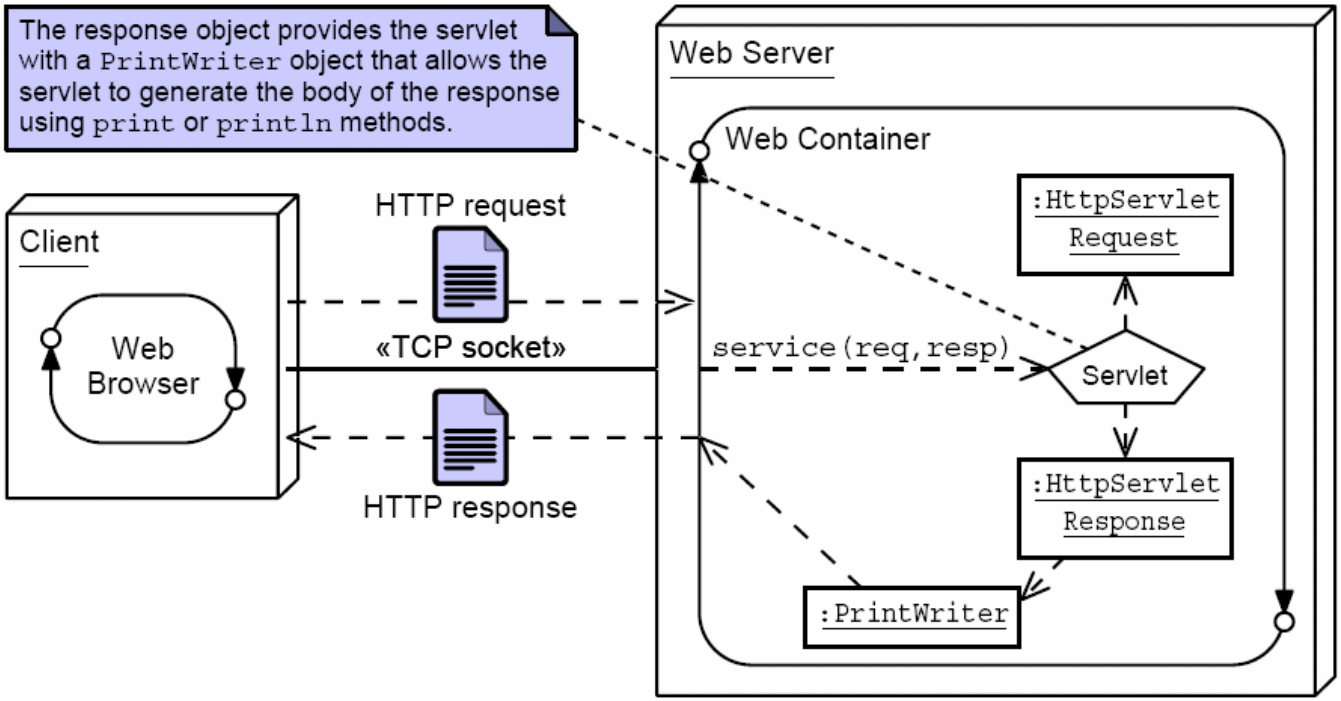
- את השרת (web container) נכתוב פעם אחת ונטען לתוכו מחלקות Java בשם servlets (שרותונים?) אשר ממשות כל אחת לוגיקה עסקית משלה
  - בפרט נוכל להתקין על אותו שרת כמה שרותים במקביל
  - כמו כן, נוכל בהמשך להחליף את השרת שכתבנו בשרת מקצועי (Tomcat, OC4J ואחרים) מבלי להחליף את השרותונים

ארכיטקטורת Web Container



- ה- web container עושה מניפולציה על פורמט הודעת GET - במקום המיועד להזנת **שם הקובץ**, הוא מצפה לקבל **שם מחלקה**

ארכיטקטורת Web Container



- על המחלקה הזו הוא יפעיל את השרות **`doGet`**, (במקרה הכללי **`service`**) וידאג להעביר לה כפרמטרים **מחלקות עזר** שבעזרתם תקרא את הפרמטרים אם הועברו כאלה בשורת הכתובת, ותייצר הודעת תשובה

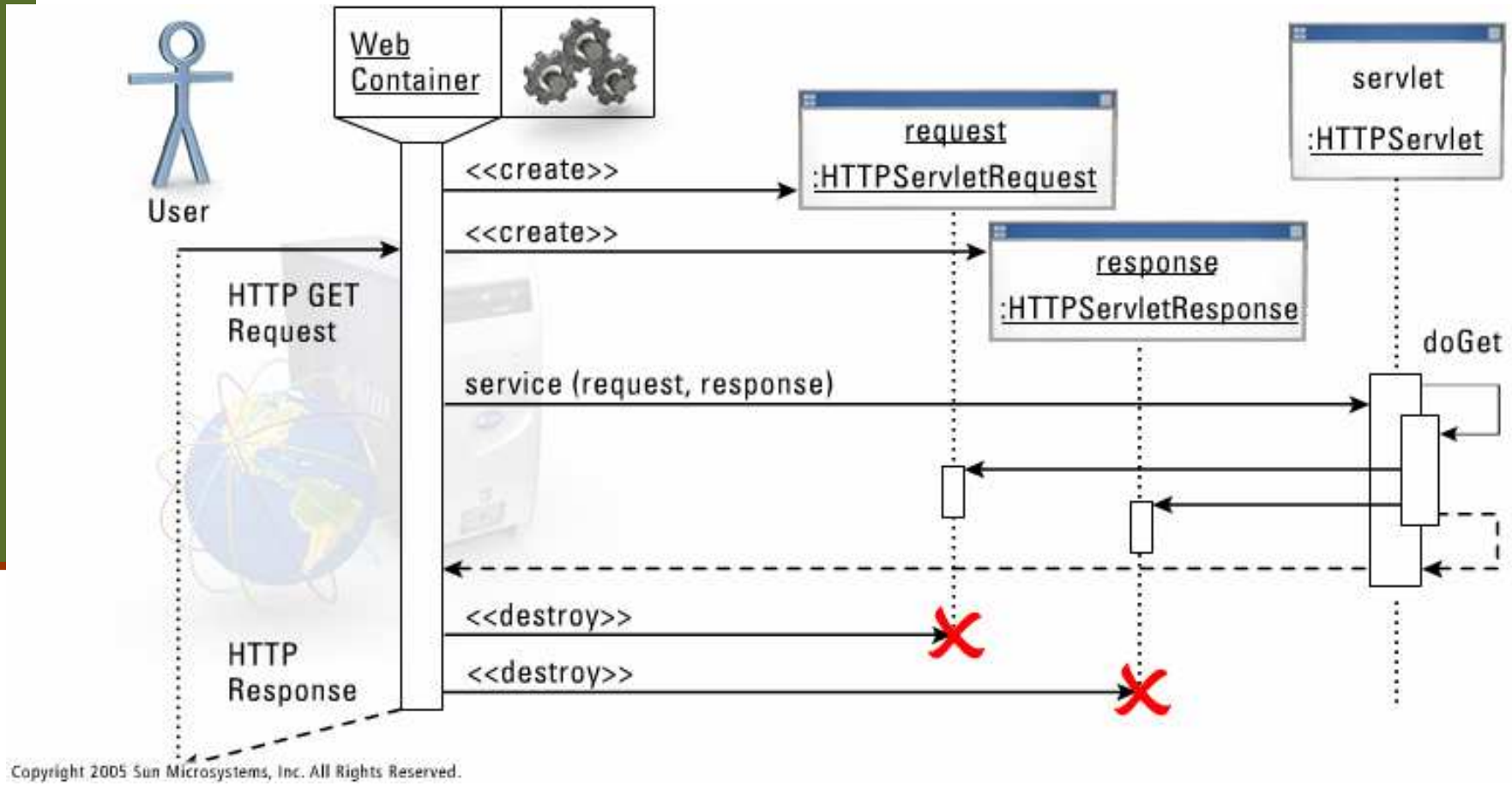
תפקידי ה Web Container

- טיפול בהבטי תקשורת:
  - יצירת שקע והמתנה ללקוחות
- ניהול חוטים:
  - יצירת Thread נפרד לטיפול בלקוחות נכנסים (accept)
- ניהול השרותונים
  - הפניית הבקשות לשרותון המתאים
  - אתחול המופעים לפי דרישה
  - שמירת מידע רלוונטי בין קריאות (session, context)

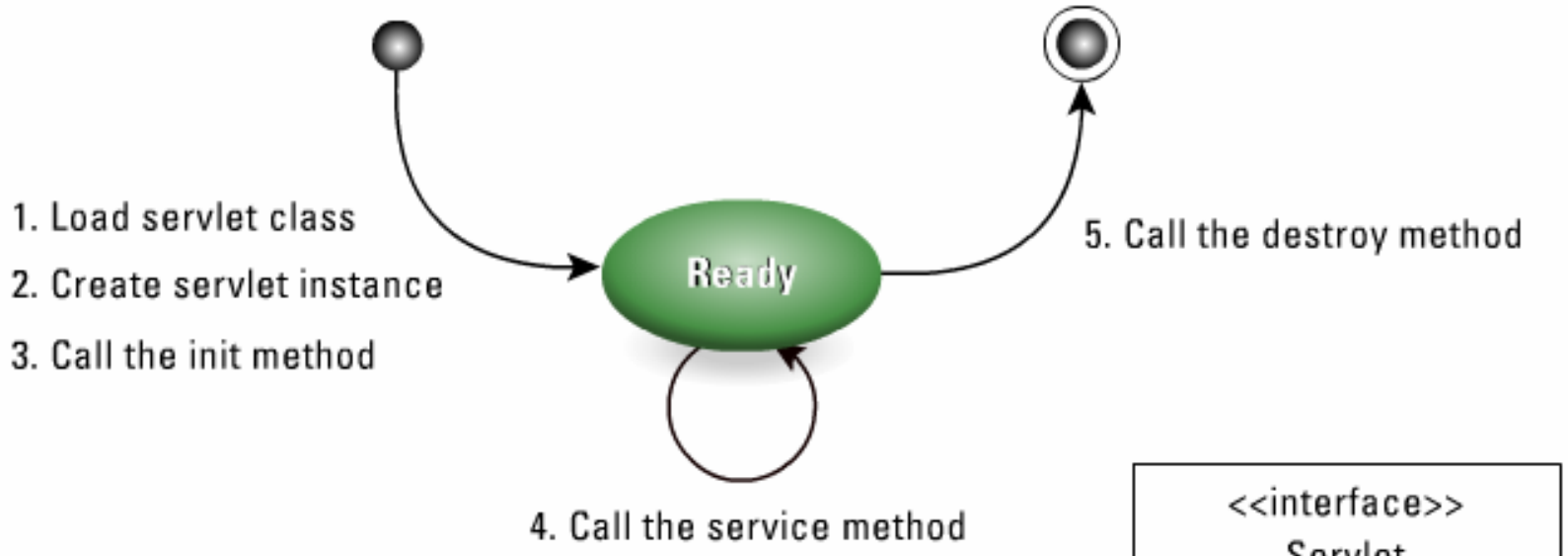
תפקידי ה Web Container

- טיפול בפרוטוקול HTTP
  - יצירת מחלקות העזר (**`HttpServletRequest`**, **`HttpServletResponse`**)
  - ניתוח כותרות ה HTTP (request)
    - ניתוח את המחרוזת במקום המיועד לשם הקובץ וחילוץ שם ה Servlet המבוקש ואת שמות המשתנים וערכיהם (לפי התווים: '?' , '&' , '=' , '/')
  - חילול כותרות ה HTTP (response)
- מיפוי מרחב השמות הלוגיים והפיזיים
- בקרת תצורה

טיפול בהודעת GET



Servlet Life Cycle



<<interface>> Servlet
init(ServletConfig) service(req,resp) destroy( )

מימוש שרת פשוט

HttpServletRequest, HttpServletResponse

- שרותי מחלקות העזר מוגדרים ע"י המנשקים הבאים:

```
public interface HttpServletRequest {
  public java.util.Set<String> getParameterNames();
  public String getParameter(String name);
}

public interface HttpServletResponse {
  public java.io.PrintWriter getWriter();
  public void setContentType(String type);
}
```

מימוש מחלקות העזר

```
public class ServletInvocation
  implements HttpServletRequest, HttpServletResponse {

  private java.util.Map<String,String> parameters;
  private PrintWriter                  writer;

  public ServletInvocation(java.util.Map<String,String> parameters,
                           PrintWriter writer) {
    this.parameters = parameters;
    this.writer     = writer;
  }

  public String getParameter(String name) { return parameters.get(name);  }

  public Set<String> getParameterNames() {  return parameters.keySet();   }

  public PrintWriter getWriter() { return writer;  }

  public void setContentType(String type) {
    writer.print("HTTP/1.1 200\r\n");
    writer.print("Content-Type: "+type+"\r\n");
    writer.print("Connection: close\r\n");
    writer.print("\r\n"); // end of header
  }

}
```

TrivialServlet

- נתחיל בדוגמא פשטנית: נכתוב את השרת **`HttpServer`**
  (שנראה אחר כך) ונתקין בו את השרותון
  `TrivialServlet` אשר מדפיס את רשימת הפרמטרים
  שקיבל

- שרותון הוא Component – הוא מקבל במתנה מהמיכל (מה-
  web container) את כל הדברים שהוא זקוק להם (aspects)
- עליו לממש רק לוגיקה עסקית בתוך השרות **`doGet`** (hook)

TrivialServlet

```
package examples.servletserver;

public class TrivialServlet extends HttpServlet {

  public TrivialServlet() {
  }

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse resp) {
    java.io.PrintWriter writer = resp.getWriter();
    resp.setContentType("text/plain");

    java.util.Set<String> parameters = req.getParameterNames();

    writer.println("This is servlet "+this.getClass().getName());
    for (String pname: parameters) {
      writer.println("parameter "+pname+" = "+req.getParameter(pname));
    }
  }

}
```

```
public class FileServlet extends HttpServlet {

  public FileServlet() {
  }

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse resp) {
    java.io.PrintWriter writer = resp.getWriter();
    String path = req.getParameter("path");
    if (path == null)
      return;

    // we really should be more clever
    if (path.endsWith(".html"))
      resp.setContentType("text/html");
    else
      resp.setContentType("text/text");

    try {
      BufferedReader in = null;
      String servlet    = null;
      java.util.Map<String,String> parameters = null;

      in = new BufferedReader(new InputStreamReader(new FileInputStream(path)));

      while (true) {
        String line = in.readLine();
        if (line==null)         break; // end of file
        writer.println(line);
      }
    } catch (IOException ioe) {
      System.err.println(ioe.getMessage());
    }
  }
}
```

מה עושה ה Servlet הזה?

ChatServlet

- השרותונים הם Singletons
  - אם שני לקוחות (דפדפנים) ניגשים לאותו ה Servlet הפניות מופנות לאותו עצם בזכרון
- ננצל עובדה זו כדי לכתוב **ChatServlet**
  - השרותון מציג טופס html (form)
  - ערכי השדות שהמשתמש מקליד משורשרים כפרמטרים לשורת הכתובת והעמוד נטען בשנית
  - השרותון מציג בנוסף לטופס את היסטורית השיחה (השמורה בפרמטרים)

```
public class ChatServlet extends HttpServlet {

  private String[] lastMessages = new String[10];
  private String[] lastNames    = new String[10];

  public ChatServlet() {
  }

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse resp) {
    java.io.PrintWriter writer = resp.getWriter();
    String name = req.getParameter("name");
    String text = req.getParameter("text");

    resp.setContentType("text/html");

    if (text != null && text.length() > 0) {
      for (int i=9; i>0; i--) {
        System.out.println("i="+i);
        lastMessages[i] = lastMessages[i-1];
        lastNames   [i] = lastNames   [i-1];
      }
      lastMessages[0]= text;
      lastNames[0]   = name == null || name.length()==0 ? "anonymous" : name;
    }
```

```
    writer.println("<HTML><HEAD>");
    writer.println("<TITLE>Chat Servlet</TITLE>");
    writer.println("</HEAD><BODY>");
    writer.println("<H1>Let's Chat!</H1>");


    writer.println("<form name=\"input\" action=\""
                   +this.getClass().getName()
                   +"\" method=\"get\">");
    writer.println("Name: <input type=\"text\" name=\"name\"");

    if (name != null && name.length() > 0)
      writer.println(" value=\""+name+"\">");
    else
      writer.println(">");

    writer.println(" Message: <input type=\"text\" name=\"text\">");
    writer.println(" <input type=\"submit\" value=\"Send\">");
    writer.println("</form>");

    for (int i=0; i<10; i++)
      if (lastNames[i] != null && lastMessages[i] != null)
          writer.println("<P>"+lastNames[i]+": " +lastMessages[i]+"</P>");

    writer.println("</BODY></HTML>");
  }
}
```

- מכיוון שכל השרותונים הם Singletons הנטענים דינאמית, מופיע מימוש תכונות אלו במחלקת הבסיס, **`HttpServlet`**

```
public abstract class HttpServlet {

  private static java.util.Map<String,HttpServlet> servlets =
                        new java.util.TreeMap<String,HttpServlet>();

  static HttpServlet getInstance(String class_name) {
    HttpServlet servlet = servlets.get(class_name);
    if (servlet != null)
      return servlet;

    try {
      java.lang.reflect.Constructor constructor =
            Class.forName(class_name).getConstructor(new Class[] { });

      servlet = (HttpServlet) constructor.newInstance(new Object[] { });
    } catch (Exception e) {
      return null;
    }

    servlets.put(class_name,servlet);
    return servlet;
  }

  protected HttpServlet() {}

  protected abstract void doGet(HttpServletRequest req, HttpServletResponse resp);
}
```

פעולת השרת: HttpServer

1. יצירת שקע והמתנה ללקוחות
2. עבור כל לקוח חדש (accept) בצע:
    1. קרא את השורה הראשונה בהודעה (header)
    2. אם מדובר בהודעת GET:
        1. נתח את המחרוזת במקום המיועד לשם הקובץ וחלץ ממנה את שם ה Servlet המבוקש ואת שמות המשתנים וערכיהם (לפי התווים: '?' , '&' , '=' , '/')
        2. קבל הפנייה לשרותון המתאים (getInstance)
        3. צור את מחלקות העזר (**`HttpServletRequest`**, **`HttpServletResponse`**)
        4. אם השרותון נמצא:
            - קרא ל **`doGet`** על השרותון עם מחלקות העזר המתאימות

            אחרת
            - הצג הודעת שגיאה

יצירת שקע והמתנה ללקוחות

```
public class HttpServer {
  public static void main(String[] arguments) {
    int port = 8888;

    ServerSocket server_socket = null; // dummy initialization
    try {
      server_socket = new ServerSocket(port);
    } catch (IOException ioe) {
      System.out.println("Failed to bind to port "+port+": "+ioe.getMessage());
      System.exit(1);
    }

    while (true) {
      Socket connection_socket = null;
      try {
        connection_socket = server_socket.accept();
      } catch (IOException ioe) {
        System.out.println("Accept failed: "+ioe.getMessage());
        System.exit(1);
      }

      BufferedReader in = null;
      String servlet    = null;
      java.util.Map<String,String> parameters = null;

      try {
        in = new BufferedReader(
                              new
    InputStreamReader(connection_socket.getInputStream()));
```

ניתוח שורת הכותרת

```
String line;

while (true) {
  line = in.readLine();
  if (line==null)         break;
  if (line.length() == 0) break;

  if (line.startsWith("GET ")) {
    String get_params[] = line.split(" ");
    String path = get_params[1];

    servlet = path.substring(1);
    int index_of_question = servlet.indexOf('?');
    int index_of_slash    = servlet.indexOf('/');

    parameters = new java.util.TreeMap<String,String>();
    if (index_of_question != -1) {
      String params = servlet.substring(index_of_question+1);
      String[] params_values = params.split("&");
      for (String pv: params_values) {
        String[] p_and_v=pv.split("=");
        parameters.put(p_and_v[0],
                       p_and_v.length==2 ? p_and_v[1] : "");

        servlet = servlet.substring(0,index_of_question);
     }
    } else if (index_of_slash != -1) {
      String path_param = servlet.substring(index_of_slash+1);
      parameters.put("path",path_param);
      servlet = servlet.substring(0,index_of_slash);
    }

  }
}
```

```
 PrintWriter out = null;
  try {
    out = new PrintWriter(connection_socket.getOutputStream());

   } catch (IOException ioe) {
    System.out.println("Could not send the output: "+ioe.getMessage());
    System.exit(1);
  }

  HttpServlet s = HttpServlet.getInstance(servlet);

  if (s != null) {
    ServletInvocation invocation = new ServletInvocation(parameters,out);
    s.doGet(invocation,invocation);
  } else {
    out.print("HTTP/1.1 404\r\n");
    out.print("Content-Type: text/html\r\n");
    out.print("Connection: close\r\n");
    out.print("\r\n"); // end of header
    out.println("<HTML><HEAD>");
    out.println("<TITLE>404 Not Found</TITLE>");
    out.println("</HEAD><BODY>");
    out.println("<H1>Not Found</H1>");
    out.println("<P>The requested URL was not found on this server.</P>");
    out.println("</BODY></HTML>");
  }

  try {
    out.close();
    in.close();
    connection_socket.close();
  } catch (IOException ioe) {
    // who cares?
  }
}
```

הפעלת השרותון המתאים

J2EE Web Containers

- שרתי Web דומים מאוד לאלו שהצגנו כאן, קיימים בצורה מסחרית (חלקם בתשלום וחלקם חופשיים)
- לדוגמא:
  - Apache Tomcat
  - Sun Java System Web Server
  - Oracle Containers for Java (OC4J)
  - Many more…
- שרתים אשר מממשים את תקן ה Servlet במלואו, מאפשרים יבילות לכותבי השרותונים
  - אנו נוכל להריץ את ה Servlets שכתבנו על כל שרת הממלא את התקן Servlet 2.4 Specification JSR-000154 JavaTM


Servlet Framework

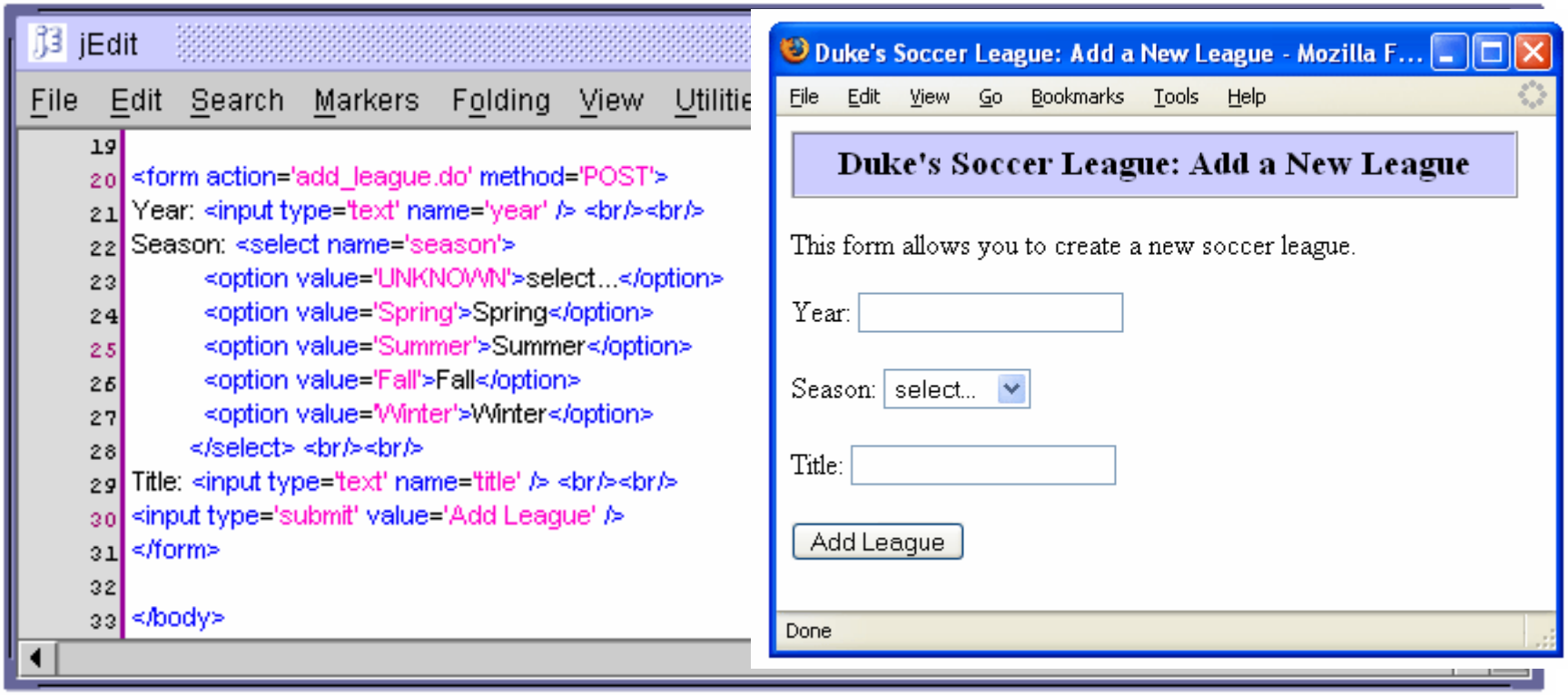
Hello World Servlet

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

public class HelloWorld extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Hello World!</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Hello World!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

טפסים (forms)


jEdit
File Edit Search Markers Folding View Utilitie
19
20 <form action='add_league.do' method='POST'>
21 Year: <input type='text' name='year' />

22 Season: <select name='season'>
23 <option value='UNKNOWN'>select...</option>
24 <option value='Spring'>Spring</option>
25 <option value='Summer'>Summer</option>
26 <option value='Fall'>Fall</option>
27 <option value='Winter'>Winter</option>
28 </select>

29 Title: <input type='text' name='title' />

30 <input type='submit' value='Add League' />
31 </form>
32
33 </body>
Duke's Soccer League: Add a New League - Mozilla F...
File Edit View Go Bookmarks Tools Help
Duke's Soccer League: Add a New League
This form allows you to create a new soccer league.
Year:
Season: select...
Title:
Add League
Done

העברת פרמטרים ב GET

```
GET /admin/add_league.do?year=2005&season=Winter&title=Westminster+Indoor HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en-US; rv:1.4
Gecko/20030624 Netscape/7.1
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain
video/x-mng,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

העברת פרמטרים ב POST

```
POST /admin/add_league.do HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en-US; rv:1.
Gecko/20030624 Netscape/7.1
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plai
0.8,video/x-mng,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://localhost:8080/controller/admin/add_league.html
Content-Type: application/x-www-form-urlencoded
Content-Length: 55
year=2005&season=Winter&title=Westminster+Indoor+Soccer
```

GET vs. POST

The HTTP GET method is used when:

- The processing of the request is idempotent
- The amount of form data is small
- You want to allow the request to be bookmarked

The HTTP POST method is used when:

- The processing of the request changes the state of the server, such as storing data in a database
- The amount of form data is large
- The contents of the data should not be visible in the URL (for example, passwords)

Session Management

- פרוטוקול HTTP הוא חסר הקשר אולם ה Web Container מספק אמצעים ליצירת הקשר
  - לדוגמא: עגלת קניות וירטואלית

- הרעיון:
  - עבור כל לקוח נשמור מידע נשמר באובייקט Java מטיפוס HttpSession
  - בכל פנייה לשרת הלקוח ישלח מזהה ייחודי
  - השרת יקבל את המזהה ויקשר את הלקוח עם המידע שנשמר בעבורו

Session Management Strategies

- עוגיות (Cookies)
  - שרת יכול להוסיף לתשובה ללקוח עוגייה – קובץ טקסט עם זוגות name=value
  - הלקוח ישלח בכל התקשרות עתידית עם השרת את כל העוגיות שהוא קיבל ממנו בעבר
  - בפועל עוגיות מכילות רק מזהה ייחודי – שאר המידע נשמר בשרת
- שכתוב קישוריות (URL Rewriting)
  - כל הקישוריות (href) בעמוד ה HTML ישוכתבו להכיל גם את המזהה הייחודי כמשתנה נוסף בשורת הכתובת
  - אסטרטגיה זו שימושית במקרה שהדפדפן חוסם עוגיות

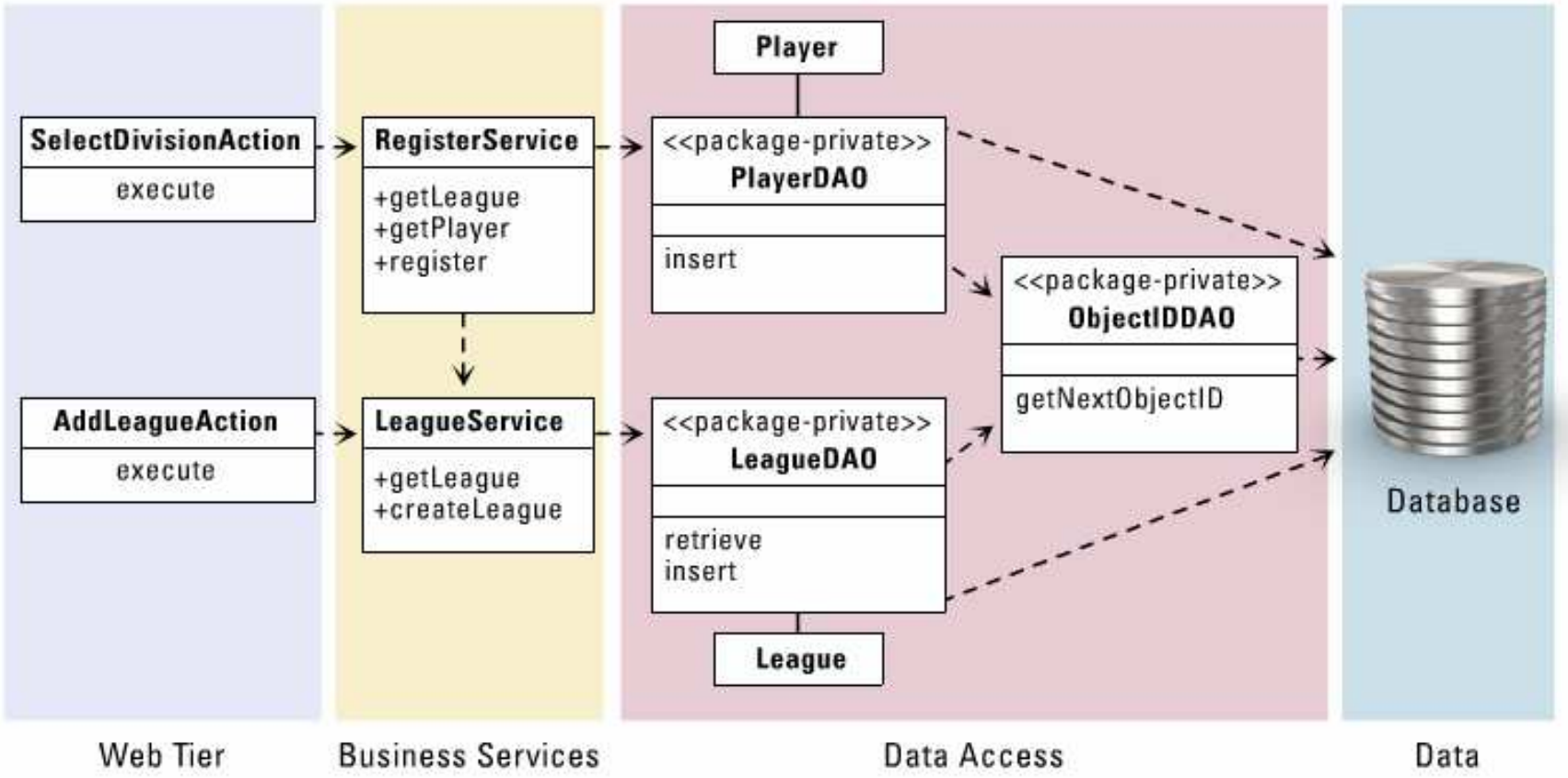
Web Container Aspects

- כל הודעה ל Servlet מיורטת ע"י השרת לצורך הוספת תמיכה ב Sessions
- ואולם ניתן ליירט הודעות גם כדי לבצע פעולות נוספות שיגדיר כותב היישום
- הדבר אנלוגי לכתיבת Aspect בסביבת Web Container

- Aspect ממומשים ע"י הגדרת Filters
  - כל Filter מגדיר את ההודעות שמעניינות אותו לפי תבנית ה URL שאליה מיועדות ההודעות
  - כל ההודעה המיועדת לכתובת שיש עליה Filter מאזין תופנה קודם כל ל Filter

- סוגי פעולות שיכול לבצע Filter הם (לדוגמא):
  - בקרת גישה, לוג, מדידות, דחיסת מידע ועוד...

Data Access Object Pattern



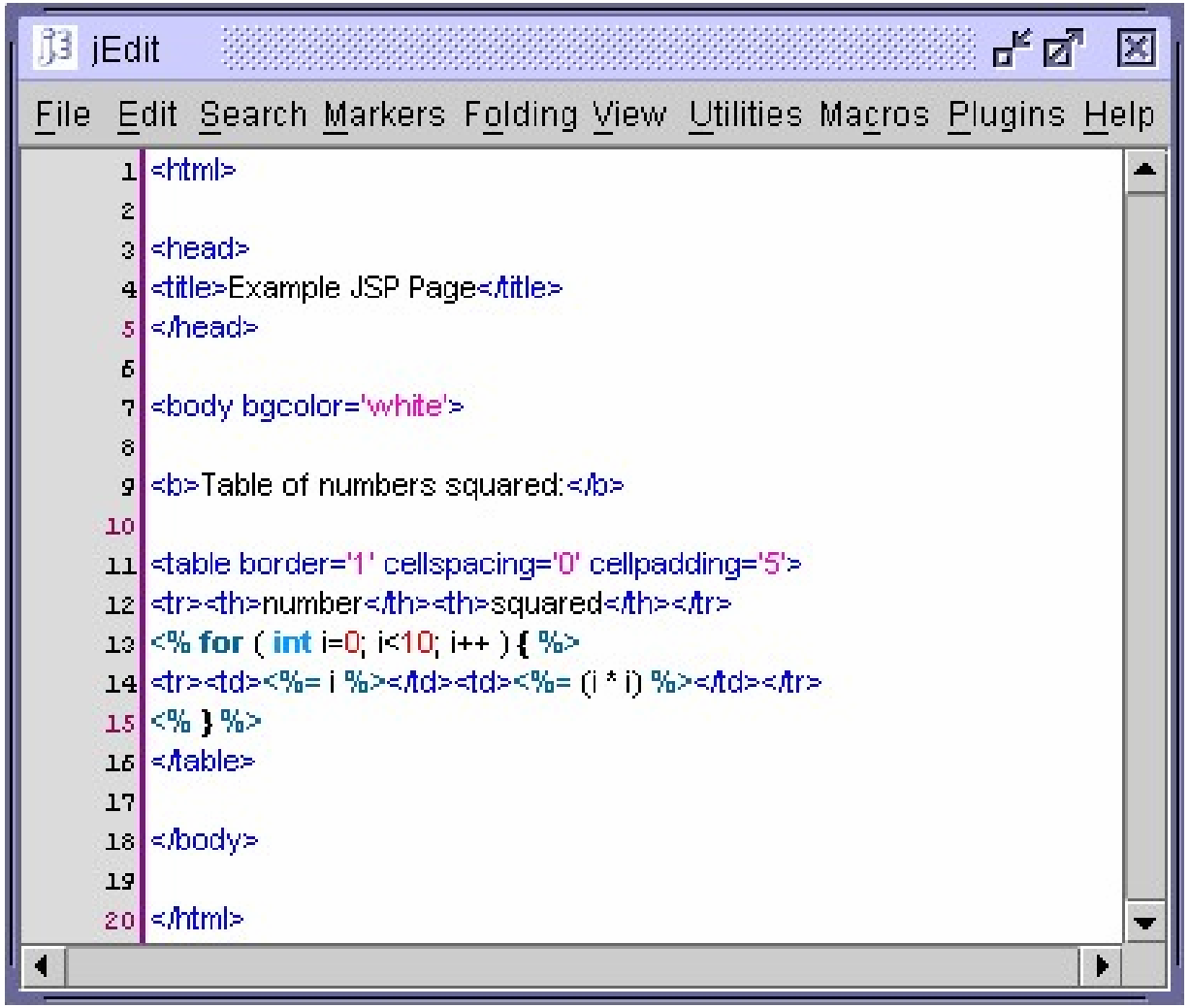
Data Access Object Pattern

- The data access object (DAO) pattern separates the business logic from the data access (data storage) logic
- The data access implementation (usually JDBC technology calls) is encapsulated in DAO classes.
- The DAO pattern permits the business logic and the data access logic to change independently.
- For example, if the DB schema changes, then you would only need to change the DAO methods, and not the business services or the domain objects.

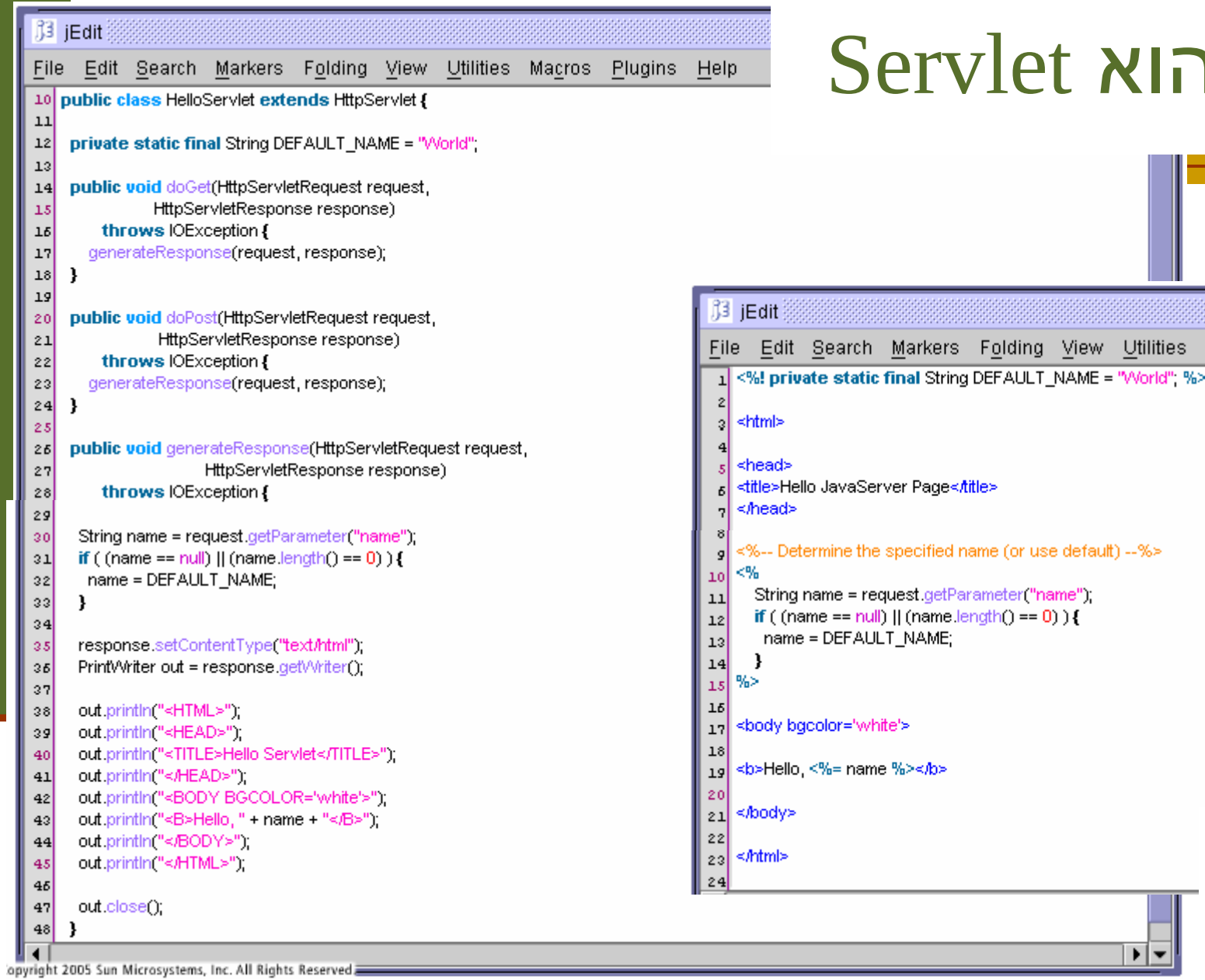
Java / HTML

- על אף שרצינו שפלט התוכנית יהיה ב HTML איננו רוצים לערבב קוד Java וקוד HTML
  - Model/View Separation
  - לפעמים נכתבים ע"י מתכנתים שונים
- יש צורך בטכנולוגיה שתאפשר פיתוח אורתוגונלי של המודל ושל ההצגה (גם אם בפועל הקוד שירוץ ממזג את שניהם)
- טכנולוגיית JSP מתיימרת להשיג מטרה זו בדיוק:
  - JSP היא שפת תגיות (בדומה ל html) המאפשרת לשלב בתוכה תוכן דינאמי
  - בזמן ריצה קובץ ה JSP עובר קומפילציה לקובץ Servlet

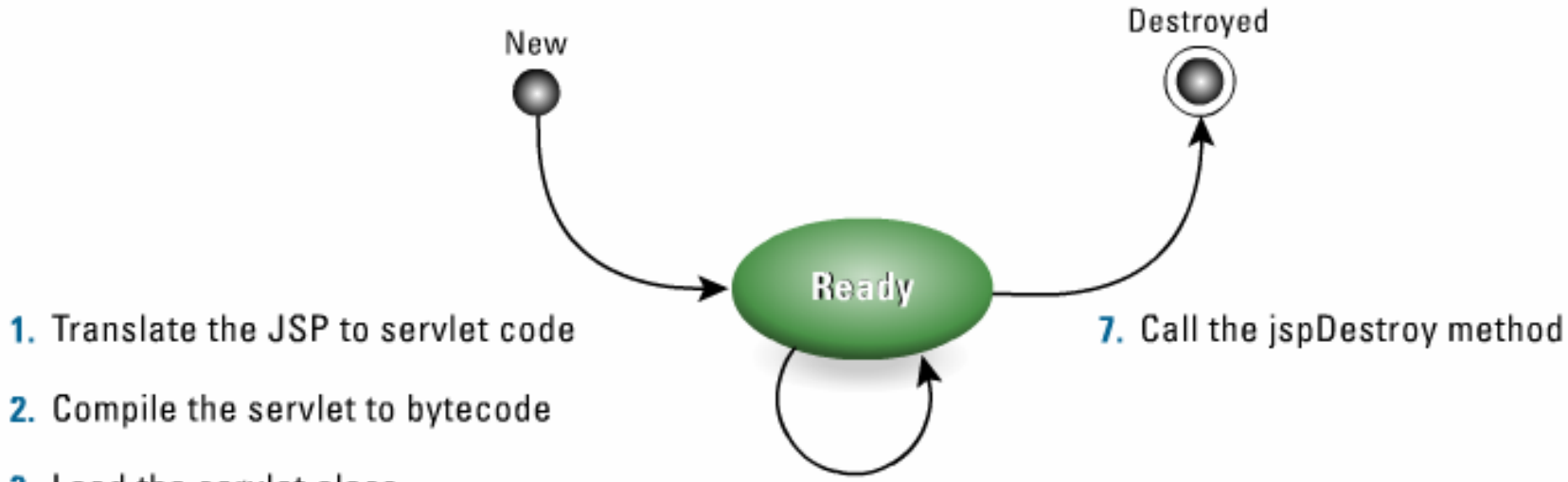
גלישה לעמוד JSP



Servlet הוא JSP



JSP Page Life Cycle



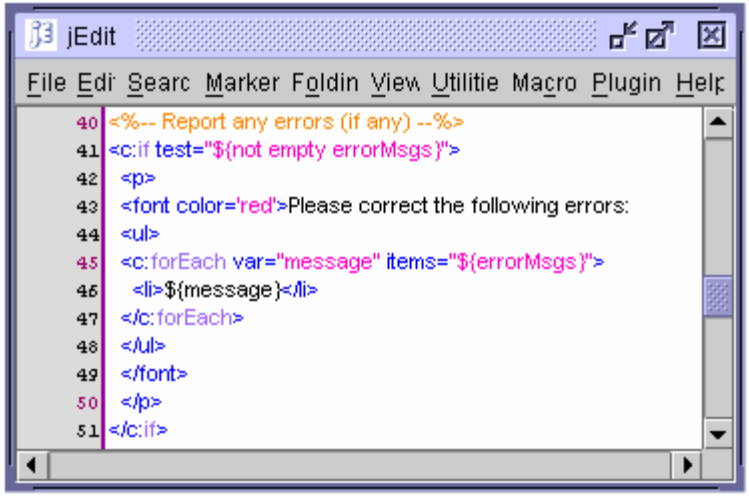
JSP / Java

- בעיות:
  - כעת במקום לזהם קוד Java ב HTML אנו מזהמים קוד JSP ב Java
  - לתחביר הסטנדרטי של JSP יכולות מוגבלות (בעיקר שילוב קוד Java ו HTML)
  - יש צורך בטכנולוגיה שתאפשר פיתוח לוגיקה ב Java ושיבוצה בתוך קובצי JSP ללא זיהום ה JSP
- תגיות JSP מתיימרות להשיג מטרה זו בדיוק – המתכנתת יכולה להגדיר בעצמה תגיות JSP
  - התגיות ימומשו בשפת Java בקובץ נפרד
  - הדבר הופך את JSP לשפת תכנות לכל דבר עם מבני בקרה, שימוש ב Java Beans ועוד...

JSP Custom Tags

```
<%-- Report any errors (if any) --%>
<%
   // Retrieve the errorMsgs from the request-scope
   List errorMsgs = (List) request.getAttribute("errorMsgs");
   if ( (errorMsgs != null) && !errorMsgs.isEmpty() ) {
%>
<p>
<font color='red'>Please correct the following errors:
<ul>
<%
     Iterator items = errorMsgs.iterator();
     while ( items.hasNext() ) {
       String message = (String) items.next();
%>
  <li><%= message %></li>
<%
     } // END of while loop over errorMsgs
%>
</ul>
</font>
</p>
<%
  } // END of if errorMsgs is not empty
%>
```

- בעזרת תגיות משתמש ניתן להשתמש במעצבים ללא ידע בתכנות לצורך כתיבת עמודי JSP



Job Roles

- ארכיטקטורת Web Applications מחזקת את הצורך לחלק את משימות הפיתוח בין מתכנתים שונים

- Web Designer – Creates view elements (web pages and layouts)

- Web Component Developer – Creates controller elements

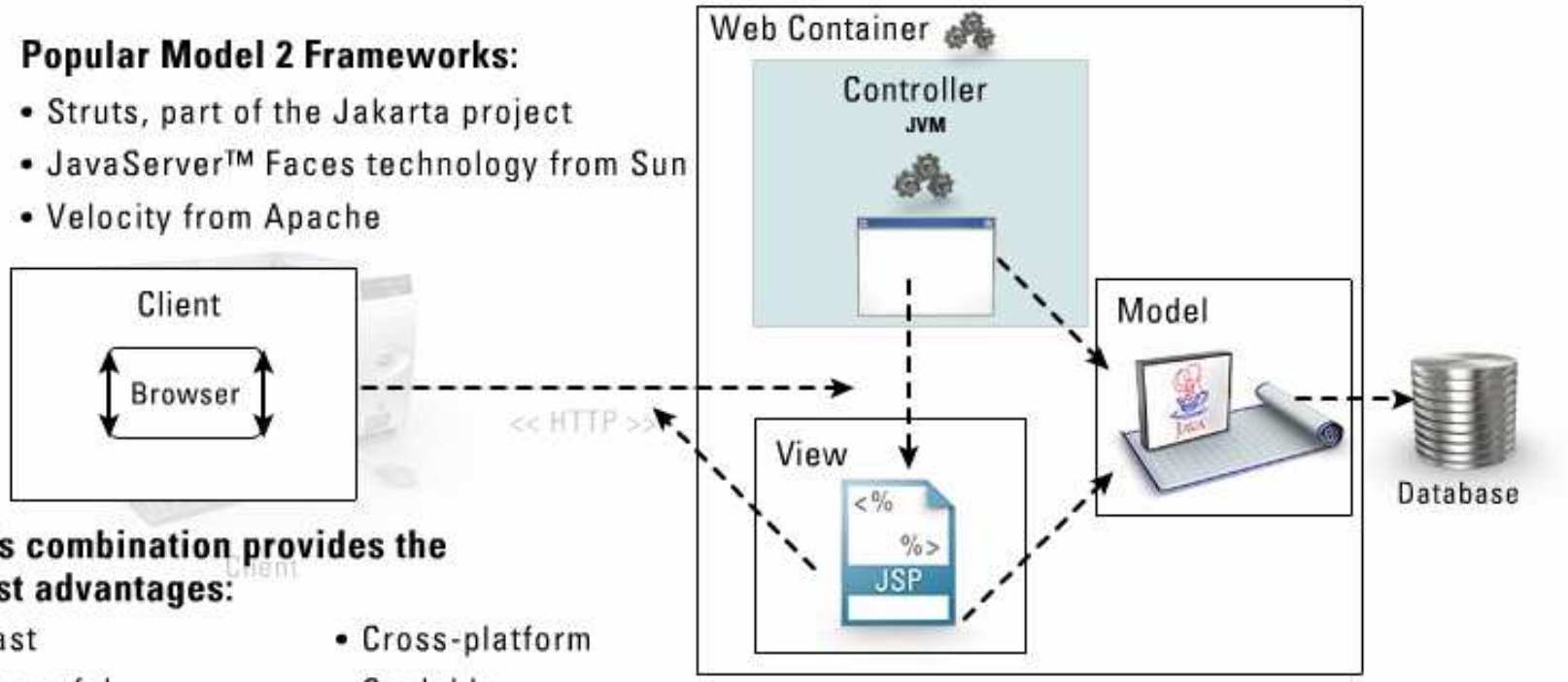
- Business Component Developer – Creates model elements

- Data Access Developer – Creates database access elements

Model 2 Framework

Popular Model 2 Frameworks:

- Struts, part of the Jakarta project
- JavaServer™ Faces technology from Sun
- Velocity from Apache



This combination provides the most advantages:

- Fast
- Powerful
- Easy to create
- Cross-platform
- Scalable
- Maintainable

Types of View Components

Data Presentation:

- Includes any data presented on a visible screen
- There are many forms of presentation including graphs, spreadsheets, listings, and so on

Data Forms:

- Forms for data entry are also considered view components

Navigational Aids:

- This type of view includes menus, hyperlinks, site maps, and so on

Informational Screens or Pop-Ups:

- This type of view includes welcome text, instructions, help screen, error messages, confirmation dialogs, and so on

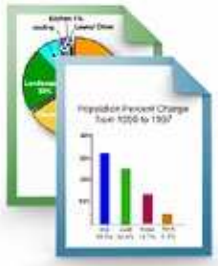
Types of Controller Components

Process input from a user

- Enter text data
- Select radio or checkbox buttons
- Drop-down lists
- Use a submit button to send the data

Screen navigation

- Hyperlinks in web pages



Preparing data for view components

- Reports present business data in a view

Form-processing controller servlets usually perform the following tasks:

1. Retrieve form parameters from the HTTP request
2. Perform any data conversion on the form parameters
3. Verify the form parameters
4. Execute the business logic
5. Dispatch to the next view components based on the results of the previous steps

Entity Component Limitations

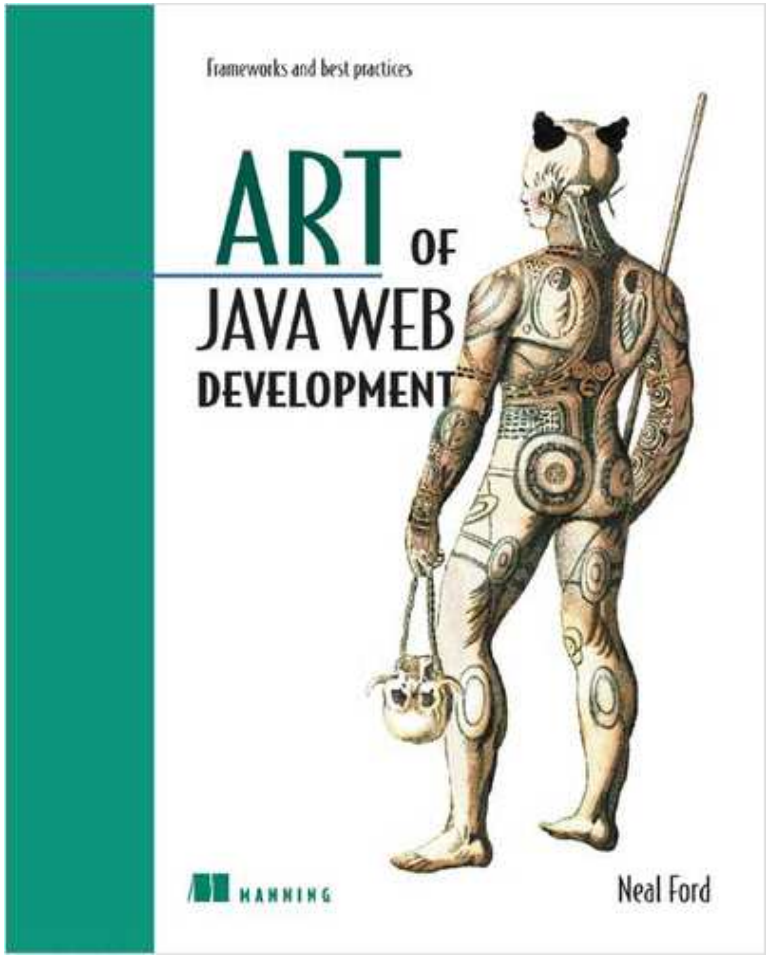
The following entity-related operations cannot be performed by the entity component itself:

- Creation – Creating a new instance of the entity
- Retrieval – Retrieving a unique instance in the data store
- Selection – Retrieving a set of instances in the data store
- Aggregation – Performing a calculation (such as an average) over a set of instances
- Deletion – Removing an instance from the data store

MVC Using Struts

- מימוש רעיון ה MVC בסביבת Web Container מכיל פעולות שחוזרות על עצמן בכל יישום:
  - ה- Controller מבצע את בדיקת תקינות הנתונים
  - ה- Controller מפנה את הבקשה עם תוצאות הבדיקה לעמוד ההצגה המתאים
- היינו מעוניינים לבצע אוטומטיזציה של פעולות אלו, בדרך זו ה Controller יתרכז רק בלוגיקה העסקית וההפרדה בין View ובין Controller תאכף ע"י הסביבה
- Struts Framework מספקת את המחלקה Action אשר עוטפת את פעולת מחלקות ה- Controller
  - בדרך זו ה Controller אפילו לא יודע את **שם המחלקה** שאליה מופנית ההודעה לאחר ביצוע הלוגיקה העסקית

Java Web Frameworks



- קיימות כמה Frameworksהמספקות תשתית לפיתוח יישומי אינטרנט מעל Servlets
- ה Frameworks השונות מסייעות לאכוף הרגלי עבודה נכונים (כגון הפרדה בין מודל ותצוגה)
- הספר

Art of Java Web Development
Struts, Tapestry, Commons, Velocity, JUnit, Axis, Cocoon, InternetBeans, WebWork
Neal Ford

סוקר חלק מהסביבות השונות, עומד על ההבדלים בינהן ומשווה בין קוד "ידני" ובין הקוד שנכתב בעזרתן

כלים וסביבות

- כאשר טכנולוגיה הופכת לפופולרית פועלים מספר כוחות:
  - כח אדם פחות מיומן צריך **כלים תומכים** כדי להשתלב במאמץ הפיתוח
  - מתגלים **שימושים חדשים** לטכנולוגיה אשר לא נתמכים בה היטב מכיוון שלא נלקחו בחשבון בתחילה
- כדי לענות על הדרישה מתפתחים כלים, סביבות ושפות ייעודיות אשר מנסים לפשט את משימות הפיתוח
- לאיזו טכנולוגיה עולה דרוש פיתוח של כלים חדשים היום?
- אילו יכולות על הכלים לספק?