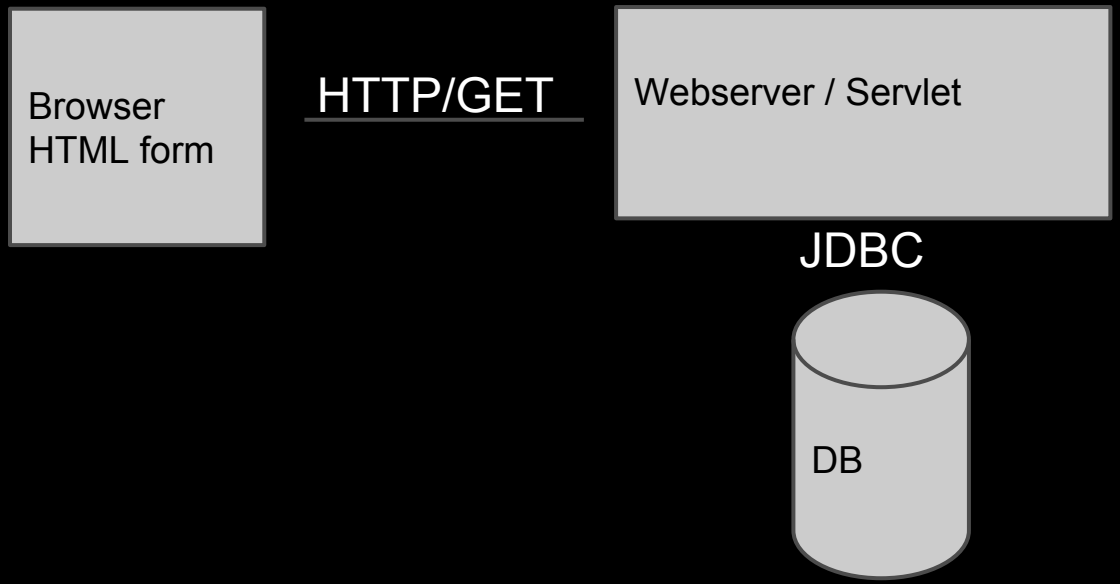
Backend

(Very) Simple server examples

Web server example



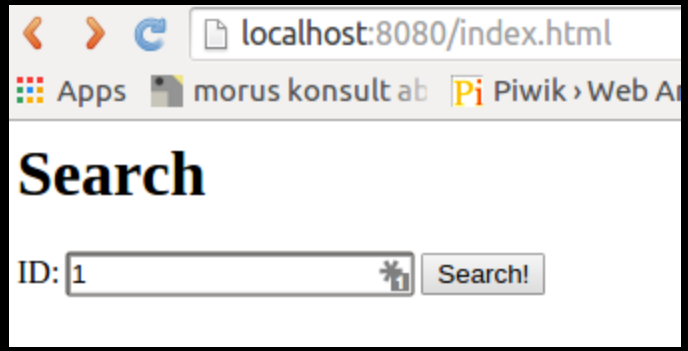
Student example

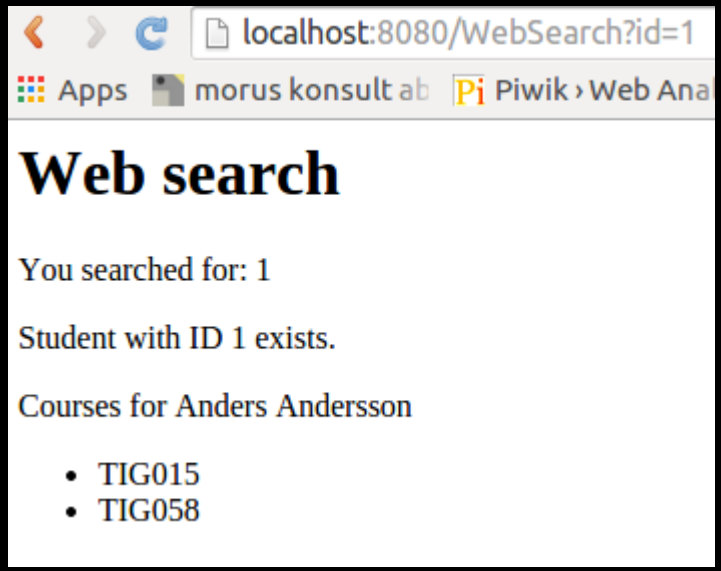
```
sqlite> .schema
CREATE TABLE students(id integer primary key asc,name varchar(30));
CREATE TABLE courses(id integer primary key asc,course_code varchar(12));
CREATE TABLE registrations(student_id int(8), course_id int(8));
sqlite> SELECT * FROM students;
1|Anders Andersson
2|Beata Bengtsson
3|Charlie Christensen
4|Dick Dale
5|Edward Eriksson
sqlite> SELECT * FROM registrations;
1|1
1|2
2|2
3|1
4|1
4|2
5|2
5|4
2|3
```

```
sqlite> SELECT * FROM courses;
1|TIG015
2|TIG058
3|TIG059
4|TIG060
```

Webserver/Servlet

Search for a student by ID





HTML FORM

```
<html>
  <body>
    <h1>Search</h1>
    <p>
      <form action="/WebSearch" method="get">
        ID: <input type="text" size="20" name="id" />
        <input type="submit" value="Search!"/>
      </form>
    </p>
  </body>
</html>
```

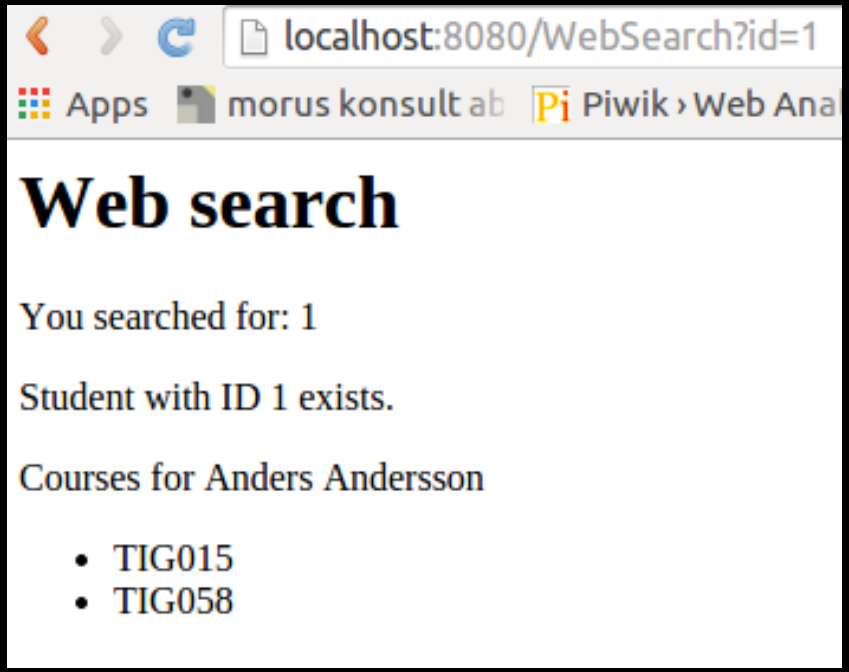
HTML FORM

```
<html>
  <body>
    <h1>Search</h1>
    <p>
      <form action="/WebSearch" method="get">
        ID: <input type="text" size="20" name="id" />
        <input type="submit" value="Search!"/>
      </form>
    </p>
  </body>
</html>
```

action=”/WebSearch”

Submit sends to /WebSearch

(on same server)



HTML FORM

```
<html>
  <body>
    <h1>Search</h1>
    <p>
      <form action="/WebSearch" method="get">
        ID: <input type="text" size="20" name="id" />
        <input type="submit" value="Search!"/>
      </form>
    </p>
  </body>
</html>
```

HTTP methods

GET -> Parameters are part of URL
HEAD
POST -> Parameters are not visible in URL
PUT
DELETE
(and then some...)

HTML FORM

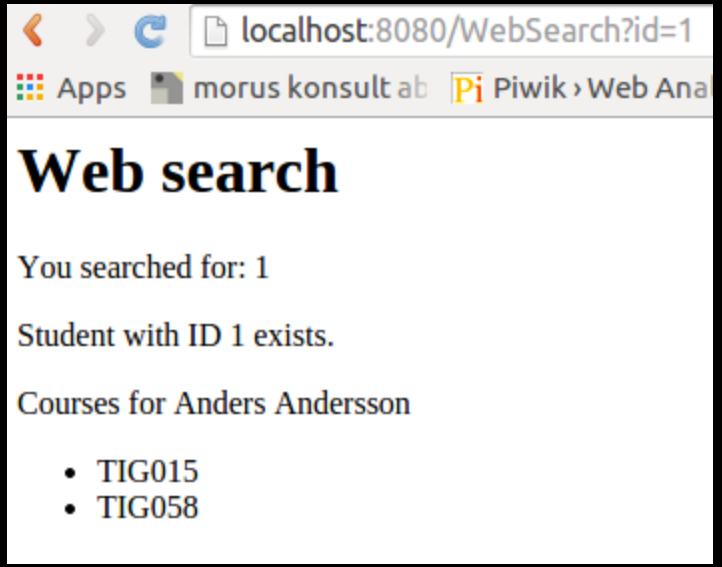
```
<html>
  <body>
    <h1>Search</h1>
    <p>
      <form action="/WebSearch" method="get">
        ID: <input type="text" size="20" name="id" />
        <input type="submit" value="Search!"/>
      </form>
    </p>
  </body>
</html>
```

HTTP GET

id is a GET parameter with value 1

GET parameters occur after ?

WebSearch?id=1



Server-side - The Servlet

```
public class WebSearch extends HttpServlet{
    private String message;
    private StudentData data = new StudentDataDB();
    public void init() throws ServletException{
        message = "Web search";
    }
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                      throws ServletException, IOException{
        String id = request.getParameter("id") ;
        if(id!=null){
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            printResult(out,id);
        }else{
            printError(response);
        }
    } // more methods...
}
```

Servlet and DB

```
public class WebSearch extends HttpServlet{
    private String message;
    private StudentData data = new StudentDataDB();
    public void init() throws ServletException{
        message = "Web search";
    }
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                      throws ServletException, IOException{
        String id = request.getParameter("id");
        if(id!=null){
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            printResult(out,id);
        }else{
            printError(response);
        }
    } // more methods...
}
```

Program against an interface

```
package student.data;
import java.util.ArrayList;

public interface StudentData{
    public ArrayList<String>getCourses(String studentID);
    public String getStudentName(String studentID);
    public boolean studentIDExists(String studentID);
    public String getCourseCode(String courseID);
    public void addCourse(String studentID, String courseID);
}

// Client code in servlet:
// private StudentData data = new StudentDataDB();
// NO JDBC HERE! Allows us to create new implementations without
// having to change the client code one bit ;-)
```

Implementation of StudentData

```
public class StudentDataDB implements StudentData{
    public ArrayList<String>getCourses(String studentID){
        ArrayList<String> courses = new ArrayList<String>();
        String query = "SELECT courses.course_code FROM students, courses "+
            "JOIN registrations ON "+
            "registrations.student_id = students.id and "+
            "registrations.course_id = courses.id and "+
            "students.id = "+studentID;
        try{
            ResultSet rs = DBUtils.executeQuery(query);
            while(rs.next()){
                courses.add(rs.getString("course_code"));
            }
        }catch(Exception e){
            System.err.println("Error getting courses: "+e.getMessage());
        }
        return courses;
    }
   // More metods...
}
```

A Servlet can be used as a service

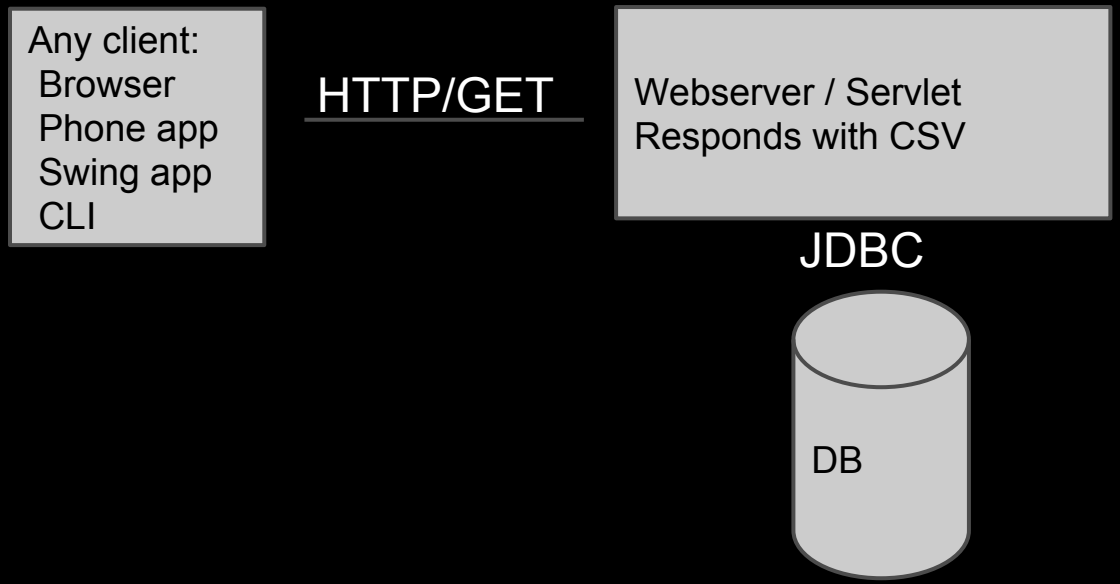
Let's say we want CSV data rather than HTML. A servlet can do that:

```
// inside doGet()...

response.setContentType("text/csv");

PrintWriter out = response.getWriter();
String id = request.getParameter("id");
if( data.studentIDExists(id) ){
    int numberOfCourses = data.getCourses(id).size();
    int count=0;
    for( String course : data.getCourses(id) ){
        out.print(course);
        if(++count < numberOfCourses){
            out.print(","); // CSV! ;-)
        }
    }
    out.println("");
}else{
    out.println(id + " " + message);
}
```

Web service example



Now, any app can use the service

Get a connection

Send GET with parameters (id=xxx)

Read response (CSV)

Parse CSV and do something with each field

Example

bash:

```
$ for course in $(GET http://localhost:8080/getbyid?id=1|tr ',' ' ')
  do echo Kurs: $course
done
Kurs: TIG015
Kurs: TIG058
$
```

Java:

```
$ java CLI 1
ID 1 has the following courses:
Course: TIG015
Course: TIG058
```

Even telnet

```
telnet localhost 8080
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET /getbyid?id=1
HTTP/0.9 200 OK
Server: Winstone Servlet Engine v0.9.10
Content-Type: text/csv;charset=ISO-8859-1
Connection: Close
Date: Fri, 01 May 2015 14:21:02 GMT
X-Powered-By: Servlet/2.5 (Winstone/0.9.10)

TIG015,TIG058
Connection closed by foreign host.
```

Java code:

```
import java.net.*;
import java.io.*;
import java.util.*;
public class CLI{
  private static final String THE_URL = "http://localhost:8080/getbyid?id=";
  public static void main(String[] args) throws Exception{
    String id = args[0];
    URL webservice = new URL(THE_URL+id);
    URLConnection con = webservice.openConnection(); // Open (with parameters)
    BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
    String line;
    while ((line = in.readLine()) != null){          // Read response
      if(line.endsWith("Not found")){
        System.out.println("ID "+id+" was not found");
      }else{
        System.out.println("ID "+id+" has the following courses:");
        StringTokenizer st = new StringTokenizer(line, ","); // Parse CSV
        while(st.hasMoreTokens()){
          System.out.println("Course: " + st.nextToken());   // Do something
        }
      }
    }
    in.close(); // Close the connection
  }
}
```

Alternatives

Use JDBC to a database server

Advantage: No need for back-end server

Disadvantages:

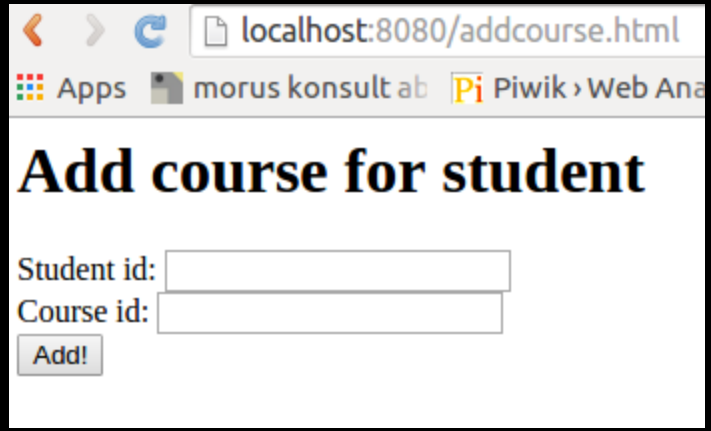
Less generic - all clients must know about the database and have access to it

You can have less business rules in a DB (whereas a servlet can do logic)

What about writing to the DB?

A Servlet (or any web service) can act on requests and write to the DB.

You can use web,
command line,
or an application
as with the previous
example.



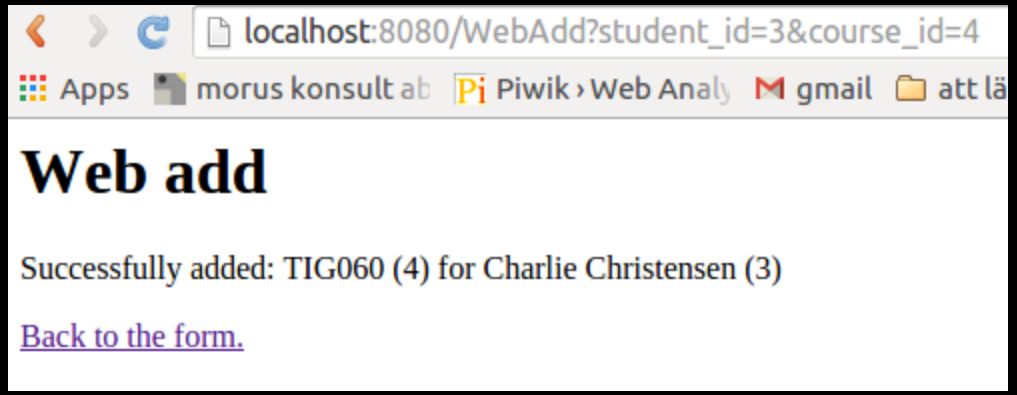
What would the Servlet have to do?

- Read parameters student_id and course_id
- Check that they are valid
- Insert into the registrations table
- Confirm to the user the result

In the real world, we would add login to avoid spam and attacks, of course!

Response

Something like this:



Java webservice

- An alternative to Servlets are WebServices
- Java API for XML Web Services (JAX-WS) (since java 6)
- SOAP based
- Create web service (listens for requests)
- Create a client (talks to the web service)

Java comes with tools for this

```
package studentws.student;  // Server-side

import studentws.student.data.*;
import javax.jws.WebService;
import java.util.List;
import java.util.ArrayList;

@WebService
public class StudentService{
    private StudentData data = new StudentDataDB();
    public List<String> getCourses(String studentID){
        return data.getCourses(studentID);
    }
}
```

Generating stub classes

```
wsgen -s src -cp . -d . studentws.student.StudentService
ls src/studentws/student/jaxws/
GetCourses.java  GetCoursesResponse.java
```

The publisher

```
package studentws.student;
import javax.xml.ws.Endpoint;

public class StudentServiceEndpointPublisher{
    public static void main(String[] args){
        Endpoint.publish
       ("http://localhost:8081/StudentWS/StudentService",
         new StudentService());
    }
}
// java studentws.student.StudentServiceEndpointPublisher&
```

Creating the client

```
// On the client computer
wsimport -s src -d . http://localhost:8081/StudentWS/StudentService?wsdl
$ tree src/
src/
└── studentws
    └── student
        ├── GetCourses.java
        ├── GetCoursesResponse.java
        ├── ObjectFactory.java
        ├── package-info.java
        ├── StudentService.java
        └── StudentServiceService.java <- I know, StudentServiceService...
```

Writing the client

```
package client;
import studentws.student.StudentService;
import studentws.student.StudentServiceService;

public class StudentClient{
    public static void main(String[] args){
        String studentID = args[0];
        StudentServiceService studentService =
                                new StudentServiceService();
        StudentService service = studentService.getStudentServicePort();
        System.out.println(service.getCourses(studentID));
    }
}

$ java client.StudentClient 1
[TIG015, TIG058]
```

How does it work?

Snippets from network traffic

```
POST /StudentWS/StudentService HTTP/1.1
<ns2:getCoursesResponse xmlns:ns2="http://student.
studentws/">
  <return>TIG015</return>
  <return>TIG058</return>
</ns2:getCoursesResponse>
```

See also: http://en.wikipedia.org/wiki/SOAP