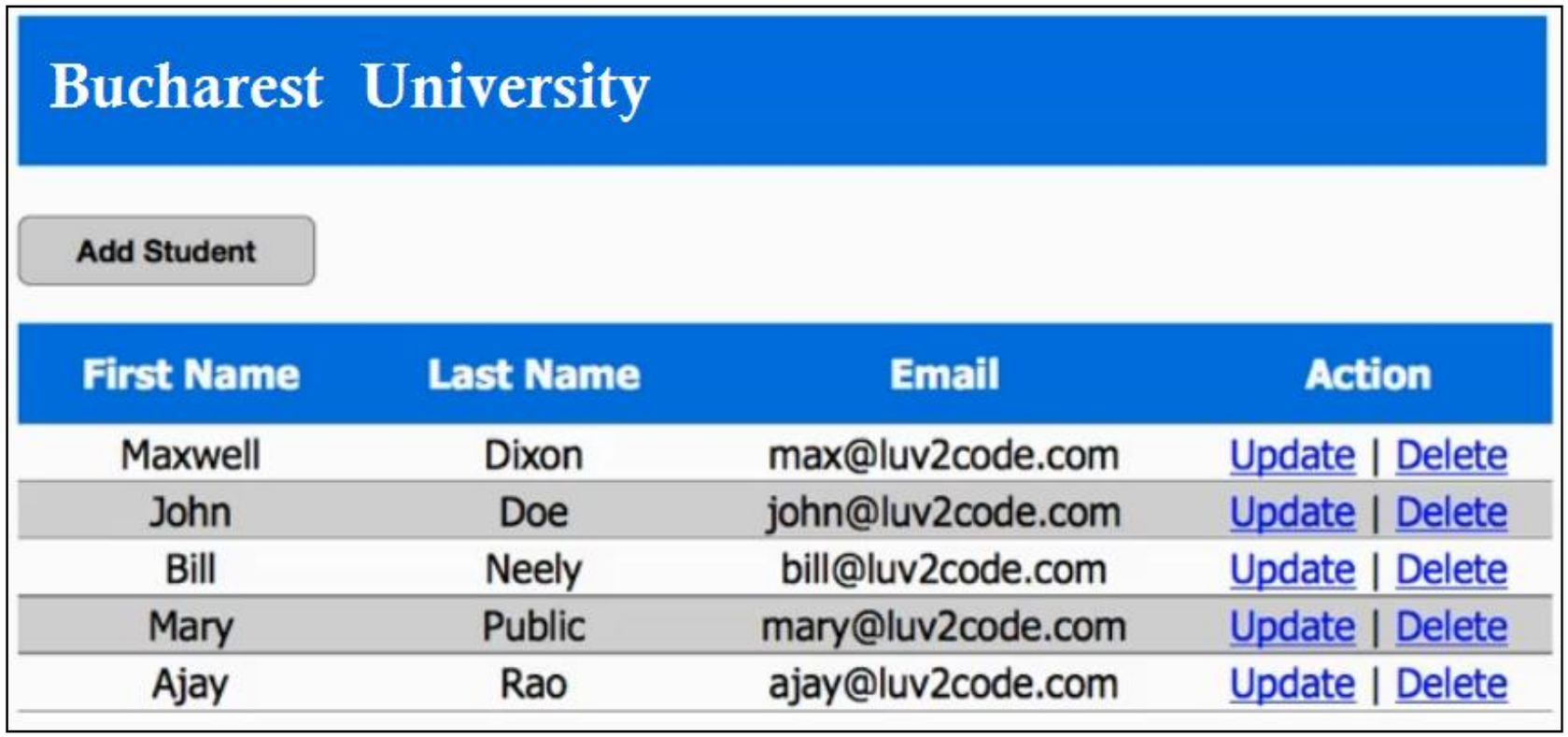
Student Tracker App

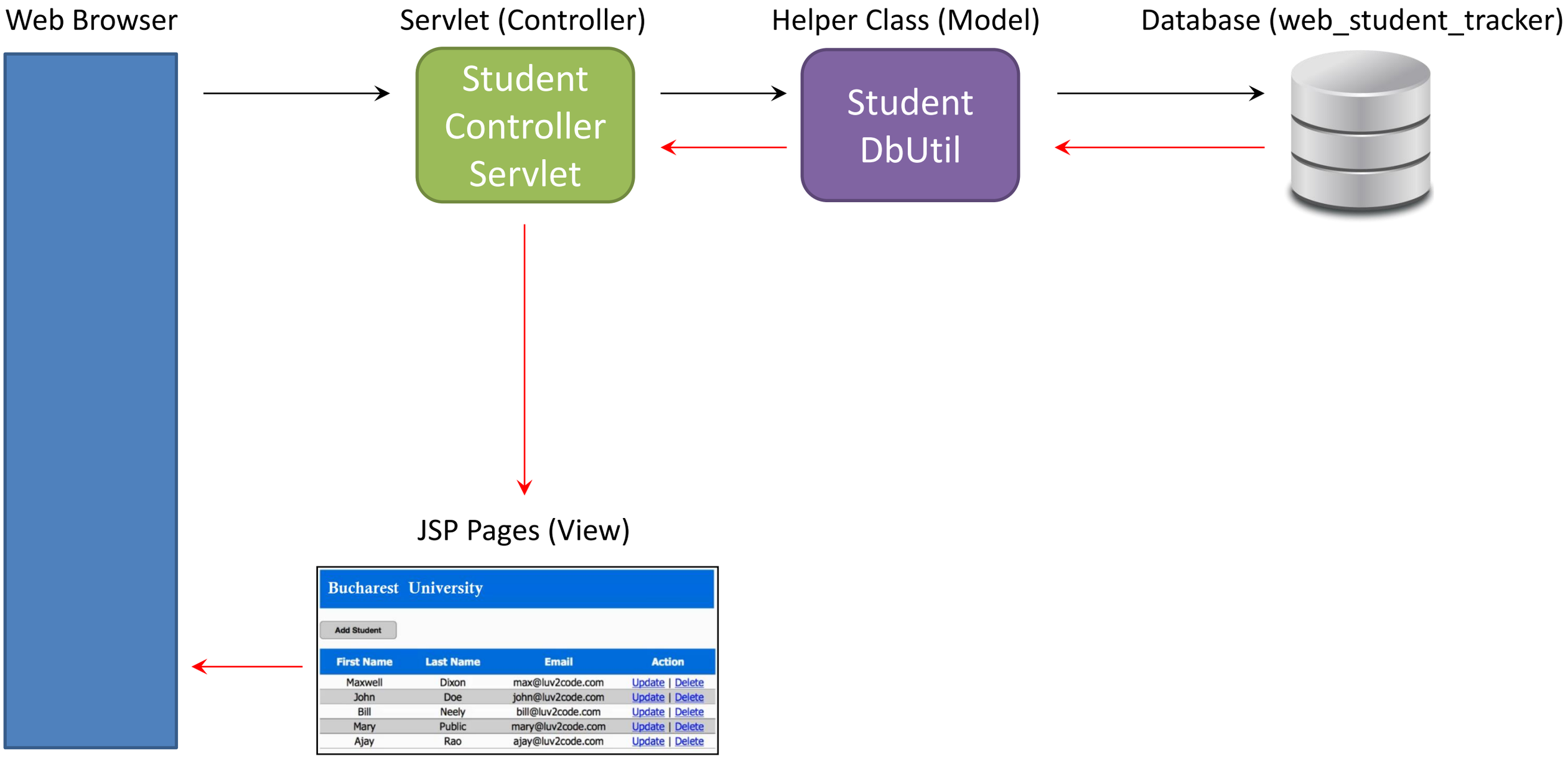
Application Features:

- List Students
- Add a new Student
- Update a Student
- Delete a Student



Student Tracker App

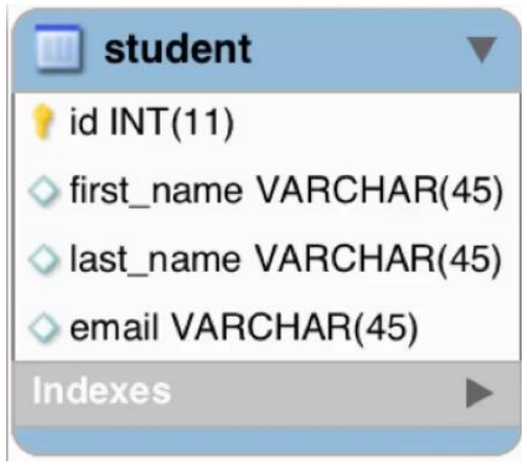
Application Architecture:



Student Tracker App

1 mySQL scripts:

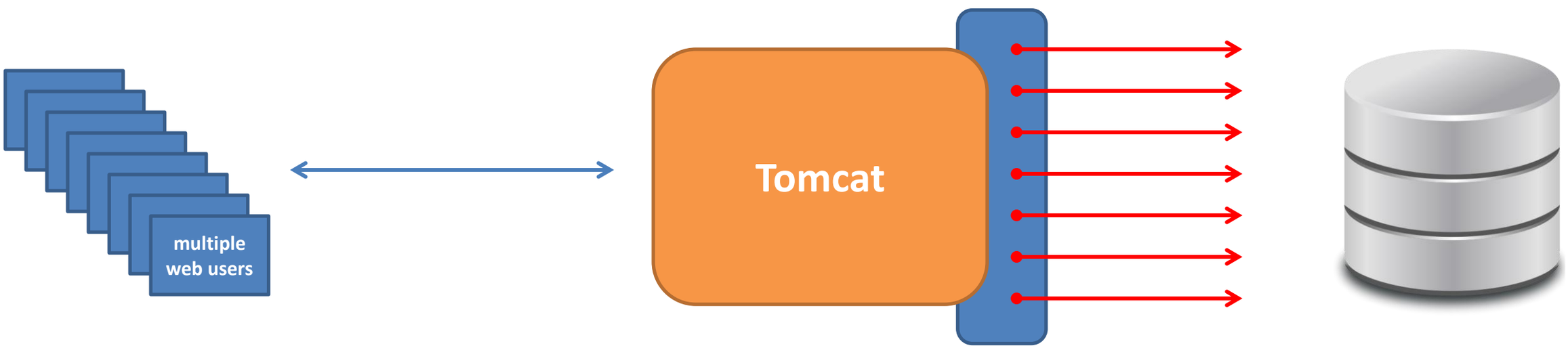
1. Create user id and password
2. Create DB and ‘student’ table
3. Populate ‘student’ table



DB: *web_student_tracker*
Table: *student*

Student Tracker App

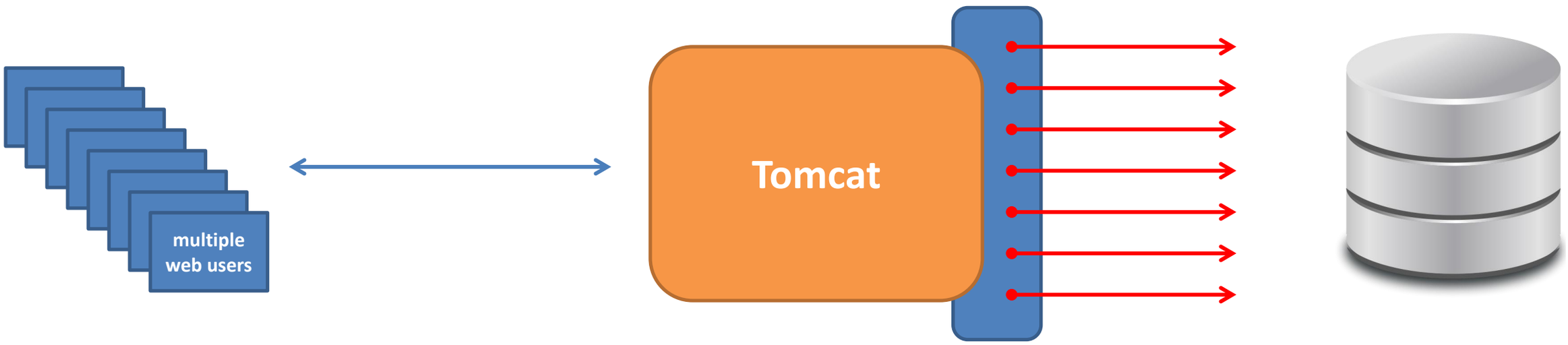
2 Setting up a Tomcat Database Connection Pool:



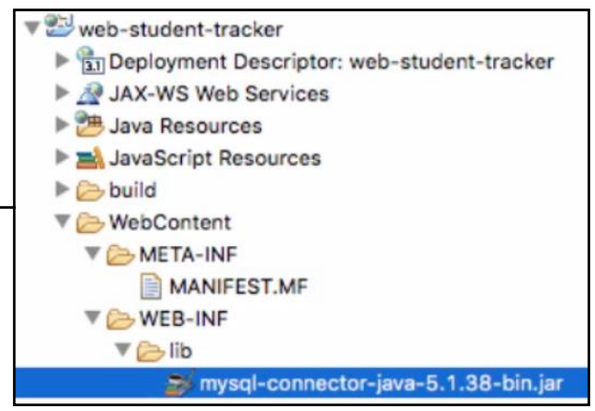
1. Download JDBC Driver JAR file
2. Define connection pool in META-INF/context.xml
3. Get connection pool reference in Java code

Student Tracker App

2.1 Setting up a Tomcat Database Connection Pool:



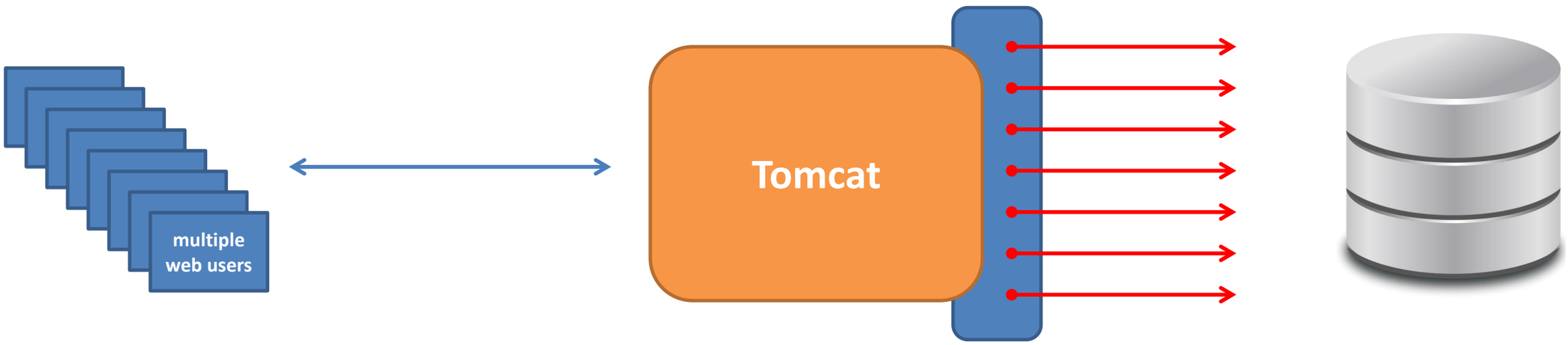
- http://dev.mysql.com/downloads
- Place the JAR file in my app's WEB-INF/lib



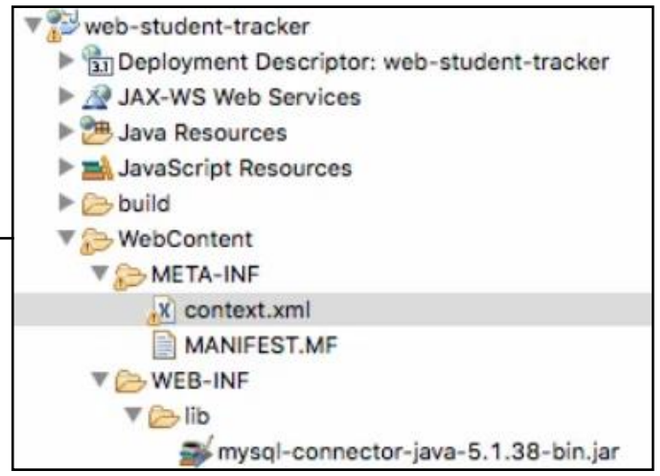
1. **Download JDBC Driver JAR file**
2. Define connection pool in META-INF/context.xml
3. Get connection pool reference in Java code

Student Tracker App

2.2 Setting up a Tomcat Database Connection Pool:



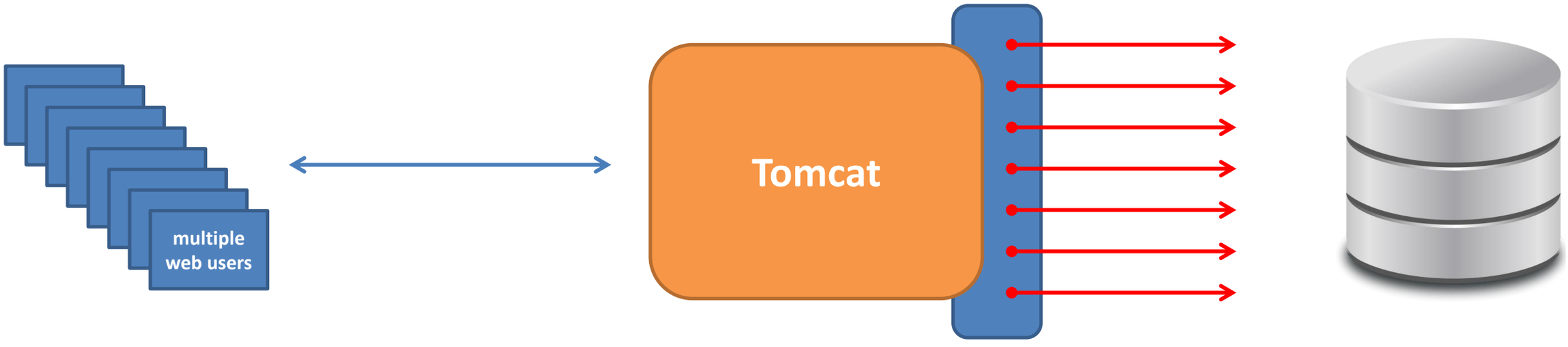

```
<Context>

  <Resource name="jdbc/web_student_tracker"
        auth="Container" type="javax.sql.DataSource"
                maxActive="20" maxIdle="5" maxWait="10000"
                username="webstudent" password="webstudent"
                driverClassName="com.mysql.jdbc.Driver"
                url="jdbc:mysql://localhost:3306/web_student_tracker"/>

</Context>
```



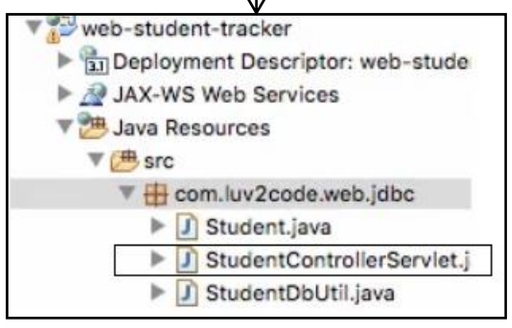
1. Download JDBC Driver JAR file
2. **Define connection pool in META-INF/context.xml**
3. Get connection pool reference in Java code

Student Tracker App

2.3 Setting up a Tomcat Database Connection Pool:



1. Download JDBC Driver JAR file
2. Define connection pool in META-INF/context.xml
3. **Get connection pool reference in Java code**
   - *3.1 create a new Servlet class*
   - *3.2 define datasource/connection pool for **Resource Injection***

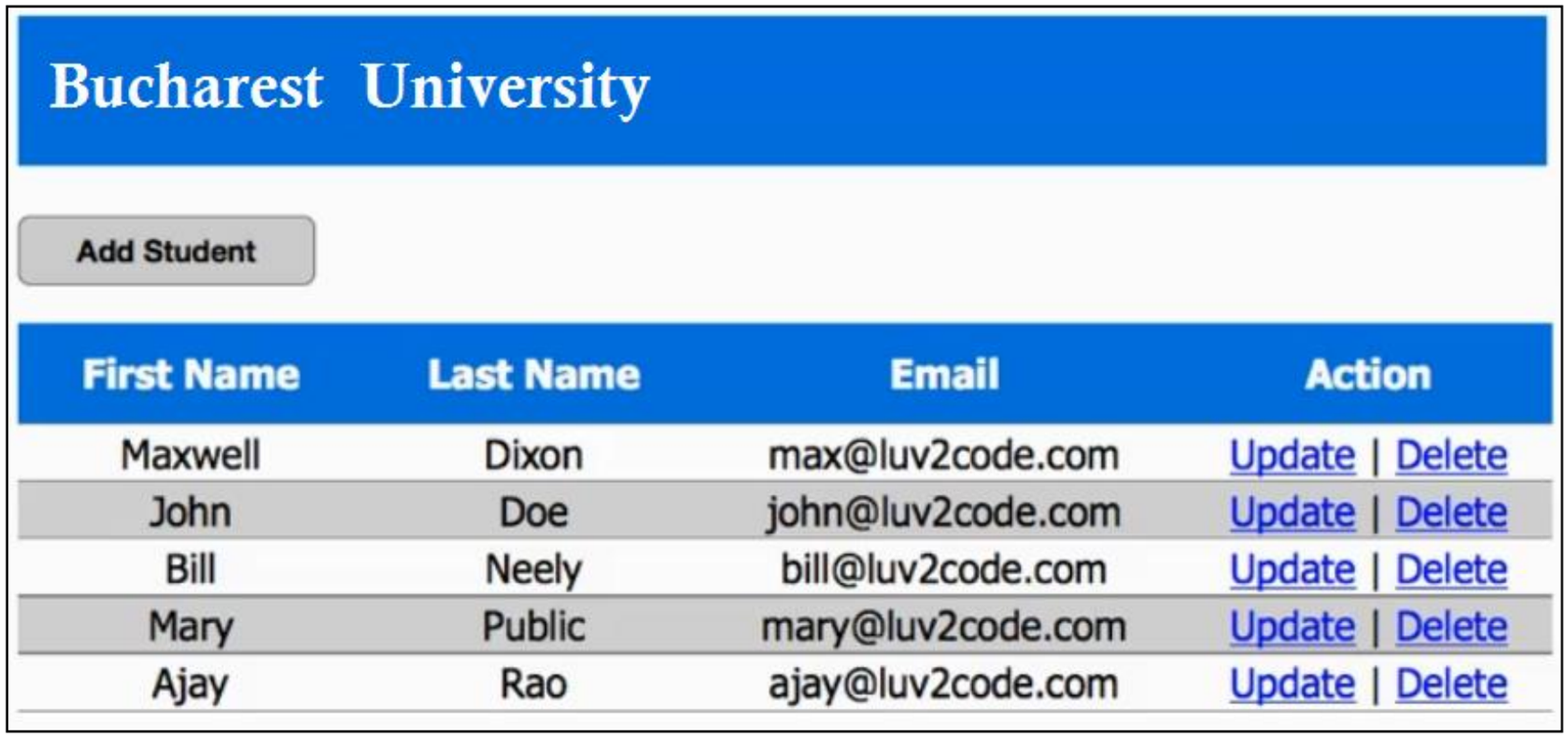



```
@WebServlet("/StudentControllerServlet")
public class StudentControllerServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    private StudentDbUtil studentDbUtil;

    @Resource(name="jdbc/web_student_tracker")
    private DataSource dataSource;
```

Student Tracker App

3 Adding Features

1. List Students
2. Add a new Student
3. Update a Student
4. Delete a Student



Student Tracker App

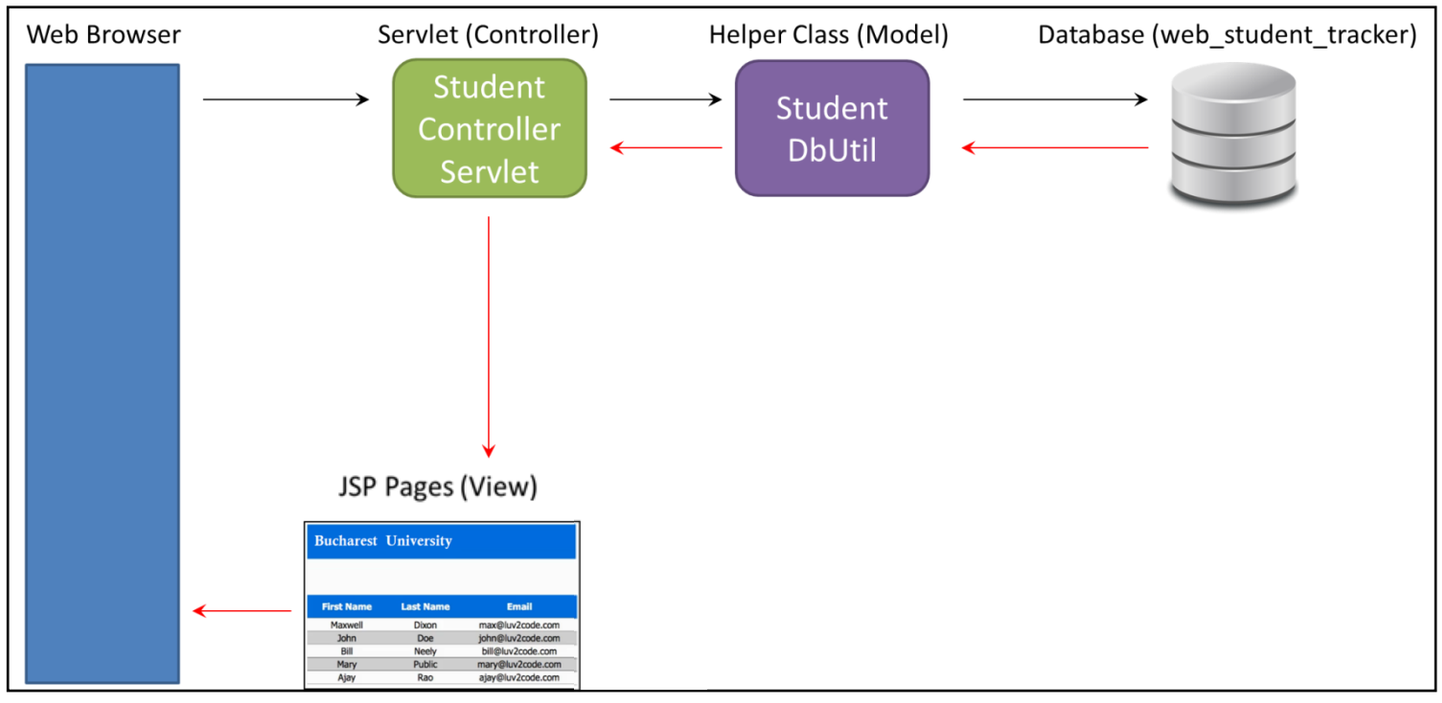
3.1 Adding Features: *List Students*

1. Create **Student.java**

2. Create **StudentDbUtil.java**

3. Create **StudentControllerServlet.java**

4. Create JSP page: **list-students.jsp**

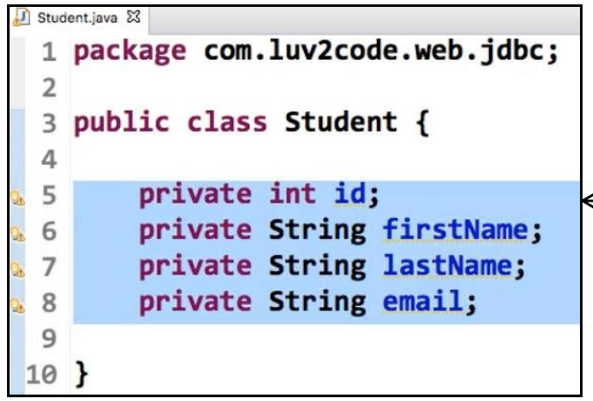


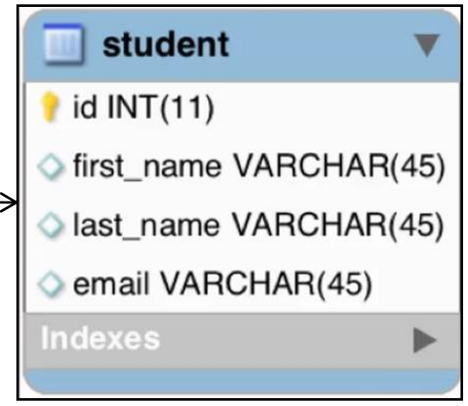
Student Tracker App

3.1.1 List Students: *Create Student.java*

1. **Create Student.java**
2. Create StudentDbUtil.java
3. Create StudentControllerServlet.java
4. Create JSP page: list-students.jsp

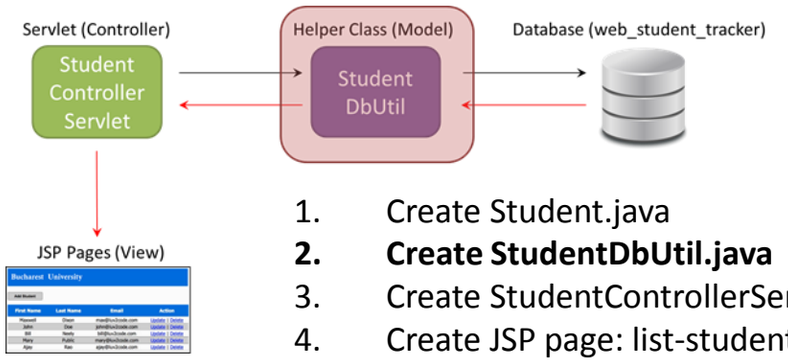
- Define fields *(as in 'student' table from DB)*
- Define constructors
- Define Getters & Setters
- Define toString() method



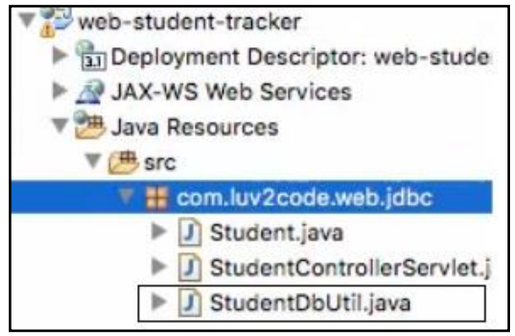


Student Tracker App

3.1.2 List Students: *Create StudentDbUtil.java (DAO)*



1. Create Student.java
2. **Create StudentDbUtil.java**
3. Create StudentControllerServlet.java
4. Create JSP page: list-students.jsp



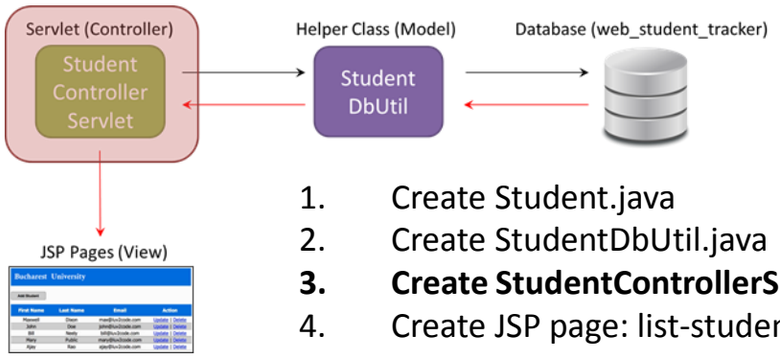
Define the **getStudents()** method:

- define JDBC objects *(Connection, Statement, ResultSet)*
- get connection
- create SQL statement
- execute query
- process result set
  (- retrieve data from result set row;
  - create new student object;
  - add it to the list of students)
- close JDBC objects

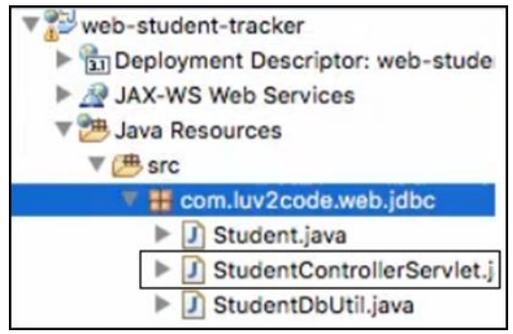
```
public List<Student> getStudents() throws Exception {
    List<Student> students = new ArrayList<>();

    Connection myConn = null;
    Statement myStmt = null;
    ResultSet myRs = null;

    try {
        myConn = dataSource.getConnection();

        String sql = "select * from student order by last_name";
        myStmt = myConn.createStatement();

        myRs = myStmt.executeQuery(sql);

        while (myRs.next()) {

            int id = myRs.getInt("id");
            String firstName = myRs.getString("first_name");
            String lastName = myRs.getString("last_name");
            String email = myRs.getString("email");

            Student tempStudent = new Student(id, firstName, lastName, email);

            students.add(tempStudent);
        }
        return students;
    }
    finally {
        close(myConn, myStmt, myRs);
    }
}
```

Student Tracker App

3.1.3 List Students: *Create StudentControllerServlet.java*



1. Create Student.java
2. Create StudentDbUtil.java
3. **Create StudentControllerServlet.java**
4. Create JSP page: list-students.jsp

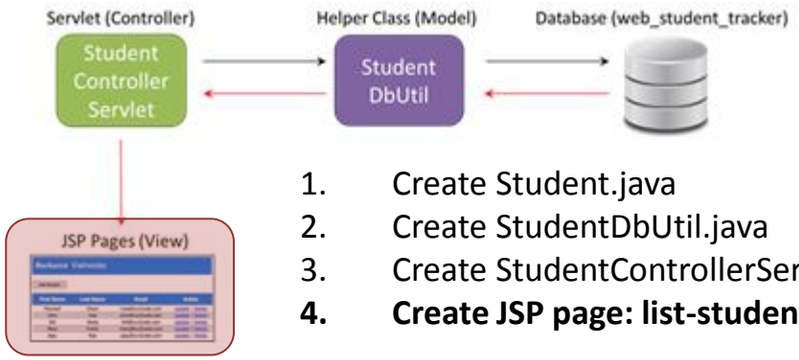


- Define a reference to StudentDbUtil *(helper class)*
- Inject DataSource
- Override **init()** method
  (create the StudentDbUtil object by passing the datasource)
- Implement **doGet()** method
- Define **listStudents()** method

```
@WebServlet("/StudentControllerServlet")
public class StudentControllerServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    private StudentDbUtil studentDbUtil;

    @Resource(name="jdbc/web_student_tracker")
    private DataSource dataSource;

    @Override
    public void init() throws ServletException {
        super.init();
        try {
            studentDbUtil = new StudentDbUtil(dataSource);
        }
        catch (Exception e ) {
            throw new ServletException(e );
        }
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
        try {
            // list the students ... in MVC fashion
            listStudents(request, response);
        }
        catch (Exception e) {
            throw new ServletException(e);
        }
    }

    private void listStudents(HttpServletRequest request, HttpServletResponse response)
        throws Exception {

        // get students from db util
        List<Student> students = studentDbUtil.getStudents();

        // add students to the request
        request.setAttribute("STUDENT_LIST", students);

        // send to JSP page (view)
        RequestDispatcher dispatcher = request.getRequestDispatcher("/list-students.jsp");
        dispatcher.forward(request, response);
    }
}
```

Student Tracker App

3.1.4 List Students: *Create list-students.jsp*



1. Create Student.java
2. Create StudentDbUtil.java
3. Create StudentControllerServlet.java
4. **Create JSP page: list-students.jsp**

- Get the students from the **request** object (sent by **servlet**)
- Set up HTML table
- Use attribute to dynamically populate the table
- CSS styling

```
<%
    // get the students from the request object (sent by servlet)
    List<Student> theStudents =
                    (List<Student>) request.getAttribute("STUDENT_LIST");
%>
```

```
<table>
    <tr>
        <th>First Name</th>
        <th>Last Name</th>
        <th>Email</th>
    </tr>
</table>
```

```
<% for (Student tempStudent : theStudents) { %>
    <tr>
        <td> <%= tempStudent.getFirstName() %> </td>
        <td> <%= tempStudent.getLastName() %> </td>
        <td> <%= tempStudent.getEmail() %> </td>
    </tr>
<% } %>
```

Bucharest University

First Name	Last Name	Email
Maxwell	Dixon	max@luv2code.com
John	Doe	john@luv2code.com
Bill	Neely	bill@luv2code.com
Mary	Public	mary@luv2code.com
Ajay	Rao	ajay@luv2code.com

```
<head>
    <title>Student Tracker App</title>
    <link type="text/css" rel="stylesheet" href="css/style.css">
</head>
```

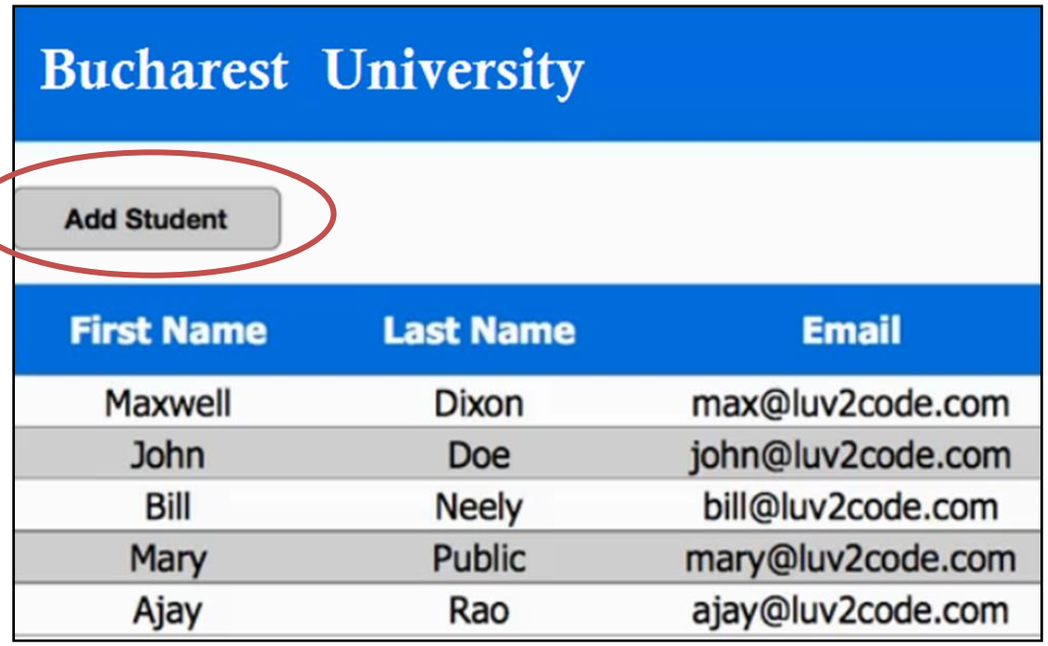
Bucharest University

First Name	Last Name	Email
Maxwell	Dixon	max@luv2code.com
John	Doe	john@luv2code.com
Bill	Neely	bill@luv2code.com
Mary	Public	mary@luv2code.com
Ajay	Rao	ajay@luv2code.com

Student Tracker App

3.2 Adding Features: *Add Students*

list-students.jsp



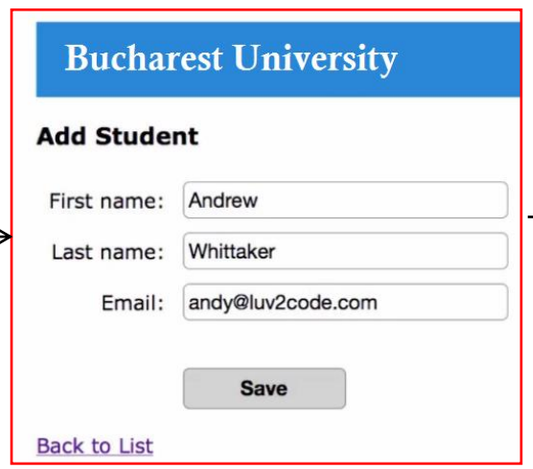
1. Update **list-students.jsp**
   *(new "**add** statement" button)*

2. Create **HTML form** for new student

3. Update **StudentControllerServlet**
   *(handle request to **add** student)*

4. Update StudentDbUtil
   *(add new JDBC method: **addStudent(…)***

add-student-form.jsp



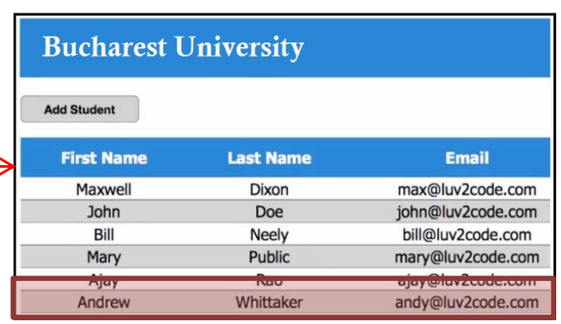
doGet(…)

command=add

Student Controller Servlet

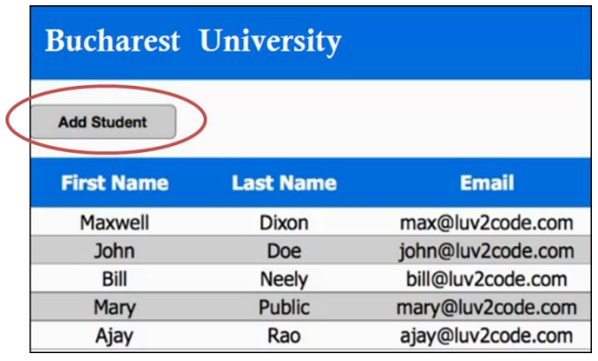
addStudent(…)

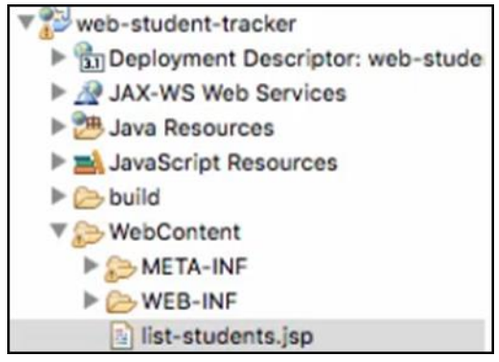
Student DbUtil



Student Tracker App

3.2.1 Add Students: *Update list-students.jsp*

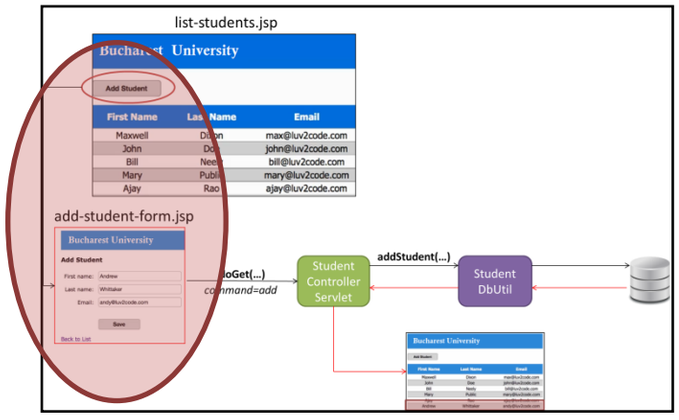


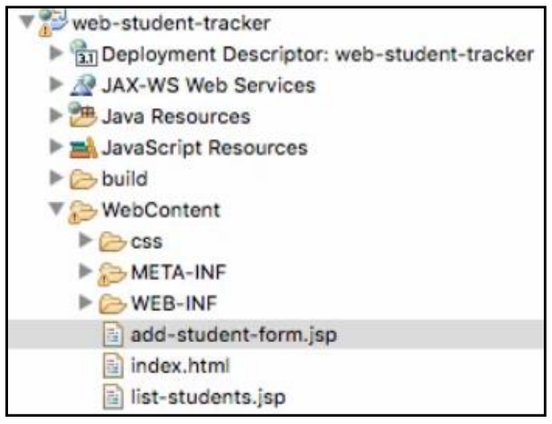



```
<div id="content">

    <!-- put new button: Add Student -->

    <input type="button" value="Add Student"
           onclick="window.location.href='add-student-form.jsp'; return false;"
           class="add-student-button"
    />
```

Student Tracker App

3.2.2 Add Students: *Create HTML form for new student*



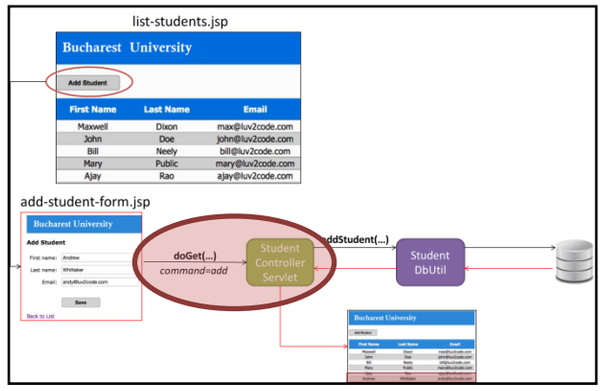


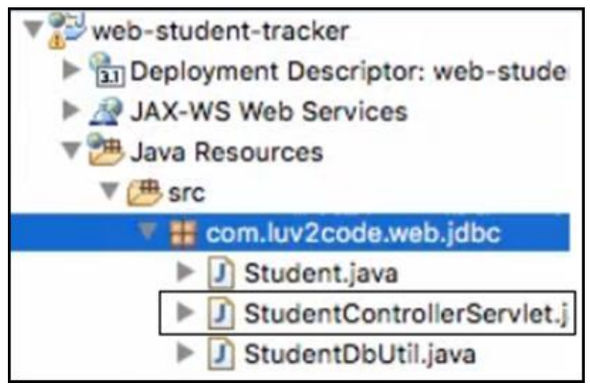
- set up “command” value to “**ADD**”
- define table fields
- add **save** option
- add link to “homepage”

```
<div id="container">
    <h3>Add Student</h3>
    <form action="StudentControllerServlet" method="GET">
        <input type="hidden" name="command" value="ADD" />
        <table>
            <tbody>
                <tr>
                    <td><label>First name:</label></td>
                    <td><input type="text" name="firstName" /></td>
                </tr>
                <tr>
                    <td><label>Last name:</label></td>
                    <td><input type="text" name="lastName" /></td>
                </tr>
                <tr>
                    <td><label>Email:</label></td>
                    <td><input type="text" name="email" /></td>
                </tr>
                <tr>
                    <td><label></label></td>
                    <td><input type="submit" value="Save" class="save" /></td>
                </tr>
            </tbody>
        </table>
    </form>
    <p>
        <a href="StudentControllerServlet">Back to List</a>
    </p>
</div>
```

Student Tracker App

3.2.3 Add Students: *Update StudentControllerServlet*





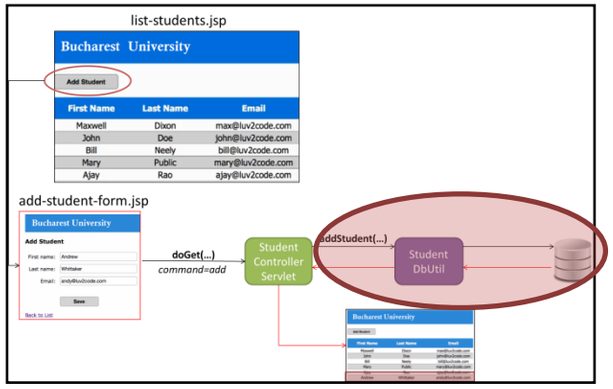
- get the “command” from the request object
- route to the appropriate method
- define addStudent(…) method

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) {
    try {
        String theCommand = request.getParameter("command");

        switch (theCommand) {

        case "LIST":
            listStudents(request, response);
            break;

        case "ADD":
            addStudent(request, response);
            break;

        default:
            listStudents(request, response);
        }
    }
    catch (Exception exc) {
        throw new ServletException(exc);
    }
}
```

```
private void addStudent(HttpServletRequest request, HttpServletResponse response) {

    // read student info from form data
    String firstName = request.getParameter("firstName");
    String lastName = request.getParameter("lastName");
    String email = request.getParameter("email");

    // create a new student object
    Student theStudent = new Student(firstName, lastName, email);

    // add the student to the database
    studentDbUtil.addStudent(theStudent);

    // send back to main page (the student list)
    listStudents(request, response);
}
```

Student Tracker App

3.2.4 Add Students: *Update StudentDbUtil*



Define addStudent(…) method:

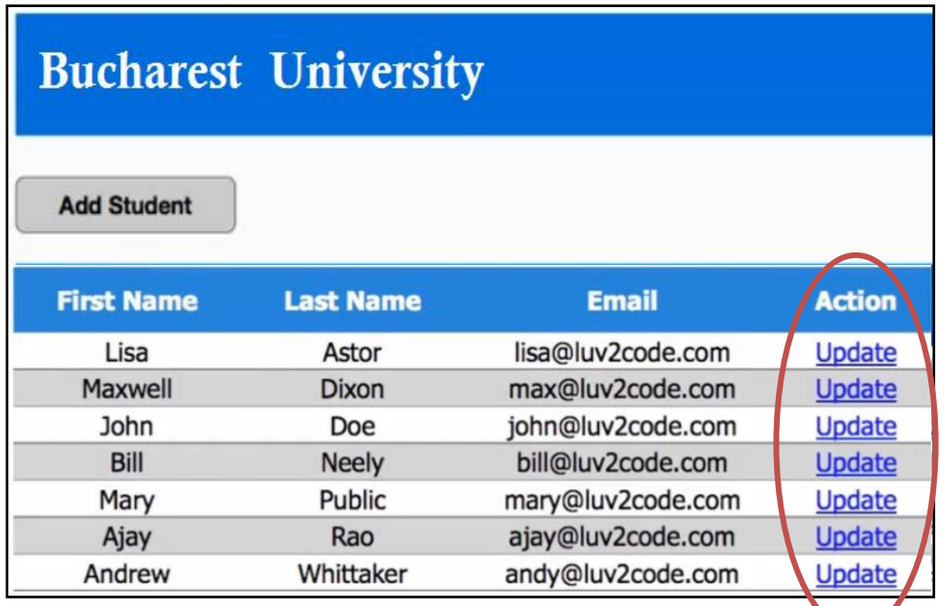
- define JDBC objects *(Connection, Statement, ResultSet)*
- get connection
- create SQL statement
- set param values for the student
- execute query
- close JDBC objects

```
public void addStudent(Student theStudent) throws Exception {

    Connection myConn = null;
    PreparedStatement myStmt = null;

    try {
        myConn = dataSource.getConnection();

        String sql = "insert into student "
                   + "(first_name, last_name, email) "
                   + "values (?, ?, ?)";

        myStmt = myConn.prepareStatement(sql);

        myStmt.setString(1, theStudent.getFirstName());
        myStmt.setString(2, theStudent.getLastName());
        myStmt.setString(3, theStudent.getEmail());

        myStmt.execute();
    }

    finally {
        close(myConn, myStmt, null);
    }
}
```

Student Tracker App

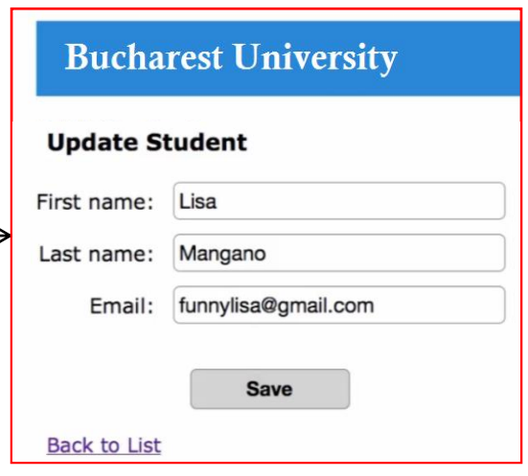
3.3 Adding Features: *Update Students*

list-students.jsp



1. Update list-students.jsp *(new "update" link)*
2. Update StudentControllerServlet *(to prepopulate the HTML form)*
3. Update StudentDbUtil *(add new JDBC method: getStudent(…))*
4. Create HTML form for updating a student
5. Update StudentControllerServlet *(handle request to update student)*
6. Update StudentDbUtil *(add new JDBC method: updateStudent(…))*

update-student-form.jsp



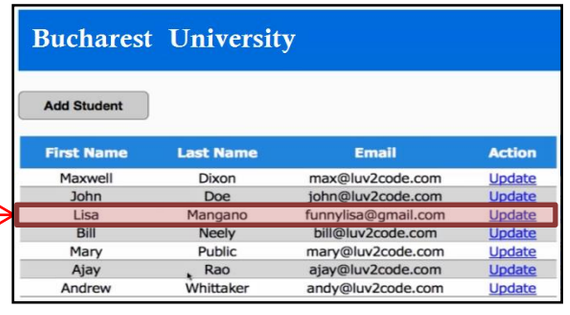
doGet(…)

command=update

Student Controller Servlet

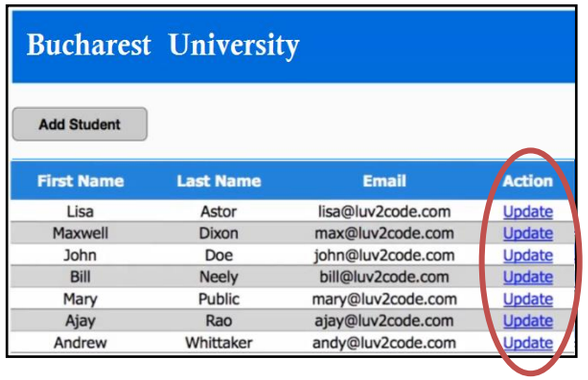
updateStudent(…)

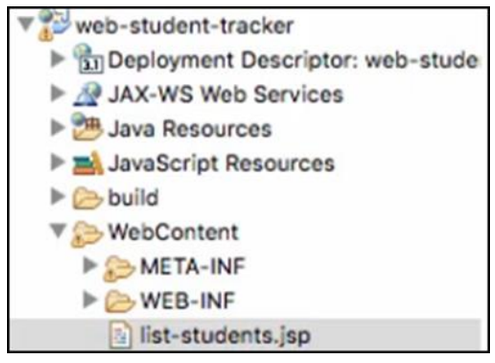
Student DbUtil



Student Tracker App

3.3.1 Update Students: *Update list-students.jsp*



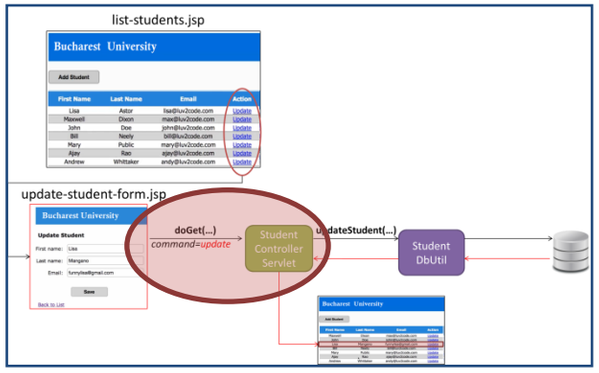


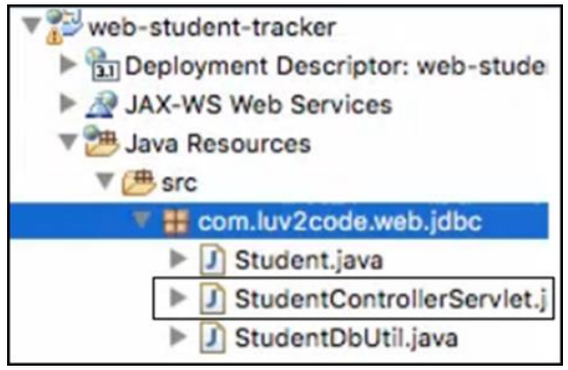
- **send params** *(command, studentId)* to the controller
- setup a link to the "**tempLink**" variable

```
<c:forEach var="tempStudent" items="${STUDENT_LIST}">

    <!-- set up a link for each student -->
    <c:url var="tempLink" value="StudentControllerServlet">
        <c:param name="command" value="LOAD" />
        <c:param name="studentId" value="${tempStudent.id}" />
    </c:url>

    <tr>
        <td> ${tempStudent.firstName} </td>
        <td> ${tempStudent.lastName} </td>
        <td> ${tempStudent.email} </td>
        <td> <a href="${tempLink}">Update</a> </td>
    </tr>

</c:forEach>
```

Student Tracker App

3.3.2 Update Students: *Update StudentControllerServlet (prepopulate the HTML form)*





- add support for the "**LOAD**" command in doGet(…)

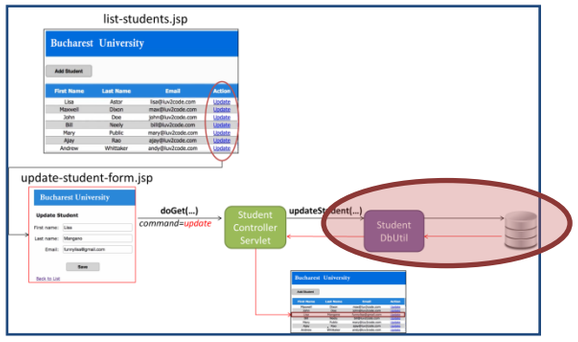
```
protected void doGet(HttpServletRequest request, HttpServletResponse response) {
    try {
        String theCommand = request.getParameter("command");

        switch (theCommand) {

        case "LIST":
            listStudents(request, response);
            break;

        case "ADD":
            addStudent(request, response);
            break;

        case "LOAD":
            loadStudent(request, response);
            break;

        default:
            listStudents(request, response);
        }
    }
    catch (Exception exc) {
        throw new ServletException(exc);
    }
}
```

- define **loadStudent(…)** method
  - read student id from data
  - get student from DB (db util)
  - place student in the request attribute
  - send to JSP page *(update-student-form.jsp)*

```
private void loadStudent(HttpServletRequest request, HttpServletResponse response) {
    String theStudentId = request.getParameter("studentId");

    Student theStudent = studentDbUtil.getStudent(theStudentId);

    request.setAttribute("THE_STUDENT", theStudent);

    RequestDispatcher dispatcher =
            request.getRequestDispatcher("/update-student-form.jsp");
    dispatcher.forward(request, response);
}
```

Student Tracker App

3.3.3 Update Students: *Update StudentDbUtil (add new JDBC method: getStudent(...))*



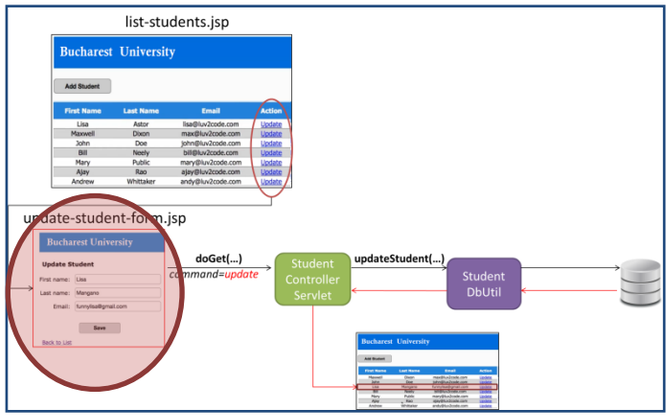
Define getStudent(...) method:

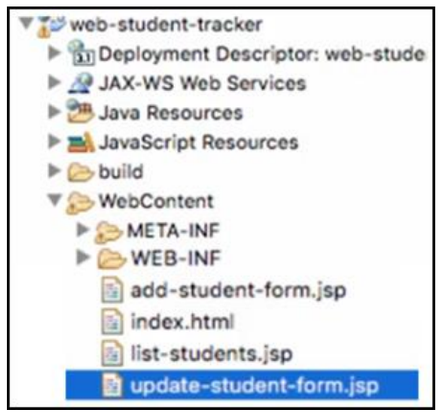
- define JDBC objects *(Connection, Statement, ResultSet)*
- convert student id to int
- get connection
- create prepared statement
- set param values for the student
- execute statement
- retrieve data from result set row
- close JDBC objects

```
public Student getStudent(String theStudentId) {

        Student theStudent = null;

        Connection myConn = null;
        PreparedStatement myStmt = null;
        ResultSet myRs = null;

        int studentId;

        try {
            studentId = Integer.parseInt(theStudentId);

            myConn = dataSource.getConnection();

            String sql = "select * from student where id=?";

            myStmt = myConn.prepareStatement(sql);

            myStmt.setInt(1, studentId);

            myRs = myStmt.executeQuery();

            if (myRs.next()) {
                String firstName = myRs.getString("first_name");
                String lastName = myRs.getString("last_name");
                String email = myRs.getString("email");

                theStudent = new Student(studentId, firstName, lastName, email);
            } else {
                throw new Exception("Could not find student id: " + studentId);
            }

            return theStudent;

        } finally {
            close(myConn, myStmt, myRs);
        }
}
```

Student Tracker App

3.3.4 Update Students: *Create HTML form for updating a student*



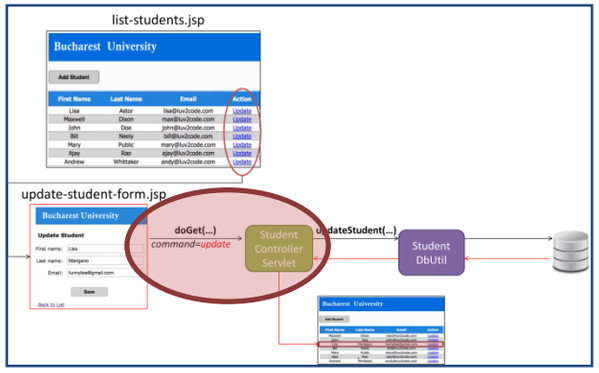


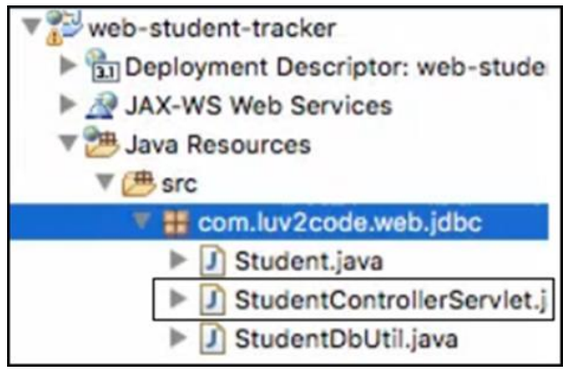
- set up “command” value to “**UPDATE**”
- set up “studentId”
- prepopulate fields
- add **save** option
- add link to “homepage”

```
<div id="container">
    <h3>Update Student</h3>
    <form action="StudentControllerServlet" method="GET">
        <input type="hidden" name="command" value="UPDATE" />
        <input type="hidden" name="studentId" value="${THE_STUDENT.id}" />
        <table>
            <tbody>
                <tr>
                    <td><label>First name:</label></td>
                    <td><input type="text" name="firstName"
                               value="${THE_STUDENT.firstName}" /></td>
                </tr>
                <tr>
                    <td><label>Last name:</label></td>
                    <td><input type="text" name="lastName"
                               value="${THE_STUDENT.lastName}" /></td>
                </tr>
                <tr>
                    <td><label>Email:</label></td>
                    <td><input type="text" name="email"
                               value="${THE_STUDENT.email}" /></td>
                </tr>
                <tr>
                    <td><label></label></td>
                    <td><input type="submit" value="Save" class="save" /></td>
                </tr>
            </tbody>
        </table>
    </form>
    <p>
        <a href="StudentControllerServlet">Back to List</a>
    </p>
</div>
```

Student Tracker App

3.3.5 Update Students: *Update StudentControllerServlet* *(handle request to update student)*





- add support for the "**UPDATE**" command in doGet(…)

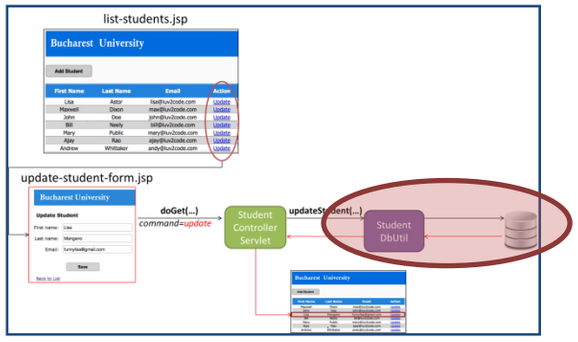
```
protected void doGet(HttpServletRequest request, HttpServletResponse response) {
    try {
        String theCommand = request.getParameter("command");
        switch (theCommand) {
        case "LIST":
            listStudents(request, response);
            break;
        case "ADD":
            addStudent(request, response);
            break;
        case "LOAD":
            loadStudent(request, response);
            break;
        case "UPDATE":
            updateStudent(request, response);
            break;
        default:
            listStudents(request, response);
        }
    }
    catch (Exception exc) {
        throw new ServletException(exc);
    }
}
```

- define **updateStudent(…)** method
  - read student info from data
  - create a new student object
  - perform update on DB
  - send them back to the "list students" page

```
private void updateStudent(HttpServletRequest request, HttpServletResponse response)
    throws Exception {

    int id = Integer.parseInt(request.getParameter("studentId"));
    String firstName = request.getParameter("firstName");
    String lastName = request.getParameter("lastName");
    String email = request.getParameter("email");

    Student theStudent = new Student(id, firstName, lastName, email);

    studentDbUtil.updateStudent(theStudent);

    listStudents(request, response);
}
```

Student Tracker App

3.3.6 Update Students: *Update StudentDbUtil (add new JDBC method: updateStudent(...))*



Define updateStudent(...) method:

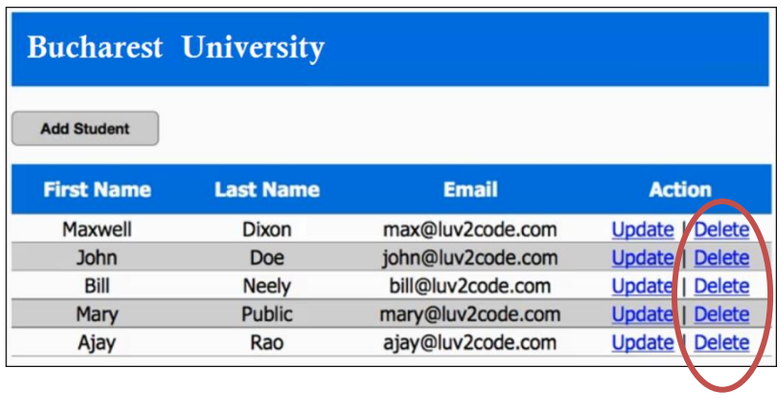
- define JDBC objects *(Connection, Statement, ResultSet)*
- get connection
- create prepared statement
- set params
- execute statement
- close JDBC objects

```
public void updateStudent(Student theStudent) throws Exception {

    Connection myConn = null;
    PreparedStatement myStmt = null;

    try {
        myConn = dataSource.getConnection();

        String sql = "update student "
                   + "set first_name=?, last_name=?, email=? "
                   + "where id=?";

        myStmt = myConn.prepareStatement(sql);

        myStmt.setString(1, theStudent.getFirstName());
        myStmt.setString(2, theStudent.getLastName());
        myStmt.setString(3, theStudent.getEmail());
        myStmt.setInt(4, theStudent.getId());

        myStmt.execute();

    } finally {
        close(myConn, myStmt, null);
    }
}
```

Student Tracker App

3.4 Adding Features: *Delete Students*

list-students.jsp



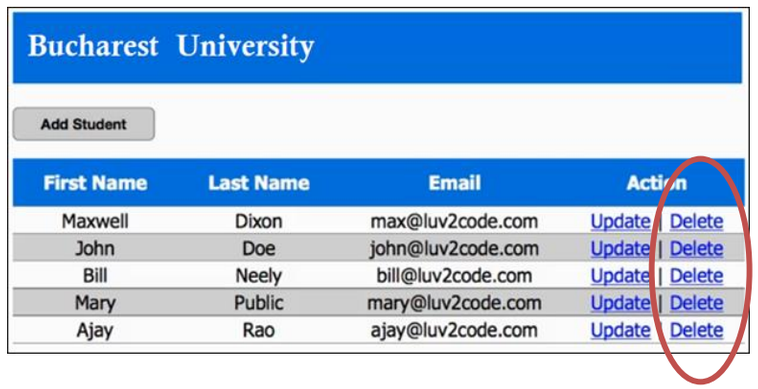
1. Add “Delete” link on JSP

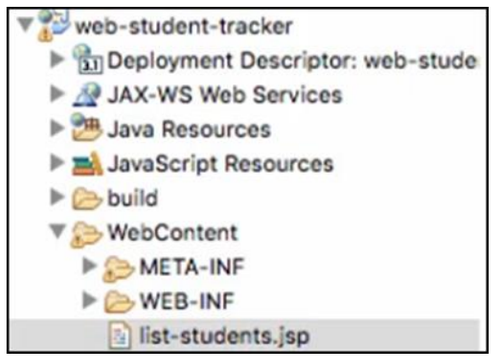
2. Add code for “Delete” to StudentControllerServlet

3. Add code for “Delete” to StudentDbUtil

Student Tracker App

3.4.1 Delete Students: *Update list-students.jsp*



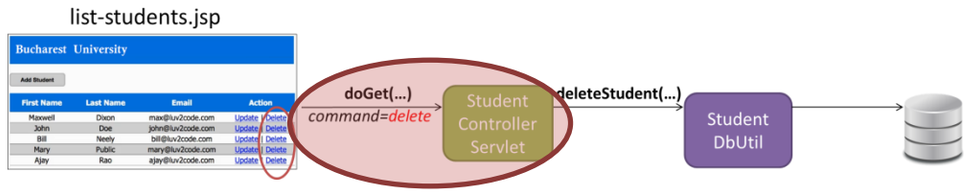


- **send params** *(command, studentId)* to the controller
- setup a link to the "**deleteLink**" variable

```
<c:forEach var="tempStudent" items="${STUDENT_LIST}">

    <!-- set up a link for each student -->
    <c:url var="tempLink" value="StudentControllerServlet">
        <c:param name="command" value="LOAD" />
        <c:param name="studentId" value="${tempStudent.id}" />
    </c:url>

    <!--  set up a link to delete a student -->
    <c:url var="deleteLink" value="StudentControllerServlet">
        <c:param name="command" value="DELETE" />
        <c:param name="studentId" value="${tempStudent.id}" />
    </c:url>

    <tr>
        <td> ${tempStudent.firstName} </td>
        <td> ${tempStudent.lastName} </td>
        <td> ${tempStudent.email} </td>
        <td>
            <a href="${tempLink}">Update</a>
            |
            <a href="${deleteLink}">Delete</a>
        </td>
    </tr>

</c:forEach>
```

Student Tracker App

3.4.2 Delete Students: *Update StudentControllerServlet*



- add support for the "**DELETE**" command in doGet(...)

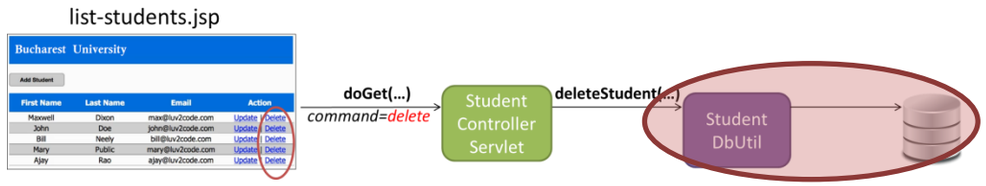
- define **deleteStudent(...)** method
  - read student id from form data
  - delete student from DB
  - send them back to "list students" page

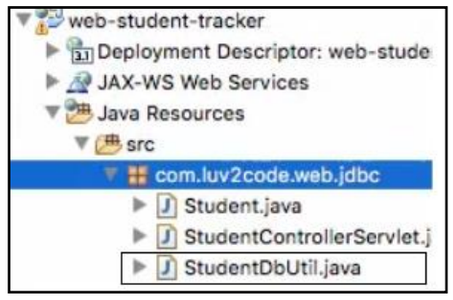
```
protected void doGet(HttpServletRequest request, HttpServletResponse response) {
    try {
        String theCommand = request.getParameter("command");

        switch (theCommand) {
          case "LIST":
              listStudents(request, response);
              break;

          case "ADD":
              addStudent(request, response);
              break;

          case "LOAD":
              loadStudent(request, response);
              break;

          case "UPDATE":
              updateStudent(request, response);
              break;

          case "DELETE":
              deleteStudent(request, response);
              break;

          default:
              listStudents(request, response);
          }

    } catch (Exception exc) {
          throw new ServletException(exc);
    }
}
```

```
private void deleteStudent(HttpServletRequest request, HttpServletResponse response) {

    String theStudentId = request.getParameter("studentId");

    studentDbUtil.deleteStudent(theStudentId);

    listStudents(request, response);
}
```

Student Tracker App

3.4.3 Delete Students: *Update StudentDbUtil*





Define deleteStudent(…) method:

- define JDBC objects *(Connection, Statement, ResultSet)*
- convert student id to int
- get connection to DB
- create prepared statement
- set params
- execute statement
- close JDBC objects

```
public void deleteStudent(String theStudentId) throws Exception {

    Connection myConn = null;
    PreparedStatement myStmt = null;

    try {
        int studentId = Integer.parseInt(theStudentId);

        myConn = dataSource.getConnection();

        String sql = "delete from student where id=?";

        myStmt = myConn.prepareStatement(sql);

        myStmt.setInt(1, studentId);

        myStmt.execute();
    } finally {
        // clean up JDBC code
        close(myConn, myStmt, null);
    }
}
```