CSE 510
Web Data Engineering

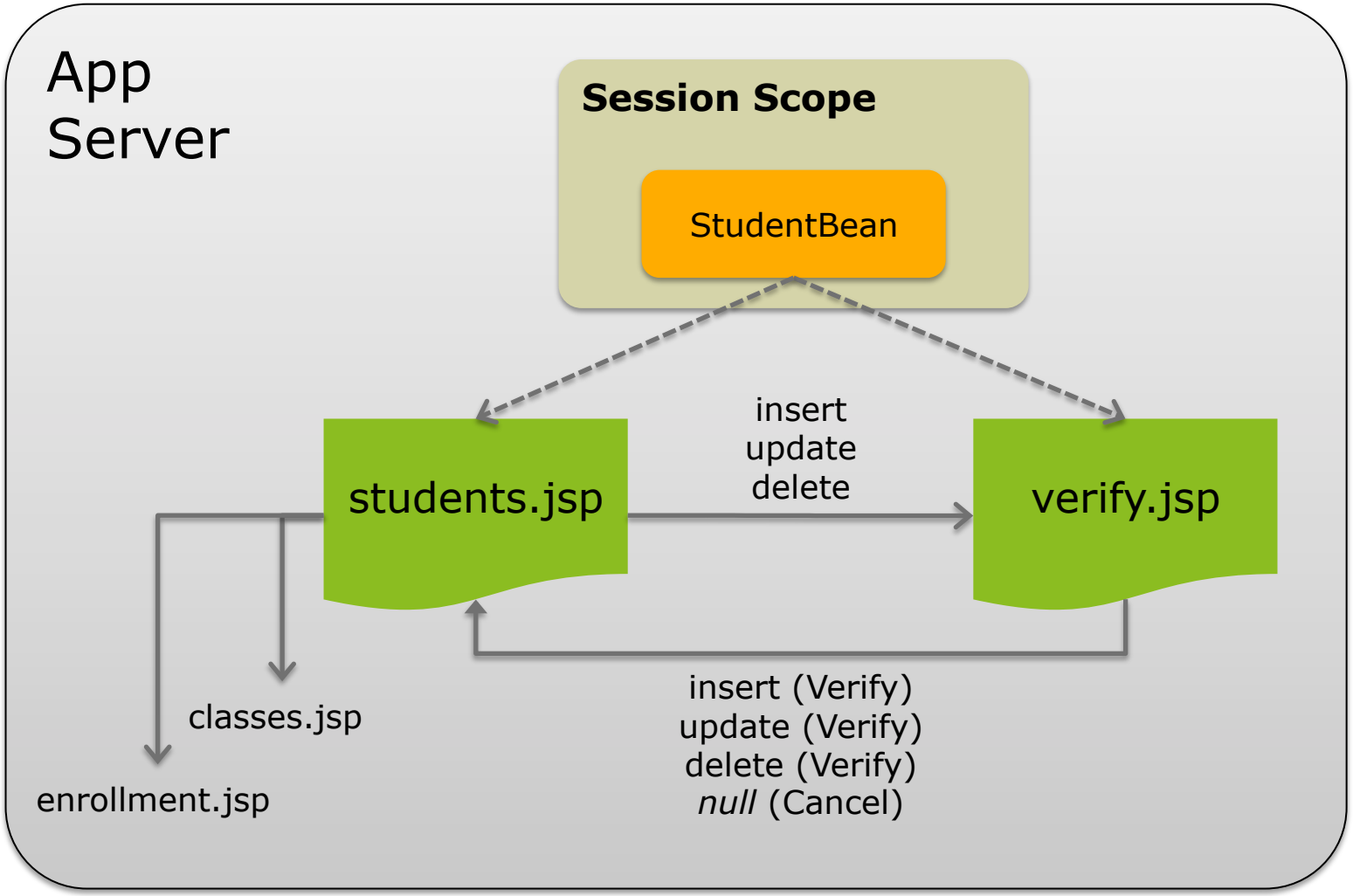
Data Access Object (DAO)
Java Design Pattern

cse@buffalo

Data Access Object (DAO) Java Design Pattern

- A Data Access Object (DAO) is a bean encapsulating database access code
- Completely separates DB access code from application logic and presentation code
- Improves code maintenance and portability
- Improves database server performance

Data Entry Example - 5th Attempt



Data Entry Example - 5th Attempt

StudentBean

```
public class StudentBean {
  private Integer id = null;
  private String first = null, middle = null, last = null;

  private String selectStr = "SELECT * FROM Students";
  private String insertStr = "INSERT INTO Students VALUES (?,?,?,?)";
  private String updateStr = "UPDATE Students SET firstName = ?, "
                + "middleName = ?, lastName = ? WHERE ubid = ? ";
  private String deleteStr = "DELETE FROM Students WHERE ubid = ?";

  public Integer getId() { return id; }
  public void setId(Integer id) { this.id = id; }
  ...
  public ResultSet getAllStudents() {...}
  public void insertStudent() {...}
  public void updateStudent() {...}
  public void deleteStudent() {...}
  public void close() { ... }
}
```

Data Entry Example - 5th Attempt

StudentBean

```
public ResultSet getAllStudents() {
  conn = DBConnectionPool.getConnection();
  pStmt = conn.prepareStatement(selectStr);
  allStudents = pStmt.executeQuery();
  return allStudents;
}
public void insertStudent() {
  conn = DBConnectionPool.getConnection();
  pStmt = conn.prepareStatement(insertStr);
  pStmt.setInt(1, id);
  ...
  pStmt.executeUpdate();
  conn.commit();
  close();
}
```

Data Entry Example - 5th Attempt

DBConnectionPool

```
public class DBConnectionPool {
  private static Context ctx = null;
  private static DataSource ds = null;

  public static Connection getConnection()
    throws NamingException, SQLException {
    if (ctx == null) {
      ctx = new InitialContext();
      ds = (DataSource)
           ctx.lookup("java:comp/env/jdbc/ClassesDBPool");
    }
    return ds.getConnection();
  }
}
```

Data Entry Example - 5th Attempt

students.jsp Code

```
<html><body><table><tr>
        <td><jsp:include page="menu.html"/></td>
        <td>
                ~~<Open Connection Code>~~
                ~~<Insertion Code>~~
                ~~<Update Code>~~
                ~~<Delete Code>~~
                ~~<Statement Code>~~
                <Application Logic Code>
                <Presentation Code>
                ~~<Close Connection Code>~~
        </td>
</tr></table></body></html>
```

Data Entry Example - 5th Attempt

students.jsp Code

```
<%@ page import="dataentry.*, java.sql.*"%>
<jsp:useBean id="student" scope="session"
   class="dataentry.StudentBean"/>
<%-- -------- Application Logic Code -------- --%>
<% String action = request.getParameter("action");
   if (action != null && action.equals("insert"))
     student.insertStudent();
   else if (action != null && action.equals("update"))
     student.updateStudent();
   else if (action != null && action.equals("delete"))
     student.deleteStudent();
   ResultSet rs = student.getAllStudents(); %>
<html>
...
```

Data Entry Example - 5th Attempt

verify.jsp Code

```
<jsp:useBean id="student" scope="session"
   class="dataentry.StudentBean"/>
<% student.clear(); %>
<jsp:setProperty name="student" property="*"/>
<html>
...
<%-- -------- Display Bean Properties -------- --%>
<tr>
 <td><jsp:getProperty name="student" property="id"/></td>
 <td><jsp:getProperty name="student" property="first"/></td>
 <td><jsp:getProperty name="student" property="middle"/></td>
 <td><jsp:getProperty name="student" property="last"/></td>
 ...
```

Data Entry Example - 5th Attempt

verify.jsp Code (Cont'd)

```
...
  <%-- Verify Button --%>
  <form action="students.jsp" method="GET">
    <input type="hidden" name="action"
           value="<%= request.getParameter("action") %>"/>
    <input type="submit" value="Verify"/>
  </form>
  <%-- Cancel Button --%>
  <form action="students.jsp" method="GET">
    <input type="submit" value="Cancel"/>
  </form>
</tr>
...
</html>
```

Data Entry Example - 5th Attempt

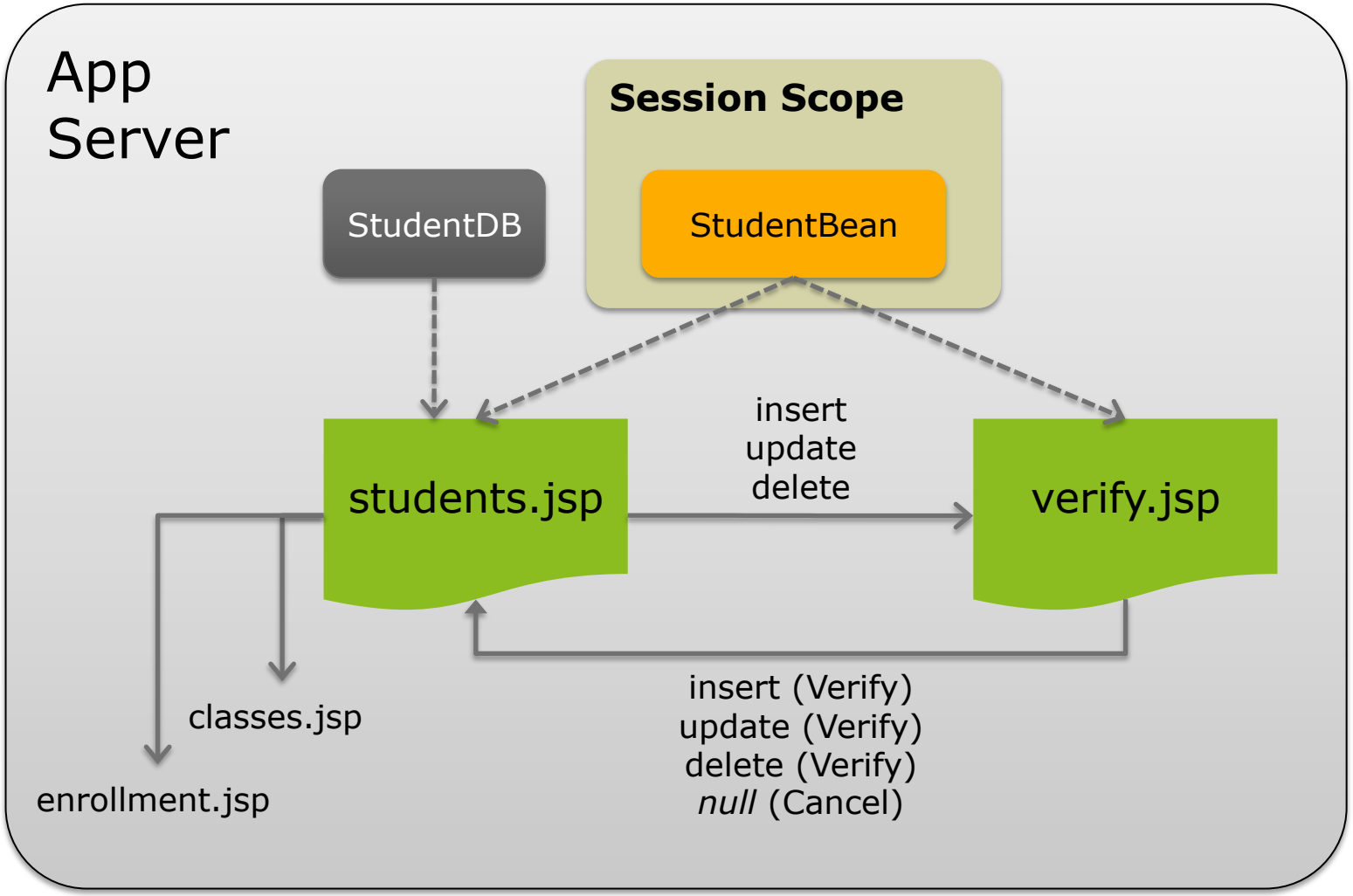
Advantages over 4th Attempt

- Moved SQL statements and connection code from students.jsp to StudentBean
  - Improves code maintenance and portability
- StudentBean makes it easy to process request parameters

Disadvantages

- StudentBean mixes DB access and session data
- Still need to manually close connection in students.jsp
- Application logic code still in students.jsp
  - An MVC framework is needed

Data Entry Example - 6th Attempt



Data Entry Example - 6th Attempt

StudentBean

```
public class StudentBean {
  private Integer id = null;
  private String first = null;
  private String middle = null;
  private String last = null;

  public Integer getId() { return id; }
  public void setId(Integer id) { this.id = id; }
  ...
  public void clear() {
    id = null;
    first = null;
    middle = null;
    last = null;
  }
}
```

Data Entry Example - 6th Attempt

StudentDB

```
public class StudentDB {
  private static String selectStr = ...;
  private static String insertStr = ...;
  private static String updateStr = ...;
  private static String deleteStr = ...;

  public static CachedRowSet getAllStudents() {...}
  public static void insertStudent(StudentBean student) {...}
  public static void updateStudent(StudentBean student) {...}
  public static void deleteStudent(StudentBean student) {...}
}
```

Data Entry Example - 6th Attempt

StudentDB

```
public static void insertStudent(StudentBean student)
  throws SQLException, NamingException {

  Connection conn = DBConnectionPool.getConnection();
  PreparedStatement pStmt = conn.prepareStatement(insertStr);

  pStmt.setInt(1, student.getId());
  pStmt.setString(2, student.getFirst());
  ...
  pStmt.executeUpdate();
  conn.commit();
  pStmt.close();
  conn.close();
}
```

Data Entry Example - 6th Attempt

StudentDB

```
public static CachedRowSet getAllStudents()
  throws SQLException, NamingException {
  Connection conn = DBConnectionPool.getConnection();
  PreparedStatement pStmt = conn.prepareStatement(selectStr);
  ResultSet allStudents = pStmt.executeQuery();

  CachedRowSet crsStudents = new CachedRowSetImpl();
  crsStudents.populate(allStudents);

  allStudents.close();
  pStmt.close();
  conn.close();
  return crsStudents;
}
```

Data Entry Example - 6th Attempt

students.jsp Code

```
<%@ page import="dataentry.beans.*, dataentry.db.*,
   javax.sql.rowset.*"%>
<jsp:useBean id="student" scope="session"
   class="dataentry.beans.StudentBean"/>
<%-- -------- Application Logic Code -------- --%>
<% String action = request.getParameter("action");
   if (action != null && action.equals("insert"))
     StudentDB.insertStudent(student);
   else if (action != null && action.equals("update"))
     StudentDB.updateStudent(student);
   else if (action != null && action.equals("delete"))
     StudentDB.deleteStudent(student);

   CachedRowSet crsStudents = StudentDB.getAllStudents(); %>
...
```

Data Entry Example - 6th Attempt

Advantages over 5th Attempt

- StudentBean stores session data only
- StudentDB executes DB access code only
- No need to manually close any connection within students.jsp

Disadvantages

- Application logic code still in students.jsp
  - An MVC framework is needed