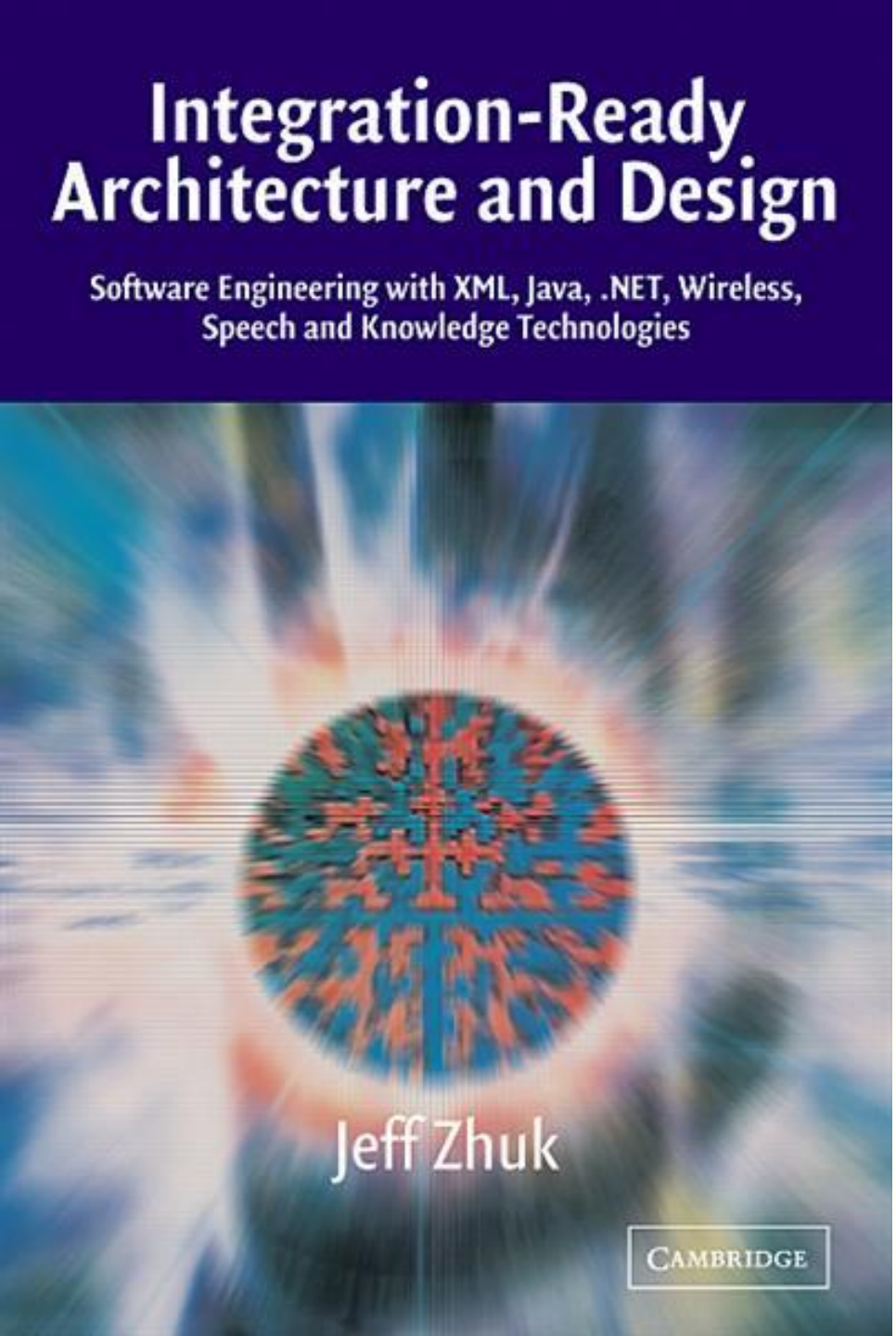


Data Service Patterns

Jeff Zhuk

From the book and beyond

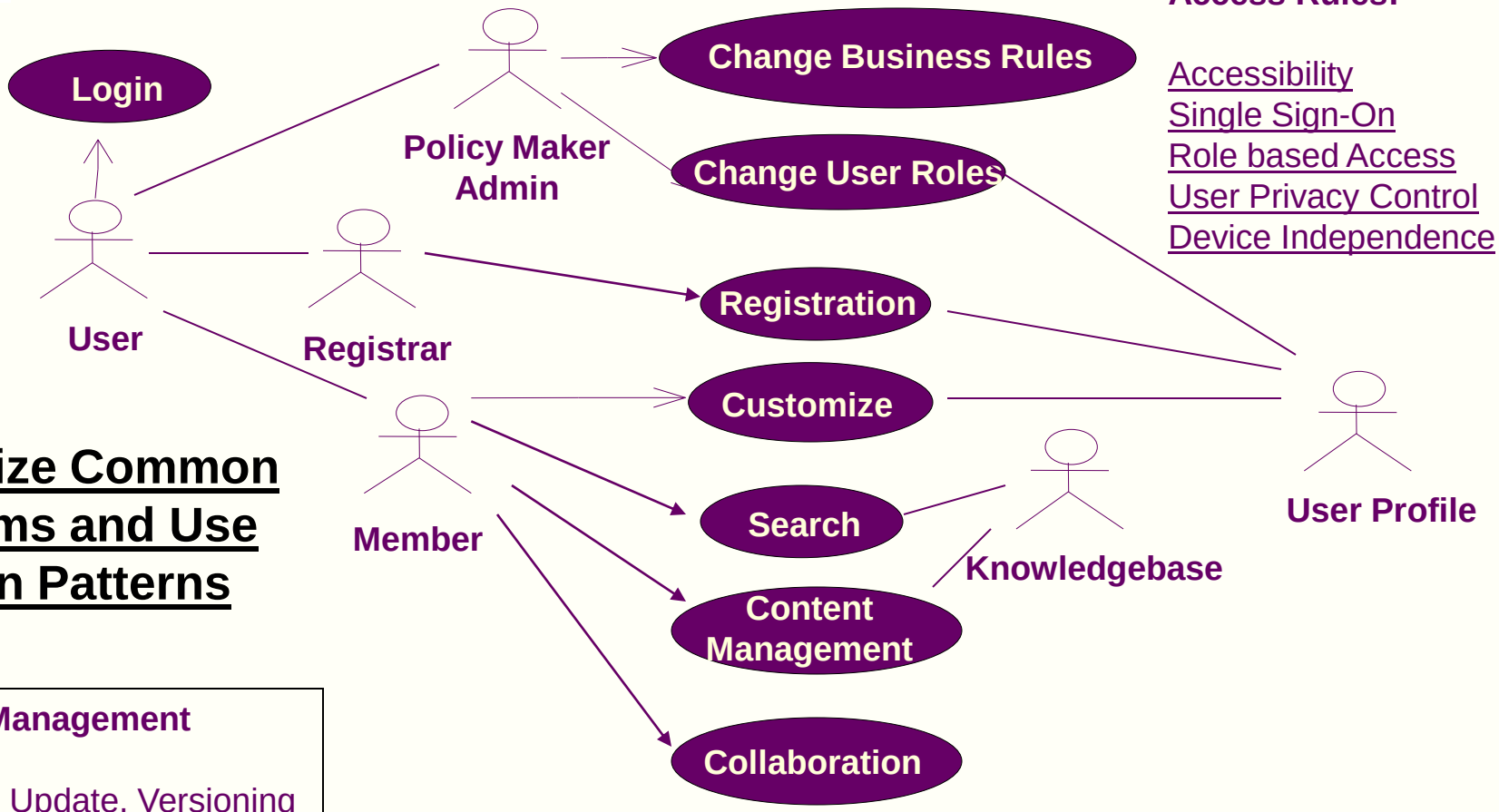
"Integration-Ready Architecture and Design"

Cambridge University Press

JavaSchool.com

Software Engineering With XML, Java, .NET, Wireless, Speech and Knowledge Technologies

Common Use Cases



Access Rules:

Accessibility
Single Sign-On
Role based Access
User Privacy Control
Device Independence

Recognize Common Problems and Use Design Patterns

Content Management

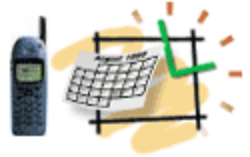
Authoring, Update, Versioning
Forms, Permits, Applications
Scheduling events/facilities
Workflow Routing
Planning and Approval
Document/Photo Imaging
Task Tracking and Reporting
Data and Service Evaluation

Collaboration

Email
Conferencing
Instant Messaging
Privilege-based Data/Service Sharing

Search

Search for Data and Services
GIS (Maps and Routes)
Linking Related Cases
Content-based Subscription



Cross Application Reuse Areas

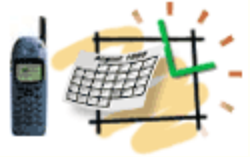
Initialization

Data Access

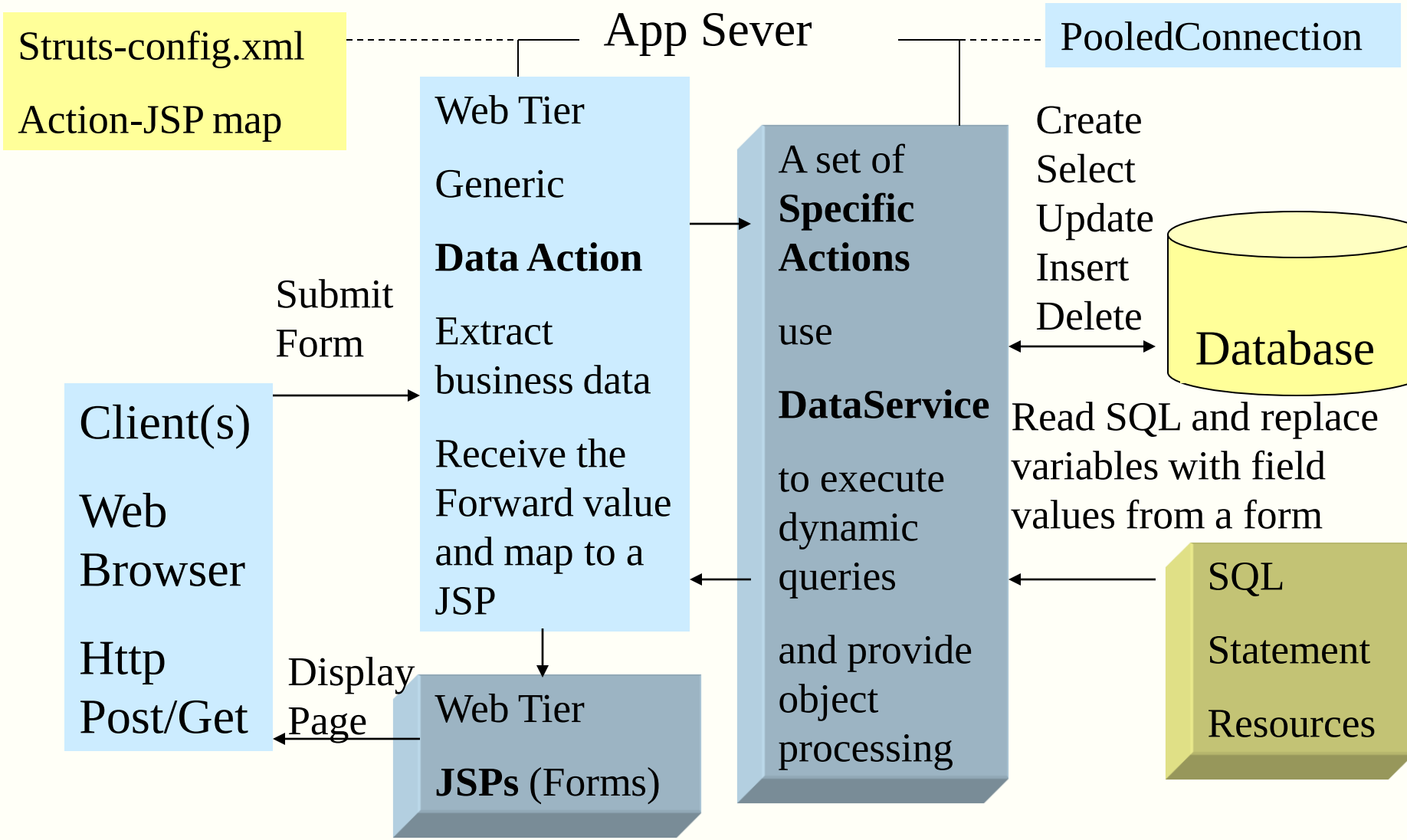
Security

More…

See http://javaschool.com/school/public/JZ.Hints.htm for design and coding hints

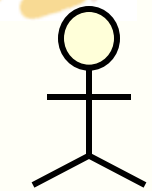


Data Intensive Web Application



The DataAction alone with the DataService implement common functions and provide generic application behavior.

Sequence Diagram



Client

DataAction | **Specific Service Action** | **DataService**

Events/Operations

Http Post/Get

Extract data from the page and call a proper service

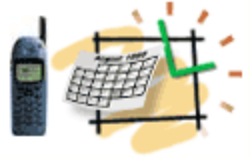
Request SQL execution with SQL statement name and map of parameters

Generic data processing: Retrieve SQL from resources; Replace run-time variables; Execute SQL, process ResultSet (if any) and return a list of object-beans

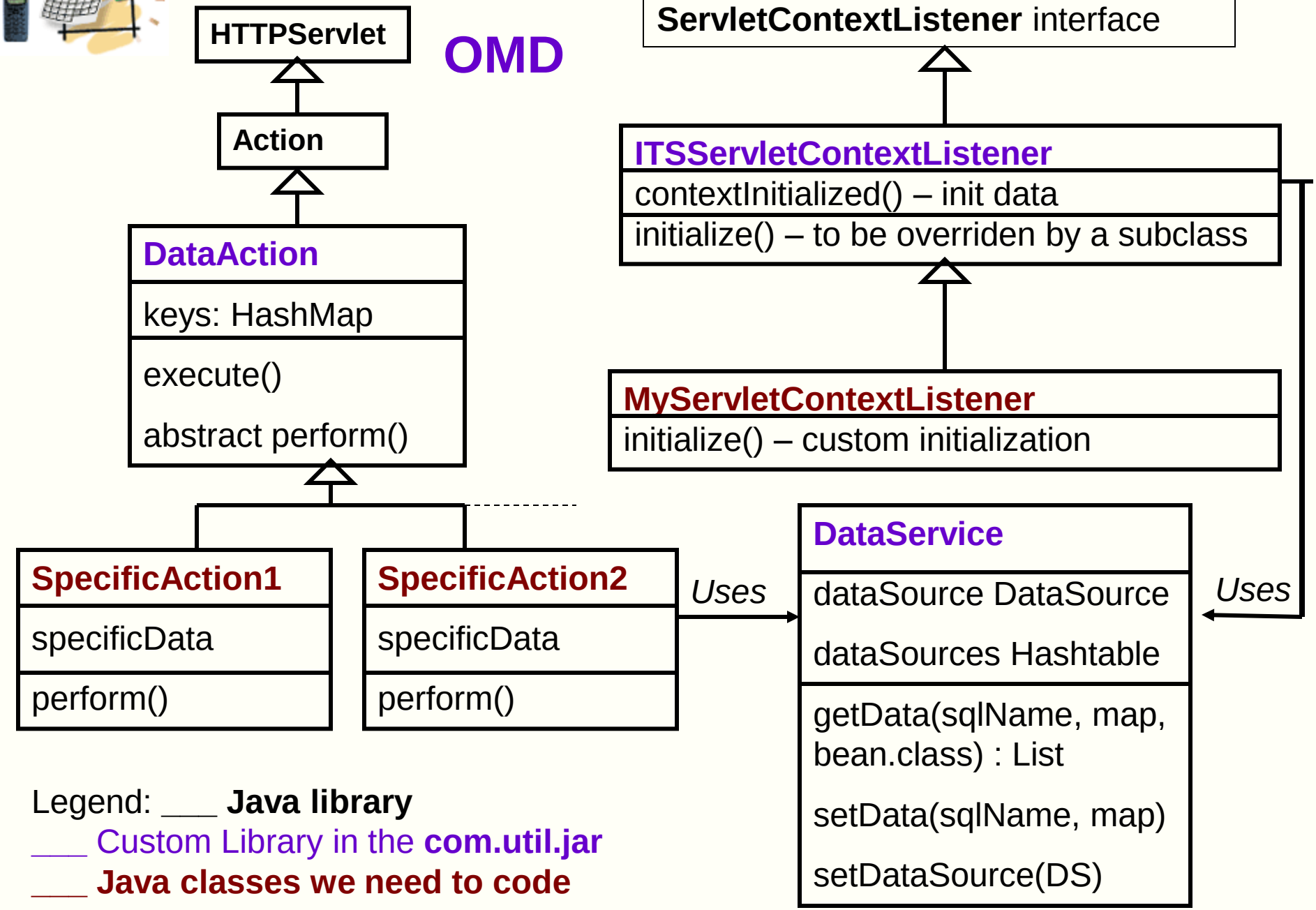
Specific processing; Work with objects and prepare data for JSP

Pass data saved in data beans, Session, or Request objects to JSP

Map the Forward value to a proper JSP (via Struts-config.xml) and display the next JSP page



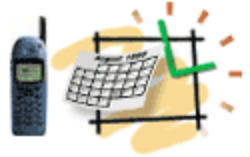
OMD



Legend: ___ **Java library**

___ Custom Library in the **com.util.jar**

___ **Java classes we need to code**

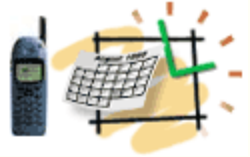


Generic DataAction Code

```
public class DataAction extends org.apache.struts.action.Action {
    private static Logger log = Logger.getLogger("DataAction");
    protected HashMap keys;
    public ActionForward execute(ActionMapping map, ….) {
        // generic code in the execute() method
        String forwardTo = null;
        keys = prepareKeys(request); // capture form and other key data
        try {    // perform specific action implemented in a subclass
            forwardTo = perform(request);
        } catch(Exception e) {
            log.severe(getClass().getName() + " e="+e);
            // prepare message for a generic error page
            request.setAttribute("msg", e.getMessage());
            throw e;
        }
        forward = map.findForward(forwardTo); // ActionMapping map
        return (forward);
    }
    public String perform(HttpServletRequest request) throws Exception {
        return null;
    }
}
```

Specific Action Implementation Example

```
public class LoginAction extends DataAction {

    public String perform(HttpServletRequest request) throws Exception {
        ……
        List beans = DataService.getData(
            "getLogin", // name of the SQL file is “getLogin.sql”
            keys,       // HashMap of key-values collected by DataAction
            LoginBean.class); // class that matches expected record structure

        if(beans.size() == 1) { // SUCCESS!
            //Create instance of SessionBean and bind it to session
            SessionBean sessionBean = new SessionBean();
            LoginBean bean = (LoginBean) beans.get(0);
            sessionBean.setUserName(bean.getLoginName());
            session.setAttribute(“sessionBean”, sessionBean);
            return “success”;
        }
        …….
        return “failure”;
    }
```

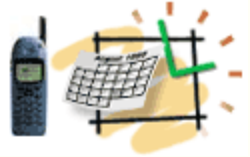


LoginBean Class Example

```
package beans;

import java.util.HashMap;
import java.util.logging.Logger;
/**
 * The LoginBean class matches the record selected by the getLogin.sql
 */

public class LoginBean
{
    private String loginName;
    private String password;

    // add getter and setter methods for data members above !!!
    public String getLoginName() {
        return loginName;
   }
   // TODO more getter and setter methods
}
```

Initialization with web.xml

```
public class ITSServletContextListener implements ServletContextListener {
   // This method is called at application start time
   public void contextInitialized(ServletContextEvent sce) {
      ServletContext servletContext = sce.getServletContext();
      String sqlLocation = servletContext.getInitParameter(“sqlLocation");
      String jndiName = "java:"+servletContext.getInitParameter("DataSource");
      …… InitialContext context = new InitialContext();
      …… DataSource dataSource = (DataSource)context.lookup(jndiName);

      // get other DataSource names, like ITSDataSource
      // JBoss deploy directory will include your oracle-ds.xml or similar file
      // that associate this name with a connection string and a driver

      com.its.util.DataService.setSqlLocation(sqlLocation);

      com.its.util.DataService.setDataSource(dataSource);

      // The initialize() method is overriden in the MyServletContextListener

      public void initialize(ServletContextEvent sce) { } // app-specific init

      public void contextDestroyed(ServletContextEvent sce) { }

}
```



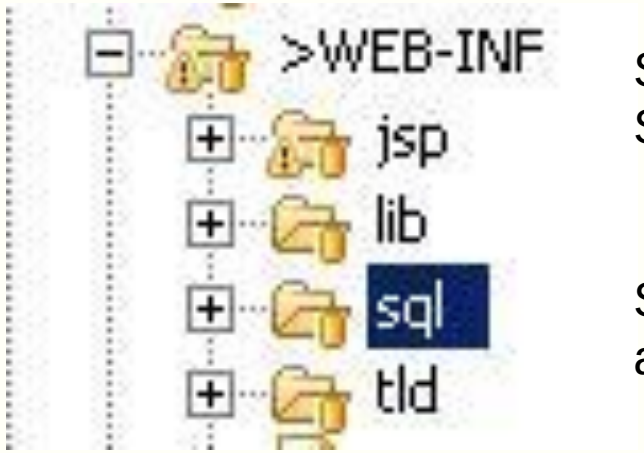
Provide DataSource Names and SQL Location in the WEB.XML file

---------- Extract from WEB.XML ----------

```
<listener>
  <listener-class>your.package.MyServletContextListener</listener-class>
 </listener>
 <context-param>
  <param-name>sqlLocation</param-name>
  <param-value>sql</param-value>
</context-param>
<context-param>
   <param-name>DataSource</param-name>
   <param-value>SpecificDataSourceNameProvidedInJBoss_ds_file
   </param-value>
 </context-param>
 <context-param>
   <param-name>AnotherDataSource</param-name>
   <param-value>AnotherSpecificDataSourceNameProvidedInJBoss_ds_file
   </param-value>
 </context-param>
```

Recommendations: Use “DataSource” for most frequently used data
Use “AnotherDataSource” names like “MyAppDataSource” for other data

SQL Statements



SQL statements are stored as separate files in the SQL directory in the WEB-INF area

Samples below demonstrate how SQL statements and their usage by DataService methods

// List beans = DataService.getData("getLogin", keyMap, LoginBean.class);
Two lines below is stored in the “getLogin.sql” file

```
select username, password from LoginTable
where username = ':loginName' and password = ':password‘
```

// int nRecords = DataService.setData(“insertUser”, map);
The line below is stored in the “insertUser.sql” file

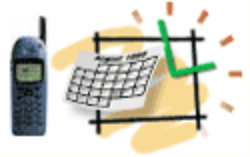
```
Insert into LoginTable values(':loginName‘, ':password‘)
```

Note that run-time variable names follow the “**:**" character

DataService API

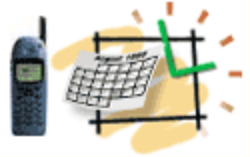
Include the library "com.its.util.jar" in the CLASSPATH and import com.its.util.DataService

```
// execute insert/delete/update SQL statements stored in the "sqlLocation"
@ param sqlStatementName for example "getLogin" stored as the "getLogin.sql"
@ param map of key-values to replace SQL <<keys>> with run-time values
@ return numberOfRowsEffected
public static int setData(String sqlStatementName, HashMap map)
// use other than "DataSource" connection pool
public static int setData(String sqlStatementName, HashMap map, String dsName)

// execute select statement and return a list of record-beans
@ param sqlStatementName for example "getLogin" stored as the "getLogin.sql"
@ param map of key-values to replace SQL <<keys>> with run-time values
@ param beanClass (e.g. LoginBean.class) supports records retrieved by the SQL statement
@ return list of objects of the beanClass
public static List getData(String sqlStatementName, HashMap map, Class beanClass)
// use other than "DataSource" connection pool
public static List getData(String sqlName, HashMap map, Class beanClass, String dsName)

@param dataSource your DataSource specified in JNDI context
@param dataSourceName of your DataSource specified as JNDI name
public static void setDataSource(DataSource dataSource, String dsName)
public static void setDataSource(DataSource dataSource)
```



More DataService API

```
// execute insert/delete/update SQL statements
@ param sqlStatement
@ return numberOfRowsEffected
public static int setData(String sqlStatement)
// use other than "DataSource" connection pool
public static int setData(String sqlStatement, String dsName)

// execute select statement and return a list of record-beans
@ param sqlStatement
@ param beanClass supports records retrieved by the SQL statement
@ return list of objects of the beanClass
public static List getData(String sqlStatement, Class beanClass)
// use other than "DataSource" connection pool
public static List getData(String sqlStatement, Class beanClass, String dsName)

// If application creates data from scratch – no SQL is needed
boolean createTable(String tableName, Class class)
Example:
DataService.createTable("LoginTable", LoginBean.class);

 int insert(String tableName, Object[] objects)

Example:
LoginBean[] logins; // array of beans populated in an action
int nRows = DataService.insert("LoginBean", logins);
```

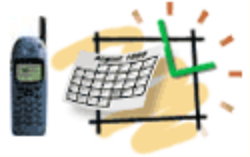


IOMaster API

- **Include the library "com.its.jar" in the CLASSPATH**

- **Import com.its.util.IOMaster;**

```
// start server daemon and assign service processing point
public static void startServerDeamon(int iPort,
   String serviceName, String methodName)

// request service from a client and receive service response
public static String clientSocketRequest(
        String iHost, int iPort, String iMessage)

// write a file with array of bytes
public static boolean writeBinFile(String iFilename, byte[] iData)

// read a file
public static byte[] readBinFile(String filename)
```



Secure Web

1. Use HTTPS in all applications
2. Re-direct all HTTP://... front pages to HTTPS://...443
3. Avoid FTP access as not secure
4. Use SSLSocket to transfer data between hosts (available since JDK1.4)
5. Prefer POST to GET method, avoid links that reveal data
6. Hide link URL with Java Script
7. Use Java Script to prevent “View Source” actions

Example:

```
<a href=http://Javaschool.com
onMouseOver=\"document.status=‘Text you want to display'; return true;\"
onMouseOut=\"document.status=''; return true;\" >Link</a>
```