ITC-Works Application Developer's Kit ™

Technical White Paper

Ver. 2.4

TABLE OF CONTENTS

Introduction

The ***ITC-Works Applications Developer Kit (ADK)*** is a rich set of Java classes and utilities (over 550) designed to ***simplify and facilitate*** the development of your business applications. ITC-Works ADK's functionality begins where the Java JDK ends. ITC-Works ADK adds higher-level API functionality to existing JDK classes as well as new application functionally that significantly reduces programming complexity and coding effort. The ADK delivers unparalleled new functionality in the areas of:

- Object Relational Mapping
- Code generation (Java and SQL)
- JDBC abstractions
- XML marshaling
- XML Schema and WSDL tools
- Generate XML Schemas from XML instance documents
- Generate Java object graphs with readers and writers from XML Schema and DTD documents
- Generate SQL Table DDL from XML Schemas
- XML preference framework for managing user preferences
- XML Roles and Privileges framework for managing GUI access and functionality
- Resource bundle, property management an internationalization
- WEB Development (Rich JSP taglibs) – auto generated java script
- Java Mail
- HTTP and HTTPS classes
- Text formatting and validation
- Extensive date handling and validation
- And many… many… more

NOTE! As part of the ITC-Works ADK download, you also get the ItcJUnit test case classes that show all the API's in context.

Please also refer to the following supplemental documents for a detailed discussion on those topics:

- ItcWorksORM++.pdf - A complete reference on the object relational mapping (ORM) and functionality of the com.itc.db package.
- JSchemaUsersGuide.pdf – ITCWorks XML Schema framework
- JWSDLUsersGuide.pdf - ITCWorks WSDL framework
- ITC-JSPTaglib-TechnicalReference.pdf - The ITC JSP taglib technical reference.

Functionality

The **ITC-Works ADK** provides the following functionality:

- **Object Relational Mapping (ORM) with easy database access, without any of the tedious, repetitive, and error-prone coding.**
- **Reverse engineers database objects and generates Java value objects and SQL needed for building and saving object graphs.**
- **Generate database record sets as XML documents**

- **A complete framework for reading and writing XML documents (into complex object graphs) without the required mapping coding required by other products.**
- **A complete framework for reading and writing XML Schemas and Web Service Definition Language (WSDL) documents.**
- **Complete toolkit for converting XML documents to Java object graphs, data objects (Maps) and vice-versa. Create XML schemas from XML instance documents. Generate Java object models from XML schema definitions.**
- **Code generators that create Java beans from XML Schemas, database tables and SQL queries.**
- **A rich set of HTTP classes for WEB and Java Servlet development**
- **Complete set of Java mail classes with mail attachment support**
- **JSP Tag library that adds complete content flow, exceptional logic processing internationalization, auto generated java script for form data scrubbing …**
- **Extensive string and date classes for parsing and formatting**
- **Resource bundle and properties management classes that can merge all loaded resources for easy access. Locale management for complete internationalization needs.**

Benefits

Integrating the **ITC-Works ADK** into your software development solutions will provide your organization with the following benefits:

- **Rapid prototyping and deployment of business applications**
- **Significant reduction of software development effort and associated costs**
- **Extensive code generation for your database and XML intrerfacing**
- **Competitive advantage in delivering business solutions "Time to Market"**

Package Summaries

The ITC-Works Framework Class Library is organized into the following packages:

- **com.itc.db**
- **com.itc.xml**
- **com.itc.xml.schema**
- **com.itc.wsdl**
- **com.itc.xml.dtd**
- **com.itc.http**

- **com.itc.util**
- **com.itc.mail**
- **com.itc.components**
- **com.itc.codegen**
- **com.itc.jsp**
- **com.itc.jsptaglib**

Each package is summarized below. Please refer to the Javadoc for the ITC-Works ADK as well as the ItcJUnit testers for the complete documentation.

The **com.itc.db** Package

The classes in this package provide Object Relational Mapping (ORM), JDBC Driver Management, Database catalog management, data source connection, connection pools, and the SQL facilities required to process SQL Statements and Stored Procedure Calls. Please refer to the ItcWorksORM++.pdf document for a complete description of this package.

Sample package listing:

Database Classes:	
ItcDbMgr	Removes an application's dependency on JDBC drivers and URL formats. These are defined in the DatasourceDrivers.xml document. The application refers only to a Datasource Name (similar to ODBC); this class manages the driver setup and connection/connection pools to the actual database. Database references can be easily modified by changing the DatasourceDrivers.xml file as opposed to changing and recompiling the application.
ItcDatabase	This class simplifies access to the database and catalog information. It provides a high-level interface for database catalog services, such as enumeration of: • tables, system tables, stored procedures or views in a database • columns and their attributes in a table or view stored procedures and their parameter names and attributes

SQL Classes:	
ItcSqlMgr	This class manages the execution of SQL statements and stored procedure calls as well as providing a high level interface to the ORM functions. The ItcSqlMgr can render queries as object graphs, XML Documents, POJO's or maps.
ItcSqlParser	This class parses SQL Statements and Stored Procedure Calls in order to obtain the columns in the table(s) that represent the input parameters or the result set attributes.
ItcSqlBatchProcessor	This class reads an SQL file containing one or more SQL statements to execute, and parses and executes each SQL statement in the file. Each statement must be terminated with a semicolon.

Additional database Utility Classes:	
ItcColInfo	This class represents the attributes of a column in a table, or of a parameter of a stored procedure.
ItcForeignKeyInfo	This class has the column(s) about foreign keys that are in a table
ItcSqlData	A container for SQL statement or stored procedure call parameters and metadata.
ItcSqlTypeEditor	The isValid() static method of this class can be used to validate a data string against a given SQL data type.

The **com.itc.xml** Package

This package contains XML data conversion classes, SOAP protocol classes and the data container classes used to hold and create XML content. Parsing XML documents into Java beans or Java object graphs is becoming common place in today's development environment. With the growth of Service Oriented Architecture (SOA), and XML messaging technologies, the need for XML parsing is paramount. At the core, the ItcXmlToBean and ItcBeanToXml classes handle these functions with just a couple of lines of code. Let's look at the following example that creates a Person object instance from an XML document.

Example: XML representation of a Person:

```
<Person id="100">
   <name>John Doe</name>
   <age>27</age>
</Person>

 public class Person
  {
    private String m_strName;
    private int        m_nAge;

    public void setName( String strName )
    ( m_strName = strName; }
    public setAge( int nAge )
    { m_nAge = nAge; }


}

ItcXmlToBean xtb = new ItcXmlToBean();
Person person = (Person)xtb.deserialize( new InputStream(  new FileReader( "\\person.xml" ) ),
Person.class, null );
```

That's all there is to it. The above example demonstrates an XML document that is organized in "Element normal form" where the data resides as text between the open and close tags. Parent XML tags are assumed to be Java classes and the child tags represent properties of that class.

ItcXmlToBean also supports "Attribute normal form" where each tag represents the Java class and the tag attributes are the properties of the class. Please refer to the ITC-JSchemaUserGuide.pdf for examples of attribute normal form documents and complete XML tool support.

Inheriting beans from the ItcXmlBeanAdapter super class allows you to construct fully loaded bean instances from XML documents as shown below.

```
Person person = new Person( new File( "\\person.xml" " ) );

String strName = person.getName();
int nAge = person.getAge();
```

Please see the TestXmlToBean in the test.itc.xml package found in the JunitForItc archive for complete examples of these classes.

Sample class listing:

ItcDataObject	An intelligent map container useful for representing XML documents with attributes
ItcDataObjList	A list container of ItcDataObjects with sort and finder methods useful for ordering and locating ItcDataObjects based on name/value keys.
ItcXmlToDataObj	A class that converts XML documents into ItcDataObjects
ItcDataObjToXml	A class that converts an ItcDataObject or ItcDataObject hierarchies to an XML document
ItcXmlToBean	A class that de-serializes XML documents into an Java object graphs.
ItcXmlBeanAdapter	A optional super class for your beans that assist in xml tags that have both text and attribute tags
ItcBeanToXml	A class that serializes java beans XML documents
ItcXmlWriter	A class API that writes (with optional formatting) XML documents
ItcXmlFileConfig	A helper utility that finds a named XML document, parses it and returns the ItcDataObject parent
ItcSoapParser	Parses XML SOAP requests into Java beans or ItcDataObjects
ItcSoapWriter	Writes SOAP request/response messages from Java Beans or ItcDataObjects

The **com.itc.http** Package

The http classes significantly simply HTTP protocol handling. The ItcHttpClient and ItcHttpsClient classes wrap the corresponding URL connection classes and simplify application coding. In addition the ItcHttpReader and ItcHttpWriter classes offer a server side HTTP protocol implementation for receiving and sending HTTP protocol streams.

The following example demonstrates the ease of sending an HTTP GET request and receiving the response. The getContentAsString simplifies having to write low level stream reading.

```
ItcHttpClient httpClient = new ItcHttpClient( “http://someHost” );

httpClient.connect( "GET" );

String strResp =  http.getContentAsString();
```

Sample class listing:

ItcHttpClient	Provides easy HTTP client support
ItcHttpsClient	Provides easy HTTPS client support
ItcHttpReader	Provides a server side HTTP protocol implementation reader class
ItcHttpWriter	Provides a server side HTTP protocol implementation writer class

The **com.itc.util** Package

The **util** package contains extensive string and date classes, file utilities, data formatting and validation, configuration, log file support, GUI components including standard dialogs, and various exception classes. This package provides:

- functionality missing from the Java classes
- useful extensions for the Java classes
- classes designed to facilitate application development

Benefits: These classes provide general-purpose support for building and deploying applications, saving many lines of low-level, error-prone code.

Sample Package Listing:

String and Date Classes:	
ItcDelimString	Delimited string class for creating and removing sub strings with user-defined delimiter sequences. Unlike the Java StringTokenizer class, this class easily creates delimited strings.
ItcExString	A robust extended string class that adds the ability to: • perform search and replace on sub strings and characters • parse a string into words (bi-directional) using user-defined tokens • validate a string against numeric data • parse a filename/directory path into individual components • search or index into a string based upon user-defined

	criteria • search for patterns • remove sub strings or characters • Strip, left or right pad with fill characters
ItcFormat	String formatter class which centers,left or right justifies text within a string of specified width.
ItcEdit	This class formats and validates data. The format is based on a user-defined edit mask. This class can also use the facilities of an ItcDataDictionary object to determine the data attributes, such as the required entry property, minimum and maximum number of characters, values and ranges, and edit masks. Once an edit mask is defined, a value can be tested for validity based upon the edit mask. This class defines many pre-built standard masks for common data types such as dates, Social Security numbers, phone numbers, file names, and addresses.
ItcTextParser	A bi-directional text parser. This parser can be taught a token grammar, comment sequences and defined delimiters. Each call to the getToken() method returns the next word/token in the string until the end of the string is encountered. The string can be traversed in either direction.
ItcDate	An extensive date class, which supports date validation for any user-defined date configuration, Julian/Gregorian date conversions, and date arithmetic.
ItcSQLTimeStamp	A class that represents the components of an SQL timestamp.

File Utilities	
ItcCfgParser	A class that parses standard ASCII configuration files into Name/Value pairs. The delimiters and characters that denote comment lines and Name/Value pairs are user-defined.
ItcIniFile	Java implementation of the popular Windows INI file format.
ItcLogger	Wrapper around log4j that allows simplification of configuration through flexible property file configuration.
ItcFileFilter	This class implements the FilenameFilter interface to provide a filter for wildcard directory listing requests.
Data Structures and Miscellaneous Utilities	
ItcStack	This class implements a blocking stack. This is very useful in multi-thread applications that both use a common stack. If the stack is empty, the client will block until some other thread pushes data in it or the allotted wait time has expired.
ItcHexDump	The dump() static method of this class can be used to format a byte array as a string with a hex dump.
ItcMath	Contains a method to compute the standard deviation.
ItcMoney	Class that handles rounding issues with floating point arithmetic
ItcStackTraceWriter	A PrintWriter that writes the current call stack.

The **com.itc.components** Package

The **components** package can be used to create wizard-like capabilities (i.e., sequenced dialogs that guide a user through the steps of an operation).

Sample Package Listing:

ItcWizardMgr	This class represents the main wizard dialog, which contains the wizard panels and the Next, Previous, and Cancel buttons. This class manages the button events and display of the wizard panels. User-defined wizard panels are added to the ItcWizardMgr by using the add() method.
ItcWizardPanel	This is the superclass for the user-defined wizard panels. One or more concrete classes can be derived and added to the ItcWizardMgr to implement specific wizard functionality.
ItcIconWindow	This is a Java Swing JComponent that represents itself as an icon. This object supports the drag and drop interface, popup menus and mouse events.
ItcTraceWindow	This object is a JPanel based window with a JTextArea used to display trace data sent over a network connection. This object creates a ServerSocket for the port specified and displays text data sent from network piers. This is a very useful monitoring and debugging aid for processes that cannot display output to the console.

The **com.itc.jsp.taglib** Package

This package provides a powerful set of JSP tag extensions that remove the need to code in line Java in JSP pages thus preserving the separation of presentation and application (J2EE best practices recommended architecture). These tags go far beyond existing libraries providing extensive logic processing with the <if>, <elseIf> and <else> tags which offer a more natural program flow. The input tags can auto generate java script for client side field validation as well as the ability to invoke bean methods that take parameters. The resource bundle and locale tags provide easy internationalization support. One of the most powerful capabilities in the ADK is the ItcServletRequestTo bean object that lets you define object graphs on a form and load/unload data with just ***ONE*** line of code. A properly constructed JSP should be manageable by a GUI design person leaving the Java programmers to provide the bean support. The following JSP snippet shows syntax without tag extensions followed by the same code with tag extensions. The first case mandates Java developer involvement because of the inline java. The Java code needs to be enclosed in scriptlet tags “<% and %>” making JSP readability difficult to maintain. The second example removes the inline java keeping the syntax clean and the development accessibility in the hands of a web designer. The tags can be thought of as html extensions. For a complete example of these tags used in context, please refer to the junitDemo.jsp file in the src/jsp folder found in the ItcJUnit archive.

```
<%
        if ( user.hasRole( “admin” )
        {
%>
        <tr>
                <td>
                        content for the admin role goes here
                </td>
        </tr>
<%      }
        else
    if ( user.hasRole( “payroll” )
        {

%>

        <tr>
                <td>
                        content for the payroll role goes here
                </td>
        </tr>
<%      }
    else
    {
%>
        <tr>
                <td>
        HTML content for all other roles go here
                </td>
        </tr>

<%      } %>
```

Using the ITC tags reduces the above complexity to this:

```
<itc:if name=”user” method=”hasRole” params=”admin” value=”true”>
  <tr>
    <td>
        HTML content for the admin role goes here
    </td>
  </tr>
</itc:if>
<itc:elseIf name=”user” method=”hasRole” params=”payroll” value=”true”>
  <tr>
    <td>
         HTML content for the payroll role goes here
    </td>
  </tr>
</itc:elseIf>
```

```
<itc:else>
  <tr>
    <td>
          HTML content for all other roles goes here
    </td>
  </tr>

</itc:else>
```

It should also be noted that the Struts tag libraries can not invoke bean methods that take parameters like the ITC tags demonstrate above.

Additional Conditional Processing Tags:

The simpleIf and simpleIfElse tags are handy when you want to assign or output a single value based on a condition. These tags do not contain body content like the if, elseIf and else tags. The following example assigns a CSS style sheet type class value based on the role of the user.

Process the attribute expression if the test attribute is true

```
 class='<itc:simpleIf name="user" property="role" value="admin" result="bold"/>'
                                  or

class='<itc:simpleIfElse name="user" property="role" value="admin" trueResult="red"
falseResult="normal"/>'
```

Ignore the body contents of the tag if the test attruibute is true

```
  <itc:excludeIf test="<%=booleanExp%>">
    …
  </itc:excludeIf>
```

Looping tags:

The <itc:counter> tag is handy for fixed count iterations or incrementing and index for array type operations.

Increments an index from zero to maxCount and processes the body of the tag until maxCount is reached

```
  <itc:counter name="x" max="maxCount" >

    …
  </itc:counter>
```

Iterates through a container of objects and process the body of the tag for each object returned. The ITC iterator tag hasmore supported collection types than the struts iterator tag does. Maps, lists and arrays, ietrator and eneration objects are supported collection types.

```
  <itc:iterator name="objName" bean="user" property="roles" >
```

…
</itc:iterator>

HTML Tags:

This tag creates a combo box or list box control and generates all the option tags for each data item in a map or List object for the html select tag. Only one line of code is needed with this tag.

<itc:select selectName=”roles” collection=”mapRoles”/>

This tag provides generated java script for required fields, as well as date and numeric data validation. The example below creates a text field that has required input and requires a valid date in the format mm/dd/yyyy.

<itc:input name=”birthDate” type=”text” required=”true” date=”mm/dd/yyyy”/>

END OF DOCUMENT