Training Course: Introduction to SmartClient

The Introduction to SmartClient training course is designed to get you up and running, productive, and able to start building powerful, cutting-edge applications. It lasts three days, spanning a weekend (Thursday, Friday and Monday) to allow time for you to try out what you have learned, and be able to ask more informed questions on the final day.

Course topics are:

- ✔ Java/JavaScript and CSS Prerequisites
- ✔ Installation & Deployment
- ✔ SmartClient Class System
- ✔ UI Components
- ✔ Data Binding
- ✔ Developer Tools
- ✔ Data Integration
- ✔ Metadata Management
- ✔ Optional Modules
- ✔ Client-Server Communication
- ✔ Event Handling
- ✔ Basic Branding
- ✔ Extending UI Components
- ✔ Performance
- ✔ Animation
- ✔ JPA, Hibernate, SQL Connectors
- ✔ Custom Server Data Integration

“The training was excellent ... we were in awe of Paul's technical ability, he just seems to know everything.”

Register Now

http://smartclient.com/services/index.jsp#training

Courses do fill up. Early booking will secure your place.
<table>
<thead>
<tr>
<th>#1 Welcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) Brief overview of Isomorphic Software</td>
</tr>
<tr>
<td>b) What is SmartClient</td>
</tr>
<tr>
<td>#2 Introducing the SmartClient SDK</td>
</tr>
<tr>
<td>a) Installation</td>
</tr>
<tr>
<td>SDK directory structure includes smartclientRuntime and</td>
</tr>
<tr>
<td>smartclientSDK directories.</td>
</tr>
<tr>
<td>• smartclientRuntime is for deployment – includes required runtime</td>
</tr>
<tr>
<td>resources only</td>
</tr>
<tr>
<td>• smartclientSDK is for development and prototyping -</td>
</tr>
<tr>
<td>includes</td>
</tr>
<tr>
<td>o embedded servlet engine (apache tomcat)</td>
</tr>
<tr>
<td>o embedded SQL database (HSQLDB)</td>
</tr>
<tr>
<td>o extensive Documentation and Examples</td>
</tr>
<tr>
<td>b) Exploring the SDK</td>
</tr>
<tr>
<td>• Home Page</td>
</tr>
<tr>
<td>• Documentation (QuickStart guide, SmartClient Reference,</td>
</tr>
<tr>
<td>Module-specific PDFs)</td>
</tr>
<tr>
<td>• Feature Explorer</td>
</tr>
<tr>
<td>• Tools including Admin Console and Visual Builder</td>
</tr>
<tr>
<td>c) Deploying and using SmartClient</td>
</tr>
<tr>
<td>• Deployment (See also “Deploying SmartClient” topic in SmartClient</td>
</tr>
<tr>
<td>Reference)</td>
</tr>
<tr>
<td>• Loading SmartClient</td>
</tr>
<tr>
<td>o “templates” directory in SDK</td>
</tr>
<tr>
<td>how make a SmartClient-enabled page</td>
</tr>
<tr>
<td>#3 Coding in SmartClient</td>
</tr>
<tr>
<td>a) Hello World example – how to create a SmartClient component.</td>
</tr>
<tr>
<td>• isc object</td>
</tr>
<tr>
<td>• class.create()</td>
</tr>
<tr>
<td>• methods and properties</td>
</tr>
<tr>
<td>• global ID</td>
</tr>
<tr>
<td>b) Introduction to SmartClient classes</td>
</tr>
<tr>
<td>• Canvas class</td>
</tr>
<tr>
<td>• Inheritance in SmartClient</td>
</tr>
<tr>
<td>• The Super method</td>
</tr>
<tr>
<td>• init and initWidget</td>
</tr>
<tr>
<td>• Inheritance and documentation</td>
</tr>
<tr>
<td>c) Declaring components in XML</td>
</tr>
<tr>
<td>• Requires SmartClient server</td>
</tr>
<tr>
<td>• Functionally equivalent to JavaScript – and ultimately translated to JS</td>
</tr>
<tr>
<td>• &lt;JS&gt; tags and &lt;!CDATA&gt; blocks</td>
</tr>
</tbody>
</table>
#3 Coding in SmartClient (Continued)

d) Creating SmartClient subclasses
   Uses of custom class vs instance properties
   class.addProperties() and class.addClassProperties()
   * revisited later

#4 SmartClient User Interface Components

a) Canvas
   • Base UI Component
   • Simple properties:
     o contents
     o backgroundColor, border, styleName
     o sizing, positioning, overflow
   • draw() / show()
   • Events
     o click, doubleClick, right click, dragResposition
     o focus and key events

b) Grids
   • ListGrid
     o Fields and Records (data List - array)
     o Events
     o User Interactions
       ▪ sorting
       ▪ editing
       ▪ filtering (query by example)
       ▪ grouping
       ▪ freezing fields
       ▪ header drag
       ▪ auto fit
       ▪ selection
       ▪ incremental data loading
     o Additional features / customization
       ▪ field type display (text, images, links, dates)
       ▪ custom cell values, styling
       ▪ printing
       ▪ nested components
       ▪ formula / summary fields fields
       ▪ grid / group summaries
       ▪ advanced filter

   • TreeGrid
     o Inherits from ListGrid
     o Hierarchical data (Tree object)
c) Forms
   • DynamicForm vs HTML Form
   • Fields and values (single records)
   • Item types
     o data items vs control items
     o data type and editor type
     o valueMap and valueIcons
     o canvasItems
   • Appearance
     o layout, hint, titles, icons, valueIcons
   • Features
     o appearance (layout, titles, hint, icons, hover)
     o show / hide, enable / disable
     o validation
     o events
     o mask / keypressFilters
   • ValuesManager
     o Similar APIs to forms but allows separated presentation

d) Additional Data Components
   • TileGrid
   • ColumnTree (Miller Columns)
   • DetailViewer
   • Calendar

Continued on next page
e) Layout Components
   • Nesting of components.
   • Canvas children
     o positioning, sizing, overflow
   • Layout members
     o Stack vs Layout
     o Nesting Layouts
     o LayoutSpacers
     o margins, alignment
     o member resizeBars
     o dragReposition within layouts
   • Windows
     o items
     o header / footer controls
     o modality
     o events (minimize, close)
   • Tabsets / Tabs
     o tabs and panes
     o align / orientation
     o selection events
     o tab bar controls
   • SectionStacks
     o sections and items
     o visibility mode
     o interactions (expand, drag)
     o section customization (showHeader, title, icon, header controls)

f) Control Components
   • Buttons
     o action, click
     o Statefulness and styling
     o ImgButtons, StretchImgButton, IButton class
   • Menus
     o menubutton / menubar vs context menus
     o dynamic content (Title, checkmark, enabled/disabled)
     o custom fields (inherits from ListGrid)
     o shortcut keys
     o submenus
     o tree menus/ tree menu item
   • Other control components: Slider, ToolStrip

Continued on next page
# DETAILED AGENDA (Continued)

## #4 SmartClient User Interface Components (Continued)

g) System Utilities.
- isc.isA, isc.History, etc.
- Extensions to native JS objects

## #5 Data Binding

a) Introduction to DataSources
- What is a DataSource
- How to define / load a dataSource
  - Inline Java definition
  - XML definition (requires SC server)
    - `loadDS jsp` tag
    - `<script src=...>` tag pointing to dataSourceLoader servlet
  - Where does data come from (overview only)
    - client only dataSource
    - JSON / XML dataSource
    - SQL DataSource

b) DataSource fields
- field properties
  - name
  - type
  - title
  - hidden
- PrimaryKey field
- foreignKey field (for hierarchical data)
- validators
  - type
  - required
  - built in validator types

c) Explicit DataSource APIs and related concepts
Allows for direct data fetch / manipulation outside the standard databound components

`fetchData() / addData() / updateData() / removeData()`

callbacks and DSRequest / DSResponse objects
Operation Types / Operation Bindings

Continued on next page
## #5 Data Binding (Continued)

### d) Component binding

- **Common databound components:**
  - DynamicForm, ListGrid, TreeGrid, DetailViewer, TileGrid
- **Combining component and dataSource fields**
- **Databinding APIs and behaviors:**
  - ListGrid / TreeGrid:
    - `fetchData() / filterData() / autoFetchData / invalidateCache`
    - `criteria objects`
    - `filterEditor`
    - `incremental data loading, server side sort`
    - `invalidateCache()`
    - `Create / Remove / Update / Delete [CRUD]:`
      - `canEdit, startEditing() / startEditingNew()`
      - `saving and autoSaveEdits`
      - `pending edit values`
      - `validation`
    - `removeData / removeSelectedData`
  - **DynamicForm:**
    - `editRecord() / editNewRecord()`
    - `saveData() / saveOperationType`
    - `getting values as criteria`
    - `AdvancedCriteria and Filter Builder`
  - **FormItem valueMaps / optionDataSource**
    - `valueField, displayField, picklistFields`
  - **DynamicForm and ValuesManager**

### e) Data Model objects

- **ResultSet / ResultTree**
- **automatically generated**
- **List Interface**
- **Intelligent cache management**

© Isomorphic Software. All rights reserved.
#6 Data Integration

a) Client-side databinding

- Client-only dataSource - client side test data
  - Note asynchronous operations, standard DS cache mgmt
- Fetching / Updating remote data by URL
  - XML / JSON operation
  - dataURL, recordXPath, valueXPath
  - dataFormat / dataProtocol
  - operationBindings for per operation URL, protocol etc.
    - (operationType / operationID)
  - transformRequest / transformResponse
  - cacheAllData / testFileName / dataURL for clientOnlyDS
- RestDataSource class
  - per operation dataURLs
  - support for meta data (start row, end row etc)
  - Documented format for server inbound data and responses
  - Existing web services -- SchemaSet loadWSDL / loadXMLSchema
    (see WsdlBinding topic)

- Options for totally custom client-side data integration:
  - ClientOnlyDataSource + getClientOnlyResponse
  - “clientCustom” operationBinding dataProtocol
  - + transformRequest / processResponse

b) Server-side databinding

- Generic server dataSource / server side features:
  - Define dataSource on server using ds.xml file
  - IDACall servlet
  - generic ds approaches (for arbitrary business logic)
    - BasicDataSource subclass plus serverConstructor attribute.
      - Implement execute... methods
      - Server-Side DSRequest / DSResponse objects

- Server side validation
  - type / built-in validation runs on client and server
  - custom validation by setting properties on DSResponse

- Velocity support
  - allows you to directly customize dataSource behavior by injecting VTL statements to be evaluated on the server at runtime.
#6 Data Integration (Continued)

- **OperationBinding** features:
  - DMI serverObject, methodName, method arguments
  - declarative authorization/authentication (requiresAuth / requiresRole / requires)
    - JAAS integration via HttpServletRequest.getRemoteUser() and isUserInRole()
    - Non-JAAS support via servlet override / rpcManager.setUserRoles()
  - outputs
  - mail

- **autoDeriveSchema + schemaBean**

- **Queued requests and “transaction chaining”**

- **SQL DataSource** (See SQLDataSource topic in docs package)
  - serverType= “sql”
  - database config - server.properties + ds tableName
  - default SQL behavior
  - custom SQL with no code
    - criteria / values objects
    - custom sql templating:
      - customSQL
      - selectClause, whereClause, valuesClause, tableClause
    - field properties:
      - customSQL
      - tableName
      - sqlStorageStrategy
  - Combining with DMI to allow custom java code
    - totally custom operations
    - running custom behavior before / after default logic
    - injecting custom velocity variables
  - autoDeriveSchema + tableName

- **Hibernate DataSource**
  - HibernateIntegration doc topic
  - serverType= “hibernate”

- **REST DataSource Servlet**
  - Allows access to any server-side dataSource via the documented client-side RESTDataSource request / response formats.
  - This means you can access SmartGWT dataSources on the server from any client side technology that via HTTP.
#7 Client-Server communication

a) Introduction to the RPCManager class
   - RPCManager handles low level client/server communications
   - Used by DataSource code
   - Can be accessed directly via documented APIs

b) Common use cases for direct RPCManager APIs
   - Start/send queue
   - default actionURL (servlet)
   - send / sendRequest non “record” data
     - atomic responses [yes/no]
     - unstructured data (HTML content, code)

c) RPCManager Features
   - automatic, bi-directional, type-safe Java<->JavaScript translation of nested structures
   - http proxying
   - request queuing

d) Additional client-server communication functionality
   - DMI class + app.xml configuration file
   - WebService class

#8 Developer Tools

a) Developer Console
   - Launching the developer console
   - Overview of tabs
   - The DOM inspector
   - Running code from the Eval area
   - Logging Categories and priorities
   - Adding logging messages to applications
   - Errors and Stack Traces
   - Debugging topic in SmartClient Reference

b) Server logs
   - Where to view server logs
   - How to customize logging sensitivity

c) The Visual Builder
   - Building application views
   - DataSource wizard
   - Loading and Saving code

Continued on next page

© Isomorphic Software. All rights reserved.
### #8 Developer Tools (Continued)

**d) Admin Console**
- Database configuration
- DataSource import

**e) DataSource Generator**

**f) Automated testing Support**
SmartClient has out of the box support for integration with Selenium. See the “tools/selenium” subdirectory for a user guide plus required files.

SmartClient also has integration support for SOASTA, and an AutoTest javascript class which may be used to generate locators / retrieve elements from the DOM making integration with other automated testing tools a possibility.

### #9 Event Handling

**a) SmartClient event model**
- SmartClient event types
- Retrieving event details via EventHandler class
- SmartClient event bubbling

**b) Page level events**
- Capturing events at the Page level
  - `setEvent()` / `clearEvent()` / `registerKey()` / `unregisterKey()`
  - cancelling events
- Page-only events – load, unload, resize, idle

**c) Component events**
- Writing event handlers and cancelling events
- Synthetic events

### #10 Customizing appearance, look and feel (skinning / branding)

**a) Localization**
- See i18n documentation topic.

**b) Skins available in the SDK**
- See isomorphic/skins directory

**c) Anatomy of a skin directory**
- `load_skin.js`
- `skin_styles.css`
- images directory

Continued on next page
### #10 Customizing appearance, look and feel (skinning / branding) [Continued]

c) Creating a custom skin
   - start with existing skin
   - component images and styles
   - loading your custom skin
   - remove / recreate compressed versions
     See “skinning” topic

### #11 Extending SmartClient

a) SmartClient components and inheritance
   - Existing components
     - superclasses
     - automatically created children

b) Creating custom components
   - See “Component Reuse” example in Feature Explorer
   - Creating children at runtime
     - initWidget override
     - use property names rather than global IDs
     - passing properties through to children
   - Embedding Custom HTML, including 3rd party widgets
   - Auto Children subsystem
     - addAutoChild vs createAutoChild

### #12 Performance

a) Simple optimizations
   - create() vs draw() vs show()
   - destroy() vs clear()
   - Memory vs CPU usage
   - application level optimizations
     - autoDraw
     - Component reuse
     - Delaying component creation vs creating up front (EG tab.pane)
     - redraws and refreshes
   - ListGrid optimization
     - draw ahead ratio
     - resultSet.resultSize

Continued on next page.
#12 Performance (Continued)

b) Network Usage
- SmartClient requirements
- Browser caching
  - version parameter on resource URL (automatically applied when using taglib)
  - version-specific skinImgDir
  - FileDownload servlet
- Compression
  - gzip resources and FileDownload Servlet
  - dynamic compression – CompressionFilter
- FileLoader Overview
  - What is it?
  - How to use it

c) Loading application resources
- ViewLoader
- HTMLFlow / HTMLPane *NOT FOR LOADING SMARTCLIENT PAGES!

#13 Optional Modules
- Analytics
- Real-Time Messaging
- Network Performance

#14 Q&A
**EXAMPLES**

Most sections of this course come with examples that you will work through to gain practical experience.

---

**Helpful Resources**

During and after training, you will find the following sources of information very useful:

**SmartClient website:** [http://www.smartclient.com](http://www.smartclient.com)
- SmartClient support forums: [http://forums.smartclient.com/](http://forums.smartclient.com/)
- Contacting SmartClient: support@isomorphic.com or [http://www.smartclient.com/company/contact.jsp](http://www.smartclient.com/company/contact.jsp)
- SmartClient blog: [http://blog.smartclient.com/](http://blog.smartclient.com/)

**SmartClient SDK deployed at localhost:8080**
- SmartClient Reference:
- Feature Explorer:

---

**Register Now**

[http://smartclient.com/services/index.jsp#training](http://smartclient.com/services/index.jsp#training)

Courses do fill up. Early booking will secure your place.