Training Course: Introduction to Smart GWT

The *Introduction to Smart GWT* 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
- ✔ 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.
# DETAILED AGENDA

## #1 Welcome

a) Brief overview of Isomorphic Software  

b) What is SmartClient?  

c) What is Smart GWT?

## #2 up your machine for SmartGwt / SmartGwtEE development

a) Installation  

Requirements:  

- Java JDK  
- GWT  
- SmartGwtEE (useful to have SmartGwt LGPL too for samples)  
- apache ant (included in SGWTEE download)

For ease of development we recommend:

- Eclipse  
- GWT Plugin for Eclipse

b) Resources  

Documentation:  

- Sample index.html  
- JavaDocs (online and local)  
  o note the special “docs” package  
- per-sample readme files  
- documentation embedded in live sample  

Support:  

- Forums (include some sticky starter-guides / how-to's)  
  o note the “FAQ's” topic  
- (support@isomorphic.com)

c) Running the samples  

- Can use ant to run directly from command line  
- Import sample project(s) to Eclipse and run using the GWT plugin.  
- Note: Showcase war also included for simple deployment

d) Anatomy of a Smart GWT project  

- Src directory  
- War directory  
  o Startic versus generated content  

*Note that the GWT site http://code.google.com/ has extensive docs GWT projects*
#3 Coding in Smart GWT

## a) Creating a first application (See also: /docs/SgwtEESetup.html in javadoc)
- 2 possibilities:
  - Create a new GWT application and add SmartGWT functionality to it
  - Copy an existing sample.

We usually recommend starting with an existing sample for a SGWTEE app unless you already have a running app you are adding SGWT functionality to.

## b) SmartGwt Classes and components:
- existing classes / APIs
  - Widget classes (E.G: Canvas) and utility classes (E.G: SC)
- SmartClient (JS) vs SmartGwt (Java)

## c) Creating SmartGWT subclasses
- Uses of custom class vs instance properties
<table>
<thead>
<tr>
<th>#4 SmartClient User Interface Components</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>a) Canvas</strong></td>
</tr>
<tr>
<td>• Base UI Component</td>
</tr>
<tr>
<td>• Simple properties:</td>
</tr>
<tr>
<td>• contents</td>
</tr>
<tr>
<td>• backgroundColor, border, styleName</td>
</tr>
<tr>
<td>• sizing, positioning, overflow</td>
</tr>
<tr>
<td>• draw() / show()</td>
</tr>
<tr>
<td>• Events</td>
</tr>
<tr>
<td>• click, doubleClick, right click, dragResposition</td>
</tr>
<tr>
<td>• focus and key events</td>
</tr>
<tr>
<td><strong>b) Grids</strong></td>
</tr>
<tr>
<td>• ListGrid</td>
</tr>
<tr>
<td>• Fields and Records (data List - array)</td>
</tr>
<tr>
<td>• Events</td>
</tr>
<tr>
<td>• User Interactions</td>
</tr>
<tr>
<td>• sorting</td>
</tr>
<tr>
<td>• editing</td>
</tr>
<tr>
<td>• filtering (query by example)</td>
</tr>
<tr>
<td>• grouping</td>
</tr>
<tr>
<td>• freezing fields</td>
</tr>
<tr>
<td>• header drag</td>
</tr>
<tr>
<td>• auto fit</td>
</tr>
<tr>
<td>• selection</td>
</tr>
<tr>
<td>• incremental data loading</td>
</tr>
<tr>
<td>• Additional features / customization</td>
</tr>
<tr>
<td>• field type display (text, images, links, dates)</td>
</tr>
<tr>
<td>• custom cell values, styling</td>
</tr>
<tr>
<td>• printing</td>
</tr>
<tr>
<td>• nested components</td>
</tr>
<tr>
<td>• formula / summary fields fields</td>
</tr>
<tr>
<td>• grid / group summaries</td>
</tr>
<tr>
<td>• advanced filter</td>
</tr>
<tr>
<td>• TreeGrid</td>
</tr>
<tr>
<td>• Inherits from ListGrid</td>
</tr>
<tr>
<td>• Hierarchical data (Tree object)</td>
</tr>
</tbody>
</table>
#4 SmartClient User Interface Components (Continued)

- CubeGrid
  - (Part of the optional Analytics module)
  - Data Structure:
    - No Fields: Every cell is a record
    - Facets and FacetValues
    - Record objects and data loading

c) Forms
- DynamicForm vs HTML Form
- Fields and values (single records)
- Item types
  - data items vs control items
  - data type and editor type
  - valueMap and valueIcons
  - canvasItems
- Appearance
  - layout, hint, titles, icons, valueIcons
- Features
  - appearance (layout, titles, hint, icons, hover)
  - show / hide, enable / disable
  - validation
  - events
  - mask / keypressFilters
- ValuesManager
  - 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
  - positioning, sizing, overflow
- Layout members
  - Stack vs Layout
  - Nesting Layouts
  - LayoutSpacers
  - margins, alignment
  - member resizeBars
  - dragReposition within layouts
- Windows
  - items
  - header / footer controls
  - modality
  - events (minimize, close)
- Tabsets / Tabs
  - tabs and panes
  - align / orientation
  - selection events
  - tab bar controls
- SectionStacks
  - sections and items
  - visibility mode
  - interactions (expand, drag)
  - section customization (showHeader, title, icon, header controls)

f) Control Components

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

#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.

... Continued on next page

© Isomorphic Software. All rights reserved.
#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.
# DETAILED AGENDA (Continued)

## #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
- 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
#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
  - Standard per-component events (click, doubleClick, etc)
  - Component level drag/drop
  - Component level focus / keyboard events
  - component-specific events:
    - ListGrid: click, doubleclick, drag/drop of records
    - FormItem events: changed, focus, blur, icon events
    - Calendar eventChanged
- EventObject - event details (source, etc), canceling events
- EventHandler class
- SmartClient event bubbling
- Page level events
  - registerKey
  - Native GWT mechanisms for capturing page level mouse events

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

a) Localization
- GWT has a standard localization mechanism (See GWT live documentation)
- inherit i18n gwt module and enable locales you want:
  - <inherits name='com.google.gwt.i18n.I18N'/>
  - <extend-property name="locale" values="en"/>
  - <extend-property name="locale" values="de"/>
- Switch to a locale via parameter on URL: ?locale=de, or via a meta data tag:
  - <meta name="gwt:property" content="locale=de"/>

© Isomorphic Software. All rights reserved.
**DETAILED AGENDA (Continued)**

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

SmartGWT ships with localized system messages for a large number of locales. The above steps will enable this functionality for the locale you selected.

To further localize your application:
- for general localization of application-specific messages, follow standard GWT localization techniques:
  - Create Constants and/or Messages interfaces
  - create per-locale .properties files to fulfill those interfaces
  - use GWT.create() to instantiate these within your app and call the APIs to get localized messages
- To override / extend localized system messages,
  - extend SmartGwtMessages
  - create .properties file(s) with the messages you want to modify
  - call i18nUtil.initMessages(...) to make use of your modified system messages

### b) Loading skins
- Loading skin resource files
- Modifying bootstrap HTML to load skin

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

### c) Creating a custom skin
- start with existing skin
- component images and styles
- loading your custom skin
  See “skinning” topic in documentation

## #11 Extending SmartClient

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

### b) Creating custom components
- Creating children at runtime
- Passing properties through to children

© Isomorphic Software. All rights reserved.
# DETAILED AGENDA (Continued)

## #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

### b) Network Usage
- Compression
  - gzip resources and FileDownload Servlet
  - dynamic compression – CompressionFilter

## #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)

- SmartGWT and SmartClient nightly builds (LGPL and Evaluation):
  - [http://www.smartclient.com/builds](http://www.smartclient.com/builds)
  - SmartClient and SmartGwt 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/)
  - SmartGwt project homepage: [http://code.google.com/p/smartgwt/](http://code.google.com/p/smartgwt/)
  - includes download links, documentation links, lgpl source code
  - SmartClient documentation: [http://www.smartclient.com/product/documentation.jsp](http://www.smartclient.com/product/documentation.jsp)

**SmartGwtEE package**

- index.html
- Javadocs for SmartGwt and SmartClient server
- Numerous samples including complete showcase
- per-sample readme files with build instructions

---

**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.

© Isomorphic Software. All rights reserved.