Software Documentation : Java Custom Client Project Lookup Javascript Bridge

This page last changed on Aug 24, 2008 by ghostar.

1. Scope

1.1 Overview

This component provides an AJAX bridge, allowing JavaScript components or web-pages to interact with the Client Project Lookup Services 1.0 component. The component has two distinct parts. The first part is a set of JavaScript functionality which provides an API mirroring that of the Client Project Lookup Services component. This API interacts with the other part, a Java servlet, through AJAX requests. The servlet translates the requests into parameters which are used to call into the Client Project Lookup Services component, and then converts returned values into an AJAXresponse which is returned to the JavaScript caller. This component is based on a previous component, the Widget Bridge, and the inner-workings and end result should be similar. The distribution for that component can be found here: widget_bridget.zip

1.1.2 Version

1.0

1.2 Logic Requirements

1.2.1 AJAX Communication Protocol

A core part of this component is to design the protocol used to encode AJAX requests and responses. The designer should specify the structure of the request and response for each API method called through the AJAX bridge. It is desired to avoid a large amount of custom code to build and decode the request/response parameters, so using JSON is required.

1.2.2 JavaScript Complex Types

Corresponding to the listed classes in the Client Project Entities DAO component, a JavaScript class should be provided (e.g Project, Client, etc). Each class mirrors the Java version but the exact structure/API is not required to be identical. Changes to better use JavaScript are permitted at the designer's discretion; all other changes must be approved by the PM. The entities required are:

- Project
- ProjectStatus
- Client
- Company
- ClientStatus

1.2.3 JavaScript Services API

1.2.3.1 API

Corresponding to each Service class in the Client Project Lookup Services component (ProjectStatusService, ClientStatusService, LookupService) a JavaScript class with the same name must be provided. The API should match the Java version as closely as the language allows, additional methods may be added at the designer's discretion but any changes to existing methods must be approved by the PM.

1.2.3.2 Making AJAX Requests

Each Service class encodes method parameters into an AJAX request, which is sent to the Java servlet (see section 1.2.4).

1.2.3.3 AJAX Responses

AJAX responses will be decoded into return values.

Because AJAX requests are asynchronous, each Javascript class should only allow one call to take place at a given time. Each Javascript class should allow a callback function to be set, which will be called after the AJAX response is received and decoded.

1.2.4 AJAX Servlet

A Java servlet decodes AJAX requests from the JavaScript part into parameters used to call API methods in the Client Project Lookup Services component, which will make web service calls. Returned values should be encoded into the AJAX response. The servlet should be thread safe, to allow for multiple calls from the clients at the same time.

1.2.5 Logging

1.2.5.1 Java

The Servlet should log all calls at the DEBUG level, including the name of the API method which is to be called, and information to identify the objects involved - object ids but not the complete parameter data. Any errors in the servlet should also be logged at the WARNING level.

1.3 Required Algorithms

- How the encoding and decoding to AJAX requests/responses is done in both JavaScript and Java parts should be covered.
- How callers are notified of return data from asynchronous AJAX responses must be described.

1.4 Example of the Software Usage

A web-page wants to get a list of projects for a client. JavaScript code calls into the bridge API which triggers an AJAX request to the server, which in turn causes the Lookup web service to be called. The return value is encoded to the AJAX response and returned to the browser, which uses it to update the page.

1.5 Future Component Direction

None at this time

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

See the attached architecture diagram.

2.1.3 Environment Requirements

- Development language: Java5.0, JavaScript
- Compile target: Java5.0

2.1.4 Package Structure

com.topcoder.clients.bridge.lookup

js.topcoder.clients.bridge.lookup

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- The configuration for the Java servlet needs to be considered, the designer should discuss the web.xml file.
- The JavaScript needs to know the URL for the Servlet, and this should be easily configured

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

- AJAX
- JSON

3.2.2 TopCoder Software Component Dependencies:

- AJAX Processor 1.0 (http://software.topcoder.com/catalog/c_component.jsp?comp=26736860)
- JSON Object 1.0 (http://software.topcoder.com/catalog/c_component.jsp?comp=22034769)
- Client Project Lookup Services 1.0

**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.3 Third Party Component, Library, or Product Dependencies:

None

3.2.4 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000
- Windows 2003

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the 'Documentation' tab of Poseidon.