[TOPCODER]
SOFTWARE

Requirements Specification

1. Scope

1.1 Overview

The URL History component maintains a dynamic list of URLs on a per user basis. i.e. Each user of this component has its own history. The URLs will be given as inputs to this component as they become available (e.g. when a user visits a new URL). At any time, the component may be queried to return a sorted list of URLs. The sorting criteria include visitation counts, last visitation date, personal ratings, and other user configurable criteria. The component is able to block certain URLs from appearing by maintaining a "blacklist".

1.2 Logic Requirements

1.2.1 Persistent Storage

- URL history data is preserved in persistent storage on a per user basis.
- Provide an association between a user-defined name and a URL.
- Provide a method to query the visitation count of a URL.
- Provide a method to query the last visitation date of a URL.
- Provide a method for a user to give personal ratings (a numerical value) to a URL.
- Provide a mechanism to associate other user-defined attributes with a URL.

Note: The designer is responsible for providing a database schema, DTD, or XSD for defining and validating the persistent storage formats.

1.2.2 Comparisons between two URLs

- By IP Addresses – two URLs are identical if they refer to the same IP host.
- By Textual Forms – a human-readable text representation of the URLs.
  - Root domains only – only the root domain names are compared.
  - Up to **n** levels deep (user configurable) – e.g. http://www.topcoder.com/files/Test1 and http://www.topcoder.com/files/Test2 are considered identical if the level is 1 (i.e. http://www.topcoder.com/files is the same in both URLs, so they are considered identical.) The root domain is considered as level 0.
  - Entire strings – the full URL strings are used for comparisons.
  - Comparisons are case-insensitive.

1.2.3 Grouping URLs

- URLs can be put into groups defined by the user.
- Groups can be nested.
- Groups can be sorted and blacklisted.

1.2.4 Sorting Criteria

- Sort URLs by Visitation Counts.
- Sort URLs by Last Visitation Date.
- Sort URLs by Personal Ratings.

- Sort by Groups.
- Sort URLs by some user-defined criteria.
- Sort URLs by multiple criteria.
- Sorting can be done in ascending or descending order.

1.2.5 Black List

- URLs added to the black list will not appear in the sorted list.
- Groups can be added to the black list. URLs in these groups will not appear in the sorted list.

1.2.6 Storage Limits

- URL history data is maintained up to a user-defined length of time.
- Storage size up to a user-defined value.
- The maximum number of unique URL entries in history (user-definable).
- URL entries and Groups can be flagged as "never-expire" so they will not be purged from the system.
- Never-expire items do not count towards the limits.

Note: the oldest entries are removed first when the limit is reached.

1.2.7 Inputs & Outputs

- Inputs: the URLs that the user visits are given to this component as they become available.
- Outputs: the sorted list of URLs can be queried at any time by the user or another component. URLs that are blacklisted will not appear in the query outputs.

1.3 Required Algorithms

1.3.1 Provide an algorithm to (partially) compare two given URLs according to Requirement 1.2.2.

1.4 Example of the Software Usage

When a user opens a web browser, a list of frequently visited web sites is displayed. This list will be dynamic as opposed to static. It is personalized based on the user web browsing history.

1.5 Future Component Direction

A web GUI front-end can be built on top of this component and present it as a personal home page. A reporting engine may use this component for statistical analyses.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

The component should expose its API in such a way that it is possible to build a GUI front-end on top to display URL history information.

2.1.2 External Interfaces

None.

2.1.3 Environment Requirements

- Development language: Java1.4
- Compile target: Java1.3, Java1.4

2.1.4 Package Structure

com.topcoder.web.urlhistory

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- URL history sorting criteria. User can define new criteria for sorting.
- URL history storage size. e.g. 10 MB
- URL history length (number of days before the data is discarded). e.g. 30 days
- Maximum number of unique URL entries. e.g. 200

Note: Each user may use different values for the above options.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None.

3.2.2 TopCoder Software Component Dependencies:

Configuration Manager 2.1.3

User Profile 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.