Review and Search Services 1.0 Component Specification

1. Design

Topcoder is going to start taking some steps to refactor and improve the TopCoder community sites. The first one we'll start with is www.topcoder.com/tc. The first improvements involve updating some of the "services" that provide contest data to the front end. The current structure forces the contest types to be split into separate pages (i.e. there is a separate active/past/stats/etc. page for each contest type.

This component implements the Review and Search Services of the application.

1.1 Design Patterns

Strategy
SearchContestsManagerAction uses SearchContestsManager implementations which are pluggable.
ReviewOpportunitiesManagerAction uses ReviewOpportunitiesManager implementations which are pluggable.
UpcomingContestsManagerAction uses UpcomingContestsManager implementations which are pluggable.
In addition, the actions are also used strategically by the Struts framework.

DAO/DTO
SearchContestsManager, ReviewOpportunitiesManager , and UpcomingContestsManager can be viewed as DAO for the DTO ContestDTO, ReviewOpportunitiesDTO, and UpcomingContestDTO.

MVC
Actions can be treated like the Controller part of the MVC pattern.

IoC
The configuration is done through Spring injection. Therefore the Inversion of Control (IoC) pattern is used.

1.2 Industry Standards

XML
JSON
HQL

1.3 Required Algorithms

1.3.1 Logging

The system will log all the errors, exceptions, warning and debug information during the application execution.
Debug should provide enough information to track the sequence of flow and Info should provide necessary information like stats (etc.). Error & Fatal should provide the stack trace of the error and a meaningful message.
The next information need to be logged:

- Method name on the application server-side (String max 50 chars, non-empty),
- Date and timestamp (String, date/time format with year/month/day and hour/minute/seconds/milliseconds, non empty),
- Method entry/exit (on the DEBUG level, serialized to String, max 4096 chars, can be empty),
- Parameters of the methods and return values (on the DEBUG level, serialized to String, max 4096 chars, can be empty),

- Stack trace for the exception (String, max 4096 chars, non empty),
- Exception name (if an exception occurred, String, max 100 chars, non empty).

New logging requirements will be implemented using log4j. No existing logging functionality will be changed.

1.3.2 HQL in UpcomingContestsManagerImpl+retrieveUpcomingContests()

The pseudo code HQL for getting upcoming contests is given below:

```
select
   ProjectGroupCategoryLookup.name as type,

   ProjectCategoryLookup.name as subType,

   (select to_char(max(scheduledStartTime), '%Y-%m-%d %R') from ProjectPhase where phaseTypeId
= 1 and projectId = p.projectId) as registrationStart,

   (select to_char(max(scheduledEndTime), '%Y-%m-%d %R') from ProjectPhase where phaseTypeId
= 2 and projectId = p.projectId) as submissionEnd,

   ROUND(((select max(scheduledEndTime) from ProjectPhase where phaseTypeId = 2 and projectId
= p.projectId) - (select max(scheduledStartTime) from ProjectPhase where phaseTypeId = 1 and
projectId = p.projectId))::interval minute(9) to minute::char(20)::decimal(10,2)/60/24, 2)
as duration,

   pi.value || ' ' || pi2.value as contestName,

   technology_list(pi4.value) as technologies,

   pi5.value::INTEGER as firstPrize,

   DECODE(ps.projectStatusId, 1, 'Draft', 2, 'Scheduled', ps.name) as status,

from Project p

join ProjectCategoryLookup on
Project.projectCategoryId=ProjectCategoryLookup.projectCategoryId // this join is for sub-type

join ProjectGroupCategoryLookup on
ProjectCategoryLookup.projectGroupCategoryId=ProjectGroupCategoryLookup.projectGroupCateg
oryId // this join is for type

join ProjectStatusLookup ps on p.projectStatusId = ps.projectStatusId // this join is for project
status

join ProjectInfo pi on p.projectId = pi.projectId and pi.projectInfoTypeId = 6  and lower(pi.value)
not like '%delete%'  // this join is for project name

join ProjectInfo pi2 on p.projectId = pi2.projectId and pi2.projectInfoTypeId = 7  // this join
is for project version

join ProjectInfo pi4 on p.projectId = pi4.projectId and pi4.projectInfoTypeId = 1  // this join
is for comp_version

join ProjectInfo pi5 on p.projectId = pi5.projectId and pi5.projectInfoTypeId = 36  // this
join is for first place prize

join DirectProject tdp on p.tcDirectProjectId = tdp.projectId and tdp.projectId not in (840)
// exclude test projects

where p.projectStatusId in (1,2,7)

  and p.projectCategoryId not in (27) --exclude spec reviews

  and not exists (SELECT 'hasEligibilityConstraints' FROM ContestEligibility ce WHERE ce.isStudio
= 0 AND ce.contestId = p.projectId)
```

```
  and (select max (scheduledStartTime) from ProjectPhase where phaseTypeId = 1 and projectId
= p.projectId) between (today -1) and (today + 365)
```

And finally depending on the value of sortingOrder, use “asc” or “desc” for the sorting by given column name. If column name is empty/null, then results should be displaying based scheduled time of contests, i.e. using date column as default.

If all the properties of the filter are null or empty, then no filtering performed and the above HQL will be used without appending further condition clauses.

If filter.types is not empty, append a condition:
```
type in (the elements of filter.types connected by ",")
```

Similarly, if filter.subTypes or catalogs are not empty, append similar IN conditions on subType and catalog.

If contestName is not null or empty, append a condition:
```
contestName LIKE {%filter.contestName%}
```

If registrationStartDate is not null
- If intervalType is BEFORE, append:
  ```
  registrationPhase.scheduledStartTime < {registrationStartDate.firstDate}.
  ```
- If intervalType is AFTER, append:
  ```
  registrationPhase.scheduledStartTime > {registrationStartDate.firstDate}.
  ```
- If intervalType is ON, append:
  ```
  registrationPhase.scheduledStartTime = {registrationStartDate.firstDate}
  ```
- If intervalType is BETWEEN_DATES, append:
  ```
  registrationPhase.scheduledStartTime between
{registrationStartDate.firstDate}            and {registrationStartDate.secondDate}
  ```
- If intervalType is BEFORE_CURRENT_DATE, append:
  ```
  registrationPhase.scheduledStartTime < CURRENT_DATE
  ```
- If intervalType is AFTER_CURRENT_DATE, append:
  ```
  registrationPhase.scheduledStartTime > CURRENT_DATE
  ```

The registrationStartDete is got from ProjectPhase where phaseTypeId = 1.

If submissionEndDate is not null, append similar conditions as above.
For submissionEndDate, use submissionEndDate column to compare.

The submissionEndDate is got from ProjectPhase where phaseTypeId = 2.

If filter.prizeStart is not Filterable.OPEN_INTERVAL, append:
```
firstPrize >= {filter.prizeStart}.
```

If filter.prizeEnd is not Filterable.OPEN_INTERVAL, append:
```
firstPrize <= {filter.prizeEnd}.
```

The firstPrize is got from project_info table where project_info_type_id = 16.

For the above two cases, note that the HQL CAST(...as...) expression should be used to convert the string value of firstPrize into integer for conversion.

1.3.3 HQL in ReviewOpportunitiesManagerImpl+retrieveReviewOpportunities()

The pseudo code SQL for getting review opportunities is given below, developers should convert them to HQL format:

```
select distinct
   ProjectGroupCategoryLookup.name as type,
   ProjectCategoryLookup.name as subType,
   rboardPayment.amount AS primaryAmount,
   rboardPayment2.amount AS secondaryAmount,
  (select count(distinct u.resource_id)
   from upload u, submission s
   where u.project_id = p.project_id
   and s.submission_type_id = 1
   and s.submission_status_id in (1,2,3,4)
   and u.upload_id = s.upload_id
   and u.upload_type_id = 1)
   as submission_count
   , c.component_name
   , reviewpp.scheduled_start_time as review_start
   , reviewpp.scheduled_end_time as review_end
   ,
  (select ph.parameter::int-round(count(*))
    from rboard_application ra
    where ra.project_id = p.project_id and ra.phase_id<1000) as available_spots
   ,
     case
       when
         exists (select * from project_phase where project_id = p.project_id and phase_type_id
= 1)
        then
           (select (scheduled_start_time + 12 units hour)
            from project_phase
            where project_id = p.project_id
            and phase_type_id = 1)
        else
           (select (scheduled_start_time + 12 units hour)
            from project_phase
            where project_id = p.project_id
            and phase_type_id = 2)
      end
   as opens_on
   , pi_prize.value::FLOAT AS prize
  from project p
    , project_info projinfo
    , comp_versions cv
    , comp_catalog c
    , categories cat
    , project_phase regpp
    , project_phase submpp
    , project_phase screenpp
    , project_phase reviewpp
    , phase_criteria ph
    , comp_version_dates cvd
    , project_info pi_prize
    , project_info pi_is_dr
    , outer project_info pi_dr_points
    join ProjectCategoryLookup on
Project.projectCategoryId=ProjectCategoryLookup.projectCategoryId // this join is for sub-type
    join ProjectGroupCategoryLookup on
ProjectCategoryLookup.projectGroupCategoryId=ProjectGroupCategoryLookup.projectGroupCateg
oryId // this join is for type
    join RboardPayment rboardPayment on project_id = rp.project_id and phase_id = rp.phase_id
and primary_ind = 1 // this join is for primary_amount
    join RboardPayment rboardPayment2 on project_id = rp.project_id and phase_id = rp.phase_id
and primary_ind = 0 // this join is for secondaryAmount
 where 1=1
   and projinfo.project_info_type_id = 1
   and projinfo.project_id = p.project_id
   and projinfo.value = cv.comp_vers_id
   and cv.phase_id in (112,113)
   AND regpp.project_id = p.project_id
   AND regpp.phase_type_id = 1
   AND submpp.project_id = p.project_id
   AND submpp.phase_type_id = 2
   AND screenpp.project_id = p.project_id
```

```
   AND screenpp.phase_type_id = 3
   and reviewpp.phase_type_id = 4
   and reviewpp.project_id =  p.project_id
   and ph.project_phase_id = reviewpp.project_phase_id
   and ph.phase_criteria_type_id = 6
   AND (submpp.phase_status_id = 2 or regpp.phase_status_id = 2 or screenpp.phase_status_id
= 2 or reviewpp.phase_status_id = 2)
   and regpp.scheduled_start_time <= CURRENT
   and cvd.comp_vers_id = cv.comp_vers_id
   and cvd.phase_id = cv.phase_id
   and cvd.status_id <> 303
   and c.component_id = cv.component_id
   and c.root_category_id = cat.category_id
   and extend((select scheduled_start_time
         from project_phase
        where project_id = p.project_id
          and phase_type_id = 2), year to hour) <= extend(current, year to hour)
   and (((select count(*)
         from rboard_application
        where project_id = p.project_id
          and phase_id = cv.phase_id) < 3)
    or not exists (select '1'
                  from rboard_application
                 where project_id = p.project_id
                   and phase_id = cv.phase_id))
 AND NOT EXISTS (SELECT 'has_eligibility_constraints' FROM contest_eligibility ce WHERE
ce.is_studio = 0 AND ce.contest_id = p.project_id)
 AND p.project_status_id = 1
 AND p.project_id = pi_prize.project_id
 AND pi_prize.project_info_type_id = 16
 AND p.project_id = pi_is_dr.project_id
 AND pi_is_dr.project_info_type_id = 26
 AND p.project_id = pi_dr_points.project_id
 AND pi_dr_points.project_info_type_id = 30
 and (select ph.parameter::int-round(count(*))
     from rboard_application ra
     where ra.project_id = p.project_id and ra.phase_id<1000) > 0
```

And finally depending on the value of sortingOrder, use “asc” or “desc” for the sorting.
The handling of filter in this method is almost the same as UpcomingContestsManagerImpl+retrieveUpcomingContests().

1.3.4 HQL in SearchContestsManagerImpl+searchContests()

```
SELECT
   ProjectGroupCategoryLookup.name as type,
   ProjectCategoryLookup.name as subType,
   pi.value as contestName
 ,cat.categoryName as catalogName
,(SELECT COUNT(*)
  FROM Resource
  WHERE project = p.projectId
  AND resourceRole.id = 1)
  as totalInquiries
 , SELECT COUNT(*)
  FROM Submission s
  WHERE s.upload.project = p.projectId
  and s.submissionTypeId = 1
  and s.submissionStatusId in (1,2,3,4)
  as totalSubmissions
, (SELECT COUNT(*)
  FROM Submission s
  WHERE s.upload.project = p.projectId
  and s.submissionTypeId = 1
  and s.submissionStatusId in (1,2,3,4)
  and s.screeningScore >= 75)
```

```
  as passedScreeningCount
  , ProjectResult.finalScore as winnerScore
  , winnerIdInfo.value as winnerExternalReferenceId
FROM Project p
join Categories cat on Project.projectCategoryId=Categories.categoryId
join ProjectCategoryLookup on
Project.projectCategoryId=ProjectCategoryLookup.projectCategoryId // this join is for sub-type
join ProjectGroupCategoryLookup on
ProjectCategoryLookup.projectGroupCategoryId=ProjectGroupCategoryLookup.projectGroupCateg
oryId // this join is for type
join ProjectInfo pi on p.project_id = pi.project_id and pi.project_info_type_id = 6  // this
join is for project name
join ProjectResult on ProjectResult.projectId=Project.projectId and ProjectResult.placed=1
join ProjectInfo winnerIdInfo on winnerIdInfo.projectId=Project.projectId and
winnerIdInfo.projectInfoTypeId=2
WHERE NOT EXISTS (SELECT 'has_eligibility_constraints' FROM ContestEligibility ce WHERE
ce.isStudio = 0 AND ce.contestId = p.projectId)
```

And finally depending on the value of sortingOrder, use “asc” or “desc” for the sorting.
The handling of filter in this method is almost the same as UpcomingContestsManagerImpl+retrieveUpcomingContests().

1.3.5 Wildcard Handling

For all “Like” conditions in HQL, the string to compare should be surrounded with “%”. So for example,

```
contestName LIKE {filter.contestName}
```

should be

```
contestName LIKE “%{filter.contestName}%”
```

1.3.6 Error Page

When the actions of this component are deployed, the configuration should ensure that if any exception is thrown by these actions, the user is redirected to an error page. Since this is handled by the Struts configuration, it’s out of scope of this component.

1.3.7 JSON Format

When calling the BaseJSONParameterAction subclass actions of this component, the caller should send a JSON string as an http parameter. This JSON object should be in the following form:

```
{ columnName:”..”,
  sortingOrder:”..”,
  pageNumber: “..”,
  pageSize: “..”,
  filter: “..” // the format of this value is the standard JSON string of any one of
the 3                // filter classes
}
```

Note that “-1” is used as the value of pageNumber and pageSize to suggest returning all pages.

For enums such as SortingOrder, the value in the JSON string should be the name of the enum, such as “ASCENDING”.

1.4 Component Class Overview

SearchContestsManager [Interface]
This interface defines the contract for getting information of contests by various filtering conditions. It has an overloaded method that supports pagination.

ReviewOpportunitiesManager [Interface]
This interface defines the contract for getting information of review opportunities by various filtering conditions. It has an overloaded method that supports pagination.

UpcomingContestsManager [Interface]
This interface defines the contract for getting information of upcoming contests by various filtering conditions. It has an overloaded method that supports pagination.

AbstractManagerImpl
This is the base manager implementation for logging. It has getter and setter for logger instance. It extends HibernateDaoSupport.

SearchContestsManagerImpl
This is the default implementation of SearchContestsManager. It extends HibernateDaoSupport to execute HQL query to get the data. It extends AbstractManagerImpl class for logging.

ReviewOpportunitiesManagerImpl
This is the default implementation of ReviewOpportunitiesManager. It extends HibernateDaoSupport to execute HQL query to get the data. It extends AbstractManagerImpl class for logging.

UpcomingContestsManagerImpl
This is the default implementation of UpcomingContestsManager. It extends HibernateDaoSupport to execute HQL query to get the data. It extends AbstractManagerImpl class for logging.

ReviewOpportunitiesDTO
This is a simple container class that has information of Review Opportunities.

AbstractContestDTO [Abstract]
This class defines the common type properties of all DTO classes of this component.

ContestDTO
This is a simple container class that has information of Contests.

UpcomingContestDTO
This is a simple container class that has information of Upcoming Contest.

ContestNameEntity [Abstract]
This class defines the common name property of all DTO and Filter classes of this component.

AbstractContestsFilter [Abstract]
This class defines the common properties of all filter classes of this component. It also implements Filterable marker interface.

UpcomingContestsFilter
This class defines the filtering condition for searching Upcoming Contests.

ReviewOpportunitiesFilter
This class defines the filtering condition for searching Review Opportunities.

ContestsFilter
This class defines the filtering condition for searching contests.

SearchContestsManagerAction
This action simply wraps around SearchContestsManager and provides its service to the front end.

ReviewOpportunitiesManagerAction
This action simply wraps around ReviewOpportunitiesManager and provides its service to the front end.

UpcomingContestsManagerAction
This action simply wraps around UpcomingContestsManager and provides its service to the front end.

1.5 Component Exception Definitions

ContestsServiceManagerException
The exception extends from BaseException. It's thrown if it has any exceptions in Manager layer.

1.6 Thread Safety

All the entity classes, DTO classes and exception classes defined in this component are mutable and not thread-safe, but this component is effectively thread-safe to use under Struts framework because dedicated instances of struts actions and interceptors will be created by the Struts framework to process each user request.

2. Environment Requirements

2.1 Environment

- Development language: Java1.6
- Compile target: Java1.6

2.2 TopCoder Software Components

- Base Exception 2.0 – used as the base for all custom exception classes.
- JSON Object 1.0 – used to convert JSON string into JSON object

NOTE: The default location for TopCoder Software component jars is../lib/tcs/COMPONENT_NAME/COMPONENT_VERSION relative to the component installation. Setting the tcs_libdir property in topcoder_global.properties will overwrite this default location.

2.3 Third Party Components

- Spring 3.0 (http://www.springsource.org/download)
- Struts 2.2.3 (http://struts.apache.org/download.cgi)
- Apache Log4j 1.2.15 (http://logging.apache.org/log4j/1.2/index.html)

NOTE: The default location for 3rd party packages is ../lib relative to this component installation. Setting the ext_libdir property in topcoder_global.properties will overwrite this default location.

3. Installation and Configuration

3.1 Package Name

com.topcoder.web.tc
com.topcoder.web.tc.implement
com.topcoder.web.tc.dto
com.topcoder.web.tc.action

3.2 Configuration Parameters

Configuration for AbstractManagerImpl:

Name	Description	Value
logger	The Logger instance for logging.	org.apache.log4j.Logger instance. It can be null. Optional.

Configuration for SearchContestsManagerAction:

Name	Description	Value
searchContestsManager	The SearchContestsManager for getting contest.	com.topcoder.web.tc.SearchContestsManager instance. It cannot be null. Required.

Configuration for ReviewOpportunitiesManagerAction:

Name	Description	Value
reviewOpportunitiesManager	The ReviewOpportunitiesManager for getting review opportunities.	com.topcoder.web.tc.ReviewOpportunitiesManager instance. It cannot be null. Required.

Configuration for UpcomingContestsManagerAction:

Name	Description	Value
upcomingContestsManager	The UpcomingContestsManager for getting upcoming contests.	com.topcoder.web.tc.UpcomingContestsManager instance. It cannot be null. Required.

3.3 Dependencies Configuration

Log4j should be configured properly.

4. Usage Notes

4.1 Required steps to test the component

Extract the component distribution.
Create a database under Informix Server.
Execute test_files\create.sql, test_files\data.sql, test_files\procedure.sql. You may configure to execute create.sql and data.sql using ant target, but procedure.sql must be run using dbaccess or 3rd party utility like Server Studio.
Execute 'ant test' within the directory that the distribution was extracted to.
Execute test_files\drop.sql to clean up the database. You may configure to execute this script using ant target.

4.2 Required steps to use the component

Please see the demo.

4.3 Demo

A sample Spring configuration file "applicationContext.xml" is attached for reference only.

4.3.1 API Usage (here is an example of using ReviewOpportunitiesManager, the other two managers are similar)

```
// 1. Create ReviewOpportunitiesFilter
ReviewOpportunitiesFilter filter = new ReviewOpportunitiesFilter();
// 2. Set filter fields
filter.setContestName("test");
List<String> list = new ArrayList<String>();
list.add("Component Design");
list.add("System Assembly");
list.add("Conceptualization");
filter.setSubtype(list);
list = new ArrayList<String>();
list.add("Component Design");
list.add("Conceptualization");
filter.setType(list);
list = new ArrayList<String>();
list.add("Design");
list.add("UI Development");
filter.setCatalog(list);
DateIntervalSpecification date = new DateIntervalSpecification();
date.setIntervalType(DateIntervalType.BEFORE);
date.setFirstDate(new Date());
filter.setRegistrationStart(date);
date = new DateIntervalSpecification();
date.setIntervalType(DateIntervalType.BEFORE_CURRENT_DATE);
filter.setSubmissionEnd(date);
date = new DateIntervalSpecification();
date.setIntervalType(DateIntervalType.BEFORE_CURRENT_DATE);
filter.setReviewEndDate(date);
filter.setPaymentStart(400);
filter.setPaymentEnd(1000);
// 3. Create ReviewOpportunitiesManager instance using Spring
ReviewOpportunitiesManager manager = (ReviewOpportunitiesManager) applicationContext
       .getBean("reviewOpportunitiesManager");
// 4. Retrieve review opportunities using manager
List<ReviewOpportunityDTO> result = manager.retrieveReviewOpportunities(filter);
```

The usage of UpcomingContestsManager and SearchContestsManager are similar to ReviewOpportunitiesManager.

4.3.2 Struts Usage

This submission comes with a fully working example which can be deployed on a web server such like tomcat. Please refer to HOW_TO_TEST_UNDER_WEB_SERVER.TXT for how to deploy this application.

5. Future Enhancements

None at this moment.