/\*\*

\* Точка входа в ЦО. Возвращает списки отчетов {@link ReportList}, а так же

\* конкретные отчеты {@link Report} по имени или коду. Возвращает информацию о

\* ходе построения конкретного запуска {@link Progress}. Является фабрикой для

\* пустого {@link DocumentSet}.

\*

\* @author EIvanov

\*

\*/

@AFClassReference(ref = ru.intertrust.cmj.reporter.impl.ReporterImpl.class)

public interface Reporter extends AFApplication.CanGetEntityByUNID<Report> {

public ReportList getReportList();

public ReportList getReportList(ReportList.Name name);

public ReportList getReportList(ReportList.Name name, Set<String> supportedTypes);

public ReportList getReportList(Name name, Set<String> supportedTypes, Set<String> keywords);

public Report getReport(String name);

public Report getReportById(String id);

public Progress getProgressById(String id);

public DocumentSet createDocumentSet();

}

/\*\*

\* Список отчетов.

\*

\* @author EIvanov

\*

\*/

public interface ReportList {

public Name getName();

public Set<String> getSupportedTypes();

public Set<String> getKeywords();

public List<Report> getReports();

public static enum Name {ALL, SELFPROPELLED, SELECTIONBASED;}

}

/\*\*

\* Представитель готового отчета в форме файла.

\*

\* @author EIvanov

\*

\*/

public interface Result {

public String getName();

public int getSize();

public InputStream getInputStream();

}

/\*\*

\* Представитель конструктива отчета. Предоставляет информацию о составе

\* диалогов параметров и возможность запустить отчет на построение.

\* Предполагается, что клиент сперва самостоятельно призведет опрос пользователя

\* по указанным диалогам и сформирует таблицу параметров, затем вызовет один из

\* методов start, затем периодически будет переполучать объект Progress для

\* отслеживания готовности.

\*

\* @author EIvanov

\*

\*/

public interface Report extends AFObject {

public String getName();

public String getFileName();

boolean isSelectionBased();

public List<String> getKeywords();

public List<ParamDialog> getParamDialogs();

public Progress start();

public Progress start(Map<String,Object> params);

public Progress start(DocumentSet collection);

public Progress start(Map<String,Object> params, DocumentSet collection);

public Progress start(Map<String, Object> params, DocumentSet collection, boolean async);

}

/\*\*

\* Представитель хода построения отчета. Предоставляет данные для отображения

\* статуса задачи и отрисовки ProgressBar.

\*

\* @author EIvanov

\*

\*/

public interface Progress extends AFObject {

/\*\*

\* @return представитель констуктива отчета.

\*/

public Report getReport();

/\*\*

\* @return код конкретного запуска отчета.

\*/

public String getId();

/\*\*

\* @return состояние хода построения. см. {@link State}

\*/

public State getState();

public String getMessage();

/\*\*

\* @return общее число "шагов" - для отрисовки ProgressBar. может быть null,

\* если не определено.

\*/

public Long getTotal();

/\*\*

\* @return число уже выполненных "шагов" - для отрисовки ProgressBar. может

\* быть null, если не определено.

\*/

public Long getDone();

/\*\*

\* @return результат построения отчета. Это свойство доступно только в

\* случае getState()==State.DONE.

\*/

public Result getResult();

/\*\*

\* отправка команды на остановку.

\*/

public void cancel();

/\*\*

\* Обновляет информацию о ходе построения.

\*/

public void refresh();

/\*\*

\* Состояние хода построения отчета

\*

\* @author EIvanov

\*

\*/

static public enum State {

/\*\*

\* Журнал отсутствует. Нештатная ситуация. Например, вручную удален

\* администратором.

\*/

UNAVAILABLE("unavailable"),

/\*\*

\* Отчет в процессе построения. Нужно ждать результат. Доступны свойства

\* {@link Progress#getMessage()}, {@link Progress#getTotal()},

\* {@link Progress#getDone()}

\*/

RUNNING("running"),

/\*\*

\* Клиент уже успешно отправил команду на отмену построения, но отчет

\* еще не принял ее. В ближайшее время остановится.

\*/

CANCELLING("cancelling"),

/\*\*

\* Отчет успешно построен. Можно забирать результат через свойство

\* {@link Progress#getResult()}.

\*/

DONE("done"),

/\*\*

\* Отчет завершился с ошибкой. Результат недоступен. Текст ошибки в

\* свойстве {@link Progress#getMessage()}, подробности в журнале

\* последнего запуска в ЦО.

\*/

FAILED("failed"),

/\*\*

\* Отчет успешно прерван по запросу клиента.

\*/

CANCELLED("cancelled");

State(String name) {this.name = name;}

private final String name;

@Override

public String toString() {

return this.name;

}

}

}

/\*\*

\* Контейнер для упаковки набора документов для передачи в отчет

\* "по выделенным документам". Предоставляет возможность добавления всех

\* документов и указанной базы, всех документов из указанной вьюшки, всех

\* документов из DocumentCollection, отдельных документов по-одному, а так же

\* полнотекстовые и @-формульные запросы. Может быть передан в качестве

\* аргумента в методы {@link Report#start(DocumentSet)},

\* {@link Report#start(java.util.Map, DocumentSet)},

\* {@link Report#start(java.util.Map, DocumentSet, boolean)}.

\*

\* @author EIvanov

\*

\*/

public interface DocumentSet {

public void addAll(String dbType, Database db);

public void addAll(String dbType, String server, String replicaId);

public void addViewByIdent(String dbType, String ident, String complect, String viewName);

public void addDocument666(String dbType, String server, String replicaId, String unid);

}

/\*\*

\* Структура данных, описывающая свойства одного диалога параметров в отчете.

\*

\* @author EIvanov

\*

\*/

public interface ParamDialog {

/\*\*

\* @return имя формы для отображения диалога параметров

\*/

public String form();

/\*\*

\* @return префикс, который будет автоматически добавлен к именам полей из

\* указанной формы с целью избежать потенциального конфликта имен

\* полей с другими диалогами. Например, если для диалога указан

\* префикс Author, а в соответствующей форме есть два поля Name и

\* Id, то в отчете буду доступны параметры с именами Author.Name и

\* Author.Id

\*/

public String prefix();

/\*\*

\* @return текстовый комментарий, который должен быть отображен пользователю

\* где-нибудь над формой. Например, "Укажите автора:"

\*/

public String caption();

}

/\*\*

\* Интерфейс представляет любой "доменный" объект.

\* <p>

\* Известные наследники: {@link ru.intertrust.cmj.af.so.SOParty}, {@link ru.intertrust.cmj.af.so.SOBeard}

\* <p>

\* Известные реализации: {@link AFObject.Mapped}

\* <p>

\* @author Gleb Nozdrachev

\*/

public interface AFObject {

/\*\*

\* Возвращает идентификатор (UNID) объекта.

\* @return не может быть <code>null</code>

\*/

String getUNID ();

/\*\*

\* Возвращает признак того, что сущность является "новой", т.е. еще не была сохранена в Хранилище.

\* @return <code>true</code> если сущность является "новой"; <code>false</code> в противном случае

\*/

boolean isNew ();

/\*\*

\* Возвращает дату-время модификации сущности.

\* @return не может быть <code>null</code>

\*/

Calendar getModified ();

}