IBM DB2 Information Integrator
OmniFind Edition

エンタープライズ・サーチ
プログラミング・ガイドおよび API リファレンス

バージョン 8.2.2

SD88-6375-01
(英文原典：SC18-9284-01)

IBM DB2 Information Integrator
OmniFind Edition

エンタープライズ・サーチ
プログラミング・ガイドおよび API リファレンス

バージョン 8.2.2

SD88-6375-01
(英文原典：SC18-9284-01)

本書および本書で紹介する製品をご使用になる前に、『特記事項』に記載されている情報をお読みください。

本書には、IBM の専有情報が含まれています。その情報は、使用許諾条件に基づき提供され、著作権により保護されています。本書に記載される情報には、いかなる製品の保証も含まれていません。また、本書で提供されるいかなる記述も、製品保証として解釈すべきではありません。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

http://www.ibm.com/jp/manuals/main/mail.html

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

http://www.ibm.com/jp/manuals/ の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原　典： SC18–9284–01
IBM DB2 Information Integrator
OmniFind Edition
Programming Guide and API Reference for Enterprise Search
Version 8.2.2

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2005.7

この文書では、平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財) 日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

目次

第 1 章 エンタープライズ・サーチ API

IBM® DB2® Information Integrator OmniFind Edition (DB2 II OmniFind Edition) は、エンタープライズ・サーチ 用の Java アプリケーション・プログラミング・インターフェース (API) を提供します。これらの API を 使用して、検索コレクションに対する照会を処理する検索アプリケーションを作成します。 また、データ・リスナー API を使用して、検索コレクションに対して文書を追加および除去することもできます。

IBM SIAPI

IBM search and index API (SIAPI) は、カスタム検索アプリケーションの作成に使用します。 SIAPI のエンタープライズ・サーチ・インプリメンテーションによって、検索サーバーへのリモート・アクセスが可能になります。検索サーバーは、エンタープライズ・サーチ・システムのコレクション・データを保管します。ユーザーは、これらの API を使用して、検索要求のサブミット、検索結果の処理、分類法ツリーのブラウズを行うアプリケーションを作成します。

DB2 II OmniFind Edition で提供されているサンプル検索アプリケーションについては、23 ページの『第 3 章 サンプル SIAPI アプリケーション』を参照してください。

データ・リスナー API

データ・リスナーは、クライアント・アプリケーションからの要求を受け入れる、エンタープライズ・サーチ・コンポーネントです。要求を受けて、コレクションにデータを追加したり、コレクションからデータを除去します。通常、データをクロールし、そのデータに対して構文解析および索引付けを行ってから、そのデータを検索で使用可能にすることによって、エンタープライズ・サーチ・コレクションを作成します。データ・リスナー・クライアント・アプリケーションを使用して、コレクションにページを追加したり、データ・ソースのクロールを待たずにコレクションから Uniform Resource Identifier (URI) を除去したり、コレクションの Web クローラーに対して Uniform Resource Locator (URL) へのアクセスまたは再アクセスを指示することができます。

DB2 II OmniFind Edition で提供されているデータ・リスナー・クライアント・サンプル・アプリケーションについては、37 ページの『データ・リスナー・クライアント・サンプル・アプリケーション』を参照してください。

検索 API のセキュリティー

検索および API 参照は、各WebSphere® 検索ノードにインストールされている ESSearchServer Enterprise アプリケーションとリモートで通信します。

グローバル・セキュリティーが使用可能になったら、WebSphere Application Server は、すべての HTTP 要求に対して、正当なユーザー名とパスワードを要求します。入力されたユーザー名とパスワードは、 WebSphere 管理コンソールを使用して構成

されるアクティブ・ユーザー・レジストリー内で有効でなければなりません。正当なユーザーの信用証明情報が含まれていない要求は、リジェクトされます。

エンタープライズ・サーチ・アプリケーションでは、 Properties オブジェクトが getSearchService メソッドまたは getBrowseService メソッド の呼び出しで渡されます。 Properties オブジェクトは、WebSphere 用に username と password と呼ばれるプロパティー名を指定します。

検索アプリケーション名およびパスワードは、WebSphere 認証に使用するリポジトリーと同じリポジトリーに保管してください。

エンタープライズ・サーチは、HTTP BASIC 認証をサポートします。HTTPS (SSL v2 または v3) に対するサポートはありません。

Java ソース・コードのコンパイル

エンタープライズ・サーチの ESSearchApplication サンプルおよびデータ・リスナーのサンプルは、IBM Software Developer's Kit 1.4.x を使用してコンパイルする必要があります。SIAPI サンプルは、1.3.x または 1.4.x のいずれかでコンパイルできます。IBM Software Developer's Kit 1.5 はサポートされていません。

Java™ ソース・コードを作成する前に、Java ベースのビルド・ツールである Apache ANT をインストールして、構成する必要があります。Apache ANT のインストールおよび構成方法について詳しくは、 http://ant.apache.org/ を参照してください。

ES_INSTALL_ROOT/samples ディレクトリーにある ESSearchApplication は、IBM

SDK バージョン 1.4 でコンパイルし、 JRE バージョン 1.4 環境で実行する必要があります。WebSphere Application Server バージョン 5.1 と WebSphere Portal バージョン 5.1 は、両方とも JRE バージョン 1.4 を提供しています。

Java ソース・コードをコンパイルするには、以下のようにします。

1. コマンド行で、以下のいずれかのディレクトリーに変更する。
   - SIAPI サンプル検索アプリケーションの場合: *ES_INSTALL_ROOT*/samples/siapi (例えば、 Linux™ および AIX® の場合は /opt/IBM/es/samples/siapi、 Windows® の場合は Program Files¥IBM¥es¥samples¥siapi)
   - データ・リスナー・クライアント・アプリケーションの場合: *ES_INSTALL_ROOT*/samples//datalistener (例えば、 Linux および AIX の場合は /opt/IBM/es/samples//datalistener、 Windows の場合は Program Files¥IBM¥es¥samples¥¥datalistener)
   - ESSearchApplication の場合: *ES_INSTALL_ROOT*/samples/ESSearchApplication (例えば、 Linux および AIX の場合は opt/IBM/es/samples/ESSearchApplication、 Windows の場合は Program Files¥IBM¥es¥samples¥ESSearchApplication)

   いずれのディレクトリーにも、ANT がファイルのビルド時に使用する `build.xml` ファイルが含まれています。
2. `ant` と入力して Enter キーを押す。

Java ソース・コードのコンパイル後、以下のメッセージが表示されます。

```
BUILD SUCCESSFUL
Total time: xx seconds
```

SIAPI Javadoc 文書

DB2 Information Integrator OmniFind Edition に添付されている SIAPI Javadoc 文書は、ご使用のエンタープライズ・サーチ・ソリューションに配置できるカスタム検索アプリケーションを作成する際に役立ちます。

検索アプリケーションを作成する際に使用できるアプリケーション・プログラミング・インターフェース (API) のリストは、Javadoc 文書を参照してください。この文書には、データ・リスナー・クライアント・アプリケーション用の API は含まれていません。

Javadoc 文書は、 *ES_INSTALL_ROOT*/docs/api/siapi ディレクトリーにあります。

第 2 章 Search and Index API (SIAPI)

IBM search and index API (SIAPI) は、ユーザーによるコレクションおよび分類法の検索あるいは参照を可能にするプログラミング・インターフェースです。

SIAPI は、ユーザーが 1 つのプログラムを作成するだけで、さまざまな IBM バックエンド検索プロダクトを検索できるようにする、統一プログラミング・インターフェースを提供します。

SIAPI は、次のようなタスクをサポートします。

- 索引の検索
- 検索結果セットに戻される情報のカスタマイズ
- 分類法の検索および参照
- 複数のコレクションを 1 つのコレクションであるかのように検索 (サーチ・フェデレーション)

SIAPI アプリケーションの構造

SIAPI アプリケーションは、検索サーバーに照会を送信し、その照会の結果を戻します。

SIAPI アプリケーションは、以下のタスクで構成されます。

- SIAPI インプリメンテーション・ファクトリー・オブジェクトの取得
- SearchService オブジェクトの取得
- Searchable オブジェクトの取得
- 照会の実行
- 照会結果の処理

SIAPI インプリメンテーション・ファクトリー・オブジェクトの取得

SIAPI ベースの検索アプリケーションでは、まず最初に、インプリメンテーション・ファクトリー・オブジェクトを取得します。

```
SearchFactory factory =
SiapiSearchImpl.createSearchFactory
    ("com.ibm.es.api.search.RemoteSearchFactory");
```

SIAPI は、ファクトリー・ベースの Java API です。検索アプリケーションで使用するすべてのオブジェクトは、 SIAPI オブジェクト・ファクトリー・メソッドを呼び出して作成されるか、またはファクトリー生成オブジェクトのメソッドを呼び出して戻されます。異なるファクトリーをロードすることによって、SIAPI インプリメンテーション間を容易にスイッチすることができます。

エンタープライズ・サーチ SIAPI インプリメンテーションは、com.ibm.es.api.search.RemoteSearchFactory クラスに提供されています。

SearchService オブジェクトの取得

SearchService オブジェクトを取得するには、ファクトリー・オブジェクトを使用します。 SearchService オブジェクトを使用して、検索可能なコレクションにアクセスすることができます。

エンタープライズ・サーチ・システムが複数のサーバーにインストールされている場合、SearchService オブジェクトの構成に、検索サーバーのホスト名とポート、さらに WebSphere グローバル・セキュリティーが使用可能ならば、有効な WebSphere ユーザー名とパスワードを指定する必要があります。

構成パラメーターは java.util.Properties に設定します。パラメーターは、SearchService オブジェクトを生成する getSearchService ファクトリー・メソッドに渡されます。

```
Properties configuration = new Properties();
configuration.setProperty("hostname", "es.mycompany.com");
configuration.setProperty("port", "80");
config.setProperty("username", "websphereUser");
config.setProperty("password", "websperePassword");
SearchService searchService =
 factory.getSearchService(config);
```

Searchable オブジェクト の取得

Searchable オブジェクトを取得するには、 SearchService オブジェクトを使用します。 Searchable オブジェクトは、検索可能なコレクションと関連しています。Searchable オブジェクトを使用して、照会を実行し、関連したコレクションに関する情報を取得することができます。各エンタープライズ・サーチ・コレクションには ID があります。

ユーザーが、 Searchable オブジェクトを要求する際には、アプリケーション ID を使用して、アプリケーションを識別する必要があります。適切なアプリケーション ID については、エンタープライズ・サーチ管理者に問い合わせてください。

```
ApplicationInfo appInfo = factory.createApplicationInfo("my_application_id");
appInfo.setPassword("my_password");
Searchable searchable =
 searchService.getSearchable(appInfo, "some_collection_id");
```

アプリケーションで使用可能なすべての Searchable オブジェクトを取得するには、getAvailableSearchables メソッドを呼び出します。

```
Searchable[] searchables =
 searchService.getAvailableSearchables(appInfo);
```

照会の実行

Searchable オブジェクトを取得したら、その Searchable オブジェクトに対して照会を実行することができます。 Searchable オブジェクトに照会を実行するには、次のようにします。

- Query オブジェクトを作成する。
- Query オブジェクトをカスタマイズする。
- Searchable オブジェクトに対して Query オブジェクトを実行する。
- ResultSet オブジェクトに示されている照会結果を取得する。

```
String queryString = "big apple";
Query query = factory.createQuery(queryString);
query.setRequestedResultRange(0, 10);
ResultSet resultSet = searchable.search(query);
```

照会結果の処理

ResultSet および Result インターフェースによって、照会結果にアクセスできます。

```
Result[] results = resultSet.getResults();
for ( int i = 0 ; i < results.length ; i++ ) {
 System.out.println
( "Result " + i + ": " + results[i].getDocumentID()
  + " - " + results[i].getTitle() );
}
```

SIAPI には、ResultSet インターフェースおよび 個々の Result インターフェース・オブジェクトと対話するためのさまざまなメソッドがあります。

照会動作の制御

QUERY インターフェースに属するメソッドを使用して、照会の処理方法やそれぞれの結果に戻されるメタデータの内容など、あらゆる分野の照会動作を制御することができます。

各メソッドの詳細は、Javadoc 文書を参照してください。

表 1. 照会動作メソッド

メソッド	説明
setLinguisticMode(int mode)	この照会の言語処理モードを設定します。以下のいずれかのモードを設定できます。 • LINGUISTIC_MODE_EXACT_MATCH: 言語処理を行わずに、入力されたままの変更されていない用語が突き合わされます。 • LINGUISTIC_MODE_BASEFORM_MATCH: 言語処理を行った後、変更されていない用語が突き合わされます。 • LINGUISTIC_MODE_ENGINE_DEFINED: エンジンの最善のポリシーに従って、変更されていない用語が突き合わされます。これが、デフォルト・モードです。
setQueryLanguage(java.lang.String lang)	検索サーバーで使用する、コレクション・デフォルト言語以外の言語を指定します。例えば、英語の場合、照会言語パラメーターは en-US です。中国語では、中国語 (簡体字) の場合は zh-CN、中国語 (繁体字) の場合は zh-TW を使用します。

表 1. 照会動作メソッド (続き)

メソッド	説明
setProperty(String name, String value)	検索可能なプロパティーの値を設定します。このメソッドには、以下のモードがあります。 • HighlightingMode: 検索結果詳細のいくつかの領域で、照会用語を強調表示可能にします。値は、次のとおりです。 – DefaultHighlighting: サマリーのみ、照会用語を強調表示します。これは、検索アプリケーションが HighlightingMode プロパティーの設定を省略した場合のデフォルトです。 – ExtendedHighlighting: 照会用語の強調表示を検索結果の他の領域に拡張します。例えば、タイトル、URL、およびその他のフィールドなどです。 • FuzzyNGramSearch: ファジー検索で N-gram コレクションにおける非制限検索の実行を可能にします。このプロパティーはブールで、値は次のとおりです。 – false: 明確な検索が行われます。これは、検索アプリケーションが FuzzyNGramSearch プロパティーの設定を省略した場合のデフォルトです。 – true: ファジー検索が行われます。FuzzyNGramSearch が true に設定されている場合は、 2 番目の関連プロパティー、ProximityWindowSize を設定できます。2 つの照会用語がこのサイズのウィンドウにある場合、文書スコアは隣接値に引き上げられます。このプロパティーの値は、符号なしの整数です。デフォルト値は 5 です。 FuzzyNGramSearch が false に設定されている場合は、ProximityWindowSize を設定できません。 • AllowStopwordRemoval: 照会構文解析時にストップワードを除去するかどうかを判別します。このプロパティーが設定されていない場合、エンジンは、独自のポリシーに従って、ストップワードの除去を適用します。このプロパティーはブールで、値は次のとおりです。 – false: ストップワードは、照会構文解析時に除去されません。 – true: ストップワードは、照会構文解析時に除去されます。

表 1. 照会動作メソッド (続き)

メソッド	説明
setRequestedResultRange(int fromResult, int numberOfResult)	戻される結果の範囲を制御します。 **fromResult** 値は、ランク付けされた文書のうち、どの文書から結果セットを開始するかを制御します。例えば、値 0 は、照会結果の最初の文書を要求していることを意味します。 **numberOfResults** 値は、結果の現行ページにいくつの結果を戻すかを制御します。最大値は 100 です。
setReturnedAttribute(int attributeType, boolean isReturned)	各 Result オブジェクトで戻される定義済み結果属性値を使用可能/使用不可に設定します。 デフォルトでは、エンタープライズ・サーチは、メタデータ・フィールド属性 (RETURN_RESULT_FIELDS) を除く、すべての定義済み結果属性値を戻します。
setReturnedFields(String[] fieldNames)	Result オブジェクトに戻されるメタデータ・フィールドを制御します。 デフォルトでは、エンタープライズ・サーチはメタデータ・フィールドを戻しません。
setSiteCollapsingEnabled(boolean value)	上位の結果に、同じ Web サイトまたはデータ・ソースからの結果を 3 個以上含めるかどうかを指定します。 例えば、特定の照会で http://www.ibm.com から 100 個の結果が戻された場合、サイト縮小が使用可能に設定されていると、ResultSet には、 上位の結果に 100 個の結果のうち 2 個の結果しか含まれません。そのサイトの他の結果は、他のサイトの結果がリストされた後に表示されます。 同じサイトからより多くの結果を検索するには、samegroupas:*result URL* 照会構文を使用するか、または照会ストリングに http://www.ibm.com サイトを追加して、同じ照会を再実行します。詳しくは、 10 ページの『照会構文』を参照してください。
setSortKey(java.lang.String sortKey)	ソート・キーを指定します。以下の事前定義ソート・キー値が、 com.ibm.siapi.search.BaseQuery に定義されています。 • SORT_KEY_NONE • SORT_KEY_DATE • SORT_KEY_RELEVANCE 上記のほかに、検索するコレクションの有効な数値フィールド名をソート・キーとして指定できます。デフォルトのソート・キーは SORT_KEY_RELEVANCE です。

表 1. 照会動作メソッド (続き)

メソッド	説明
setSortOrder(int sortOrder)	SORT_ORDER_ASCENDING または SORT_ORDER_DESCENDING のいずれかのソート順序を指定します。 ソート・キーが SORT_KEY_RELEVANCE または SORT_KEY_NONE の場合には、ソート順序は無視されます。
setSortPoolSize(int sortPoolSize)	上位関連結果のいくつをソートして、結果セットに戻すかを制御します。値は 1 から 500 までです (デフォルトのソート・プール・サイズは 500 です)。それ以外の値を指定すると、検索サーバーによって SiapiException がスローされます。 sortKey が SORT_KEY_RELEVANCE または SORT_KEY_NONE の場合には、ソート・プール・サイズは無視されます。
setPredefinedResultsEnabled (boolean value)	照会結果で、通常の結果のほかに、定義済みリンクを含めるかどうかを指定します。デフォルトでは、定義済みリンクが使用可能になります。
setSpellCorrectionEnabled(boolean enable)	照会結果に照会のスペル修正の提案を含めるかどうかを指定します。デフォルトでは、スペル修正は使用不可になります。
setResultCategoriesDetailLevel (int detailLevel)	照会結果に要求されるカテゴリー詳細レベルを指定します。このメソッドは、カテゴリー属性 (RETURN_RESULT_CATEGORIES) が使用可能な場合に使用されます。デフォルト値は、RESULT_CATEGORIES_ALL です。
setSynonymExpansionMode (int mode)	照会に対して同義語の拡張モードを設定します。以下のいずれかのモードを設定できます。 • SYNONYM_EXPANSION_OFF: この定数を setSynonymExpansionMode メソッドに渡して、照会に同義語演算子が含まれていても、同義語が拡張されないようにします。 • SYNONYM_EXPANSION_MANUAL: この定数を setSynonymExpansionMode メソッドに渡して、同義語演算子による影響を受ける照会用語についてのみ同義語を拡張します。 • SYNONYM_EXPANSION_AUTOMATIC: この定数を setSynonymExpansionMode メソッドに渡して、*最大限の努力* をして、該当するすべての照会用語を拡張します。
setACLConstraints(java.lang.String aclConstraints)	セキュアな検索を行うために、この照会にアクセス制御に関する制約を設定します。

照会構文

照会で特定の文字を使用することにより、検索結果を絞り込むことができます。

単純照会構文文字

以下に、照会結果を絞り込むために検索アプリケーションで使用できる文字を示します。

フリー・スタイル照会構文

フリー・スタイル照会構文は、明示的な変換処理が無い照会を記述する際に使用します。この照会には、通常、正符号 (+)、曲折アクセント記号 (^)、負符号 (-) が付いておらず、デフォルトの動作が定義されていない用語が含まれます。

照会: `computer software`

結果: この照会は、 *computer* または *software* という用語、その両方、あるいはインプリメンテーションのセマンティクスによってはそれ以外の用語を含む文書を戻します。

~ (接尾部)

用語の後に波形記号 (~) を付けると、言語学上の基本型が照会用語と同じである用語 (見出し語または語幹ともいいます) が含まれている文書を検索することを示します。

照会: `apples~`

結果: この照会は、*apples* または *apple* という用語が含まれている文書を検索します (apple は apples の基本型であるため)。

+

用語の前に正符号 (+) を付けると、その用語に一致する用語が含まれている文書を検索することを示します。

照会: `+computer +software`

結果: この照会は、 *computer* と *software* という用語が含まれている文書を戻します。

–

用語の前に負符号 (-) を付けると、その用語が含まれていない文書を検索することを示します。

照会: `computer -hardware`

結果: この照会は、 *computer* という用語が含まれており、*hardware* という用語が含まれていない文書を戻します。

^

用語の前に曲折アクセント記号 (^) を付けると、曲折アクセント記号 (^) を付けて指定した用語と、曲折アクセント記号を付けずに指定した用語が少なくとも 1 つ以上含まれている文書を検索することを示します。

照会: `^computer science software`

結果: この照会は、 `computer` と *science* または *software* という用語が含まれている文書を戻します。

照会: `cats dogs ^$language::en ^$doctype::html`

結果: この照会は、 *cats* および *dogs* という用語のうち、少なくとも 1 つが含まれている英語の HTML 文書を戻します。

*****

用語の後にワイルドカード記号 (*) を付けると、この用語で始まる単語を含む文書を検索することを示します。

照会: `app*`

結果: この照会は、 *apple*、*application* などの用語が含まれている文書を検索します (これらは、すべて *app* で始まる語であるため)。

( )
括弧 ( ) を使用すると、括弧内の用語のうち 1 つ以上が含まれる文書を 検索することを示します。

括弧内には、正符号 (+)、負符号 (-)、曲折アクセント記号 (^) を使用しないでください。

括弧内の 用語は、OR や垂直バー (|) で区切ります。

照会: `+computer (hardware OR software)`

照会: `+computer (hardware | software)`

結果: 上記の照会は、両方とも、*computer* という用語と、 *hardware* または *software* という用語が少なくとも 1 つ以上含まれている文書を検索します。

OR 条件は、必要条件 (+) または必要不十分条件 (^) にすることができますが、 禁止条件 (-) にすることはできません。これによって、照会言語機能が制限されることはありません。 `-(dogs OR cats)` は、`-dogs -cats` と表すことができます。

OR 条件は、デフォルトでは必要条件 (+) になります。よって、前述の照会は、 `+computer +(hardware | software)` と同じになります。

″ ″
二重引用符 (″) を使用すると、二重引用符内に指定した句とまったく同じ句が含まれている文書を検索することを示します。

照会: `"computer software programming"`

結果: この照会は、`computer software programming` とまったく同じ句が含まれている文書を検索します。

デフォルトでは、句は必要条件になります。したがって、`building "new york"` および `building +"new york"` という 2 つの照会は同じになります。句は、禁止条件 (-) および必要不十分条件 (^) にすることもできます。

句の中の語には、見出し語処理は行われません。

~ (接頭部)
用語の前に波形記号 (~) を付けると、その用語またはその用語のシノニム (同義語) が含まれている文書を検索することを示します。

照会: `~fort`

照会: この照会は、 `fort` という用語またはそのシノニム (`garrison` や `stronghold` など) が含まれている文書を検索します。

=
用語の前に等号 (=) を付けると、その用語に完全一致する用語が含まれている文書を検索することを示します (見出し語処理はできません)。

照会: `=apples`

結果: この照会は、 複数形 `apples` が含まれている文書のみを戻します。

site:*text*
Web コンテンツが含まれているコレクションを検索する場合に、 `site` キーワードを使用して、特定のドメインを検索します。例えば、特定の Web サイトの全ページを戻すことができます。

サイト照会では、接頭部に http:// は付けないでください。

照会: +laptop site:www.ibm.com

結果: この照会は、laptop という用語が含まれている、www.ibm.com ドメイン上のすべての文書を検索します。

url:*text*

Web コンテンツが含まれているコレクションを検索する場合に、 url キーワードを使用して、URL の一部に特定の語が含まれている文書を検索します。

照会: url:support

結果: この照会は、 URL の一部に support という値が含まれている文書 (http://www.ibm.com/support/fr/ など) を検索します。

link:*text*

Web コンテンツが含まれているコレクションを検索する場合に、 link キーワードを使用して、特定の Web ページへのハイパーテキスト・リンクが少なくとも 1 つ以上含まれている 文書を検索します。

照会: link:http://www.ibm.com/us

結果: この 照会は、ページ http://www.ibm.com/us へのリンクが 1 つ以上含まれている文書をすべて検索します。

*field***:***text*

コレクション内の文書にフィールド (または列) が含まれており、コレクション管理者がこれらのフィールドをフィールド名によって検索可能にしている場合に、コレクション内の特定のフィールドを照会できます。

照会: lastname:smith div:software

結果: この照会は、 ソフトウェア部門 (div:software) で働いており、 ラストネームが Smith (lastname:smith) である従業員に関する文書をすべて検索します。

docid:*documentid*

特定の URI (または文書 ID) の文書を検索するには、docid キーワードを使用します。通常、コレクション内で特定の URI に一致する文書は多くても 1 つしかありません。

照会: (docid:http://www.ibm.com/solutions/us/ OR docid:http://www.ibm.com/products/us/)

結果: この照会は、URI が http://www.ibm.com/solutions/us/ または http://www.ibm.com/products/us/ のすべての文書を検索します。

samegroupas:*URI*

デフォルトで、エンタープライズ・サーチは、ホスト名が同じ URL は同じグループに属しているとみなします。また、同じスレッドのニュース記事は同じグループに属しているとみなします。その他すべてのデータ・ソースの URI の場合、 各 URI は独自のグループを形成します。ただし、エンタープライズ・サーチを使用して、特定の接頭部に一致する URI をグループ化することができます。例えば、以下のグループ定義を構成することができます。

```
http://mycompany.server1.com/hr/         hr
http://mycompany.server2.com/hr/         hr
http://mycompany.server3.com/hr/         hr
http://mycompany.server1.com/finance/    finance

file:///myfileserver1.com/db2/sales/      sale
file:///myfileserver1.com/websphere/sales/  sale
file:///myfileserver2.com/db2/sales/      sale
file:///myfileserver2.com/websphere/sales/  sale
```

この例では、接頭部が http://mycompany.server1.com/hr/ または http://mycompany.server2.com/hr/ または http://mycompany.server3.com/hr/ の URI は、すべて *hr* という 1 つのグループに属することになります。接頭部が http://mycompany.server1.com/finance/ の URI は、すべて別のグループ *finance* に属することになります。さらに、接頭部が file:///myfileserver1.com/db2/sales/ または file:///myfileserver1.com/websphere/sales/ または file:///myfileserver2.com/db2/sales/ または file:///myfileserver2.com/websphere/sales/ の URI は、すべて *sale* という別のグループに属することになります。コレクション内の URI が file:///myfileserver2.com/websphere/sales/mypath/mydoc.txt である場合、 検索条件として以下を指定した照会 (この照会は、1 行にする必要があります)

```
samegroupas:file:///myfileserver2.com/websphere/sales/mypath/mydoc.txt
```

は、検索を *sale* グループ内の URI に限定します。この照会のすべての結果には、以下のいずれかの接頭部が付きます。

```
file:///myfileserver1.com/db2/sales/
file:///myfileserver1.com/websphere/sales/
file:///myfileserver2.com/db2/sales/
file:///myfileserver2.com/websphere/sales/
```

照会: `samegroupas:http://www.ibm.com/solutions/us/`

結果: この照会は、`http://www.ibm.com/solutions/us/` と同じグループに属する URI (この場合は URL) の文書をすべて検索します。

taxonomy_ID::*category_ID*

エンタープライズ・サーチ用に作成されたカテゴリーが含まれているコレクションを検索する場合に、特定の分類法の特定のカテゴリーに属する文書を検索することができます。エンタープライズ・サーチは 2 つのタイプのカテゴライザーをサポートしており、各カテゴライザーは固有の分類法 ID を持っています。

カテゴリー ID を検索するには、*ES_NODE_ROOT*/*col_xyz*.parserdriver/CategoryTree.xml ディレクトリーに移ります。 CategoryTree.xml ファイルにカテゴリー ID が含まれています。

分類法 ID

rulebased

この分類法 ID を使用して、文書のカテゴリー化に文書コンテンツ規則または文書 URI 規則を使用しているカテゴリーに属する文書を検索します。ルール・ベースのカテゴリーの構成については、「*DB2 Information Integrator OmniFind Edition エンタープライズ・サーチの管理*」を参照してください。

照会:`rulebased::c1`

結果: この照会は、 c1 というルール・ベースのカテゴリー ID に属している文書を戻します。

modelbased

この分類法 ID を使用して、WebSphere Portal 分類法に定義されているカテゴリーに属する文書を検索します。モデル・ベースのカテゴリーまたは分類法のマイグレーションおよび使用については、「*DB2 Information Integrator OmniFind Edition エンタープライズ・サーチの管理*」を参照してください。

照会:`modelbased::c3`

結果: この照会は、 c3 というモデル・ベースのカテゴリー ID に属している文書を戻します。

taxonomy_ID`::`*category_ID* 照会用語は、指定された *category_ID* またはそのサブカテゴリーに属する文書を検索します。これは、分類法 ID の前に波形記号 (~) を付けることによって、明示的に指定できます。

照会で、指定されたカテゴリーに属する文書を戻し、そのサブカテゴリーに属する文書を戻さない場合は、 *taxonomy_ID::category_ID* の前に等号 (=) をつけて、 `=rulebased::c1` のように指定します。

scopes::*scope_name*

有効範囲 (索引内の URI の範囲) に属する文書を検索するには、有効範囲名を使用します。有効範囲の構成については、「*DB2 Information Integrator OmniFind Edition エンタープライズ・サーチの管理*」を参照してください。

照会:`scopes::research "computer science"`

結果: この照会は、 `computer science` という句が含まれている research という有効範囲に属する文書を戻します。

$source::*source_type*

特定のデータ・ソース・タイプの文書を検索するには、`$source` キーワードを使用します。ソース照会は、複数のソースの文書が含まれているコレクションにおいて便利です。

コレクションに有効なソース・タイプのリストを入手するには、そのコレクションの検索可能オブジェクトの getAvailableAttributeValues(Searchable ATTRIBUTE_SOURCE) メソッドを呼び出します。

照会: `$source::DB2 "computer science"`

結果: この照会は、`computer science` という句が含まれている、DB2 クローラーによってコレクションに追加された文書を検索します。

$language::*language_id*

特定の言語で書かれている文書を検索するには、`$language` キーワードを使用します。

コレクションに有効な言語 ID のリストを入手するには、そのコレクションの検索可能オブジェクトの getAvailableAttributeValues(Searchable.ATTRIBUTE_LANGUAGE) メソッドを呼び出します。

照会: `$language::en "computer science"`

結果: この照会は、`computer science` という句が含まれている英語の文書を検索します。

$doctype::*document_type*

特定の文書フォーマットまたは MIME タイプの文書を検索するには、`$doctype` キーワードを使用します。

コレクションに有効な文書タイプのリストを入手するには、そのコレクションの検索可能オブジェクトの getAvailableAttributeValues(Searchable.ATTRIBUTE_DOCTYPE) メソッドを呼び出します。

照会: `$doctype::application/pdf "computer science"`

結果: この照会は、`computer science` という句が含まれている Portable Document Format (PDF) 文書を検索します。

#*field***::=***value*

指定した数値に等しい値が数値フィールドにある文書を検索するには、パラメトリック制約構文を使用します。

照会: `#price::=1700 laptop`

結果: この照会は、`laptop` および `price` フィールドの値が 1700 である文書を検索します。

#*field***::>***value*

指定した数値より大きい値が数値フィールドにある文書を検索するには、パラメトリック制約構文を使用します。

照会: `#price::>1700 laptop`

結果: この照会は、`laptop` および `price` フィールドの値が 1700 より大きい文書を検索します。

#*field***::<***value*

指定した数値より小さい値が数値フィールドにある文書を検索するには、パラメトリック制約構文を使用します。

照会: `#price::<1700 laptop`

結果: この照会は、`laptop` および `price` フィールドの値が 1700 より小さい文書を検索します。

#*field***::>=***value*

指定した数値より大きいか等しい値が数値フィールドにある文書を検索するには、パラメトリック制約構文を使用します。

照会: `#price::>=1700 laptop`

結果: この照会は、`laptop` および `price` フィールドの値が 1700 以上である文書を検索します。

#*field*::<=*value*

指定した数値より小さいか等しい値が数値フィールドにある文書を検索するには、パラメトリック制約構文を使用します。

照会: `#price::<=1700 laptop`

結果: この照会は、`laptop` および `price` フィールドの値が 1700 以下である文書を検索します。

#*field*::>*value1*<*value2*

指定した数値の範囲内の値 (指定した数値を含まない) が数値フィールドにある文書を検索するには、パラメトリック制約構文を使用します。

照会: `#price::>1700<3900 laptop`

結果: この照会は、`laptop` および `price` フィールドの値が 1700 より大きく 3900 より小さい文書を検索します。

#*field*::>=*value1*<=*value2*

指定した数値の範囲内の値 (指定した数値を含む) が数値フィールドにある文書を検索するには、パラメトリック制約構文を使用します。

照会: `#price::>=1700<=3900 laptop`

結果: この照会は、`laptop` および `price` フィールドの値が 1700 以上 3900 以下の文書を検索します。

#*field*::>*value1*<=*value2*

指定した数値の範囲内の基準に一致する値が数値フィールドにある文書を検索するには、パラメトリック制約構文を使用します。

照会: `#price::>1700<=3900 laptop`

結果: この照会は、`laptop` および `price` フィールドの値が 1700 より大きく 3900 以下の文書を検索します。

#*field*::>=*value1*<*value2*

指定した数値の範囲内の基準に一致する値が数値フィールドにある文書を検索するには、パラメトリック制約構文を使用します。

照会: `#price::>=1700<3900 laptop`

結果: この照会は、`laptop` および `price` フィールドの値が 1700 以上で 3900 より小さい文書を検索します。

ACL constraints: (*security_tokens*)

セキュリティーに関して、照会ストリングにアクセス制御制約を指定することはできません。照会のアクセス制御制約を指定するには、QUERY インターフェースの setACLConstraints(String aclConstraints) メソッドを使用します。 括弧、正符号 (+)、負符号 (-)、曲折アクセント記号 (^)、 XML セキュリティー・コンテキスト・ストリングを ACL 制約ストリング (@SecurityContext::'securityContext') に指定することができます。securityContext ストリング構文については、setACLContstraints メソッド について記載されている Javadoc 文書を参照してください。各シンボルの意味は、前述の構文記述に示されているものと同じです。

setACLConstraints メソッドにおける ACL 制約ストリング : `michelle_c | dev_group`

setACLConstraints メソッドにおける ACL 制約ストリング: `michelle_c @SecurityContext::'securityContext'`

照会: `thinkpad`

結果: この照会は、最初のケースでは `thinkpad` という用語と セキュリティー・トークン `michelle_c` または `dev_group` が含まれている文書を、2 番目のケースでは `thinkpad` という用語と `michelle_c` および指定されたセキュリティー・コンテキスト制約が含まれている文書を検索します。

不透明条件の照会構文文字

エンタープライズ・サーチを使用して、2 つのタイプの不透明条件に関する照会構文を作成することもできます。不透明条件を使用すると、照会の一部を XML フラグメントおよび XPath などの他の言語で表すことができます。 XML フラグメントおよび XPath は、XML 照会言語の 2 つのタイプです。XML フラグメントは、UIMA 構造の照会にも使用できます。不透明条件の記号は、*@xmlf2::* で表されます。 XML フラグメントは、単一引用符 (’ ’) で囲みます。

xmlf2 表記は XML フラグメントに、*xmlxp* 表記は XPath 条件に使用されます。不透明条件の 構文は @syntax_name::’value’ のようになります。式は @ 記号で始まり、 構文名 (xmlf2 または xmlxp)、2 つのコロン (::)、および単一引用符 (’ ’) で囲まれた値の順になります。’value’ の前に +、–、^ が付くことがあります。式の値セクションで単一引用符を使用する必要がある場合には、円記号 (¥) を使用して単一引用符を拡張します。例えば、¥’ のようにします。

@xmlf2::<tag1><.depth value=’$number’><tag2> ... </tag2></.depth></tag1> (または <.depth value=″$number″>))

この照会構文は、tag1 から $number レベル下の tag2 のオカレンスを探します。 depth は、照会において、語 w またはタグ t と、それに最も近いタグ t’ との間のレベル数を表します。

$number は正の整数です。数値を単一引用符 (’ ’) または二重引用符 (″ ″) で囲むことができます。 この照会構文は UIMA には適用されません。

照会:
```
@xmlf2::'<author>Albert Camus<.depth
value='1'><publisher>Carey Press</publisher></.depth></author>'
```

結果: この 照会は、author より 1 レベル下の publisher の文書を検索します。以下のような XML エレメントを持つ文書があるとします。

```
<author>Albert Camus<ISBN>002-12345</ISBN><country>USA
<publisher>Carey Press</publisher></country><author>
```

この文書は、上記の照会例では戻されません。 publisher (<publisher>) エレメントが author (<author>) エレメントより 2 レベル下にあるからです。

@xmlf2::’<tag1><@tag1> ... </@tag1></tag1>’

エレメントと属性を区別することができます。属性は、エレメント内に明示的に指定するか、または @ 記号の後にサブエレメントとして指定します。これによって、同じ名前である可能性があるエレメントと属性を区別することができます。

これらのタグ内に他のタグ、語、句を定義できます。これらの語または句は、通常の照会での用語と同じ機能をサポートします。

照会: `@xmlf2::'<author country="USA"></author>'`

照会: `@xmlf2::'<author><@country>USA</@country></author>'`

結果: この照会は、USA 出身の author が含まれている文書を検索します。

照会:

```
@xmlf2::'<author><@country>USA</@country>
<firstName>Michelle</firstName>
<lastName>Ropelatto</lastName></author>'
```

結果: この 照会は、USA 出身の author 名 Michelle Ropelatto が含まれている文書を検索します。

@xmlf2::'+text1 ... +text2 -text3 ... -text4 text5'

語または句の前に、正符号 (+) または負符号 (-) を使用します (通常、引用符 (″ ″) で囲みます)。各照会レベルで (エレメントの下のデータによって、新しくネストされた照会レベルが作成されます)、 ″+″ はその用語が含まれていなければならないことを意味し、 ″-″ はその用語が含まれていてはならないことを意味します。それ以外の用語はオプションであり、ランキング情報にのみ寄与します。 ″+″ が付いた用語が無い場合、少なくとも 1 つ以上のオプションの用語がなければなりません。

照会: `@xmlf2::'+"Graph Theory" -network'`

結果: この照会は、`Graph Theory` という句が含まれており、network という用語が含まれていない文書を検索します。

@xmlf2::'<tag1> <.or> ... </.or> <.and> ... </.and> </tag1>'

フラグメント照会で、AND (<.and>) および OR (<.or>) スコープを表すために絶対ブールを使用します。

照会: `@xmlf2::'<book><.or><author>Sylvia Plath</author><title>XML -Microsoft</title></.or></book>'`

結果: この照会は、book の author が Sylvia Plath であるか、 または book の title に XML という語が含まれており Microsoft という語が含まれていない文書を検索します。

@xmlf2::'<annotation1+annotation2> ... </annotation1+annotation2>'

フラグメント照会で、エレメントの開始タグと終了タグの間に正符号 (+) を使用することで、連続する注釈の連結を表すことができます。連続する注釈は、少なくとも 1 語はオーバーラップしていなければなりません (交差していなければなりません)。複数の重複した注釈の連結は、個々の注釈によってカバーされるテキスト全体をカバーする新規の仮想注釈になります。

照会: `@xmlf2::'<Report+HoldsDuring> +Pakistan +March +Reuters</Report+HoldsDuring>'`

結果: この 照会は、Report と HoldsDuring によって形成される連結注釈に含まれている、 3 月 (March) にパキスタン (Pakistan) で発生した事件に関するロイター (Reuters) の文書を検索します。

@xmlf2::'<annotation1*annotation2> ... </annotation1*annotation2>'

エレメントの開始タグと終了タグの間にアスタリスク記号 (*) を使用するこ

とで、フラグメント照会において注釈の交差を表すことができます。複数の重複した注釈の交差は、重複した注釈の交差によってカバーされるテキストのみをカバーする新規の仮想注釈になります。

照会: `@xmlf2::'<Inhibits* Activates>Aspirin</Inhibits*Activates>'`

結果: この照会は、「Inhibits」と「Activates」の両方の注釈に Aspirin という用語がある文書を検索します。

@xmlxp::'/tag1/@tag1'
エレメント (XML 開始および終了タグ) と属性を区別することができます。 属性を @ 記号の後に明示的に指定します。これによって、同じ名前である可能性があるエレメントと属性を区別することができます。

照会: `@xmlxp::'/author[@country="USA"]'`

結果: この照会は、 author が USA 出身である文書を検索します。

@xmlxp::'/tag1[tag2 or tag3 and tag4]'
XPath 照会で AND および OR スコープを表すには、絶対ブールを使用します。

照会: `@xmlxp::'book[author ftcontains("Jose Perez") or title ftcontains("XML -Microsoft")]'`

結果: この 照会は、book の author が Jose Perez であるか、 または book の title に XML という語が含まれており Microsoft という語が含まれていない文書を検索します。

@xmlxp::'tag1//tag2/tag3'
下層ノード (//) と下位ノード (/) を区別することができます。

照会: `@xmlxp::'/books//book/name'`

結果: この照会は、book エレメントが books エレメントの子孫で、 name エレメントが book エレメントの直接の子である文書を検索します。

@xmlf2::'text1 <tag1> ... </tag1>'
フラグメント照会を新規 SIAPI 不透明条件として表すには、@xmlf2:: 接頭部を使用して、 照会を単一引用符で囲みます。

照会: `@xmlf2::'<title>"Data Structures"</title>'`

結果: この照会は、 *title* という索引付き注釈のスパン内に *Data Structures* という句が含まれている文書を検索します。

@xmlxp::'tag1/.../tagn'
XPath 照会を SIAPI 不透明条件として表すには、@xmlxp:: 接頭部を使用して、 照会を単一引用符で囲みます。

照会: `@xmlxp::'books[booktitle ftcontains("Data Structures")]'`

結果: この照会は、「title」という索引付き注釈のスパン内に「Data Structures」という句が含まれている文書を検索します。

SIAPI フェデレーター

フェデレーターを使用して、異機種間で検索可能なコレクションにフェデレーテッド・サーチ要求を発行したり、統一された文書結果セットを入手します。

フェデレーターは、サービスのリクエスターとそのサービスを実行するエージェントとの間に介在する仲介コンポーネントです。フェデレーターは、単一要求から生成される多数の検索を管理するために設計された、特殊な目的のリソース・コーディネーターです。

エンタープライズ・サーチは、以下の 2 つのタイプの SIAPI フェデレーターを提供します。

- Local Federator
- Remote Federator

Local Federator および Remote Federator は、SIAPI 検索可能オブジェクトです。複数レベルのフェデレーションが可能ですが、フェデレーションのレベルが多すぎると検索のパフォーマンスが低下します。

Local Federator は、SIAPI SearchFactory クラスの createLocalFederator メソッドを使用して作成します。照会対象となる一連の検索可能コレクションは、フェデレーターの作成時に指定します。検索呼び出し時に、検索可能オブジェクトのサブセットを指定することもできます。

Remote Federator は、サーバー上で稼動し、サーバーのリソースを消費します。Remote Federator では、入力コレクション ID を一致する検索可能オブジェクトにマップする際に、追加のステップが必要です。

Local Federator

Local Federator は、検索可能な一連のオブジェクトをフェデレート (統合) するクライアント・サイドのフェデレーターです。

LocalFederator を作成する前に、SIAPI 検索可能オブジェクトを作成または検索する必要があります。これは、通常、SIAPI SearchFactory を使用します。LocalFederator に渡される SIAPI 検索可能オブジェクトは、追加情報を使用せずに検索できる状態でなければなりません。 Local Federator は、検索可能オブジェクトを使用して、フェデレーテッド・サーチ要求を発行します。この要求を完了するには、さまざまな検索可能オブジェクトを使用するために必要なすべてのソフトウェア・コンポーネントが Local Federator 環境になければなりません。

以下のコードの断片は、LocalFederator の作成方法および検索要求の発行方法を示しています。

```
Searchable[] finalSearchables;

// create searchables

// create a query and set query options
Query query = searchFactory.createQuery(queryString);
query.setRequestedResultRange(0, 100),
query.setQueryLanguage("en_US");
query.setSpellCorrectionEnabled(true);
query.setPredefinedResultsEnabled(true);

// create the local federator and call search
LocalFederator federator =
 factory.createLocalFederator(finalSearchables);
ResultSet rs = federator.search(query);
```

Remote Federator

Remote Federator は、検索可能な一連のオブジェクトをフェデレート (統合) するサーバー・サイドのフェデレーターです。

RemoteFederator は、SIAPI AdminService インターフェースを使用して作成します。RemoteFederator の構成時に、コレクション ID のセットが渡されます。 コレクション ID は、RemoteFederator によって SIAPI 検索可能オブジェクトに内部的にマップされます。 Remote Federator 環境では、Remote Federator をアクセス可能にする小規模なプロキシー以外に、検索可能な関連ソフトウェア・コンポーネントは必要ありません。

各検索アプリケーションには、固有のフェデレーターがあるため、フェデレーター ID は ApplicationInfo ID の値と同じ値です。

以下のコードの断片は、RemotelFederator の作成方法および検索要求の発行方法を示しています。

```
// get collection IDs
String[] collectionIDs;

// create a query object
Query query = searchFactory.createQuery(queryString);
query.setRequestedResultRange(0, 100),
query.setQueryLanguage("en_US");
query.setSpellCorrectionEnabled(true);
query.setPredefinedResultsEnabled(true);

// create a remote federator
RemoteFederator federator = getFederator(appinfo, appinfo.getID());
// search
ResultSet rs = federator.search(query);
```

第 3 章 サンプル SIAPI アプリケーション

SIAPI には、単純検索アプリケーションや拡張検索アプリケーションの作成方法を示すいくつかのサンプル・アプリケーションが含まれています。

以下のサンプル・アプリケーションには、さまざまな検索タスクの実行方法が示されています。

単純検索
SearchExample クラスは、検索サーバーに検索をサブミットするのに必要な最小必要要件の簡単な例を提供します。

基本的な分類法ブラウズおよびナビゲーション
BrowseExample クラスは、コレクションの分類法ツリーにアクセスし、基本的なナビゲーション・プロパティーをいくつか表示する例を提供します。

すべての検索結果 の取得
このサンプル・アプリケーション (コードの断片) は、ソートされていない結果を戻し、照会結果をループする照会の設定方法を示します。

フェデレーテッド・サーチ
FederatedSearchExample クラスは、検索サーバーにフェデレーテッド・サーチをサブミットするのに最低限必要なタスクの簡単な例を提供します。

拡張検索
AdvancedSearchExample クラスは、拡張照会設定および結果処理オプションの使用例を提供します。

サンプル SIAPI 検索アプリケーションのコンパイル

サンプル SIAPI 検索アプリケーションは、Ant スクリプトを実行してコンパイルします。

SIAPI サンプル・アプリケーションをコンパイルして実行するには、以下のようにします。

1. 以下のディレクトリーに変更する。(これらは、デフォルトのインストール・ディレクトリーです。)

   Linux および AIX: /opt/IBM/es/samples/siapi

   Windows: C:¥Program Files¥IBM¥es¥samples¥siapi
2. Ant スクリプトを実行する。
3. 以下のコマンドを実行して、アプリケーションを実行する。

   Linux および AIX:

   ```
   java -classpath ES_INSTALL_ROOT/lib/esapi.jar:ES_INSTALL_ROOT>/lib/siapi.jar:.
   SearchExample
   ```

   Windows:

```
java -classpath "ES_INSTALL_ROOT¥lib¥esapi.jar;ES_INSTALL_ROOT>¥lib/siapi.jar;."
SearchExample
```

コマンド行から以下のコマンドを 実行することにより、クラスを実行することができます。

Linux および AIX:

```
java -classpath ES_INSTALL_ROOT/lib/esapi.jar:ES_INSTALL_ROOT>/lib/siapi.jar
SearchExample
```

Windows:

```
java -classpath ES_INSTALL_ROOT¥lib¥esapi.jar;ES_INSTALL_ROOT>¥lib¥siapi.jar
SearchExample
```

SearchExample をコンパイルする他のサンプル検索アプリケーションに置き換えてください。

単純サンプル検索アプリケーションと拡張サンプル検索アプリケーション

SearchExample クラスは、検索サーバーに検索照会を実行するのに最低限必要なアプリケーションの簡単な例を提供します。 AdvancedSearchExample クラスは、簡単な例と同じタスクを示していますが、 ResultSet の一部の値だけでなく、すべての値を印刷します。

単純サンプル・アプリケーションでは、以下の方法が示されています。

- サービスへのアクセス
- コレクションの指定
- アプリケーションの指定
- 照会の実行
- 戻される結果の処理

拡張サンプル・アプリケーションは、単純サンプルと同じタスクを実行しますが、戻された結果の処理が単純サンプルとは異なります。

単純サンプル・アプリケーション (SearchExample.java) と拡張サンプル・アプリケーション (AdvancedSearchExample.java) は、以下のディレクトリーにあります。

- Linux および AIX: /opt/IBM/es/samples/siapi
- Windows: C:¥Program Files¥IBM¥es¥samples¥siapi

ブラウズおよびナビゲーション・サンプル・アプリケーション

BrowseExample クラスは、コレクションの分類ツリーにアクセスし、一部の基本ナビゲーション・ プロパティーを表示するサンプル・アプリケーションを提供します。

このサンプルでは、以下の方法が示されています。

- ブラウズ・ファクトリーの取得
- ブラウズ・サービスの取得
- ブラウザー参照の取得

- ルート・カテゴリーの検出および表示
- ルートの最初の子カテゴリーの検出
- 子カテゴリーおよびそのルートからのパスの表示

サンプル BrowseExample.java アプリケーションは、以下のディレクトリーにあります。

- Linux および AIX: /opt/IBM/es/samples/siapi
- Windows: C:¥Program Files¥IBM¥es¥samples¥siapi

すべての検索結果取得のサンプル

このサンプル・コードは、ソートされていない結果を戻し、照会結果をループする照会の設定方法を示します。 照会に対して、ソートされた結果は最大500 個のみ得ることができます。 ただし、ソートされていない結果はすべて得ることができます。

以下のサンプル・コードでは、次の方法が示されています。

- SearchFactory および Searchable オブジェクトの取得
- 新規照会オブジェクトの作成
- ソートされていない結果を戻す照会の設定
- 検索の実行

SearchFactory および Searchable オブジェクトの取得

24 ページの『単純サンプル検索アプリケーションと拡張サンプル検索アプリケーション』 の例で示されているように、SearchFactory および Searchable オブジェクトを取得します。

```
SearchFactory factory;
Searchable searchable;

... // obtain a SearchFactory and Searchable object
```

新規照会オブジェクトの作成

```
Query q = factory.createQuery("big apple");
```

ソートされていない結果を戻す照会の設定

```
q.setSortKey(Query.SORT_KEY_NONE);
```

検索の実行

照会をループして実行し、一度に 1 ページ分の結果を取得します。エンタープライズ・サーチで許可される最大結果ページ・サイズは 100 です。

結果ページを受け取る際に、ソートされた照会結果の場合とは異なる方法で、getAvailableNumberOfResults メソッドと getEstimatedNumberOfResults メソッドを解釈する必要があります。

- エンタープライズ・サーチは、ソートされていない結果に対する結果数の見積もりを提供しないため、 getEstimatedNumberOfResults メソッドは常に 0 を戻します。
- getAvailableNumberOfResults メソッドは、これが最終結果ページである場合には 0、さらに結果が続く場合は 1 を戻します。
- getResults メソッドによって戻される配列の長さで、この結果ページ内にいくつの結果があるかがわかります。

```
int fromResult = 0;
int pageSize = 100;
boolean moreResults = true;

// loop over query results, pageSize results at a time
while (moreResults) {

 // set the result range for the next page of results
 q.setRequestedResultRange(fromResult, pageSize);

 // execute the search
 ResultSet resultPage = s.search(q);

 // loop over the results from the ResultSet
 Result[] results = resultPage.getResults();
 for (int i=0;i<results.length;i++) {
... // process result
 }

 // check if there are more available results
 moreResults = (resultPage.getAvailableNumberOfResults() == 1);

 // modify the range for getting the next page of results
 fromResult += pageSize;
}
```

フェデレーテッド・サーチ・サンプル・アプリケーション

FederatedSearchExample クラスは、検索サーバーにフェデレーテッド・サーチ をサブミットするのに最低限必要なタスクの簡単な例を提供します。

FederatedSearchExample アプリケーションでは、以下の方法が示されています。

- フェデレーター ID を使用した RemoteFederator オブジェクトの取得。 この ID は、ApplicationInfo オブジェクト ID と同じです。
- 新規照会オブジェクトの作成
- 結果範囲の設定
- RemoteFederator オブジェクトのデフォルト search メソッドの呼び出しによる検索の実行

FederatedSearchExample.java ファイルは、以下のディレクトリーにあります。

- Linux および AIX: /opt/IBM/es/samples/siapi
- Windows: C:¥Program Files¥IBM¥es¥samples¥siapi

第 4 章 データ・リスナー

データ・リスナー・アプリケーション・プログラミング・インターフェース (API) を使用して、データ・リスナーにデータを送信するクライアント・アプリケーションを作成します。

データ・リスナーは、クライアント・アプリケーションからの要求を受け入れる、エンタープライズ・サーチ・コンポーネントです。要求を受けて、コレクションにデータを追加したり、コレクションからデータを除去します。通常、1 つのコレクションに対して 1 つ以上のクローラーを作成します。これらのクローラーは、Web や DB2 Content Manager データベースなどの特定のデータ・ソースからデータを検索します。データ・リスナー・クライアント・アプリケーションを使用すると、クローラーを作成せずに、コレクションにページを追加することができます。また、コレクションから Uniform Resource Identifier (URI) を除去したり、コレクションの Web クローラーに対して Uniform Resource Locator (URL) へのアクセスまたは再アクセスを指示することもできます。

図 1 に、データ・リスナー・クライアント・アプリケーションを使用しない場合のエンタープライズ・サーチ・システムの検索動作の概要を示します。ユーザーは検索コンポーネントに照会を実行し、索引は定期的に検索コンポーネント内のデータをリフレッシュします。 (これらのコンポーネントは、1 つのサーバー上にあっても、複数の接続されたサーバー上にあってもかまいません。このトピックの説明では、複数のサーバー・システムを使用していることを想定しています。)

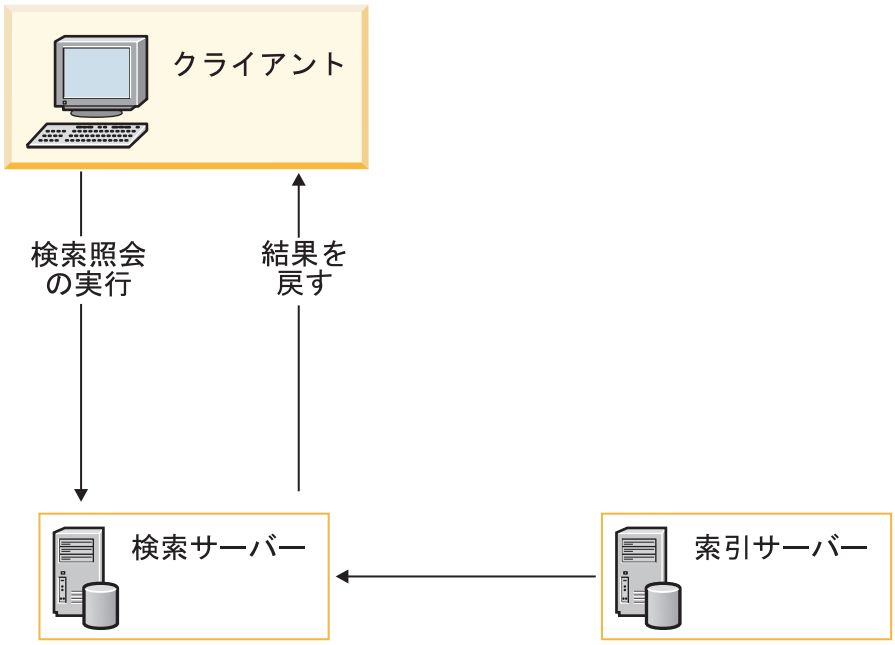

図 1. 照会はエンタープライズ・システムに どのように送信されるか

データ・リスナー・クライアント・アプリケーションを使用して、データ・リスナーに要求をサブミットすることができます。要求のタイプに応じて、データ・リスナーは、後続処理のために、その要求を他のエンタープライズ・サーチ・コンポーネントに送信します。

クライアント・アプリケーションがデータ・リスナーに接続すると、データ・リスナーは、指定されたパスワードがクライアントID およびパスワードに対して有効であるか検査します。また、データ・リスナーは、クライアント・アプリケーションが、指定されたコレクションに対してデータの追加または除去を許可されているかについても検査します。

図 2 に、データ・リスナー・クライアント・アプリケーションがエンタープライズ・サーチ・システムに要求を送信する方法を示します。

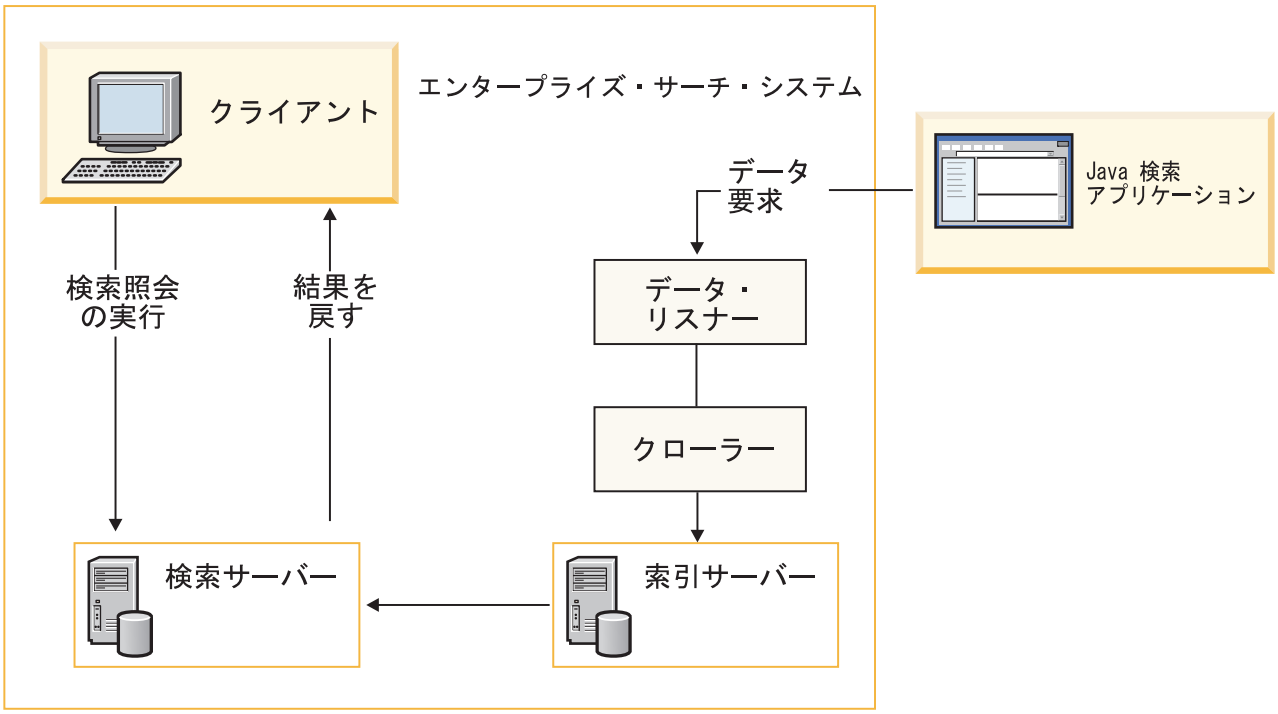

図 2. データ・リスナー API はエンタープライズ・サーチ・システムとどのように連動するか

データ・リスナー API は、以下の JAR ファイルにパッケージされています。

- es.dl.client.jar
- es.oss.jar

これらの JAR ファイルは、ご使用のオペレーティング・システムによって、以下のディレクトリーにあります。

- Linux および AIX: *INSTALL_ROOT*/IBM/es/lib
- Windows: *INSTALL_ROOT*¥IBM¥es¥lib

JAR ファイルをデフォルトのインストール・ディレクトリーにインストールしている場合は、以下の値を CLASSPATH 変数に追加してください。

- Linux および AIX: opt/IBM/es/lib/es.oss.jar:/opt/IBM/es/lib/es.dl.client.jar

- Windows: c:/Program Files/IBM/es/lib/es.oss.jar;c:/Program Files/IBM/es/lib/es.dl.client.jar

JAR ファイルをデフォルトのディレクトリーにインストールしていない場合は、適切なディレクトリーを CLASSPATH に追加してください。

データ・リスナー API によるデータの除去

データ・リスナーを使用して、特定の URI または URI のパターンを除去することによって、文書をコレクションから除去することができます。

特定の URI を除去すると、除去した URI は即時に検索コンポーネントからなくなります。 URI パターンを除去すると、次回の索引再編成時に、除去したパターンに一致する文書が除去されます。

検索コンポーネントおよび索引から URI を除去する

データ・リスナー API を使用して、 特定の URI を除去することができます。以下の図に示されているように、要求がデータ・リスナーに送信され、URI が索引サーバーおよび検索サーバーから除去されます。

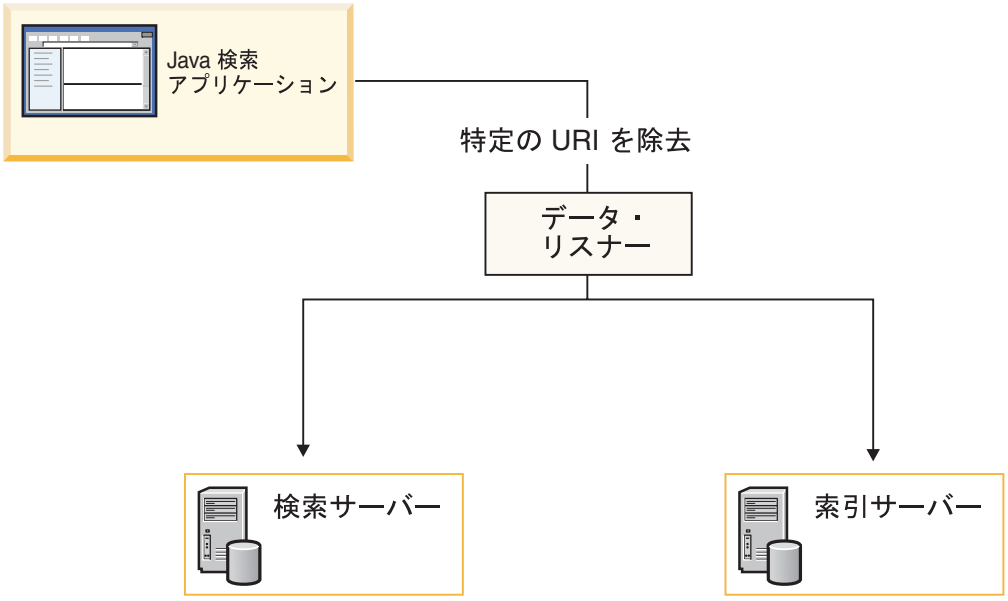

図 3. 特定の URI の除去

removeURI メソッドを使用して、指定されたコレクションからデータを削除します。

索引から URI パターンを除去する

データ・リスナー API を使用して、 URI パターンを除去することができます。例えば、URI として http://www.ibm.com/*.html を指定して URI パターンの除去要求をサブミットすると、索引サーバーは以下の URL を除去します。

- http://www.ibm.com/home.html
- http://www.ibm.com/family.html

- http://www.ibm.com/pics.html

重要: URI パターンの除去要求は、注意して使用してください。除去されたコンテンツはリカバリーできません。クローラーまたはデータ・リスナー・クライアント・アプリケーションによって、コンテンツを索引に再度、追加する必要があります。

データ・リスナー API によるデータの追加

データ・リスナーを使用して、コンテンツと共に URI を追加したり、URL にアクセスまたは再アクセスすることにより、検索コレクションにデータを追加あるいはプッシュすることができます。

コンテンツと共に URI を追加する

データ・リスナー API を使用して、 pushData メソッドに content パラメーターを組み込むことによって、URI とそのコンテンツを追加することができます。このメソッドは、指定されたデータを指定されたコレクションに送信します。

追加された URI とそのコンテンツは、索引の更新または再編成後に検索で使用可能になります。

URL にアクセスまたは再アクセスする

データ・リスナー API を使用して、revisitURLs メソッドによって、特定の URL または URL パターンをコレクションの Web クローラーに追加することができます。このメソッドは、Web コンテンツに対してのみ有効です。クローラーは、URL をクロールし、そのデータ (URL、メタデータ、およびコンテンツ) をコレクションに追加します。URL は、クローラーにとって新規のもの (アクセス) でも、クローラーが既に検出しているもの (再アクセス) でもかまいません。 URL パターンを指定すると、指定されたパターンに一致する、クローラーが既に検出済みの URL に再アクセス することを要求します。

データ・リスナー・クライアント・アプリケーションの作成

データ・リスナーに要求を送信するには、クライアント・アプリケーションを作成して、 コレクションに関する特定のプロパティー (ターゲット・コレクションのコレクションID への要求には指定しなければならない) と 認証データを指定します。 認証データには、クライアント ID とパスワードが含まれます。 クライアント ID は、コレクションを更新する権限を持っている必要があります。

手順

データ・リスナー・アプリケーションを作成するには、以下のようにします。

1. 変更するコレクションを決定する。
2. そのコレクションの ID を入手する。コレクション ID は、エンタープライズ・サーチ管理コンソールでコレクションを作成した際に作成されます。コレクション ID は「コレクション設定」ページで見ることができます。エンタープライズ・サーチ管理コンソールで「コレクション」ビューの「一般」ページから「**コ**

レクション設定の表示」をクリックします。エンタープライズ・サーチ管理コンソールにログインするか、またはこの ID をエンタープライズ・サーチの管理者に問い合わせる必要があります。

3. コレクションのクライアント ID とパスワードを指定する。クライアントとは、コレクションにアクセスするデータ・リスナー・アプリケーションです。アクセスするコレクションに対するクライアント ID とパスワードを作成して、コレクションにアクセスする許可をデータ・リスナー・アプリケーションに与える必要があります。クライアント ID とパスワードを指定するには、エンタープライズ・サーチの管理者に依頼するか、または以下のステップに従ってください。
   a. エンタープライズ・サーチ管理コンソールにログインする。
   b. 「**システム**」 → 「**データ・リスナー**」 → 「**データ・リスナーの構成**」の順にクリックする。
   c. 「**データ・リスナー・クライアント ID の追加**」をクリックする。
   d. データ・リスナー・クライアント ID、パスワードを入力して、コレクションを選択する。「**OK**」をクリックします。

      さらに、他のコレクションに対するクライアント ID とパスワードを追加する場合は、「**データ・リスナー・クライアント ID の追加**」 をクリックして、同様の操作を行います。

DLResponse クラス

クライアント・アプリケーションがデータ・リスナーから受け取るオブジェクトを定義します。

クライアント・アプリケーションがデータ・リスナーに要求を送信すると、 データ・リスナーは DLResponse オブジェクトを戻します。オブジェクトには、要求の状況 に関する情報が含まれています。また、このクラスは、オブジェクトに関して起こりうる戻りコードも定義します。

タイプ	値
Class	public class DLResponse

DLResponse クラスは、データ・リスナーから以下のタイプの状況を戻す可能性があります。

値	説明
public final static int SUCCESS = 0;	要求された操作が正常に行われました。
public final static int COMMUNICATION_ERROR = 1;	通信エラーが発生しました。
public final static int BAD_REQUEST = 2;	データ・リスナーは要求を理解しませんでした。
public final static int PERMISSION_DENIED = 3;	クライアント ID はコレクションを更新する権限を持っていません。
public final static int FAIL_TO_SAVE = 4;	データ・リスナーは要求の保管に失敗しました。

値	説明
public final static int UNSUPPORTED_REQUEST_VERSION = 5;	DB2 II OmniFind Edition のクライアントのバージョンが、サーバーのバージョンと一致していません。
public final static int UNSUPPORTED_RESPONSE_VERSION = 6;	DB2 II OmniFind Edition のクライアントのバージョンが、サーバーのバージョンと一致していません。
public final static int INVALID_PASSWD = 7;	指定したパスワードは指定したクライアント ID に対して無効です。
public final static int UNKNOWN_COLLECTION = 8;	指定したコレクション ID が無効です。
public final static int FAIL_TO_ADD_TO_CRAWLER = 9;	指定した URL を保管できませんでした。
public final static int FAIL_TO_REMOVE_FROM_COLLECTION = 10;	データ削除要求が失敗しました。
public final static int UNKNOWN_URICODE = 11;	データ・リスナーは要求された操作を実行できませんでした。

getCode メソッド

データ・リスナーから状況コードを戻します。

構文

```
public int getCode()
```

戻り

データ・リスナーから戻りコードに対応する数値を戻します。 31 ページの『DLResponse クラス』を参照してください。

getCodeName メソッド

戻りコードに応じてメッセージ・ストリングを戻します。

構文

```
public String getCodeName()
```

戻り

データ・リスナーのいずれかの戻りコードからストリングを戻します。 31 ページの『DLResponse クラス』を参照してください。

DLDataPusher クラス

このクラスは、コレクションに対してデータを追加する API およびコレクションから URI を除去するAPI を提供します。 また、コレクションの Web クローラーに URL へのアクセスまたは再アクセスを要求する API も提供します。

共通クラス `DLDataPusher`

これは、データ・リスナー API の main クラスです。

これらの API を呼び出す場合は、データ・リスナーのホスト名とポートを指定し、認証用のクライアント ID とパスワードを指定する必要があります。クライアント ID、パスワード、およびコレクション ID の入手方法については、 30 ページの『データ・リスナー・クライアント・アプリケーションの作成』を参照してください。

removeURIs メソッド

指定されたコレクションから URI を除去します。

コレクションから除去する URI を別々に指定するか、またはコレクションから除去する URI パターン を指定することができます。このメソッドを使用して個別に URI を除去すると、 URI は即時に検索で使用不可になります。しかし、このメソッドを使用して URI パターンを除去 すると、パターンに一致する URI が、索引の次回再編成時に除去されます。索引が再編成されるまで、 URI は検索可能です。

構文

```
public static DLResponse removeURIs(String hostname,
                                    int port,
                                    String id,
                                    String pw,
                                    String uris,
                                    String col)
```

パラメーター

hostname
データ・リスナーが稼動しているサーバーのホスト名。

port
データ・リスナーのポート番号。

ID ユーザーまたはエンタープライズ・サーチの管理者が、エンタープライズ・サーチ管理コンソールで特定のコレクションに対して指定したクライアント ID。

pw クライアント ID のパスワード。

uris
コレクションから除去する特定の URI、複数の URI、または URI パターン。各 URI または URI パターンを空白で区切る必要があります。URI パターンを指定する場合は、 URI の最後または途中にワイルドカード文字 (*) を使用します。例えば、 cm://enterprise/finance/* または cm://enterprise/*/sales のように指定します。

col 変更するコレクションの ID。コレクション ID は、エンタープライズ・サーチ管理コンソールで見ることができます。

revisitURLs メソッド

指定されたコレクションの Web クローラーに、URL を追加する、 または URL に再アクセスするように指示します。

このメソッドで URL パターンを指定すると、Web クローラーは、パターンに一致する検出済みの URL に即時に再アクセスしようとします。 URL を個別に指定す

ると、Web クローラーは、指定された URL に即時にアクセスします (または、クローラーが既にそれらの URL を検出済みの場合は再アクセスします)。

revisitURLs メソッドは、Web データ・ソース (URL) に対してのみ使用できます。

構文

```
public static DLResponse revisitURLs(String hostname,
                                      int port,
                                      String id,
                                      String pw,
                                      String urls,
                                      String col)
```

パラメーター

hostname
データ・リスナー・サーバーのホスト名。

port
データ・リスナーのポート番号。

id クライアント ID。

pw クライアントのパスワード。

urls
空白文字で区切られた URL または URL パターン。URL パターンを指定する場合は、 URL の最後または途中にワイルドカード文字 (*) を使用します。例えば、 http://www.ibm.com/* または http://www.ibm.com/*/software のように指定します 。

col コレクションの ID。

pushData メソッド

データ・リスナーにデータをプッシュします。

構文

```
public static DLResponse pushData(String hostname,
                                   int port,
                                   String id,
                                   String pw,
                                   String uri,
                                   String col,
                                   byte[] content)
```

パラメーター

hostname
データ・リスナー・サーバーのホスト名。

port
データ・リスナーのポート番号。

id クライアント ID。

pw クライアントのパスワード。

uri データの URI。

col コレクションの ID。

metadata
文書のメタデータを記述するオブジェクト。

content
文書のコンテンツ。

メタデータ・オブジェクトの作成:

データ・プッシュ要求を送信する前に、メタデータ・オブジェクトを作成する必要があります。 メタデータには、文書に関する情報が含まれており、作成者、変更日付、作成日付などの情報を組み込むことができます。

事前定義されたフィールドを使用してメタデータ・オブジェクトを作成するには、createDataSourceMetadata API を使用します。データ・ソースに固有のその他のフィールドを追加するには、addMetaField API を使用します。

createDataSourceMetadata メソッド:

メタデータ・オブジェクトを作成します。

構文

```
public static DataSourceMetadata createDataSourceMetadata(String ds,
String cid,
                                                   String dsName,
                                                   int score,
                                                   Date dt,
                                                   String language,
                                                   String securityACLs,
                                                   String contentType,
                                                   String charSet,
                                                   byte[] content)
```

パラメーター

ds この文書の元となるデータ・ソース。

cid データ・リスナー・アプリケーション・クライアント ID。

dsName
データ・ソース名。DB2 や Notes など (ストリング)。

score
このパラメーターは使用されません。

dt この文書の現行性を示す日付オブジェクト。この文書のランキングを左右するために使用されます。

language
文書の言語。「en」、「en_US」、「zh」など。エンタープライズ・サーチは、文書の言語の判別に失敗した場合に、このパラメーターを使用します。

securityACLs
コンマで区切られたセキュリティー・トークンのストリング。値がヌルの場合、文書にはセキュリティーに関する制限がなく、誰でもアクセスすることができます。例えば、文書にトークン A、B、および C がある場合、セキュリティー・トークン A、B、または C を持っているユーザーのみが文書にアクセスできます。

contentType
この文書の MIME タイプ。

charSet
文書の文字セット。UTF-8 など。

content
文書のコンテンツのバイト配列。

addMetaField メソッド:

メタデータ・オブジェクトにエレメントを追加します。

構文

```
public static void addMetaField(DataSourceMetadata metadata,
                                 String fieldName,
                                 String fieldValue,
                                 boolean searchable,
                                 boolean partOfResult,
                                 boolean fieldSearchable,
                                 boolean parametricSearchable
              boolean isContent)
```

パラメーター

metadata
新規フィールドの追加先となるメタデータ・オブジェクト。

fieldName
フィールドの名前。

fieldValue
フィールドの値。

searchable
このフィールドが検索可能かどうかを示します。

partOfResult
このフィールドが検索結果の一部であるかどうかを示します。

fieldSearchable
このフィールドに対してフィールド検索がサポートされるかどうかを示します。

parametricSearchable
このフィールドに対してパラメトリック検索がサポートされるかどうかを示します。

isContent
このフィールドがコンテンツの一部とみなされるかどうかを示します。

setKnownLanguageFor メソッド:

文書の言語を指定します。

このメソッドを使用して言語を指定しないと、エンタープライズ・サーチは、文書の言語を判別しようとします。

構文

```
static public void setKnownLanguageFor(DataSourceMetadata dsmd,
String knownLanguage);
```

パラメーター

dsmd
メタデータ・オブジェクト。

knownLanguage
言語 ID の名前。「en」、「en_US」、「zh」など。

データ・リスナー・クライアント・サンプル・アプリケーション

DB2 II OmniFind Edition は、 データ・リスナー API を使用して、エンタープライズ・サーチ・コレクションにデータを追加する方法またはエンタープライズ・サーチ・コレクションからデータを除去する方法を示すサンプル・クライアント・アプリケーションを提供します。

以下のサンプル・クライアント・アプリケーションは、次のタスクを行う方法を示します。

DLRemoveURIs.java
コレクションから URI を除去します。

DLPushData.java
コレクションに URI およびコンテンツを追加します。

DLRevisitURLs.java
Web クローラーに URL への再アクセスを指示します。

DLSampleClient.java
すべてのタスクを URI の除去、URI の追加、および URL の再アクセスを行う 1 つのアプリケーションに結合します。

データ・リスナー・クライアント・サンプル・アプリケーション: コレクションからの URI の除去

DLRemoveURIs クラスには、コレクションの URI または URI パターン を除去するクライアント・アプリケーション API の例が含まれています。

DLRemoveURIs クラスは、main と removeURIsExample の 2 つの静的メソッドを定義します。

main メソッドは、まず、ホスト名、ポート、クライアント ID、パスワード、およびコレクション ID を準備します。次に、 main メソッドは、これらのパラメーターを指定して removeURIsExample メソッドを呼び出します。

removeURIsExample メソッドは、パラメーターとしてホスト名、ポート、クライアント ID、パスワード、およびコレクション ID を受け取って、 URI または URI パターン・ストリングを準備します。

URI または URI パターンを準備したら、 removeURIsExample は、DLDataPusher クラスの removeURIs メソッドを呼び出します。その後、removeURIsExample メソッドは、データ・リスナーからの応答を検査して印刷します。

```
public class DLRemoveURIs {

    /**
     *  To remove URIs or URI patterns from a collection, you
     *  need to specify the host name and the port of the
     *  data listener server. You also need to provide
     *  the client ID and password for authentication.
     *  You need to specify which collection the data is
     *  applied to by providing the collection ID.
     *
     *  This function shows how to use the data listener
     *  APIs to remove URIs from a collection. In enterprise
     *  search, you can remove specific URIs and URIs that
     *  match patterns. If you remove specific URIs, those
     *  URIs are removed from the index. If you remove URI patterns,
     *  those URIs that match those patterns are not removed until
     *  the next index reorganization, and those URIs are still
     *  searchable. After the URIs are removed from a collection, those
     *  URIs will not be searchable until the they are re-crawled by
     *  the crawler or added through the data listener later.
     */
    static void removeURIsExample(String hostname, int port,
    String clientID, String passwd, String collectionID) {

        // In this example, you will remove some URIs from
        // one collection.
        // URIs can be either specific URIs or URIs that match patterns.
        StringBuffer sb = new StringBuffer();
        sb.append("uri1");
        sb.append("¥n");
        sb.append("uri*pattern1");
        sb.append("¥n");

        sb.append("uri2");
        sb.append("¥n");
        sb.append("uri*pattern2");

        String uris = sb.toString();

        // Now, you call the data listener API to remove URIs
        // from the collection.
        // Note that you can specify multiple URIs and URI patterns
        // in a single call. The URIs or URI patterns are seperated
        // by a newline character. In your application, you need
        // to replace the URIs and URI patterns from the sample
        // application with the real URIs that make sense for a
        // real collection.
        DLResponse dlRes = DLDataPusher.removeURIs(hostname, port,
        clientID, passwd, uris, collectionID);

        // Check the result.
        // The DLResponse object contains a result code and
        // a message that indicates the result of the operation.
        // See the documentation for the DLResponse class for the
        // code values and their meanings.
        if (dlRes != null) System.out.println(dlRes.toString());
    }

    public static void main (String [] argv) {
        if (argv.length > 0 && argv[0].equalsIgnoreCase("-h")) {
            System.out.println("usage: java DLRemoveURIs [hostname [port]]");
            return;
        }
```

```
            // First, you get the host name and the port number
            // for the data listener.
            // You need to ensure that the data listener is running on
            // the host server and listening to the port.
            // Otherwise, you might get a connection refused exception.
            String hostname = "localhost";
            if (argv.length > 0) hostname = argv[0];
            int port = 6668;
            if (argv.length > 1) {
                try {
                    port = Integer.parseInt(argv[1]);
                }
                catch (Exception e) {
                   port = 6668;
                }
            }

            // Assume that client_id_1 is a valid client id with password_1.
            // If not, you will get INVALID_PASSWD as the result code

            // Assume that client_id_1 has the authorization to update
            // collection with collection ID collection_id_1.
            // If not, you will get PERMISSION_DENIED as the result code.
            //
            // Assume that the collection  with collection ID collection_id_1
            // is a valid collection. Otherwise, you will get
            // UNKNOWN_COLLECTION as the result code.
            //
            // You need to contact the enterprise search administrator
            // to find out what client ID, password, and collection ID
            // are valid and use those valid values here.
            //
            removeURIsExample(hostname, port, "client_id_1", "password_1",
            "collection_id_1");
        }
    }
```

データ・リスナー・クライアント・サンプル・アプリケーション: コレクションへの URI および コンテンツの追加

DLPushData クラスには、URI および各 URI のコンテンツをコレクションに 追加またはプッシュするクライアント・アプリケーション API の例が含まれています。

DLPushData クラスは、main と dataPushExample の 2 つの静的メソッドを定義します。

main メソッドは、まず、ホスト名、ポート、クライアント ID、パスワード、およびコレクション ID を準備します。次に、 main メソッドは、これらのパラメーターを指定して dataPushExample メソッドを呼び出します。

URI およびそのコンテンツをコレクションに追加するには、文書 URI、コンテンツ、およびメタデータを準備します。URI およびコンテンツを準備するには、シンプル・テキスト・ストリングを追加します。文書のメタデータを準備するには、DLDataPusher クラスから DataSourceMetadata オブジェクトを作成します。その後、DataSourceMetadata オブジェクトにフィールドを追加します。

pushData メソッドを呼び出して、データ・リスナーからの応答を確認します。

```
public class DLPushData {

    /**
     *  To push data to a data listener, you need to specify the host name
     *  and the port of the data listener server.
     *  You also need to provide the client ID and password for authentication.
     *  You need to specify which collection the data is applied to by providing the
     *  collection ID.
     *
     *  This function shows how to use the data listener APIs to push a document
     *  to an enterprise search collection for indexing. A document consists of
     *  three parts: the URI, the metadata and the content.
     *  The content is the raw data of the document.
     *  The metadata contains the attributes values of
     *  the document.
     */
    static void dataPushExample(String hostname, int port, String clientID,
    String passwd, String collectionID) {

        // Prepare the data.
        // Suppose this is a very simple document with "almost empty" content
        String content = "almost empty";
        // Suppose the URI for the document is "myURI"
        String uri = "myURI";

        // Use data listener client APIs to prepare metadata.
        //
        // First, create a DataSourceMetadata object.
        //
        // The first argument indicates the data source type
        // of this document. The second argument specifies the client ID.
        // The third argument is a string that specifies the data source name.
        // The fourth argument is a number that indicates the quality of this
        // this document. This number is not currently used.
        // The fifth argument is a date that indicates
        // the currency (how up to date it is) of this document. It can be
        // used to influence the ranking of this document in the search results.
        // The sixth argument indicates language of the document.
        // Normally, enterprise search will detect the language ID of a
        // given document.
        // However, if the detection fails, the enterprise search system
        // will assume the document is in this specified language.
        // The seventh argument specifies security tokens.
        // If it is null, then this document is assumed to be available
        // to anyone. Otherwise, the document is accessible only by
        // a user with a security token that is specified here.
        // The eigthth argument specifies the MIME type of the document.
        // The ninth argument specifies the character set of the content.
        // Finally, the tenth argument is the content.
        DataSourceMetadata md =
  DLDataPusher.createDataSourceMetadata("CustomerDataSource",
   clientID,
                        "CustomerDataSourceName",
                        90,
                        new Date(),
                        "en",
                        "securityToken",
                        "text/plain",
                        "iso-8859-1",
                        content.getBytes());

        // Second, you add more fields to the metadata.
        // Each field is a name/value pair.
        // The fourth argument specifies whether the field value is searchable.
        // The fifth argument specifies whether the field value will
        // be part of the search result.
        // The sixth argument specifies whether to support field search.
```

```
            // The seventh argument specifies whether the field is
            // parametric searchable.
            // The eight argument specifies whether the field is part of the content.
            DLDataPusher.addMetaField(md,
                                      "fieldName1", "fieldValue1",
                                      true, false, true, false, true);
            DLDataPusher.addMetaField(md,
                                      "fieldName2", "fieldValue2",
                                      true, false, true, false, true);
            System.out.println("metadata:¥n" +  md.generateXML().toString());

            // Call the pushData method.
            DLResponse dlRes = DLDataPusher.pushData(hostname, port, clientID,
            passwd, uri, collectionID, md, content.getBytes());

            // Check the result from the data listener.
            if (dlRes != null) System.out.println(dlRes.toString());
        }

        public static void main (String [] argv) {
            if (argv.length > 0 && argv[0].equalsIgnoreCase("-h")) {
                System.out.println("usage: java DLPushData [hostname [port]]");
                return;
            }

            // First, you get the host name and the port number for the data listener.
            // You need to ensure that the data listener is running on the host server
            // and listening to the port.
            // Otherwise, you might get a connection refused exception.
            String hostname = "localhost";
            if (argv.length > 0) hostname = argv[0];
            int port = 6668;
            if (argv.length > 1) {
                try {
                    port = Integer.parseInt(argv[1]);
                }
                catch (Exception e) {
                   port = 6668;
                }
            }

            // Assume that client_id_1 is a valid client ID with password_1.
            // Assume that client_id_1 has the authorization to update the
            // collection with collection ID collection_id_1.
            //
            // You need to contact the enterprise search administrator to find out what
            // client ID, password, and collection ID are valid and
            // use those valid values here.
            //
            dataPushExample(hostname, port, "client_id_1", "password_1",
                            "collection_id_1");
        }
    }
```

データ・リスナー・クライアント・サンプル・アプリケーション: URL への再アクセス

DLRevisitURLs クラスには、Web クローラーに特定の URL または URL パターンへの再アクセスを指示するクライアント・アプリケーション API の例が含まれています。

DLRevisitURLs クラスは、main と revisitURLsExample の 2 つの静的メソッドを定義します。

main メソッドは、まず、ホスト名、ポート、クライアント ID、パスワード、およびコレクション ID を準備します。次に、 main メソッドは、これらのパラメーターを指定して revisitURLsExample メソッドを呼び出します。

revisitURLsExample メソッドは、パラメーターとしてホスト名、ポート、クライアント ID、パスワード、およびコレクション ID を受け取って、 URL または URL パターン・ストリングを準備します。

URL または URL パターンを準備したら、 revisitURLsExample メソッドは、DLDataPusher クラスの revisitURLs メソッドを呼び出します。その後、revisitURLsExample メソッドは、データ・リスナーからの応答を検査して印刷します。

```
public class DLRevisitURLs {

    /**
     *  To revisit URLs, you need to specify the host name
     *  and the port of the data listener server.
     *  You also need to provide the client ID and password for
     *  authentication.
     *  You need to specify which collection the data is applied
     *  to by providing the collection ID.
     *
     *  This function shows you how to use the data listener APIs
     *  to instruct the Web crawler of a collection to revisit
     *  URLs or URLs that match patterns.
     */
    static void revisitURLsExample(String hostname, int port,
    String clientID, String passwd, String collectionID) {

        // You will revisit some URLs from one collection.
        // URLs can be either individual URLs or URLs that
        // match patterns.
        StringBuffer sb = new StringBuffer();
        sb.append("uri1");
        sb.append("¥n");
        sb.append("uri*pattern1");
        sb.append("¥n");

        sb.append("uri2");
        sb.append("¥n");
        sb.append("uri*pattern2");

        String uris = sb.toString();

        // Now, you call the data listener API to revisit URLs.
        // Note that you can specify multiple URLs and URL
        // patterns in a single call. The URLs (URL patterns)
        // are seperated by a newline character.
        DLResponse dlRes = DLDataPusher.revisitURLs(hostname, port,
        clientID, passwd, uris, collectionID);

        // Check the result from the data listener.
        // The DLResponse object contains a result code and a
        // message that indicates the result of the operation.
        // See the documentation for the DLResponse class for
        // the code values and their meanings.
        if (dlRes != null) System.out.println(dlRes.toString());
    }

    public static void main (String [] argv) {
        if (argv.length > 0 && argv[0].equalsIgnoreCase("-h")) {
            System.out.println("usage: java DLRevisitURLs [hostname [port]]");
            return;
```

```
            }

            // First, you get the host name and the port number for
            // the data listener. You need to ensure that the data
            // listener is running on the host server and listening
            // to the port.
            // Otherwise, you might get a connection refused exception.
            String hostname = "localhost";
            if (argv.length > 0) hostname = argv[0];
            int port = 6668;
            if (argv.length > 1) {
                try {
                    port = Integer.parseInt(argv[1]);
                }
                catch (Exception e) {
                   port = 6668;
                }
            }

            // Assume that client_id_1 is a valid client ID with password_1.
            // If not, you will get INVALID_PASSWD as the result code.

            // Assume that client_id_1 has the authorization to update
            // collection with collection ID collection_id_1.
            // If not, you will get PERMISSION_DENIED as the result code
            //
            // Assume that the collection  with collection id collection_id_1
            // is a valid collection. Otherwise, you will get
            // UNKNOWN_COLLECTION as the result code.
            //
            // You need to contact the enterprise search administrator
            // to find out what  client ID, password, and collection ID
            // are valid and use those valid values here.
            //
            revisitURLsExample(hostname, port, "client_id_1", "password_1",
            "collection_id_1");
        }
    }
```

データ・リスナー・クライアント・サンプル・アプリケーション: コレクションに対するデータの追加、除去、および再アクセス

DLSampleClient クラスは、1 つ以上のコレクションに対して データを追加、除去、および再アクセスする方法を示すサンプル・コードを提供します。

DLSampleClient は、URI の追加、URL の除去、および URL の再アクセス用のサンプル・クライアント・アプリケーションを結合しています。

```
import java.io.*;
import java.net.*;
import java.util.Date;

import com.ibm.es.datalistener.client.*;
import com.ibm.es.datalistener.common.*;
import com.ibm.es.util.DataSourceMetadata;

/**
 * This class is sample code that shows you how to use the data listener APIs
 * to update collections.
 */
public class DLSampleClient {

    /**
     *  This example shows how to use data listener APIs to revisit URLs of a
     *  collection. You need to specify the host name
```

```
 *  and the port of the data listener server.
 *  You also need to provide the client ID and password for authentication.
 *  You need to specify which collection this operation is applied to
 *  by providing the collection ID.
 */
static void dataPushExample_1(String hostname, int port, String clientID,
String passwd, String collectionID) {

    // You will visit (add) and revisit several URLs of one collection.
    // You can revisit URLs or URLs that match patterns. URLs and
    // URL patterns are separated by a newline character. A URL pattern
    // is a string with a wildcard character (*).
    //
    StringBuffer sb = new StringBuffer();
    sb.append("url1");
    sb.append("¥n");
    sb.append("url*pattern1");
    sb.append("¥n");

    sb.append("url2");
    sb.append("¥n");
    sb.append("url*pattern2");

    String urls = sb.toString();

    // Now, you call the revisitURLs method of DLDataPusher class.
    DLResponse dlRes = DLDataPusher.revisitURLs(hostname, port, clientID,
    passwd, urls, collectionID);

    // Check the response from the data listener.
    if (dlRes != null) System.out.println(dlRes.toString());
}

/**
 *  This example shows how to use data listener APIs to remove URIs of a
 *  collection. You need to specify the host name
 *  and the port of the data listener server.
 *  You also need to provide the client ID and password for authentication.
 *  You need to specify which collection this operation is applied to
 *  by providing the collection ID.
 */
static void dataPushExample_2(String hostname, int port, String clientID,
String passwd, String collectionID) {

    // You now remove some URIs from a collection.
    // URIs are separated by a  newline characters.
    // Those URIs will be removed immediately so that they do
    // not appear in the search result.
    StringBuffer sb = new StringBuffer();
    sb.append("url1");
    sb.append("¥n");
    sb.append("url2");
    String urls = sb.toString();

    // Now, you call the removeURIs method of DLDataPusher class.
    DLResponse dlRes = DLDataPusher.removeURIs(hostname, port, clientID,
    passwd, urls, collectionID);
    // Check the response from the data listener.
    if (dlRes != null) System.out.println(dlRes.toString());



    // You can also remove URIs that match patterns.
    // URI patterns are separated by newline characters.
    // Those URIs that match these patterns will be removed
    // during the next index reorganization. Note that
    // these results might still appear in the search result until
```

```
                    // the next index reorganization.
                    sb = new StringBuffer();
                    sb.append("url*pattern1");
                    sb.append("¥n");
                    sb.append("url*pattern2");
                    String url_patterns = sb.toString();

                    // Now, you call the removeURIs method of DLDataPusher class.
                    dlRes = DLDataPusher.removeURIs(hostname, port, clientID, passwd,
                    url_patterns, collectionID);
                    // Check the response from the data listener.
                    if (dlRes != null) System.out.println(dlRes.toString());

                    // You can even remove both individual URLs and URLs that match
                    // patterns in the same request.
                    sb = new StringBuffer();
                    sb.append("url3");
                    sb.append("¥n");
                    sb.append("url*pattern4");
                    sb.append("¥n");
                    sb.append("url5");

                    // Now, you call the removeURIs method of DLDataPusher class.
                    dlRes = DLDataPusher.removeURIs(hostname, port, clientID, passwd,
                    sb.toString(), collectionID);
                    // Check the response from the data listener.
                    if (dlRes != null) System.out.println(dlRes.toString());

                }

                /**
                 *  This example shows how to use data listener APIs to push
                 *  documents to a collection. You need to specify the host name
                 *  and the port of the the data listener server.
                 *  You also need to provide the client ID and password for authentication
                 *  You need to specify which collection the data is applied to
                 *  by providing the collection ID.
                 */
                static void dataPushExample_3(String hostname, int port, String clientID,
                String passwd, String collectionID) {

                    // Prepare the content.
                    String content = "Almost empty";

                    // Prepare the URI.
                    String uri = "myURI2";

                    // Prepare the metadata.
                    //
                    // First, create a DataSourceMetadata object
                    DataSourceMetadata md =
            DLDataPusher.createDataSourceMetadata("CustomerDataSource",
             clientID,
                                  "CustomerDataSourceName",
                                  90,
                                  new Date(),
                                  "en",
                                  "securityToken",
                                  "text/plain",
                                  "iso-8859-1",
                                  content.getBytes());

                    // Second, add more fields to the metadata.
                    // Each field is a name/value pair.
                    // The fourth argument specifies whether the field value is
                    // searchable.
                    // The fifth argument specifies whether the field value will be part
```

```
                // of the search result.
                // The sixth argument specifies whether to support field search.
                // The seventh argument specifies whether the field is parametric
                // searchable.
                // The eight argument specifies whether the field is part of the
                // content.
                DLDataPusher.addMetaField(md,
                                          "fieldName1", "fieldValue1",
                                          true, false, true, false, false);
                DLDataPusher.addMetaField(md,
                                          "fieldName2", "fieldValue2",
                                          true, false, true, false, false);
                System.out.println("metadata:¥n" +  md.generateXML().toString());

                // Call the pushData method
                DLResponse dlRes = DLDataPusher.pushData(hostname, port, clientID,
                passwd, uri, collectionID, md, content.getBytes());

                // Check the response from the data listener.
                if (dlRes != null) System.out.println(dlRes.toString());

                // Push the same result again. This one will overwrite the previous one.
                dlRes =  DLDataPusher.pushData(hostname, port, clientID, passwd, uri,
                collectionID, md, content.getBytes());
                if (dlRes != null) System.out.println(dlRes.toString());
            }

            public static void main (String [] argv) {
                if (argv.length > 0 && argv[0].equalsIgnoreCase("-h")) {
                    System.out.println("usage: java DLSampleClient [hostname [port]]");
                    return;
                }

                // First, you obtain the host name and the port number for the data
                // listener.
                // You need to ensure that the data listener is running on the
                // host server and listening to the port.
                // Otherwise, you might get a connection refused exception.
                String hostname = "localhost";
                if (argv.length > 0) hostname = argv[0];
                int port = 6668;
                if (argv.length > 1) {
                    try {
                        port = Integer.parseInt(argv[1]);
                    }
                    catch (Exception e) {
                       port = 6668;
                    }
                }

                // Assume that client_id_1 is a valid client ID with password_1.
                // Assume that client_id_1 has the authorization to update the
                // collection with collection ID collection_id_1.
                //
                // You need to contact the enterprise search administrator to find
                // out what client ID, password, and collection ID are valid and
                // use those valid values here.
                //
                dataPushExample_1(hostname, port, "client_id_1", "password_1",
                "collection_id_1");

                dataPushExample_2(hostname, port, "client_id_1", "password_1",
                "collection_id_1");


                // Assume that client_id_2 is a valid client ID with password_2.
                // Assume that client_id_2 has the authorization to update the
```

```
                            // collection with collection ID collection_id_2.
                            //
                            dataPushExample_2(hostname, port, "client_id_2", "password_2",
                            "collection_id_2");

                            dataPushExample_3(hostname, port, "client_id_1", "password_1",
                            "collection_id_1");
                        }
                    }

```

第 5 章 言語サポート

エンタープライズ・サーチは、大部分のインド・ヨーロッパ語族の言語、および日本語などの アジア言語によるテキスト文書に対する言語検索サポートを提供します。

検索時の言語サポートの目的は、文書検索結果の精度を上げることです。つまり、照会に一致する文書の中で最も有効なコレクションに到達することです。

言語処理は、2 つのステージで行われます。文書が処理されて索引に追加される際と、検索時にユーザーが照会を入力する際です。

エンタープライズ・サーチには、入力文書の言語の判別および文書入力ストリームの語またはトークンへのセグメント化に必要なおおまかな言語機能しか組み込まれていません。

必要な検索が、主に、基本的なキーワード検索または文書構造を使用するネイティブ XML 検索に限られる場合、検索要件はエンタープライズ・サーチに組み込まれている言語処理で十分カバーされます。

ただし、以下の場合のように、単に文書内の語の検索でなく、より特定化した検索が必要な場合には、このタイプの処理だけでは必ずしも十分ではありません。

- 常に明確に示されているわけではない情報の検索を含むコラボレーションの場合。例えば、特定の住所または電話番号の検索など。
- 競合に関する情報。競合相手およびその競合相手が提供する製品が記載されている文書の検索、あるいは競合相手の Web サイトが過去 3 カ月にある製品セットから別の製品セットにシフトしたことの認識など。
- カスタマー・リレーションシップ・マネージメント。サンフランシスコ地区の修理店における特定の車の故障情報について記載されている文書の検索など。
- 研究分野。特定のたんぱく質とそれに関係する少なくとも 1 つの病気が同じパラグラフにある文書の検索など。

これらのシナリオにおいて、今日存在する膨大な情報ソースの集合体の中から必要なものを検索するには、エンタープライズ・サーチが提供するセグメンテーション・レベルおよび辞書ベースの分析よりさらに複雑な分析が新たに必要であることを示しています。興味のある情報のほとんどは、オリジナル文書で何らかの方法で明確にタグ付けされたり、マークされたりしていません。 代わりに、興味のある概念 (例えば、個人、組織、場所、機能、製品などの特定のエンティティーやこれらの概念相互の考えられる関係など) を認識し探し出すために、情報を分析する必要があります。

IBM Unstructured Information Management Architecture (UIMA) は、文書コレクションの中から興味のある情報を検出および探索するために必要な拡張分析機能をエンタープライズ・サーチにおいて作成する際に役立つ、アーキテクチャーおよびソフトウェア・フレームワークです。

関連概念

51 ページの『第 6 章 カスタム・テキスト分析の組み込み』
非構造化情報管理アーキテクチャー (UIMA) は、分析機能の作成、発見、構成、および配置を サポートするソフトウェア・アーキテクチャーです。 UIMA を使用して、ユーザー固有のカスタム・テキスト分析を作成することができます。

52 ページの『非構造化情報管理アーキテクチャー (UIMA)』
非構造化情報管理アーキテクチャー (UIMA) は、文書コレクション内の特定の情報を検索できる 拡張分析機能の構築を支援するアーキテクチャーおよびソフトウェア・フレームワークです。

第 6 章 カスタム・テキスト分析の組み込み

非構造化情報管理アーキテクチャー (UIMA) は、分析機能の作成、発見、構成、および配置を サポートするソフトウェア・アーキテクチャーです。 UIMA を使用して、ユーザー固有のカスタム・テキスト分析を作成することができます。

UIMA は、概念的に異なる分析機能ごとにコンポーネントを識別し、これらのコンポーネントの再使用や他のコンポーネントとの結合を容易にするオープン・プラットフォームです。

UIMA の中心となる概念 は *分析エンジン* です。これは、テキスト文書における分析内容を明確に表します。分析ロジック・コンポーネントはアノテーター と呼ばれます。アノテーターは、分析タスクのみに焦点をあて、他の処理には関与しません。分析エンジンには、1 つのアノテーターが含まれているか、あるいは、それぞれにアノテーターが含まれている多数のエンジンから構成されます。

分析エンジンにより導き出された情報は、*分析結果* と呼ばれます。理想的には、分析結果が、ユーザーが検索したい情報に一致します。

高機能言語分析には、多数の異なる分析タスクが含まれています。分析では、例えば、最初に言語の検出およびセグメンテーションを行い、続いて個々の品詞の認識を行い、その後より詳しい文法的な構文解析を行います。最終ステップには、例えば、科学的な本質と特定の徴候の出現との間の関係などの識別が含まれます。分析プロセスの各ステップは、後続のステップで必要になります。

UIMA は、ユーザー固有の分析エンジンを作成、テスト、および展開するための基本的な構築ブロックを提供します。UIMA 環境に配置できる事前構成済みの分析エンジンとして、言語分析機能を提供するものではありません。

UIMA Software Development Kit (SDK) には、UIMA コンポーネントのインプリメンテーション、記述、構成および配置を行う UIMA フレームワークの Java インプリメンテーションが組み込まれています。また、UIMA を使用するための一連のツールおよびユーティリティーが含まれている Eclipse ベースの開発環境も提供します。 Eclipse について詳しくは、www.eclipse.org を参照してください。

UIMA を使用するには、UIMA Software Development Kit をインストールする必要があります。この Development Kit は、IBM developerWorks® で入手可能です。詳しくは、WebSphere Information Integrator (http://www.ibm.com/developerworks/db2/zones/db2ii/) にアクセスしてください。
Eclipse 対話式開発環境における UIMA Software Development Kit のインストール方法については、UIMA 文書を参照してください。

関連概念

49 ページの『第 5 章 言語サポート』
エンタープライズ・サーチは、大部分のインド・ヨーロッパ語族の言語、および日本語などの アジア言語によるテキスト文書に対する言語検索サポートを提供します。

『非構造化情報管理アーキテクチャー (UIMA)』
非構造化情報管理アーキテクチャー (UIMA) は、文書コレクション内の特定の情報を検索できる 拡張分析機能の構築を支援するアーキテクチャーおよびソフトウェア・フレームワークです。

非構造化情報管理アーキテクチャー (UIMA)

非構造化情報管理アーキテクチャー (UIMA) は、文書コレクション内の特定の情報を検索できる 拡張分析機能の構築を支援するアーキテクチャーおよびソフトウェア・フレームワークです。

フィーチャー構造 とは、分析の結果を表す基礎となるデータ構造です。フィーチャー構造は、属性-値の構造をしています。各フィーチャー構造が 1 つのタイプになっており、Java クラスのように、すべてのタイプに、指定された一連の有効なフィーチャーまたは属性 (プロパティー) があります。フィーチャーには、フィーチャーが使用する値のタイプ (例えば、String など) を示す範囲タイプがあります。

ほとんどの分析プログラム (アノテーターとも呼ばれています) は、分析結果を注釈形式で生成します。注釈は、言語分析処理用に指定される特殊な種類のフィーチャー構造です。フィーチャー構造は、1 つの入力テキストをカバーし、入力テキストの先頭位置と終了位置によって定義されます。

例えば、通貨表示を認識するアノテーターは、「100.55 US Dollars」というテキストに対して、 `currencySymbol` フィーチャーを「$」に設定して、テキストをカバーする `monetaryExpression` タイプの注釈を作成します。

UIMA のすべてのアノテーターは、フィーチャー構造を使用して情報の保管および読み取りを行います。つまり、すべてのデータがフィーチャー構造としてモデル化されます。

タイプ・システムは、Java のクラス階層のように、タイプとフィーチャーから考えられるすべてのフィーチャー構造を定義します。

すべてのフィーチャー構造は、*共通分析構造* と呼ばれる中央データ構造で表されます。すべてのデータ交換は、共通分析構造を使用して扱われます。

共通分析構造には、以下のオブジェクトが含まれます。

- テキスト文書
- タイプ、サブタイプ、フィーチャーを示すタイプ・システム記述
- 文書または文書の領域を示す分析結果
- 分析結果へのアクセスおよび反復をサポートする索引リポジトリー

関連概念

49 ページの『第 5 章 言語サポート』
エンタープライズ・サーチは、大部分のインド・ヨーロッパ語族の言語、および日本語などの アジア言語によるテキスト文書に対する言語検索サポートを提供します。

51 ページの『第 6 章 カスタム・テキスト分析の組み込み』
非構造化情報管理アーキテクチャー (UIMA) は、分析機能の作成、発見、構

成、および配置を サポートするソフトウェア・アーキテクチャーです。 UIMA を使用して、ユーザー固有のカスタム・テキスト分析を作成することができます。

カスタム分析の組み込みのワークフロー

カスタム・テキスト分析アルゴリズムは、UIMA Software Development Kit を使用して、作成および テストした後、配置し、エンタープライズ・サーチの文書コレクションにおいて実行します。

ユーザー固有の分析アルゴリズムを作成し、その機能をエンタープライズ・サーチに取り込む 際のメイン・ステップは、以下のとおりです。

1. 計画および設計
   a. 検索したい情報を決定する。検索したい文書は何ですか ?
   b. 検索したい文書で情報を取得するために必要なテキスト分析の種類を指定する。
   c. コレクションに XML 文書がある場合は、ソリューションで XML マークアップを活用するかどうかを決める。エンタープライズ・サーチでは、以下の 2 つのうちいずれかの方法で XML マークアップを使用することができます。
      - カスタム分析で XML マークアップを使用できる場合 (例えば、文書に、要約またはカテゴリー化アノテーターにおいて便利な `<summary>` または `<topic>` エレメント が含まれている場合) は、XML と共通分析構造とのマッピングを定義します。
      - 照会で、XML マークアップを、文書にある通りに使用したい場合は、ネイティブ XML マッピングを 使用可能にします。
   d. 共通分析構造に保管するテキスト分析結果情報を決定する。この情報に、セマンティック検索を使用してアクセスできるようにします。共通分析構造と索引とのマッピングを定義します。
2. 作成: UIMA Software Development Kit を使用したアクティビティー
   a. 個々の分析ステップを定義する。
   b. マッピングおよび分析ロジックのタイプ・システムを記述する。
   c. 分析ステップごとに分析ロジック (アノテーター) を作成し、UIMA Software Development Kit を使用して、分析エンジンにアノテーターを組み込む。アノテーターを作成する際に、エンタープライズ・サーチ・テキスト分析パッケージの基本機能 (言語識別およびトークン化) の上にカスタム分析をビルドします。
   d. UIMA で分析アルゴリズムをテストしたら、分析エンジンを PEAR (Processing Engine Archive) ファイルとしてパッケージする。アーカイブには、ユーザーの分析アルゴリズムのみが含まれるようにし、エンタープライズ・サーチの基本言語機能は含まれないようにしてください。
3. 配置: エンタープライズ・サーチを使用したアクティビティー
   a. 分析エンジンのアーカイブ (.pear) をエンタープライズ・サーチにアップロードする。分析コンポーネントに名前を付けます。エンタープライズ・サーチで参照する際には、この名前を使用します。

b. 1 つ以上の文書コレクションを分析エンジンに関連付ける。
c. コレクションごとに、カスタム分析用に定義した、XML と共通分析構造とのマッピングをアップロードおよび選択する。
d. コレクションごとに、セマンティック検索用に定義した、共通分析構造と索引とのマッピングをアップロードおよび選択する。

テキスト分析アルゴリズム

UIMA Software Development Kit には、アノテーター (タイプ・システム記述が含まれている 分析アルゴリズム) を作成して、分析エンジンに組み込むための API およびツールが組み込まれています。

UIMA 文書には、これらのコンポーネントの作成に役立つチュートリアル・スタイルのガイドが組み込まれています。 Software Development Kit には、テストや結果の表示を行うユーティリティーや、分析結果の索引を作成する小規模なセマンティック検索エンジンが組み込まれています。索引に保管されている情報について、より高度な検索を行うこともできます。

UIMA Software Development Kit には、事前に構成されている分析エンジンはありません。ただし、 UIMA 環境においてエンタープライズ・サーチで提供される基本的な言語サポートを使用することはできます。ご使用の UIMA 環境でテキスト分析アルゴリズムを作成する前に、言語検出機能およびトークン化機能を組み込む方法を UIMA 文書で確認してください。

UIMA Software Development Kit を使用して分析エンジンの開発およびテストを完了し、エンタープライズ・サーチでドキュメント・コレクションにおいてそのアルゴリズムを実行するには、 PEAR (Processing Engine ARchive) ファイルを作成する必要があります。このアーカイブ・ファイルには、エンタープライズ・サーチにおける分析エンジンとしてユーザーが作成したカスタム分析機能を配置する際に必要なリソースがすべて含まれています。アーカイブ作成に必要なすべての処理ステップは、Software Development Kit に含まれている UIMA 文書に記載されています。

アーカイブには、必ずカスタム分析が含まれている必要があります。そのカスタム分析が、エンタープライズ・サーチで提供されている基本的な言語機能に基づいている場合でも、アーカイブに含まれていなければなりません。カスタム分析を実行する前に、常に、基本的なエンタープライズ・サーチの分析ステップが実行されます。

XML 文書構造の共通分析構造へのマッピング方法

文書内の XML 構造の情報を、UIMA アノテーターを作成せずに、共通分析構造に直接マップすることが できます。

構成ファイルを使用して、より詳細な分析を行うための基礎を形成するコンテンツが含まれている入力文書の XML エレメント、あるいは文書のメタデータ (例えば、作成者 や日付など) が含まれているエレメントを判別することができます。

特定のエレメントのコンテンツおよびセマンティクスが明確に認識されており、それ以上分析を行わなくても情報を特定のフィーチャー構造に直接マップできること

があります。例えば、請求に関する文書のエレメント <addressee> に、常に、顧客名が含まれている場合、これらの名前と価格情報を組み合わせて「個人 - 価格」の関係またはトランザクションを自動的に推測できます。

文書のコレクションに対して複数の XML 構成を使用することができます。 どの構成をどの XML 文書に使用するかは、<identifier> エレメントで判別します。 構成ファイル内の <identifier> エレメントは、XML 文書のルート・エレメントに一致している必要があります。例えば、ユーザー文書のルート・エレメントが doc である場合、構成ファイル内の <identifier> エレメントの値 も「doc」でなければなりません。

一致するものが見つからない場合、プログラムは、構成ファイル内で「Default」に設定されている <identifier> エレメントを検索します。デフォルト構成が見つからない場合は、文書の本文 (タグ情報が無い部分) が共通分析構造の文書注釈にマップされます。

マッピングには、基本的に、2 つのタイプがあります。

「エレメント - データ・タイプ」のマッピング

コレクション内の文書が同じ XML 構造 (同じエレメント) を共有していることがわかっている場合、 これらの XML エレメントを UIMA フィーチャー構造に直接マップすることができます。 例えば、エレメント <addressee> をタイプ person の フィーチャー構造から派生しているタイプ customer のフィーチャー構造にマップすることができます。

また、フィーチャー構造の属性を設定することもできます。例えば、コレクション内のすべての文書に、ある特定のバージョンの製品を所有している顧客が記載されていることがわかっている場合は、属性 product version をその値に設定することができます。その場合、分析プロセスでこれを明示的に行う必要はありません。

構成時のみに値を設定する代わりに、マップされるエレメントのコンテンツに応じて値を設定することができます。例えば、エレメントのコンテンツがストリング値でなければならない場合などです。

また、マップするエレメント内で他のエレメントによってカバーされているテキスト (ネストされたエレメントと呼ばれます) を使用して、フィーチャー値をネストされたエレメントのテキストに設定することもできます。例えば、エレメント <firstName>、<middleName>、および <lastName> がすべてエレメント <author> 内に含まれている場合は、フィーチャー author の値を、ネストされたエレメントによってカバーされているテキスト全体に設定することができます。

さらに、マッピングに関する条件を指定することができます。例えば、エレメントが文書構造内の特定の位置にある (<addressee> エレメントが <shipping> エレメント内にあるかどうかにかかわらず、<body> 内のすべての <addressee> エレメントのコンテンツを抽出する)、あるいはエレメントが特定の属性 (または特定の値) を持っている、などの条件を指定します。

文書の関係のある部分のみに含まれている情報を抽出し、関係の無い部分は無視したい場合は、文書内のどの XML エレメントに関連情報が含まれているかを指定するだけでかまいません。これは、コンテンツ抽出と呼ばれま

す。 例えば、title および body エレメントに指定されている入力情報を抽出して、author、date、ID、 publisher は無視することができます。

コンテンツ抽出は、以下のタイプの XML 文書の分析処理を向上させることができます。

- 分析に適合しないコンテンツが大量にある文書 (例えば、バイナリー添付ファイルなど)。コンテンツ抽出を使用することにより、文書サイズがかなり削減され、処理が速くなり、不適合データによる分析エラーが回避されます。
- 文書テキストと関係の無いテキストが混在している文書 (例えば、<note> タグ内に編集情報が含まれている文書など)。この情報を無視することにより、文書コンテンツの分析時により良い結果が得られます。

ネイティブ XML マッピング

このプロセスには、文書からすべての XML タグ付け情報を取り除く処理、およびより詳細な分析のために文書のコンテンツを残しておく処理が含まれます。

文書に関する XML 構造化情報を保持するために、com.ibm.es.tt.MarkupTag を使用して、エレメントの名前、その属性および値がエンタープライズ・サーチに保管されます。

このように、XML 情報は、ネイティブ XML 検索でもアクセスすることができます。ネイティブ XML マッピングは、マッピング構成ファイルを必要としません。エンタープライズ・サーチの管理コンソールを使用して、ネイティブ XML マッピングを使用可能にすることができます。

ネイティブ XML マッピングと、エレメントとタイプのマッピングのコンテンツ抽出オプションは、すべてのコンテンツを対象とするか、指定されたコンテンツのみを対象とするかという点で互いに矛盾しています。コンテンツ抽出オプションを指定すると、ネイティブ XML マッピングは無視されます。コンテンツ抽出を使用しない場合は、エレメントとタイプのマッピングと、ネイティブ XML マッピング の両方を使用することができます。

XML マッピング構成ファイルで使用するすべてのタイプおよびフィーチャーは、カスタム分析ステップのシステム記述に示されていなければなりません。ご使用の UIMA 環境で、 Component Descriptor Editor Eclipse プラグインを使用して、タイプ・システム記述子を作成することができます。このプラグインによって、必要な XML 構文に煩わされることなく、記述子ファイルを作成することができます。

カスタム分析の作成およびテストが完了したら、 UIMA PEAR (Processing Engine ARchive) 生成ウィザードを使用して、タイプ・システム記述が組み込まれたカスタム分析ファイルが含まれるアーカイブを作成します。

その後、エンタープライズ・サーチの管理コンソールを使用して、カスタム分析アーカイブと XML マッピング構成ファイルをエンタープライズ・サーチにアップロードします。

関連タスク

57 ページの『XML マッピング構成ファイル』
XML マッピング構成ファイルでは、XML を共通分析構造にマッピングするための 全範囲の構成オプションを使用できます。

関連資料

60 ページの『XML マッピング・サンプルおよび出力結果』
XML 文書構造のカスタム分析構造へのマッピングを、簡単なサンプル文書に基づいて示します。

XML マッピング構成ファイル

XML マッピング構成ファイルでは、XML を共通分析構造にマッピングするための全範囲の構成オプションを使用できます。

構成ファイルは、特定のスキーマに従った XML ファイルでなければなりません。XML 構文エラーを防ぐために、XML エディターを使用して構成ファイルを作成してください。

XML マッピング・スキーマは、以下の 2 つのパラメトリック・セクションに分割されます。

- **<contentElements>**: 特定のコンテンツを抽出する場合、例えば、 文書の <Abstract> および <Body> セクションのコンテンツを抽出する場合には、このエレメントを使用します。文書内のその他のセクションはすべて無視されます。
- **<elementToTypeMappings>**: 文書内の個々のどの XML エレメント (<elementToTypeMapping> エレメントで指定されています) を共通分析構造のどのフィーチャー構造にマップするかを指定するには、このエレメントを使用します。

  コンテンツ抽出オプションを使用する場合には、<elementToTypeMappings> セクション に指定されている XML エレメントが、<contentElements> セクション に指定されている XML エレメント内に含まれていなければならないことに注意してください。

サンプル構成ファイルは、以下のとおりです。

```
<?xmlversion="1.0"?>
<xmlCasInitializerConfiguration
 xmlns="http://www.ibm.com/2005/uima/jedii_ci_xml">

 <identifier>Default</identifier>
 <description>Sample configuration</description>

 <contentElements>
   <element>/doc/Title</element>
   <element>/doc/Abstract</element>
   <element>/doc/Body</element>
 </contentElement>

 <elementToTypeMappings>
   <elementToTypeMapping>
     <element>//Employees//IBMer</element>
     <type>example.Person</type>
     <featureValueAssignment>
       <feature>employer</feature>
       <basicValue>default="IBM"</basicValue>
     </featureValueAssignment>
     <featureValueAssignment>
       <feature>age</feature>
       <basicValue default="34"/>
     </featureValueAssignment>
```

```
   </elementToTypeMapping>
 </elementToTypeMappings>

</xmlCasInitializerConfiguration>
```

各 <elementToTypeMapping> には、以下のエレメントが含まれている必要があります。

- <element> エレメント。これは、XML エレメントのパスを指定するために 使用され、XPath 構文に従います。先頭の「/」は、絶対パスが指定されていることを意味します。例えば、ルート・エレメント doc の下の Title がこれに該当します。「//」は、パス・サブセットを意味します。例えば、IBMer は Employees 内になければなりませんが、他のエレメントはこれら 2 つの間にあればかまいません。
- <type> エレメント。これは、タイプ・システム記述に定義されているタイプを指定します。
- ゼロ以上の <featureValueAssignment> エレメント。

<featureValueAssignment> エレメントでは、フィーチャーが <feature> エレメントで指定されており、値が <basicValue> エレメントで割り当てられている必要があります。<basicValue> エレメントは、以下の属性を持つことができます。useAttributeValue 属性と useContentValue 属性は、相互に排他的である点に注意してください。

- <basicValue useAttributeValue="date" default="UNKNOWN"> これは、定義されたフィーチャーに date 属性の値をとります。属性が無い場合は、デフォルト値 UNKNOWN を使用します。
- <basicValue useContentValue="date" trim="yes" default="UNKNOWN"> これは、定義されたフィーチャーの値として、ブランクが挿入された XML エレメント date のコンテンツを使用します。 date にコンテンツが無い場合は、デフォルト値 UNKNOWN を使用します。
- <basicValue default="UNKNOWN"> これは、常にデフォルト値 (この場合は UNKNOWN) をとります。

フィーチャーの値として属性の値を使用したい場合は、useAttributeValue を使用します。 例えば、以下の構成断片の場合、

```
<elementToTypeMapping>
  <element>/Doc//IBMer</element>
  <type>example.Person</type>
  <featureValueAssignment>
    <feature>age</feature>
    <basicValue useAttributeValue="age"/>
  </featureValueAssignment>
</elementToTypeMapping>
```

結果は、次のようになります。

- Doc タグ内の IBMer タグごとに、 example.Person タイプのフィーチャー構造が作成されます。
- IBMer タグに age 属性が含まれている場合、新規に作成されたフィーチャー構造のフィーチャー age は、属性の値に設定されます。

フィーチャーの値としてコンテンツを追加するには、useElementContent 属性を使用します。例えば、book の author エレメントによってカバーされるテキストは、フィーチャー author の値になり、前後のブランクはすべて削除されます。

```
<elementToTypeMapping>
  <element>//book</element>
  <type>example.Book</type>
  <featureValueAssignment>
    <feature>author</feature>
    <basicValue useElementContent="author" trim="true"/>
  </featureValueAssignment>
</elementToTypeMapping>
```

以下のケースでは、<values> エレメントの間に複数の値が指定されています。

- フィーチャーが、配列タイプである場合。以下の例では、7 と 12 は、整数配列タイプの フィーチャー kidsAges にある値です。

```
<elementToTypeMapping>
  <element>/Doc/Body/Father</element>
  <type>example.Father</type>
  <featureValueAssignment>
    <feature>kidsAges</feature>
    <values>
      <basicValue default="7"/>
      <basicValue default="12"/>
    </values>
  </featureValueAssignment>
</elementToTypeMapping>
```

- 多数のストリングが 1 つのストリングに連結され、ストリング・タイプのフィーチャーにマップする場合。以下の例では、Mr. というタイトルが定数で、ファーストネームが属性の値で、ラストネームが XML エレメントによってカバーされます。

```
<elementToTypeMapping>
  <element>//IBMer</element>
  <type>example.Person</type>
  <featureValueAssignment>
    <feature>fullName</feature>
    <values concatenate="true" delimiter=" ">
      <basicValue default="Mr."/>
      <basicValue useAttributeValue="firstName"/>
      <basicValue useElementContent="IBMer"/>
    </values>
  </featureValueAssignment>
</elementToTypeMapping>
```

ストリング・フィーチャー値は、構成ファイルからそのまま抽出されます。値は、先頭および末尾のブランクが入ったままになります。ただし、タイプおよびフィーチャーの名前には、ブランクが挿入されます。

<condition> エレメントを使用して、属性に条件を設定することもできます。例えば、 example.IBMFather タイプのフィーチャー構造は、 employer 属性が ibm に設定されており、2 番目の sonsName 属性を持つ Employee が文書内にある場合にのみ作成されます。

```
<elementToTypeMapping>
  <element>//Employee</element>
  <type>example.IBMFather</type>
  <condition attribute="employee" value="ibm"/>
  <condition attribute="sonsName"/>
</elementToTypeMapping>
```

関連概念

54 ページの『XML 文書構造の共通分析構造へのマッピング方法』
文書内の XML 構造の情報を、UIMA アノテーターを作成せずに、共通分析構造に直接マップすることが できます。

関連資料

『XML マッピング・サンプルおよび出力結果』
XML 文書構造のカスタム分析構造へのマッピングを、簡単なサンプル文書に基づいて示します。

XML マッピング・サンプルおよび出力結果

XML 文書構造のカスタム分析構造へのマッピングを、簡単なサンプル文書に基づいて示します。

サンプル XML 入力文書は、以下のような構造になっています。

```
<Doc>
By
  <Head>
  <Author><FirstName>Nina</FirstName><LastName>Eisenberg</LastName></Author>
and
  <Reviewer>Tanja Stahlhugel</Reviewer>
  <Date type="text">26 November 2004</Date>
  <GMT type="gmt">26 November 2004</GMT>
A publication of
  <Publisher>The San Diego Journal</Publisher>
  <Title>Chip, heal themself</Title>
  </Head>
  <Abstract>I.B.M. invents chips that reconfigure themselves.</Abstract>
  <Body>At I.B.M., researchers have designed chips with built-in fuses
that can do some self-repair jobs, said
    <IBMer>Subramanian S. Iyer</IBMer>, an inventor of the technology. Miss
    <OtherEmployee  company="Siemens" firstName="Gundula"
      middleInitial="O." age="56">Baumgarten</OtherEmployee>
from Siemens congratulated. Another employee, Mister
    <OtherEmployee firstName="Titus">Jones</OtherEmployee> said ...

Mister
<Father lastName="Clark">Clark is father of <Child>Clara</Child></Father>
</Body>
</Doc>
```

共通分析構造サンプル構成に対応する XML は、以下のとおりです。

```
<?xmlversion="1.0"?>
<xmlCasInitializerConfiguration>
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="XMLCasInitSchema.xsd">

 <identifier>Doc</identifier>
 <description>Sample configuration</description>

 // content elements from which to extract the text. Abstract is ignored
 <contentElements>
   <element>/Doc/Title</element>
   <element>/Doc/Body</element>
   <element>/Doc/Head</element>
 </contentElement>

 // single constant feature values
 <elementToTypeMappings>
   <elementToTypeMapping>
```

```
  <element>/Doc//IBMer</element>
  <type>example.Person</type>
  <featureValueAssignment>
    <feature>employer</feature>
    <basicValue useAttributeValue="company" trim="true"
      default="IBM"/>
  </featureValueAssignment>
  <featureValueAssignment>
    <feature>age</feature>
    <basicValue useAttributeValue="age" default="-1"/>
  </featureValueAssignment>
 </elementToTypeMapping>
 <elementToTypeMapping>
  <element>/Doc//OtherEmployee</element>
  <type>example.Person</type>
  <featureValueAssignment>
    <feature>employer</feature>
    <basicValue useAttributeValue="company" trim="true"
       default="IBM"/>
         </featureValueAssignment>
  <featureValueAssignment>
    <feature>age</feature>
    <basicValue useAttributeValue="age" default="-1" />
  </featureValueAssignment>
  <featureValueAssignment>
    <feature>fullName</feature>
    <values concatenate="true" delimiter=",">
      <basicValue useAttributeValue="firstName" />
      <basicValue useElementContent="OtherEmployees"
        default="UNKNOWN"/>
    </values>
  </featureValueAssignment>
 </elementToTypeMapping>
 // annotate nested elements
 <elementToTypeMapping>
  <element>//Head</element>
  <type>example.Book</type>
  <featureValueAssignment>
    // the feature author is set to the contents of the element Author
    <feature>author</feature>
    <basicValue useElementContent="Author"/>
  </featureValueAssignment>
  <featureValueAssignment>
    <feature>publisher</feature>
    <basicValue useElementContent="Publisher" trim="true"
       default="UNKNOWN"/>
  </featureValueAssignment>
  <featureValueAssignment>
    <feature>title</feature>
    <basicValue useElementContent="Title" />
  </featureValueAssignment>
</elementToTypeMapping>
<elementToTypeMapping>
  <element>/Doc//Father</element>
  <type>example.Father</type>
  <featureValueAssignment>
    <feature>kidsNames</feature>
    <values>
      <basicValue useElementContent="Child"/>
      <basicValue default="Sarah"/>
    </values>
  </featureValueAssignment>
  <featureValueAssignment>
    <feature>kidsAges</feature>
      <values>
        <basicValue default="5" />
      </values>
```

```
    </featureValueAssignment>
  </elementToTypeMapping>
  // annotate the date if the attribute type=text is true
  <elementToTypeMapping>
     <element>/Doc//Date</element>
     <type>example.Date</type>
     <condition attribute="type" value="text" />
     <featureValueAssignment>
       <feature>date</feature>
       <basicValue useElementContent="Date"/>
     </featureValueAssignment>
     <featureValueAssignment>
       <feature>gmt</feature>
       <basicValue default="-1" />
     </featureValueAssignment>
  </elementToTypeMapping>
  // annotate the date if the attribute type=long is true.
  // In input text this condition is not true, so no annotation is
  // created.
  <elementToTypeMapping>
     <element>/Doc//GMT</element>
     <type>example.Date</type>
     <condition attribute="type" value="long" />
     <featureValueAssignment>
       <feature>gmt</feature>
       <basicValue useElementContent="GMT"/>
     </featureValueAssignment>
     <featureValueAssignment>
       <feature>date</feature>
       <basicValue default="UNKNOWN" />
     </featureValueAssignment>
  </elementToTypeMapping>

 </elementToTypeMappings>

</xmlCasInitializerConfiguration>
```

構成ファイルに基づいて、共通分析構造に以下の注釈が生成されます。

```
uima.tcas.DocumentAnnotation:
 begin=0
 end=479
 covered text:
"  Nina Eisenberg
and
Tanja Stahlhugel
26 November 2004
26 November 2004
A publication of The San Diego Journal
Chip, heal themself

At I.B.M., researchers have designed chips with
built-in fuses that can do some self-repair jobs,
said Subramanian S. Iyer,
an inventor of the technology.

Miss Baumgarten from Siemens congratulated.

Another employee,
Mister Jones
said ......

Mister  Clark is the father of Clara."

example.Book:
 begin=0
 end=166
```

```
 author= "Nina Eisenberg"
 publisher="The San Diego Journal"
 title="Chip, heal themself"
 covered text:
"  Nina Eisenberg
and
Tanja Stahlhugel
26 November 2004
26 November 2004

A publication of The San Diego Journal
Chip, heal themself"

example.Date:
 begin=55
 end=71
 date="26 November 2004"
 gmt=-1
 covered text: "26 Novmber 2004"

example.Person:
 begin=277
 end=296
  employer = "IBM"
 fullName = null
 age = -1
 covered text: "Subramanian S. Iyer"

example.Person:
 begin=342
 end=352
  employer = "Siemens"
 fullName = "Gundula,Baumgarten"
 age = 56
 covered text: "Baumgarten"

example.Person:
 begin=414
 end=419
  employer = "IBM"
 fullName = "Titus,Jones"
 age = -1
 covered text: "Jones"

example.Father:
 begin=446
 end=477
 kidsNames = [Clara,Sarah]
 kidsAges = [5]
 covered text: " Clark is father of Clara. "
```

関連概念

54 ページの『XML 文書構造の共通分析構造へのマッピング方法』
文書内の XML 構造の情報を、UIMA アノテーターを作成せずに、共通分析構造に直接マップすることが できます。

関連タスク

57 ページの『XML マッピング構成ファイル』
XML マッピング構成ファイルでは、XML を共通分析構造にマッピングするための 全範囲の構成オプションを使用できます。

カスタム分析結果の索引付けの方法

文書のコレクションにカスタム分析を実行した後、 エンタープライズ・サーチの検索エンジンを使用して、カスタム分析アルゴリズムによって生成される 共通分析構造に保管された情報から索引を作成することができます。

エンタープライズ・サーチ索引で、分析結果をフィールド、テキストのスパン、および属性にマッピングすることにより、照会でその情報を使用できるようになります。カスタム分析を語とテキストのスパンの両方の索引付け機能を持つエンタープライズ・サーチと結合することにより、セマンティック検索が可能になります。

索引作成構成ファイルを使用して、共通分析構造内のどの分析結果の索引付けを行うかを判別することができます。

共通分析構造内のフィーチャー構造をエンタープライズ・サーチ索引にマップするには、さまざまなスタイルがあります。

注釈 注釈スタイルを使用して、共通分析構造内のフィーチャー構造の索引付けを行うと、指定したタイプのすべての注釈が、検索可能なスパンとして索引に保管されます。

例えば、テキストの一定範囲に渡るフィーチャー構造が `person` タイプで、注釈スタイルを使用して索引付けを行う場合には、以下の照会が可能です。

表 2. 照会例

必要な情報	可能な照会
少なくとも 1 人の個人 (person) 名が含まれている文書を検索する	<person/>
個人 (person) 注釈に上司 (boss) が含まれている文書を検索する	<person>boss</person>
言葉 (Lang) が競合相手 (competitors) のいずれかと同じセンテンス (sentence) に記載されている文書をすべて検索する	<sentence><person>Lang</person> <competitor/></sentence>

フィーチャー構造の属性も、スパンの一部として索引付けることができます。例えば、車を検出し、`car` 注釈の `make` フィーチャーとして車の製造会社を保管するアノテーターの場合を考えてみます。この場合、「Chevrolet が製造した車が記載されている文書を探す」という照会が可能になります。

フィールド

エンタープライズ・サーチのフィールド検索機能を使用して、検索時にフィーチャー構造のコンテンツをアクセス可能にする場合は、このスタイルを使用します。この方法では、フィーチャー構造のコンテンツを検索結果に表示したり、パラメトリック検索で使用したりすることができます。

例えば、薬の服用量をパラメトリック・フィールドにマップすると、「服用時に 100 ミリグラムを超える薬について記載されている文書をすべて探す」という照会を行うことができます。

ブレーク

特定のフィーチャー構造を明確な区切りとして解釈する (例えば、セクショ

ンやパラグラフなど) 場合に、このスタイルを使用します。エンタープライズ・サーチは、デフォルトで、センテンス (文) およびパラグラフ (段落) を検出します。このスタイルは、カスタム分析が、文書内で、個別に解釈したい追加の構造化エレメントを検出する場合にのみ使用します。

索引作成構成ファイルを作成したら、エンタープライズ・サーチの管理コンソールを使用して、索引作成構成ファイルをアップロードすることができます。

フィーチャー・パスの定義

索引作成仕様は、フィーチャー・パスを使用して、共通分析構造内のフィーチャー構造を指定します。これは、XML 文書で XML エレメントにアクセスする際に使用する XPath ステートメント に似ています。

索引作成仕様内の `<feature>` エレメントにコード化されるフィーチャー・パスは、共通分析構造内の単純な値のフィーチャー、つまり `uima.cas.Float`、`uima.cas.Integer`、`uima.cas.String` タイプのフィーチャーに対するパスを定義します。

最も簡単な形式では、フィーチャー・パスはマッピング・エレメントに記述されているタイプのフィーチャーの名前になります。タイプが、値としてフィーチャー構造を持っているような複雑なフィーチャーを定義している場合、フィーチャー・パスを使用して、共通分析構造内の単純なフィーチャー値にアクセスすることができます。

例えば、車と車の製造会社を示すアノテーターの場合を考えてみます。アノテーターは、`make` という属性を持つ `car` タイプの注釈を作成します。ただし、`make` には、実際の会社名 (例えば、 Chevrolet など) は含まれていませんが、それ自体が `companyname` というストリング値属性 を持つ `Company` というタイプのフィーチャー構造が含まれています。車の名前と会社名を結合するセマンティック照会を可能にするには、フィーチャー・パス `make/companyname` を使用して、値 `companyname` を車の注釈用に生成される車のスパンに結び付けます。これにより、'/car[@make="Chevrolet"]' という 構文を使用して「Chevrolet 社製造の車が含まれている文書を探す」という照会を行うことができます。

フィーチャー・パスは、以下のプロパティーを持つ一連のフィーチャー名 (f1/.../fn) です。

- パス内の最後のフィーチャー fn は、単純な値のフィーチャー、つまり `uima.cas.Float`、`uima.cas.Integer`、または `uima.cas.String` タイプでなければなりません。
- f1 から fn-1 までのパス内のフィーチャーは、単純な値のフィーチャーであってはなりません。
- オプションで、フィーチャー名にタイプを指定できます。フィーチャー名の前に、完全修飾タイプ名を指定し、コロンで区切る必要があります。例えば、`f1/com.ibm.es.SomeType:f2/.../fn` のように指定します。

  これは、特定のフィーチャーのタイプの有効範囲を絞り込む際に使用できます。例えば、`uima.cas.TOP` タイプの `additionalInfo` というフィーチャーの場合を考

えてみます。 additionalInfo フィーチャーの値が実際に EmployeeInfo であることがわかっている場合には、 EmployeeInfo/additionalInfo:fn と指定することができます。

配列値を持つフィーチャーは、以下の追加プロパティーを持ちます。

- フィーチャー・パスの次のエレメントには、タイプを指定しなければなりません。タイプ名は、配列内のエレメントのタイプです。例えば、Info タイプの フィーチャー構造の場合を考えてみます。このタイプには、範囲が FSArray である companies というフィーチャーがあります。配列のエレメントは Company タイプです。Company には、 profit というフィーチャーがあります。3 番目の会社の収益 (profit) 情報を獲得するには、 companies[3]/Company:profit (通常、完全修飾タイプ名を使用します) と指定します。
- 大括弧 ([ ]) を使用して、配列内の 1 つのエレメントを選択しなければなりません。配列はゼロ (0) から開始します。例えば、companies 配列内の最初のエレメントを選択する場合には、 Company:companies[0] と指定します。配列のサイズに関係なく、配列内の最後のエントリーを選択する場合は、特殊マーカー [last] を使用できます。例えば、 Company:companies[last] のように指定します。
- パスの最後のフィーチャー fn が配列値である場合は、uima.cas.FloatArray、uima.cas.IntegerArray、および uima.cas.StringArray 配列タイプのみが許可されます。
- パス内の f1 から fn-1 までのフィーチャーは、uima.cas.FloatArray、uima.cas.IntegerArray、または uima.cas.StringArray タイプであってはなりません。
- パスの最後のフィーチャー fn が配列値である場合は、空の大括弧 ([]) を使用して「すべてのエレメント」を表します。これらのエレメントは連結され、単一のものとして索引に書き込まれ、multi-term 属性になります。例えば、それぞれが名前のようなものを指定している nameCandidates という配列がある場合、すべての候補者名が特定の属性 (またはフィールド) にマップされます。

索引作成構成ファイルの作成

索引作成構成ファイルは、特定のスキーマに従った XML ファイルでなければなりません。

サンプル索引作成構成ファイルは、以下のようになっています。

```
<?xmlversion="1.0" encoding="UTF-8"?>
<indexBuildSpecification
xmlns:namespace="http://www.ibm.com/of/822/consumer/index/xml">
 <skipCondition>
   <type>com.ibm.uima.tt.DocumentAnnotation</type>
   <filter syntax="FeatureValue">toBeprocessed = 0</filter>
 </skipCondition>

 <indexBuildItem>
   <typeName>com.ibm.uima.tt.PersonAnnotation</typeName>
   <indexRule>
     <style name="Annotation">
       <attribute name="fixedName" value="Person"/>
       <attributeMappings>
         <mapping>
           <feature>title</feature>
           <indexName>title</indexName>
         </mapping>
```

```
        <mapping>
          <feature>firstName</feature>
          <indexName>name</indexName>
        </mapping>
      </attributeMappings>
    </style>
    <style name="Field">
      <attribute name="fixedName" value="People"/>
      <attribute name="parametric" value="false"/>
      <attribute name="fieldSearchable" value="true"/>
      <attribute name="returnable" value="true"/>
    </style>
  </indexRule>
  <filter syntax="FeatureValue">confidence = 0.7</filter>
</indexBuildItem>
 <indexBuildItem>
  <name>com.ibm.uima.tt.GeneralEntity</name>
  <indexRule>
    <style name="Annotation">
      <attribute name="nameFeature" value="categoryName" />
    </style>
    <style name="Field">
      <attribute name="nameFeature" value="categoryName" />
      <attribute name="parametric" value="false" />
      <attribute name="fieldSearchable" value="true" />
      <attribute name="returnable" value="true" />
    </style>
  </indexRule>
</indexBuildItem>
<indexBuildItem>
  <name>com.ibm.uima.tt.DrugDosage</name>
  <indexRule>
    <style name="Field">
      <attribute name="fixedName" value="dosage" />
      <attribute name="parametric" value="true" />
      <attribute name="fieldSearchable" value="true" />
      <attribute name="returnable" value="false" />
     </style>
    <style name="Field">
      <attribute name="fixedName" value="make" />
      <attribute name="valueFeature" value="make/companyname" />
      <attribute name="parametric" value="false" />
      <attribute name="fieldSearchable" value="true" />
      <attribute name="returnable" value="false" />
    </style>
  </indexRule>
  <filter syntax="FeatureValue">confidence >= 0.7</filter>
</indexBuildItem>
</indexBuildSpecification>
```

<skipCondition> エレメント

<skipCondition> エレメントはオプションで、一定のフィーチャー値に基づいて、特定の文書については索引付けを行わないようにする場合に使用します。上記の例では、toBeProcessed というフィーチャーがゼロに設定されている com.ibm.uima.tt.DocumentAnnotation タイプのデータ構造が含まれている文書は、索引付けが行われません。

<indexBuildItem> エレメント

索引作成仕様構成ファイルには、1 つ以上の <indexBuildItem> エレメントが含まれています。各エレメントは、共通分析構造内のある特定のフィーチャーを索引内の構造 (スパンまたはフィールド) にマッピングするための情報を示します。

<name> エレメントには、フィーチャー構造タイプが含まれています。タイプの指定には 2 つの方法があります。

- 完全タイプ名を指定する。例えば、com.ibm.uima.tt.DrugDosage のように指定します。
- ワイルドカードを使用する。例えば、com.ibm.uima.tt.* のように指定します。ワイルドカード文字は、タイプ指定の最後にのみ追加できます。

タイプ A がタイプ B のサブタイプで (「Person」は「Entity」のサブタイプ)、両方のタイプに <indexBuildItem> エレメント Ia と Ib が定義されている場合、処理は以下のようになります。

- タイプ B のフィーチャー構造は Ib に応じて処理される。
- タイプ A のフィーチャー構造は Ia と Ib の組み合わせを使用して処理される。
- Ia と Ib の両方に同じ <attributemappings> エレメントが定義されている場合は、エラーが発生する。Ia は、A に指定されている追加属性のみを扱う、Ib の「true」拡張として定義されなければなりません。

索引作成項目として、uima.tcas.Annotation のサブタイプのみを使用してください。uima.tcas.Annotation のサブタイプが サブタイプ uima.cas.TOP のフィーチャー構造である場合 (かつ、uima.tcas.Annotation のフィーチャー構造でない場合) は、引き続きフィーチャー・パスを使用してこのフィーチャー構造にアクセスできます。

<filter> エレメントはオプションで、 <indexBuildItem> によるマッピングを特定の属性値を持つフィーチャー構造のみに制限する際に使用します。これは、何に対して索引付けを行うかについてのスイッチとして属性を使用する場合に便利です。例えば、個人 (person) と組織 (organization) が EntityAnnotation タイプの注釈に記載されているとします。 その type というフィーチャーは、person または organization のいずれかに設定されます。 個人 (person) のみを抽出し、組織 (organization) は抽出しないようにするには、以下のフィルターを追加します。

```
<filter syntax="FeatureValue">type = "person"</filter>
```

さらに、個人 (person) と組織 (organization) を異なるスパン名、 例えば person と organization で索引付けを行うことができます。このようにするには、EntityAnnotation タイプの 2 つの<indexBuildItem> エレメントを定義し、 type フィーチャーで 2 つのフィルターを使用して、個人 (person) または組織 (organization) のいずれかをトリガーするようにします。

いずれのフィルター式も以下の形式になります。

```
<FeaturePath> <Operator> <Literal>
```

ここで、

- FeaturePath は、共通分析構造内のフィーチャー構造の名前です。
- Operator は =、!=、<、<=、> または >= です。< (< のみ) は < と表記することに注意してください。
- Literal は、整数、浮動小数点数 (指数構文はサポートされていません) または二重引用符で囲まれたストリング・リテラル (ストリング内の引用符および円記号は円記号を付けて拡張します) です。

<FeaturePath>、<Operator> および <Literal> は、ブランク・スペースで区切ってください。

以下に有効なフィルターの例を示します。

```
フィーチャー foo に、hello world というストリングが含まれている
<filter syntax="FeatureValue"> foo = "hello world" </filter>

フィーチャー foo に、整数値 42 が含まれている
<filter syntax="FeatureValue"> foo &lt; 42 </filter>

フィーチャー make に値が Chevrolet のフィーチャー company が
あるフィーチャー構造が含まれているフィーチャー・パス make/company
<filter syntax="FeatureValue"> make/company = "Chevrolet" </filter>

フィーチャー bar7 に、浮動小数点値 0.5 が含まれている
<filter syntax="FeatureValue"> bar7 >= 0.5 </filter>
```

<indexRule> エレメント

各 <indexBuildItem> エレメントには、 1 つの <indexRule> エレメントが含まれています。各 <indexRule> エレメントには、共通分析構造内のフィーチャー構造をフィールド、注釈、ブレーク・スタイルとして索引にマップするために必要な情報がすべて含まれています。注釈スタイルおよびフィールド・スタイルは、多くの属性をサポートします。エンタープライズ・サーチの UIMA でサポートされている条件スタイルは使用できません (条件スタイルはスキップされます)。

注釈スタイルおよびフィールド・スタイルの場合、索引に注釈名またはフィールド名を指定する際に、次のような代替手段があります。

- 各フィーチャー構造が索引内で同じ名前でアクセスできる場合は、fixedName を使用します。以下の例で、PersonAnnotation タイプの各フィーチャー構造は、索引内で「Person」というスパンにマップされます。

  ```
  <indexBuildItem>
     <name>com.ibm.tt.PersonAnotation</name>
     <indexRule>
       <style name="Annotation">
         <attribute name="fixedName" value="Person" />
       </style>
     </indexRule>
  <indexBuildItem>
  ```

  これにより、「個人名として Boss が含まれている文書を探す」というような照会が可能になります。 照会は、次のように XML フラグメントを使用して表されます。 `@xmlf2::'<person>Boss</person>'<attribute name="fixedName" value="Person" />`

- 注釈の特定のフィーチャーの値に基づいて異なるスパンを使用してアクセスできるさまざまなエンティティーが注釈にある場合は、nameFeature を使用します。以下の例で、EntityAnnotation は、type というフィーチャーの値に基づいて、person スパンまたは organization スパンとして索引付けが行われます。フィーチャーはフィーチャー・パスであっても構いません。

  ```
  <indexBuildItem>
     <name>com.ibm.tt.EntityAnotation</name>
     <indexRule>
       <style name="Annotation">
  ```

```
      <attribute name="nameFeature" value="type" />
    </style>
  </indexRule>
<indexBuildItem>
```

これにより、「WHO に関する文書を探す」(英単語 who ではなく) というような照会が可能になります。 照会は、限定 XPath 構文では次のように表されます。 `@xmlp::'/organization[ftcontains="WHO"]'`

- 上記の属性がなにも使用されない場合は、<indexBuildItem> エレメント内の注釈タイプのショート・ネームが使用されます。これはデフォルトです。例えば、以下のようになります。

```
<indexBuildItem>
   <name>com.ibm.uima.tutorial.RoomNumber</name>
   <indexRule>
     <style name="Annotation" />
     <style name="Field" />
   </indexRule>
 </indexBuildItem>
```

この <indexBuildItem> エレメントは、 com.ibm.uima.tutorial.RoomNumber によってカバーされるテキストがある RoomNumber という注釈およびフィールドになります。

<style name="Annotation" /> エレメント

<style> エレメントの Annotation は、エンタープライズ・サーチにおけるスパン情報へのアクセス方法を指定します。 fixedName および nameFeature 属性が使用できる以外に、このスタイルは、<attributemappings> エレメントもサポートします。このエレメント内で、フィーチャー構造の値を索引内の結果スパンの属性にマップすることができます。以降、検索式でそのエレメントを使用することができます。

各マッピングは、それぞれ別個の <mapping> エレメント内で行われます。 <feature> エレメントにはフィーチャー・パスが含まれ、<indexName> エレメントには <feature> の値を保管するために索引で使用される属性の名前が含まれます。例えば、以下のようになります。

```
<mapping>
  <feature>make/companyname</feature>
  <indexName>company</indexName>
</mapping>
```

この <mapping> エレメントは、 パス make/companyname にあるフィーチャーの値を索引属性 company に直接保管します。

フィーチャー値の索引属性へのマッピングは、ネストされたフィーチャー構造が多く含まれているなど、テキスト分析時に使用されるタイプ・システムが複雑な場合に特に便利です。 <mapping> エレメントを使用して、関係のある属性を明らかにすることにより、オリジナルのタイプ・システム構造の詳細を知らなくても、照会でこれらの属性を使用することができます。

<style name="Field" /> エレメント

<style> エレメントの Field は、エンタープライズ・サーチにおけるフィールド情報へのアクセス方法を指定します。 fixedName および nameFeature 属性以外にも、以下の属性を設定することができます。以下の属性が設定されていない場合、デフォルトは false になります。

parametric
true に設定すると、パラメトリック検索を使用して、フィールド値を検索できます (例えば、#dosage:>100)。

fieldSearchable
true に設定すると、検索でフィールド値を使用できます (例えば、make:Bayer)。

returnable
true に設定すると、フィールドとその値が検索結果に戻されます。

フィールド情報は、常に検索可能なコンテンツです。つまり、フィールド情報は、通常のキーワード検索でアクセス可能です。

オプション属性 valueFeature は、フィールド値として、どのフィーチャー値をとるかを定義します。フィーチャー構造が注釈で、属性が設定されていない場合、注釈のカバー・テキストがフィールド値として使用されます。以下の例では、

```
<indexBuildItem>
  <name>com.ibm.uima.tt.DrugDosage</name>
  <indexRule>
    <style name="Field">
      <attribute name="fixedName" value="dosage" />
      <attribute name="parametric" value="true" />
      <attribute name="fieldSearchable" value="true" />
      <attribute name="returnable" value="false" />
    </style>
    <style name="Field">
      <attribute name="fixedName" value="make" />
      <attribute name="valueFeature" value="make/companyname" />
      <attribute name="parametric" value="false" />
      <attribute name="fieldSearchable" value="true" />
      <attribute name="returnable" value="false" />
    </style>
  </indexRule>
</indexBuildItem>
```

DrugDosage に対して 2 つのフィールドが生成されます。 dosage というフィールドには、カバー・テキスト (例えば 100) が含まれます。この場合、「#dosage::>100」を使用して照会することができます。もう 1 つのフィールドには、フィーチャー・パス make/companyname の属性 companyname の値が含まれます。この場合、「make:Bayer」を使用して照会することができます。

<style name="Breaking" /> エレメント

<style> エレメントの値 Breaking には、これ以外のエレメントは含まれません。

エンタープライズ・サーチに定義されているタイプおよびフィーチャー

エンタープライズ・サーチに定義されているタイプ・システムは、文書メタデータ処理 および基本的な言語分析をカバーします。

文書の言語認識およびセグメンテーションを行う基本的な言語分析は、カスタム分析が選択されているかどうかにかかわらず、文書の索引付けを行う際に常に実行されます。基本的な文書の分析時に、カスタム分析で使用できる共通分析構造に以下の情報が追加されます。

- 文書メタデータ (com.ibm.es.tt.DocumentMetaData タイプ)。
- トークン、センテンス、およびパラグラフ注釈 (uima.tt.TokenAnnotation、uima.tt.SentenceAnnotation および uima.tt.ParagraphAnnotation タイプ)トークン注釈には、フィーチャー lemma が含まれています。

エンタープライズ・サーチに定義されているタイプ・システムには、テキスト分析に固有の複雑なタイプやフィーチャーは含まれていません。これらは、UIMA 環境でユーザー固有のカスタム分析を作成する際に使用 (および拡張) できる UIMA タイプ・システムに含まれます。

ユーザーのカスタム分析に必要な新しいタイプを組み込むために拡張する UIMA タイプ・システムと違い、エンタープライズ・サーチ・タイプ・システムはほとんど拡張する必要がありません。

エンタープライズ・サーチ・タイプ・システムは、 UIMA Software Development Kit (SDK) には定義されていません。UIMA でアノテーターを作成する際にエンタープライズ・サーチ・タイプ・システムのいずれかのタイプを使用する場合 (文書のセキュリティー情報にアクセスする場合や、クローラー・タイプや文書タイプにアクセスする場合など) には、分析エンジンのシステム記述にこれらのタイプを再度定義する必要があります。

エンタープライズ・サーチには、以下のタイプおよびフィーチャーが定義されています。

uima.tcas.Annotation
注釈は以下のタイプからなります。

uima.tcas.DocumentAnnotation
文書注釈には、以下のフィーチャーがあります。

esDocumentMetaData
com.ibm.es.tt.DocumentMetaData タイプの文書メタデータが含まれています。

com.ibm.es.tt.ContentField
コンテンツ・フィールド注釈には、以下のフィーチャーがあります。

parameters
com.ibm.es.tt.CommonFieldParameters タイプのコンテンツ・フィールド・パラメーター。

com.ibm.es.tt.Anchor
HTML 文書のアンカー・テキスト用のアンカー注釈。以下のフィーチャーがあります。

uri アンカー・テキストのターゲット URI。フィーチャー値は、uima.cas.String タイプです。

com.ibm.es.tt.MarkupTag

マークアップ情報注釈。例えば、XML タグの注釈など。マークアップ情報は、以下のフィーチャーに保管されています。

name マークアップ・タグの名前。フィーチャー値は、uima.cas.String タイプです。

depth ネストの深さ。フィーチャー値は、uima.cas.Integer タイプです。

attributeName

フィーチャー属性の名前。フィーチャー値は、uima.cas.StringArray タイプです。

attributeValues

属性の値のストリング。フィーチャー値は、uima.cas.StringArray タイプです。

uima.CAS.TOP

タイプ・システムのルート。以下のタイプがあります。

com.ibm.es.tt.DocumentMetaData

文書メタデータには、以下のフィーチャーがあります。フィーチャーは、文書注釈フィーチャー esDocumentMetaData に結び付けられます。

crawlerId

クローラー名。フィーチャー値は、uima.cas.String タイプです。

dataSource

以下のいずれかのデータ・ソース・タイプ。

- Web (Web クローラーによる文書)
- NNTP (ニュース・グループ・クローラーによる文書)
- DB2 (DB2 クローラーによる文書)
- Notes® (Notes クローラーによる文書)
- CM (コンテンツ・マネージメント・クローラーによる文書)
- FS (UNIX® ファイル・システム・クローラーによる文書)
- WinFS (Windows ファイル・システム・クローラーによる文書)
- Exchange (Exchange クローラーによる文書)
- VBR (VeniceBridge クローラーによる文書)

フィーチャー値は、uima.cas.String タイプです。

dataSourceName

クローラー (データ・ソース) の名前。フィーチャー値は、uima.cas.String タイプです。

charset

文書コード・ページ。フィーチャー値は、uima.cas.String タイプです。

docType

以下のいずれかの文書タイプ。

- text/html
- application/postscript
- application/pdf
- application/x-mspowerpoint
- application/msword
- application/x-msexcel
- application/rtf
- application/vnd.lotus-wordpro
- application/x-lotus-123
- application/vnd.lotus-freelance
- text/xml
- text/plain
- application/x-js-taro (一太郎)

フィーチャー値は、`uima.cas.String` タイプです。

securityTokens

文書のセキュリティー・トークン。フィーチャー値は、`uima.cas.StringArray` タイプです。

date 文書の日付。フィーチャー値は、`uima.cas.String` タイプです。

baseUri

ページの基本 URI。フィーチャー値は、`uima.cas.String` タイプです。

metaDataFields

フィーチャー値は、`uima.cas.FSArray` タイプです。この配列の各エレメントは、`com.ibm.es.tt.MetaDataField` タイプです。

redirectUrl

リダイレクトされた URL。フィーチャー値は、`uima.cas.String` タイプです。

contentLanguage

メタデータ設定を使用してユーザーが定義したコンテンツの言語。フィーチャー値は、`uima.cas.String` タイプです。

hasSeparateContent

文書にコンテンツおよびメタデータがあるかどうかを示すフラグ。

mimeType

MIME タイプ、または文書タイプ。例えば、XML など。フィーチャー値は、`uima.cas.String` タイプです。

metaLanguage
メタデータの言語。フィーチャー値は、uima.cas.String タイプです。

url 文書の URL。フィーチャー値は、uima.cas.String タイプです。

com.ibm.es.tt.CommonFieldParameters
共通フィールド・パラメーターには、以下が含まれています。

searchable
フィールドが検索可能かどうかを示すフラグ。

fieldSearchable
フィールドがフィールドとして検索可能かどうかを示すフラグ。

parametric
パラメトリック検索を示すフラグ。

showInSearchResult
検索結果詳細に注釈付きのデータが含まれているかどうかを示すフラグ。

resolveConflict
MetadataPreferred、ContentPreferred、および Coexist 間のメタデータの競合を解決するフラグ。フィーチャー値は、uima.cas.String タイプです。

name フィールドの名前。フィールド名を使用して、このフィールドを検索することができます。フィーチャー値は、uima.cas.String タイプです。

com.ibm.es.tt.MetaDataField
メタデータ・フィールド・データは、文書コンテンツの一部ではありませんが、「text」フィーチャーに保管されます。

parameters
タイプ com.ibm.es.tt.CommonFieldParameters のメタデータ・フィールド・パラメーター。

text メタデータ・テキストは、タイプ uima.cas.String のこのフィーチャーに保管されます。

UIMA に定義されているタイプおよびフィーチャー

UIMA Software Development Kit は、テキスト分析時に文書内で検出される可能性がある いくつかの基本的な言語タイプおよびフィーチャーを定義します。

各分析エンジンには、分析エンジンに含まれているアノテーターの入力要件および出力タイプを示す、固有のタイプ・システム記述子があります。タイプ・システム記述は、ドメインおよびアプリケーションに固有のものです。

UIMA タイプ・システムにユーザー独自のタイプおよびフィーチャーが含まれるように拡張することができます。UIMA 環境には、アノテーターのタイプ・システム

記述子の編集を支援する Eclipse プラグインがあります。 Component Descriptor Editor プラグインのインストールおよび使用についての詳細は、UIMA の資料を参照してください。

UIMA 環境で、分析エンジンの開発およびテストを完了したら、分析エンジン・ファイルが含まれるユーザー作成のアーカイブ・ファイル (.pear) にもタイプ・システム記述を組み込みます。

以降のセクションに、UIMA に定義されているタイプおよびフィーチャーをリストします。

uima.tcas.Annotation
注釈は以下のタイプからなります。

uima.tcas.DocumentAnnotation

uima.tt.TTAnnotation

uima.tcas.DocumentAnnotation
文書注釈には、以下のフィーチャーがあります。

categories
文書のカテゴリー名またはラベルのリスト。フィーチャー値は、`uima.cas.FSList` タイプです。

languageCandidates
文書の言語参照のリスト。フィーチャー値は、`uima.cas.FSList` タイプです。

id 文書識別形式。例えば URL など。フィーチャー値は、`uima.cas.String` タイプです。

uima.tt.TTAnnotation
TT 注釈には、以下のタイプがあります。

uima.tt.DocStructureAnnotation
文書に関する構造的な情報。文書構造注釈には、以下のタイプがあります。

uima.tt.SentenceAnnotation
開始句読点および終了句読点を含むセンテンス。これには、以下のフィーチャーがあります。

sentenceNumber
パラグラフ内のセンテンスのシーケンス番号。各パラグラフの先頭で 1 にリセットします。フィーチャー値は、`uima.cas.Integer` タイプです。

uima.tt.ParagraphAnnotation
パラグラフ。これには、以下のフィーチャーがあります。

paragraphNumber
パラグラフのシーケンス番号。フィーチャー値は、`uima.cas.Integer` タイプです。

uima.tt.LexicalAnnotation
文書に関するコンテンツ情報。字句の注釈は、以下のタイプからなります。

uima.tt.CompPartAnnotation
複合語の一部。多くのゲルマン言語の複合語は、ブランクで区切らずに一緒に書かれています。例えば、ドイツ語の ″Abteilungsleiter″ (部門管理者) という語は、 ″Abteilung″ (部門) と ″Leiter″ (管理者) から構成されています。

uima.tt.TokenAnnotation
空白で囲まれていないトークン。これには、以下のフィーチャーがあります。

lemma lemmaEntries に指定されているトークンに対して有効なすべての見出し語のうちの 1 つの見出し語。各項目は、トークンに有効なディクショナリー項目です。

lemmaEntries
指定されたトークンに対して有効なすべての見出し語のリスト。各項目は、トークンに有効なディクショナリー項目です。

tokenNumber
センテンス内のトークンのシーケンス番号。各センテンスの先頭で 1 にリセットします。フィーチャー値は、`uima.cas.Integer` タイプです。

tokenProperties
トークンのプロパティー。例えば、uppercase、numerics など。フィーチャー値は、`uima.cas.Integer` タイプです。

stopwordToken
ストップワードとしてマークされているトークン。フィーチャー値は、`uima.cas.Integer` タイプです。

synonymEntries
`uima.tt.Synonym` タイプの項目に対する参照リスト。各項目は、トークンに有効なシノニム項目です。

normalizedCoveredText
注釈によってカバーされるテキストの正規化表現。フィーチャー値は、 `uima.cas.String` タイプです。

uima.CAS.TOP
タイプ・システムのルート。以下のタイプがあります。

uima.tt.KeyStringEntry
以下のフィーチャーがあるストリング。

key 実際のストリング。

uima.tt.Lemma
以下の形態素情報を持つディクショナリー項目。

partOfSpeech
見出し語の品詞の整数エンコード。

morphID
形態素情報の整数エンコード。

uima.tt.Synonym
指定された uima.tt.keyStringEntry タイプの語に対するシノニム項目。

uima.tt.LanguageConfidencePair
文書の言語選択を示す以下のフィーチャーがあるタイプ。

uima.tt.LanguageConfidencePair

languageConfidence
選択された言語が文書の言語に実際にどの程度適合するかを示す標識 (0 から 1 の間の浮動小数点)。

language
文書の言語 (ISO 値)。値は、uima.cas.String タイプです。

languageID
言語 ID。値は、uima.cas.Integer タイプです。

uima.tt.CategoryConfidencePair
文書のカテゴリー選択を示す以下のフィーチャーがあるタイプ。

uima.tt.CategoryConfidencePair
カテゴリーには、以下のフィーチャーがあります。

categoryString
カテゴリーの名前。値は、uima.cas.String タイプです。

categoryConfidence
カテゴリーが文書にどの程度適合するかを示す標識。値は浮動小数点タイプです。

mostSpecific
カテゴリーが文書に最も適しているかどうかを示すフラグ (uima.cas.Integer タイプ)。

taxonomy
カテゴリーが属する分類法の名前。文書は、異なる分類法のカテゴリーを持つことができます。値は、uima.cas.String タイプです。

セマンティック検索アプリケーション

Search and Index API (SIAPI) インターフェースを使用して検索アプリケーションで照会できる、 エンタープライズ・サーチ索引に保管される文書情報には、4 つのタイプがあります。

以下の 4 つの異なるタイプの情報があります。

- 文書で検出されるテキストの語。例えば、computer software など。

- スパン名。例えば、<author>James</author> が含まれている XML 文書では、スパン <author> が生じます。
- 属性名。例えば、<author countryOfBirth=USA>James</author> が含まれている XML 文書では、属性「countryOfBirth」が生じます。
- 属性値。例えば、USA は属性「countryOfBirth」の値です。

SIAPI 照会言語は、セマンティック検索照会条件を組み込めるように拡張されています。条件は、twig (小枝) のパターンを指定します。twig とは、小さなツリーで、その葉が上記の 4 つのタイプの語になります。ツリーの内部ノードが、文書内のオカレンスの相互関係を指定します。関連を指定する内部ノードには、以下の 5 つのタイプがあります。

- and
- or
- not
- in_the_span_of
- attribute_in_the_span_of

文書に葉のオカレンスが含まれており、内部ノードによって指定された制約 (定義されている関係) が成り立つ場合に、その文書は指定されたセマンティック検索条件を満たしていることになります。

セマンティック検索照会条件は、より的確な文書の検索に役立ちます。語および注釈のブール組み合わせを使用した検索だけでなく、例えば、author というスパンに James が含まれている文書を検索したり、同じセンテンス内に「ibm」と「search」という用語がある文書を検索することができるようになります。

関連概念

『セマンティック検索照会』
セマンティック検索照会条件は、不透明条件と呼ばれます。

セマンティック検索照会

セマンティック検索照会条件は、不透明条件と呼ばれます。

Search and Index API (SIAPI) で不透明条件を表す構文には、以下の 2 つの形式があります。

- XML フラグメント
- 限定 XPath

正しく定義された XML 文書フラグメントのように見えます。 XML フラグメント照会条件は、不透明条件記号 `@xmlf2::` の後に単一引用符 ('...') で囲まれた XML フラグメント式が続きます。

これに対して、限定 XPath 照会用語は、`@xmlxp::` の後に 単一引用符 ('...') で囲まれた XPath 照会が続きます。

Search and Index API (SIAPI) インターフェースの一般的な照会用語と同じように、各用語に出現修飾子を付けることができます。

正符号 (+)
必ずその用語がなければなりません。

= (接頭部)
用語が完全一致していなければなりません。

波形記号 (~) (接頭部)
照会用語のシノニムも考慮します。

波形記号 (~) (接尾部)
照会用語と同じ見出し語を持つ語も考慮します。

以下に XML フラグメント照会の例を示します。

@xmlf2::'<title>″Data Structures″</title>'
「Data Structures」という句が含まれているスパン (注釈) title が含まれている文書を 検索します。

@xmlf2::'<author country=″USA″></author>'
作成者 (author) が USA 出身の文書を検索します。

@xmlf2::'<book><.or><author>John Smith</author><title>XML -Microsoft</title><!.or></book>'
本 (book) の作成者 (author) が John Smith であるか、 または 本 (book) の 表題 (title) に XML という語が含まれており Microsoft® という語が含まれていない文書を検索します。

対応する XPath 照会には、以下の構造があります。

@xmlxp::'/booktitle[ftcontains(″Data Structures″)]'
「Data Structures」という句が含まれているスパン (注釈) booktitle が含まれている文書を 検索します。

@xmlxp::'//author[@country=″USA″]'
作成者 (author) が USA 出身の文書を検索します。

@xmlxp::'/book[author[ftcontains(″Jane Smith″)] or title[ftcontains(″XML -Microsoft″)]]'
本 (book) の作成者 (author) が Jane Smith であるか、 または 本 (book) の 表題 (title) に XML という語が含まれており Microsoft という語が含まれていない文書を検索します。

関連概念

78 ページの『セマンティック検索アプリケーション』
Search and Index API (SIAPI) インターフェースを使用して検索アプリケーションで照会できる、 エンタープライズ・サーチ索引に保管される文書情報には、4 つのタイプがあります。

第 7 章 エンタープライズ・サーチに組み込まれているテキスト分析

エンタープライズ・サーチに組み込まれているテキスト分析は、言語検出とセグメンテーション から構成されています。

文書処理時に、エンタープライズ・サーチは、文書の言語を判別し、入力テキストのストリームを別個の単位またはトークンに分割します。

検索時に、ユーザー、つまりアプリケーション・デフォルトは、手動で照会言語を選択する必要があります。照会ストリングは、セグメント化され、分析され、索引内で検索されます。

文書分析も照会ストリング分析も、以下に分割されます。

- 基本的な非辞書ベースのサポート。これには、空白によるセグメンテーションと N-gram セグメンテーションがあります。
- 辞書ベースの言語サポート。これには、語およびセンテンス・セグメンテーションと見出し語処理があります。

言語処理では、字句解析が行われます。これは、入力テキストの代替表記を作成する処理で、有効なすべての辞書データを、入力テキストにおいて認識されたトークンに関連付けます。拡張言語処理を使用することにより、検索品質は一段と向上します。

関連概念

『言語の識別』
エンタープライズ・サーチでは、語およびセンテンスのセグメンテーション、文字の正規化、 見出し語処理を行う前に、ソース・ドキュメントの言語を判別 する必要があります。

82 ページの『非辞書ベース・セグメンテーションに関する言語サポート』
言語検出および字句解析テクノロジーによってサポートされない言語による文書の場合、 エンタープライズ・サーチは、Unicode ベースの空白によるセグメンテーションおよび N-gram セグメンテーション の形式で基本サポートを提供します。

言語の識別

エンタープライズ・サーチでは、語およびセンテンスのセグメンテーション、文字の正規化、 見出し語処理を行う前に、ソース・ドキュメントの言語を判別 する必要があります。

エンタープライズ・サーチは、以下の言語を自動的に検出することができます。

アラビア語
中国語 (繁体字および簡体字)
チェコ語
デンマーク語
フランス語
ドイツ語
ギリシャ語
ヘブライ語
韓国語
ポーランド語
ポルトガル語
ロシア語

オランダ語		ハンガリー語	スペイン語
英語		イタリア語	スウェーデン語
フィンランド語		日本語	トルコ語

エンタープライズ・サーチの言語処理では、照会処理時ではなく、索引作成時にソース・ドキュメントの言語を検出します。

言語を自動的に検出できない文書は、言語に依存しない基本的なテクノロジーで処理されます。

エンタープライズ・サーチの言語検出テクノロジーは、単一言語文書に最も適しています。文書が複数の言語で書かれている場合は、その文書で最も多く使用されている言語を判別します。ただし、その場合、分析結果が常に満足できるものであるとは限りません。

エンタープライズ・サーチの言語検出テクノロジーを使用して、検索結果を特定言語で書かれている文書のみに制限することができます。例えば、 Jacques Chirac に関する文書を検索する場合、検索結果にフランス語で書かれている文書のみが含まれるように指定することができます。

関連概念

81 ページの『第 7 章 エンタープライズ・サーチに組み込まれているテキスト分析』
エンタープライズ・サーチに組み込まれているテキスト分析は、言語検出とセグメンテーション から構成されています。

『非辞書ベース・セグメンテーションに関する言語サポート』
言語検出および字句解析テクノロジーによってサポートされない言語による文書の場合、 エンタープライズ・サーチは、Unicode ベースの空白によるセグメンテーションおよび N-gram セグメンテーション の形式で基本サポートを提供します。

非辞書ベース・セグメンテーションに関する言語サポート

言語検出および字句解析テクノロジーによってサポートされない言語による文書の場合、 エンタープライズ・サーチは、Unicode ベースの空白によるセグメンテーションおよび N-gram セグメンテーション の形式で基本サポートを提供します。

Unicode ベースの空白によるセグメンテーション
この言語処理方式は、語間の空白 (またはブランク・スペース) を語の区切り文字として使用します。

N-gram セグメンテーション
この言語処理方式は、*n* 文字のオーバーラップするシーケンスを単一の語として扱います。この簡単なセグメンテーション方式は、多くの検索タスクで十分に使用できます。

これらの方式は、言語辞書に依存せず、基本型への変換などの複雑な言語処理テクノロジーも組み込まれていません。

N-gram セグメンテーションは、区切り文字としてブランク・スペースを使用しないタイ語などの言語に使用されます。同じ方式が、ヘブライ語やアラビア語に適用されます。これらの 2 つの言語は、空白区切り文字を使用しますが、N-gram セグメ

ンテーションを使用した方が、Unicode ベースの空白によるセグメンテーションの基本形式を使用するより、良い結果が得られます。

関連概念

81 ページの『第 7 章 エンタープライズ・サーチに組み込まれているテキスト分析』
エンタープライズ・サーチに組み込まれているテキスト分析は、言語検出とセグメンテーション から構成されています。

81 ページの『言語の識別』
エンタープライズ・サーチでは、語およびセンテンスのセグメンテーション、文字の正規化、 見出し語処理を行う前に、ソース・ドキュメントの言語を判別 する必要があります。

辞書ベース・セグメンテーションに関する言語サポート

文書の言語が正しく検出され、言語固有の辞書が使用可能である場合には、 該当する言語処理が適用されます。

セグメンテーションとは、入力テキストを個別の字句単位にブレークダウンするプロセスのことです。このプロセスは、以下のいくつかの言語処理アクティビティーから構成されます。

語のセグメンテーション

語のセグメンテーションは、日本語や中国語のように、語の間に空白 (または区切り文字) を使用しない言語に使用されます。

見出し語処理

見出し語処理は、テキスト内のそれぞれの語形の見出し語を判別する言語処理形式です。語の *見出し語* は、語の基本型に加えて、同じ品詞を共用する語形変化型も含めます。 例えば、見出し語 go には、go、goes、went、gone、および going が含まれます。名詞グループの見出し語には、単数形と複数形が含まれます (calf と calves など)。形容詞グループの見出し語には、比較級、最上級形が含まれます (good、better、best など)。代名詞グループの見出し語には、同じ代名詞のさまざまな格が含まれます (I、me、my、mine など)。

見出し語処理では、索引付けと検索の両方に辞書が必要です。

エンタープライズ・サーチは、見出し語と語形変化した語の索引付けを行い、照会内のすべての語形変化した語に見出し語を対応させます。見出し語処理は、照会において、さまざまな語形変化した語が含まれている文書を検出することにより、検索の品質を向上させます。例えば、照会に *mouse* という語が含まれている場合、 *mice* という語が含まれている文書も検出されます。

短縮形のo分割

短縮形を識別して、それをコンポーネント・パーツに分割することによって、検索の品質が向上します。例えば、以下のようになります。

wouldn't は *would* と *not* に分割されます。
Horse's は *Horse* と *is* または *'s* に分割されます (照会のあいまいさを考慮するため)

接語の識別

接語は特殊な形式の短縮形で、接語の構成要素を判別することにより、検索の品質が向上します。*接語* は、接辞および語のような性質を持っています。ただし、接語は、語形成の一部でもあるため、識別が難しくなります。他の形態構造的な (語構造) 事象と異なり、接語は統語的な構造内にあり、語に結び付いている接語は、語形成規則の一部ではありません。例えば、以下のようになります。

reparti-lo-emos には、 *repartir* と *lo* と *emos* の構成要素があります。
l'avenue には、*le* と *avenue* の構成要素があります。
dell'arte には、*dello* と *arte* の構成要素があります。

英字以外の文字認識

言語処理は、英字以外の文字を認識します。英字以外の文字は、内部的な言語依存ロジックに従って、さまざまなタイプの個別の字句単位として戻されたり、グループ化されたりします。

例えば、アポストロフィやハイフンは、接語内にある場合は語の一部とみなされ、不明な省略形内にある場合は終止符 (またはピリオド) とみなされます。また、言語処理は、一部の特殊なシーケンスの文字 (例えば、URL、E メール・アドレス、日付など) をトークンとして認識します。

省略形の認識

言語処理は、辞書にある省略形を 1 つの字句単位として認識します。省略形が辞書に無い場合、その省略形は字句項目として認識されますが、その省略形には関連した辞書情報がありません。

省略形を正しく認識することは、文の認識においてきわめて重要です。例えば、省略形の語尾にあるピリオドは、必ずしもセンテンスの終わりを示すものではありません。

センテンスの終わりを示すマーカーの認識

言語処理は、センテンスのセグメンテーションのためにセンテンスの終わりを示すマーカーを正しく識別します。

辞書ベースの言語サポートは、以下の言語で有効です。

中国語 (繁体字および簡体字)	イタリア語
チェコ語	日本語
デンマーク語	韓国語
オランダ語	ノルウェー語 (ブークモールおよびニーノシュク)
英語	ポーランド語
フィンランド語	ポルトガル語 (本国およびブラジルで使用されているもの)
フランス語 (本国およびカナダで使用されているもの)	ロシア語
ドイツ語 (本国およびスイスで使用されているもの)	スペイン語
ギリシャ語	スウェーデン語

関連概念

85 ページの『日本語における語のセグメンテーション』
テキスト文書または照会ストリングが日本語であると認識されると、 エンタープライズ・サーチは、日本語に最適化された形態学的な分析を使用して、 適切な語のセグメンテーションを行います。

『日本語における変種文字』
日本語は、多数の変種文字を使用します。カタカナは、外来語のスペルや 発音によく使用されるため、最も重要です。 日本語では、多数のカタカナがよく使用されます。

日本語における語のセグメンテーション

テキスト文書または照会ストリングが日本語であると認識されると、 エンタープライズ・サーチは、日本語に最適化された形態学的な分析を使用して、 適切な語のセグメンテーションを行います。

この最適化の例が語分解です。日本語は、多数の複合語を使用します。これらの語は、検索結果の精度を向上させるために、最適なサイズのトークンに分解されます。語形変化した語句や接頭語も、検索効率を上げるために、分解されます。

関連概念

83 ページの『辞書ベース・セグメンテーションに関する言語サポート』
文書の言語が正しく検出され、言語固有の辞書が使用可能である場合には、 該当する言語処理が適用されます。

『日本語における変種文字』
日本語は、多数の変種文字を使用します。カタカナは、外来語のスペルや 発音によく使用されるため、最も重要です。 日本語では、多数のカタカナがよく使用されます。

日本語における変種文字

日本語は、多数の変種文字を使用します。カタカナは、外来語のスペルや 発音によく使用されるため、最も重要です。 日本語では、多数のカタカナがよく使用されます。

エンタープライズ・サーチは、変種文字対応辞書を使用して、標準的なカタカナをその基本型 (見出し語に類似したもの) にマップし、照会ストリングにカタカナが含まれている文書を含む、すべての文書を検索できるようにします。

また、エンタープライズ・サーチは、漢字の語尾にひらがなで書かれている標準的な送り仮名もサポートします。

関連概念

83 ページの『辞書ベース・セグメンテーションに関する言語サポート』
文書の言語が正しく検出され、言語固有の辞書が使用可能である場合には、 該当する言語処理が適用されます。

『日本語における語のセグメンテーション』
テキスト文書または照会ストリングが日本語であると認識されると、 エンタープライズ・サーチは、日本語に最適化された形態学的な分析を使用して、 適切な語のセグメンテーションを行います。

ストップワードの除去

エンタープライズ・サーチでは、検索効率を上げるために、 複数語照会からすべてのストップワード (例えば *a* や *the* などの共通の語) が除去されます。

日本語におけるストップワードの認識は、文法的な情報に基づいて行われます。例えば、エンタープライズ・サーチは、他の言語では特殊なリストに基づいてストップワードを認識しますが、日本語については、語が名詞であるか動詞であるかに基づいてストップワードを認識します。

関連概念

『文字の正規化』
文字の正規化は、想起性を改善するプロセスです。文字を正規化して想起性を改善すると、文書が照会に完全に一致していなくても、より多くの文書が検索されることになります。

文字の正規化

文字の正規化は、想起性を改善するプロセスです。文字を正規化して想起性を改善すると、文書が照会に完全に一致していなくても、より多くの文書が検索されることになります。

エンタープライズ・サーチは、アジア言語の全角および半角文字の正規化を含む Unicode 互換の正規化を使用します。

例えば、日本語では、全角の英数字は半角文字に正規化され、半角のカタカナは全角文字に正規化されます。また、エンタープライズ・サーチは、日本語で複合語の区切り文字として使用されるカタカナの中黒を除去します。

文字の正規化のその他の形式には、以下のものがあります。

大/小文字の正規化
例えば、*usa* と指定された検索で *USA* が含まれている文書を検索します。

ウムラウトの拡張
例えば、*schön* と指定された検索で *schoen* が含まれている文書を検索します。

アクセントの除去
例えば、*e* と指定された検索で *é* が含まれている文書を検索します。

その他の発音符の除去
例えば、*c* と指定された検索で *ç* が含まれている文書を検索します。

合字の拡張
例えば、*ae* と指定された検索で *Æ* が含まれている文書を検索します。

すべての正規化は、両方向に作用します。*USA* と指定した検索で *usa* が含まれている文書を検索することも、*é* と指定した検索で *e* の付く語が含まれている文書を検索することもできます。これらの正規化を組み合わせて使用することもできます。例えば、*METEO* と指定した検索で *météo* が含まれている文書を検索することができます。

正規化は、Unicode 文字特性に基づいており、言語に依存しません。例えば、エンタープライズ・サーチは、ヘブライ語における発音符の除去、およびアラビア語における合字の拡張をサポートします。

関連概念

85 ページの『ストップワードの除去』
エンタープライズ・サーチでは、検索効率を上げるために、 複数語照会からすべてのストップワード (例えば *a* や *the* などの共通の語) が除去されます。

DB2 Information Integrator の資料

ここでは、DB2 Information Integrator の資料についての情報を提供します。

次のトピックの表は、正式な資料名、資料番号、および PDF 文書の場所を示しています。ハードコピー版の資料を注文するには、正式な資料名または資料番号が必要です。 DB2 Information Integrator のリリース情報とインストール要件の資料名、ファイル名、および場所についても、以下のトピックの中に含まれています。

z/OS 上の DB2 Universal Database のイベント・パブリッシング機能に関する資料

z/OS 上の DB2 Universal Database のイベント・パブリッシング機能に関する資料

目的

z/OS 上の DB2 Universal Database のイベント・パブリッシング機能に関する資料。

表 3. z/OS 上の DB2 Universal Database のイベント・パブリッシング機能に関する DB2 Information Integrator の資料

資料名	資料番号	場所
ASNCLP Program Reference for Replication and Event Publishing	なし	DB2 Information Integrator Support の Web サイト
レプリケーションとイベント・パブリッシング 入門	GC88-9895	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
レプリケーションとイベント・パブリッシング ガイドおよびリファレンス	SC88-9893	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
Tuning for Replication and Event Publishing Performance	なし	DB2 Information Integrator Support の Web サイト
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	なし	• 「DB2 インフォメーション・センター」で、「**製品概要**」>「**インフォメーション・インテグレーション**」>「**DB2 Information Integrator 概説**」>「**問題、予備手段、および資料の更新**」 • DB2 Information Integrator のインストール・ランチパッド • DB2 Information Integrator Support の Web サイト • *DB2 Information Integrator* 製品 CD

z/OS 上の IMS および VSAM のイベント・パブリッシング機能に関する資料

z/OS 上の IMS および VSAM のイベント・パブリッシング機能に関する資料

目的

z/OS 上の IMS および VSAM のイベント・パブリッシング機能に関する資料。

表 4. z/OS 上の IMS および VSAM のイベント・パブリッシング機能に関する DB2 Information Integrator の資料

資料名	資料番号	場所
Classic Federation および Classic Event Publishing のクライアント・ガイド	SD88-7512	DB2 Information Integrator Support の Web サイト
Classic Federation および Classic Event Publishing の Data Mapper ガイド	SD88-7515	DB2 Information Integrator Support の Web サイト
Classic Event Publishing はじめに	GD88-7516	DB2 Information Integrator Support の Web サイト
Classic Federation および Classic Event Publishing のインストール・ガイド	GD88-7517	DB2 Information Integrator Support の Web サイト
Classic Event Publishing オペレーション・ガイド	SD88-7510	DB2 Information Integrator Support の Web サイト
Classic Event Publishing 計画ガイド	SD88-7511	DB2 Information Integrator Support の Web サイト
Classic Federation および Classic Event Publishing の管理ガイド	SD88-7509	DB2 Information Integrator Support の Web サイト
Classic Federation および Classic Event Publishing のシステム・メッセージ	SD88-7514	DB2 Information Integrator Support の Web サイト
IBM DB2 Information Integrator Classic Event Publisher for IMS リリース情報	なし	DB2 Information Integrator Support の Web サイト
IBM DB2 Information Integrator Classic Event Publisher for VSAM リリース情報	なし	DB2 Information Integrator Support の Web サイト

Linux、UNIX、および Windows におけるイベント・パブリッシングおよびレプリケーション機能に関する資料

Linux、UNIX、および Windows におけるイベント・パブリッシングおよびレプリケーション機能に関する資料

目的

Linux、UNIX、および Windows におけるイベント・パブリッシングおよびレプリケーション機能に関する資料。

表 5. Linux、UNIX、および Windows 上のイベント・パブリッシングおよびレプリケーション機能に関する DB2 Information Integrator の資料

資料名	資料番号	場所
ASNCLP Program Reference for Replication and Event Publishing	なし	DB2 Information Integrator Support の Web サイト
インストール・ガイド *(Linux、UNIX、Windows 版)*	GC88-9562	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
レプリケーションとイベント・パブリッシング 入門	GC88-9895	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
Migrating to SQL Replication	なし	DB2 Information Integrator Support の Web サイト
レプリケーションとイベント・パブリッシング ガイドおよびリファレンス	SC88-9893	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
SQL レプリケーション・ガイドおよびリファレンス	SC88-9163	DB2 Information Integrator Support の Web サイト
Tuning for Replication and Event Publishing Performance	なし	DB2 Information Integrator Support の Web サイト
Tuning for SQL Replication Performance	なし	DB2 Information Integrator Support の Web サイト
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	なし	• 「DB2 インフォメーション・センター」で、「**製品概要**」>「**インフォメーション・インテグレーション**」>「**DB2 Information Integrator 概説**」>「**問題、予備手段、および資料の更新**」 • DB2 Information Integrator のインストール・ランチパッド • DB2 Information Integrator Support の Web サイト • *DB2 Information Integrator* 製品 CD

Linux、UNIX、および Windows におけるフェデレーテッド機能に関する資料

Linux、UNIX、および Windows におけるフェデレーテッド機能に関する資料

目的

Linux、UNIX、および Windows におけるフェデレーテッド機能に関する資料。

表 6. Linux、UNIX、および Windows 上のフェデレーテッド機能に関する DB2 Information Integrator の資料

資料名	資料番号	場所
アプリケーション開発者向けガイド	SC88-9609	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
ラッパー開発における C++ API リファレンス	SC88-9921	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
データ・ソース構成ガイド	なし	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
フェデレーテッド・システム・ガイド	SC88-9614	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
Guide to Configuring the Content Connector for VeniceBridge	なし	DB2 Information Integrator Support の Web サイト
インストール・ガイド (Linux、UNIX、Windows 版)	GC88-9562	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
ラッパー開発における Java API リファレンス	SC88-9922	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
マイグレーション・ガイド	SC88-9610	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
ラッパー開発者向けガイド	SC88-9923	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト

表 6. *Linux、UNIX、および Windows 上のフェデレーテッド機能に関する DB2 Information Integrator の資料 (続き)*

資料名	資料番号	場所
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	なし	• 「DB2 インフォメーション・センター」で、「**製品概要**」>「**インフォメーション・インテグレーション**」>「**DB2 Information Integrator 概説**」>「**問題、予備手段、および資料の更新**」 • DB2 Information Integrator のインストール・ランチパッド • DB2 Information Integrator Support の Web サイト • *DB2 Information Integrator* 製品 CD

z/OS におけるフェデレーテッド機能に関する資料

z/OS におけるフェデレーテッド機能に関する資料

目的

z/OS におけるフェデレーテッド機能に関する資料。

表 7. *z/OS 上のフェデレーテッド機能に関する DB2 Information Integrator の資料*

資料名	資料番号	場所
Classic Federation および Classic Event Publishing のクライアント・ガイド	SD88-7512	DB2 Information Integrator Support の Web サイト
Classic Federation および Classic Event Publishing の Data Mapper ガイド	SD88-7515	DB2 Information Integrator Support の Web サイト
Classic Federation はじめに	GD88-7508	DB2 Information Integrator Support の Web サイト
Classic Federation および Classic Event Publishing のインストール・ガイド	GD88-7517	DB2 Information Integrator Support の Web サイト
Classic Federation および Classic Event Publishing の管理ガイド	SD88-7509	DB2 Information Integrator Support の Web サイト
Classic Federation および Classic Event Publishing のシステム・メッセージ	SD88-7514	DB2 Information Integrator Support の Web サイト
Classic Federation トランザクション・サービス・ガイド	SD88-7513	DB2 Information Integrator Support の Web サイト
IBM DB2 Information Integrator Classic Federation for z/OS リリース情報	なし	DB2 Information Integrator Support の Web サイト

z/OS におけるレプリケーション機能に関する資料

z/OS におけるレプリケーション機能に関する資料

目的

z/OS におけるレプリケーション機能に関する資料。

表 8. z/OS 上のレプリケーション機能に関する DB2 Information Integrator の資料

資料名	資料番号	場所
ASNCLP Program Reference for Replication and Event Publishing	なし	DB2 Information Integrator Support の Web サイト
レプリケーションとイベント・パブリッシング 入門	GC88-9895	DB2 Information Integrator Support の Web サイト
Migrating to SQL Replication	なし	DB2 Information Integrator Support の Web サイト
レプリケーションとイベント・パブリッシング ガイドおよびリファレンス	SC88-9893	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
Replication Installation and Customization Guide for z/OS	SC18-9127	DB2 Information Integrator Support の Web サイト
SQL レプリケーション・ガイドおよびリファレンス	SC88-9163	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support の Web サイト
Tuning for Replication and Event Publishing Performance	なし	DB2 Information Integrator Support の Web サイト
Tuning for SQL Replication Performance	なし	DB2 Information Integrator Support の Web サイト
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	なし	• 「DB2 インフォメーション・センター」で、「**製品概要**」>「**インフォメーション・インテグレーション**」>「**DB2 Information Integrator 概説**」>「**問題、予備手段、および資料の更新**」 • DB2 Information Integrator のインストール・ランチパッド • DB2 Information Integrator Support の Web サイト • *DB2 Information Integrator* 製品 CD

Linux、UNIX、および Windows におけるエンタープライズ・サーチ機能に関する資料

Linux、UNIX、および Windows におけるエンタープライズ・サーチ機能に関する資料

目的

Linux、UNIX、および Windows におけるエンタープライズ・サーチ機能に関する資料。

表 9. Linux、UNIX、および Windows 上のエンタープライズ・サーチ機能に関する DB2 Information Integrator の資料

資料名	資料番号	場所
エンタープライズ・サーチの管理	SD88-6374	DB2 Information Integrator Support の Web サイト
エンタープライズ・サーチ インストール・ガイド	GD88-6373	DB2 Information Integrator Support の Web サイト
エンタープライズ・サーチ プログラミング・ガイドおよび API リファレンス	SD88-6375	DB2 Information Integrator Support の Web サイト
エンタープライズ・サーチ リリース・ノート	なし	DB2 Information Integrator Support の Web サイト

リリース情報およびインストール要件

リリース情報には、製品のリリースとフィックスパック・レベルに特有の情報が入っています。 また、それぞれのリリースの資料に対する最新の訂正も含まれています。インストール要件には、 製品のリリースに特有の情報が入っています。

表 10. DB2 Information Integrator のリリース情報とインストール要件

資料名	ファイル名	場所
Installation Requirements for IBM DB2 Information Integrator Event Publishing Edition, Replication Edition, Standard Edition, Advanced Edition, Advanced Edition Unlimited, Developer Edition, and Replication for z/OS	Prereqs	• *DB2 Information Integrator* 製品 CD • DB2 Information Integrator のインストール・ランチパッド
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	ReleaseNotes	• 「DB2 インフォメーション・センター」で、「**製品概要**」>「**インフォメーション・インテグレーション**」>「**DB2 Information Integrator 概説**」>「**問題、予備手段、および資料の更新**」 • DB2 Information Integrator のインストール・ランチパッド • DB2 Information Integrator Support の Web サイト • *DB2 Information Integrator* 製品 CD
IBM DB2 Information Integrator Classic Event Publisher for IMS リリース情報	なし	DB2 Information Integrator Support の Web サイト

表 10. *DB2 Information Integrator* のリリース情報とインストール要件 (続き)

資料名	ファイル名	場所
IBM DB2 Information Integrator Classic Event Publisher for VSAM リリース情報	なし	DB2 Information Integrator Support の Web サイト
IBM DB2 Information Integrator Classic Federation for z/OS リリース情報	なし	DB2 Information Integrator Support の Web サイト
エンタープライズ・サーチ リリース・ノート	なし	DB2 Information Integrator Support の Web サイト

リリース情報およびインストール要件の表示

リリース情報およびインストール要件の表示

目的

Windows オペレーティング・システム上で 、CD に入っているインストール要件およびリリース情報を表示するには、次のように入力します。

x¥doc¥*%L*

パラメーター

x Windows CD ドライブ名

%L
使用したい資料のロケール。例えば、en_US

目的

UNIX オペレーティング・システム上で 、CD に入っているインストール要件およびリリース情報を表示するには、次のように入力します。

/*cdrom*/doc/*%L*

パラメーター

cdrom
CD の UNIX マウント・ポイント

%L
使用したい資料のロケール。例えば、en_US

PDF 文書の表示および印刷

PDF 文書の表示および印刷

DB2 PDF Documentation CD から DB2 Information Integrator PDF ブックを表示および印刷するには、次のようにします。

1. *DB2 PDF Documentation* CD のルート・ディレクトリーから、index.htm ファイルをオープンします。

2. 使用したい言語をクリックします。
3. 表示したい文書のリンクをクリックします。

DB2 Information Integrator の資料へのアクセス

DB2 Information Integrator の資料へのアクセス

すべての DB2 Information Integrator ブックおよびリリース情報の PDF ファイルは、 www.ibm.com/software/data/integration/db2ii/support.html にある DB2 Information Integrator Support の Web サイトから入手できます。

DB2 Information Integrator Support の Web サイトから、最新の DB2 Information Integrator 製品資料にアクセスするには、 98 ページの図 4 に示すように、「Product Information」リンクをクリックします。

図 4. DB2 Information Integrator Support の Web サイトの「Product Information」リンク

「Product Information」リンクから、サポートされるすべての言語の最新の DB2 Information Integrator の資料にアクセスできます。

- DB2 Information Integrator 製品資料 (PDF ファイル)
- リリース情報も含めた、フィックスパック製品資料
- Linux、UNIX、および Windows の DB2 Information Center のダウンロードとインストールの説明
- DB2 Information Center オンラインへのリンク

DB2 Information Integrator Support の Web サイトは、 サポート資料、IBM Redbooks、白書、製品のダウンロード、ユーザー・グループへのリンク、および、DB2 Information Integrator についてのニュースも提供します。

アクセス支援

アクセス支援機能は、身体に障害のある (身体動作が制限されている、視力が弱いなど) ユーザーがソフトウェア製品を十分活用できるように支援します。 DB2® バージョン 8 製品に備わっている主なアクセス支援機能は、以下のとおりです。

- すべての DB2 機能は、マウスの代わりにキーボードを使ってナビゲーションできます。詳細については、『キーボードによる入力およびナビゲーション』を参照してください。
- DB2 のユーザー・インターフェースのフォント・サイズおよび色をカスタマイズすることができます。詳細については、 100 ページの『アクセスしやすい表示』を参照してください。
- DB2 製品は、Java™ Accessibility API を使用するアクセス支援アプリケーションをサポートします。詳細については、 100 ページの『支援テクノロジーとの互換性』を参照してください。
- DB2 資料は、アクセスしやすい形式で提供されています。詳細については、 100 ページの『アクセスしやすい資料』を参照してください。

キーボードによる入力およびナビゲーション

キーボード・フォーカス

キーボード・フォーカス

UNIX® オペレーティング・システムでは、アクティブ・ウィンドウの中で、キー・ストロークによって操作できる領域が強調表示されます。

キーボード入力

キーボード入力

キーボードだけを使用して DB2 ツールを操作できます。マウスを使って実行できる操作は、キーまたはキーの組み合わせによっても実行できます。標準のオペレーティング・システム・キー・ストロークを使用して、標準のオペレーティング・システム操作を実行できます。

キーまたはキーの組み合わせによって操作を実行する方法について、詳しくは、「キーボード・ショートカットおよびアクセラレーター: Common GUI help」を参照してください。

キーボード・ナビゲーション

キーボード・ナビゲーション

キーまたはキーの組み合わせを使用して、 DB2 ツールのユーザー・インターフェースをナビゲートできます。

キーまたはキーの組み合わせによって DB2 ツールをナビゲートする方法の詳細については、「キーボード・ショートカットおよびアクセラレーター: Common GUI help」を参照してください。

アクセスしやすい表示

アクセスしやすい表示

目的

アクセスしやすい表示

フォントの設定

フォントの設定

「ツール設定」ノートブックを使用して、メニューおよびダイアログ・ウィンドウに使用されるテキストの色、サイズ、およびフォントを選択できます。

フォント設定に関する詳細情報は、「メニューおよびテキストのフォントを変更する: Common GUI help」を参照してください。

色に依存しない

色に依存しない

本製品のすべての機能を使用するために、ユーザーは必ずしも色を識別する必要はありません。

支援テクノロジーとの互換性

支援テクノロジーとの互換性

DB2 ツールのインターフェースは、Java Accessibility API をサポートします。これによって、スクリーン・リーダーその他の支援テクノロジーを DB2 製品で利用できるようになります。

アクセスしやすい資料

アクセスしやすい資料

DB2 の資料は、ほとんどの Web ブラウザーで表示可能な XHTML 1.0 形式で提供されています。XHTML により、ご使用のブラウザーに設定されている表示設定に従って資料を表示できます。さらに、スクリーン・リーダーや他の支援テクノロジーを使用することもできます。

シンタックス・ダイアグラムはドット 10 進形式で提供されます。この形式は、スクリーン・リーダーを使用してオンライン資料にアクセスする場合にのみ使用できます。

IBM と連絡を取る

IBM へのお問い合わせ先は、Web サイト www.ibm.com/planetwide にある「IBM Directory of Worldwide Contacts」をご覧下さい。

製品情報

DB2 Information Integrator 製品に関する情報は、 Web により入手できます。

Web サイト www-6.ibm.com/jp/software/data/ をご覧ください。

このサイトには、次の最新情報が入っています。

- 技術ライブラリー
- 資料の注文方法
- 製品のダウンロード
- ニュースグループ
- フィックスパック
- ニュース
- Web リソースへのリンク

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものであり、米国以外の国においては本書で述べる製品、サービス、またはプログラムを提供しない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。 本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、製造元によって明示的に指定されたものを除き、他社の製品, プログラムまたはサービスを使用した場合の評価と検証はお客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032
東京都港区六本木3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム（本プログラムを含む）との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003 U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。お客様は、IBM のアプリケーション・プログラミング・インターフェースに準拠し

たアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、 IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生した創作物には、次のように、著作権表示を入れていただく必要があります。

商標

ここでは、IBM の商標と、特定の IBM 以外の商標をリストします。

以下は、IBM Corporation の商標です。

IBM
AIX
AIX 5L
DB2
DB2 Universal Database
Domino
Domino.doc
Hummingbird
Informix
Lotus
Lotus Notes
Notes
OmniFind
POWER4
POWER5
RISC System/6000
Tivoli
WebSphere
Workplace
xSeries
z/OS

以下は、それぞれ各社の商標または登録商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Intel、Intel Inside (ロゴ)、MMX および Pentium は、Intel Corporation の米国およびその他の国における商標です。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

A

B

C

D

F

G

J

L

N

P

R

S

U

X

Printed in Japan

SD88-6375-01

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12