



Splunk デベロッパーマニュアル

バージョン : 4.0.3

作成日 : 2009 年 8 月 26 日 午前 9 時 59 分

Copyright Splunk, Inc. All Rights Reserved

目次

概要	1
デベロッパーマニュアルの概要	1
App のしくみ	2
App コンポーネント	3
App の構築	5
App の構築	5
カスタム UI の構築	6
App ビルダの使用	8
App ディレクトリ構造	9
3.X App のマイグレート	10
デフォルト App の設定	12
ダッシュボードの構築	13
ダッシュボードとは何か	13
ダッシュボードの構築方法	14
簡単ダッシュボード	15
高度なダッシュボード設定	22
フォーム検索の構築	26
フォーム検索とは何か	26
簡単フォーム検索	26
ダイナミックフォーム検索	30
高度なフォーム検索	31
検索ビューの構築	36
検索ビュー	36
検索ビューの開始	36
設定とナレッジオブジェクトの追加	40
開発者のための設定とナレッジオブジェクトガイド	40
App.conf	42
保存済み検索	43
App セットアップ画面の設定	44
App 権限の設定	47

App のルック・アンド・フィールのカスタマイズ	49
App のルック・アンド・フィールのカスタマイズ	49
ナビゲーションメニューのカスタマイズ	49
ウェブリソースをビューに追加	53
App のスタイルの変更.....	56
モジュールのしくみ	57
モジュールレファレンス	63
高度なトピック	92
モジュールの構築.....	92
mako テンプレートの構築.....	92
Splunk ダッシュボードエレメントを他社製ソフトウェアに埋め込む	92
最終仕上げ	95
App をパッケージにする	95
Splunk Base の配布	95
拡張例	96
拡張例	96
ダッシュボードの例	96
フォーム検索の例	108

概要

デベロッパーマニュアルの概要

デベロッパーマニュアルの概要

Splunk デベロッパーマニュアルによろこそ。このマニュアルは、Splunk の表示形式から API の拡張にいたるまで、Splunk の開発について説明しています。このマニュアルでは主に、Splunk Web のカスタマイズとアプリケーション (App) の構築を中心に説明します。

App とは何か？

大まかに言えば、App は、各企業個別の使用を解決するワークスペースです。例えば、1 つの Splunk のインストールで、ヘルプデスク App、ビジネス分析 App、WindowsApp および検索 App のすべてを同時に実行することができます。

App は：

- ユーザーのデータ特性に注目します。
- 使用ケースを中心に構築されます。
- 多様なユーザーグループと役割をサポートします。
- 並列で実行します。
- 任意の数の Splunk の設定およびナレッジオブジェクトを含みます。
- フロントエンドからバックエンドに至るまで、完全なカスタマイズが可能です。
- HTML、CSS および JavaScript などのウェブアセットを含むことができます。
- なぜ App なのか？

App により、Splunk インスタンスの上に異なる環境を構築し維持することが可能になります。1 つの Splunk のインストールで、複数の App を実行することができます。このようにして、複数の異なるグループが、同じ Splunk インスタンスを、互いに鉢合わせすることなく使用することができます。

例えば、ヘルプデスクグループにいるユーザーが Splunk にログインすると、そのサポートケースを追跡することができます。カスタマイズ環境をそこに見つけることができます。マーケティンググループのユーザーがログインすると、ビジネス分析 App をそこに見つけ、ビジネストレンドとウェブアクティビティに関するレポートを実行することができます。一方で、Splunk 管理者は、インストールされたすべての App を維持管理するとともに、構築とインストールを追加することができます。

App を構築することにより、1 つの組織内にいる各 Splunk ユーザーグループ別の環境を別々に作成することができます。例えば、Eメールサーバーのトラブルシューティング用の App、ビジネストレンド分析用の App などです。このように、すべての人は同じ Splunk インスタンスを使用しますが、それぞれの業務に関連のあるデータのみを見ることができます。あるグループが複数 App にアクセスする一方で、別のグループは 1 つの App にアクセスすることができます。App はカスタマイズ度が高く、したがって、誰が何を使用し、それがどのように機能するかを決定することができます。

このマニュアルは誰が使用するべきか？

このマニュアルは、Splunk Web の変更、ダッシュボード、フォーム検索、App の構築、REST API を使用して Splunk の拡張を行いたい人のためのものです。Splunk の開発は容易です。Splunk App 構築の開発者である必要はありません。開発者であることは、Splunk の設定コンポーネントの理解の助けになりますが、本書からそれらの知識をピックアップすることもできます。App フレームワークは十分にわかりやすいため、個々の Splunk インストールを使用用途に応じて素早く設定することができます。

App のしくみ

App のしくみ

App は設定の集合体です。その範囲は、スクリプト、ナレッジオブジェクト、カスタム UI からバンクエンド設定まであります。各 App がそれぞれのディレクトリを備えているため、パッケージ化と配布を容易に行うことができます。

App は、企業内の様々なグループのためのカスタムダッシュボードを複数作成したり、Splunk のウェブサーバーのための独自のウェブページを作成するといったことを簡単に行うことができます。検索/入力スクリプトを書き、カスタムロールを作成し、Splunk Web の UI の変更を行うなど、多くのことを行うことができます。Splunk はカスタマイズ度が高く、個々の Splunk インストールの動作を希望どおり正確に設定することができます。

App フレームワーク

バージョン 4.0 では、App の作成をすべてのレベルでサポートする新たな App フレームワークが導入されました。これにより、コードを書くのではなく、設定により、強力な App を構築することが可能です。

Splunk の App フレームワークは、次のような機能を提供します：

- カスタム分析のためのカスタム UI(保存済み検索、レポート、イベントタイプ)
- カスタム設定(データ入力、ユーザー、インデックス)
- 高い適応性を備えた UI(スキン、フォーム検索、ダッシュボード)
- カスタム html、css および JavaScript をサポートする組み込みウェブサーバー

App は何を備えているか？

各 App は、カスタマイズされた設定で関連データを提供するワークスペースです。App は以下を備えることができます。

- ナレッジオブジェクト(保存済み検索、レポート、イベントタイプ、タグ、フィールドなど)
- ナレッジオブジェクトを提供するためのダッシュボード
- 検索ビュー
- 入力およびその他の設定
- カスタム検索コマンド
- ウェブアセットおよびカスタムウェブページ
- カスタマイズされた設定画面(App ユーザーが各 App の重要設定を閲覧するためのもの)

どのように App を構築するのか？

本書と Splunk の組み込み App フレームワークを使用して自身の App を作成します。App の構築は、複数の保存済み検索とそれら保存済み検索を表示するためのビューの作成など、簡単に行うことができます。App 構築の概要については、このセクションの次頁を引き続きご覧ください。

App コンポーネント

App コンポーネント

このページは App のすべてのアイテムに関する概略紹介です。データに適した事項を選択することができます。また、このページでは、本書全体の理解に必要な用語を定義します。

完全にカスタマイズ可能な UI

各ユーザーおよび使用用途に応じたカスタム UI を作成するには、Splunk の App フレームワークを使用します。Splunk の UI(Splunk Web)は完全にカスタマイズが可能です。したがって、Splunk のルック・アンド・フィールを少し変更することも、完全にデザインし直すことも可能です。以下のようなカスタマイズを Splunk に施すことができます。

Splunk Web のカスタムページを作成します。以下が可能です。

- ダッシュボードの構築
 - ◆ ダッシュボードは、様々な検索のサマリーを視覚的に提供するのに便利です。
 - ◆ ダッシュボードの詳細は、「ダッシュボードとは何か」を参照してください。
- フォーム検索の構築
 - ◆ フォーム検索は、検索インタフェースを制限して、バックで実行しているより複雑な検索を伴う 1 つまたは複数の検索ボックスを表示します。
 - ◆ Add_an_IFrame の詳細は、「フォーム検索とは何か」を参照してください。
- 検索ビューの構築
 - ◆ 検索ビューは、検索を実行し検索結果との相互作用を可能にするページです。
 - ◆ デフォルト検索ビューは Flash 時間軸ビューです。これは、ユーザーがよく知っているであろう標準的な Splunk 検索 UI です。
 - ◆ 検索ビューの詳細は「検索ビュー」を参照してください。

さらに高度に App のルック・アンド・フィールをカスタマイズできます。

- メニューレイアウトからバックグラウンド画像まで、すべてを変更できます。
- また、自分のカスタム html と JavaScript を App に組み込むことができます。
- 何ができるかに関する詳細は、「App のルック・アンド・フィールのカスタマイズ」を参照してください。

設定とナレッジオブジェクト

特定のデータタイプを収集および管理することにより、さらに App をカスタマイズすることができます。ナレッジを

データに追加し、ユーザーと使用用途に対応します。Splunk の設定のほとんどは、Splunk Web のマネージャインタフェースから利用が可能です。Splunk マネージャにより、以下が可能になります。

- 入力とインデックスを追加して、データを収集および保存する。
- 保存済み検索やレポート、フィールドなどのオブジェクトを通じてナレッジを追加する。
- App およびオブジェクトに対する権限を設定する。
- 新しいビューおよびナビゲーションメニューを作成/編集する。
- ユーザーおよび役割を追加し、それらを App に組み込む。

カスタム設定、オブジェクトおよび権限に関する詳細は「デベロッパー設定およびナレッジオブジェクトガイド」を参照してください。

App の構築

App の構築

App の構築

App の構築のためのオプションは山ほどあります。何よりもまず、App の使用目的とその用途について決定する必要があります。まず始めに、Splunk の App の構築方法に関する概略を説明します。

必要なもの

- テキストエディタ
 - ◆ App は Splunk Web のみですべて構築することができます。ただし、カスタム XML や css、html を作成する場合は、テキストエディタを使用する必要があります。テキストエディタには Komodo Edit を推奨します。無料でクロスプラットフォームであり、さまざまな標準フォーマットでテキストのハイライト表示をサポートしています。
- Splunk ナレッジオブジェクト
 - ◆ ほとんどの App は少なくとも 1 つの保存済み検索またはレポートを持っています。App 作成プロセスの一部として、これらを追加することができます。
- 便利で適切なデータ
 - ◆ App に表示させたいデータをインデックスします。
 - ◆ このデータの周囲にナレッジを構築し、このナレッジを App の UI を通じて表示させることができます。
- ウェブ開発ツール
 - ◆ ブラウザ依存のこれらツールは、使用している JavaScript、CSS および html のトラブルシューティングで使用されます。
 - ◆ 高度なカスタマイゼーションを行う場合は、これらツールのいずれかをインストールして使用することを強く推奨します。
 - ◆ Firefox を使用する場合、Firebug を使用することができます。
 - ◆ IE8 は、**デベロッパーツール**のもとにある**ツールメニュー**にビルトインコンソールがあります。
 - ◆ Safari 4 にもビルトインコンソールがあります。まず始めに、**プレファレンス > 詳細設定**タブにある、**メニューバーの開発メニュー**を有効にする必要があります。メニューバーの中の**デベロッパーの表示**メニューをチェックします。次に、**開発 > ウェブインスペクタの表示**を選択します。

App 構築概要

大まかには、App の構築は次のステップに従います。

1. App ワークスペースを作成する

- App ビルダを使って、App ワークスペースを作成します。

- Appビルダについては「Appビルダの使用」を参照してください。
2. データおよびナレッジを App に追加する
 - カスタム設定を追加して個々のデータをインデックス化して処理します。
 - ナレッジを追加してデータの表示方法をカスタマイズします。
 - Appにおけるデータおよびオブジェクトの設定の方法については、「デベロッパー設定およびナレッジオブジェクトガイド」を参照してください。
 3. Appユーザーにデータを提供する
 - インデックスするデータを表示するビューとダッシュボードを追加します。
 - AppのUIのカスタマイズ方法の詳細は、ここを参照してください。
 4. ユーザーに App を設定してもらう
 - カスタム設定をユーザーに提示します。
 - Appのセットアップ画面の設定方法の詳細は、「カスタムUIの構築」を参照してください。
 5. ユーザーがナレッジを App に追加する
 - ユーザーに権限を設定して Appユーザーがナレッジオブジェクトを App に追加できるようにします。
 - 権限の詳細は、「App権限」を参照してください。
 - ユーザーおよびナレッジマネージャマニュアルには、ナレッジオブジェクトの追加および設定に関する詳細が記載されています。

カスタム UI の構築

カスタム UI の構築

Splunk の App フレームワークは、カスタマイズが可能な表示レイヤを内蔵しており、コンポーネントのライブラリを備えた完全なものです。Splunk Web のページのすべてはカスタマイズ可能です。これらのウェブページ、つまりビューは、XML ファイルであり、App のビューディレクトリに保存されています。ビューはモジュールライブラリから作成されます。各モジュールは実際には CSS、JavaScript、HTML のディレクトリであり、Python および Flash のディレクトリである場合もあります。

ビューは、各仕様に応じて変更することができます。また、独自のビューを作成することもできます。以下に、ビューを設定するための基本概念の一般概要を記します。

1. ビューに含めたいモジュールを決定します。
2. `<view_name>.xml` で各モジュールを設定します。
3. Appディレクトリ：`$SPLUNK_HOME/etc/apps/<App_name>/data/ui/views/`にあるビューディレクトリに `<view_name>.xml` を置きます。
4. Appで複数のビューがある場合、「ビューコレクションの作成」にある説明に従って UI でそれらを調整します。

5. ビューの CSS を変更するには、App スタイルを変更します。

注意： Splunk 管理の外部にある XML を修正して App をカスタマイズする場合、次の場所に移動して設定を再ロードすることができます。

新規 App を表示：

```
https://<splunkserver>:<splunkmgmtport>/services/apps/local?refresh=true
```

特定のアプリケーションを再ロード：

```
https://<splunkserver>:<splunkmgmtport>/services/apps/local/<appname>?refresh=true
```

すべてのビューを再ロード：

```
http://<splunkserver>:<splunkwebport>/app/<appname>/
```

ビューの種類

ビューには、ダッシュボード、フォーム検索、検索ビューの 3 つのタイプがあります。各タイプのビューは、`$(SPLUNK_HOME)/share/splunk/search_mrsparkle/templates/view/`にある Mako テンプレートに定義されています。Mako テンプレートは、Python をサポートする HTML ファイルです。Splunk のテンプレートは、ページレイアウトを定義します。基本的には、1 つのページに適用する要素の数を決めます。

ダッシュボード

- ダッシュボードの構築
 - ◆ ダッシュボードは、様々な検索のサマリーを視覚的に提供するのに便利です。
 - ◆ ダッシュボードの詳細は「ダッシュボードとな何か」を参照してください。

フォーム検索

- フォーム検索の構築
 - ◆ フォーム検索は、検索インタフェースを制限し、バックで実行しているより複雑な検索を伴う 1 つまたは複数の検索ボックスを表示します。
 - ◆ 詳細は「フォーム検索とは何か」を参照してください。

検索ビュー

- 検索ビューの構築
 - ◆ 検索ビューは、検索を実行し検索結果と相互作用を可能にするページです。
 - ◆ 検索ビューの構築の方法に関する詳細は「検索ビュー」を参照してください。

Splunk Web の詳細なカスタマイゼーション

App のルック・アンド・フィールをカスタマイズします。例えば、以下が行えます。

- メニューレイアウトの変更
- 独自のカスタム HTML と JavaScript の追加

- App をカスタム CSS でスキン

何ができるかに関する詳細は「App のルック・アンド・フィールのカスタマイズ」を参照してください。

モジュールの種類

ビューの構築に使用できるモジュールはたくさんあります。例えば、ページ内の検索バーはモジュールの 1 つです。また、モジュールには、グラフやチャート、テキスト入力ボックス、リンク、ドロップダウンメニュー、その他のコンポーネントがあります。モジュールの機能に関する概要は、「モジュールのしくみ」を参照してください。

モジュールは `$SPLUNK_HOME/share/splunk/search_mrsparkle/modules/` にあります。全リストは、モジュールレファレンスを参照してください。また、各ページのモジュールの階層を見るには、ビューの URL に、`?showsourc=true`、を付加します。例えば：

```
http://localhost:8000/en-US/app/search/charting?showsourc=true
```

注意： localhost:8000 は使用するインストールホストおよびポートで置き換えてください。

App ビルダの使用

App ビルダの使用

App ビルダを使って、App ワークスペースを作成およびカスタマイズします。App ビルダは、App 構築プロセスを迅速化するためのテンプレートを提供します。App テンプレートは、App の作成に必要なすべての基本的設定を有効にしたワークスペースを提供します。App ビルダは以下の機能を供します。

- 2 つのダッシュボード
- 3 つのフォーム検索
- 2 つの検索ビュー
- 大量の保存済み検索
- サンプル sendmail データ

また、バックエンドから App を作成することもできます。ただし、App ビルダは、App ディレクトリ構造と必要なファイルをユーザー用に自動作成します。

sample_app

現在のところ、sample_app は App ビルダでのみ利用できるテンプレートです。sample_app テンプレートは、sample_app によりインデックス化されたサンプル sendmail データ上に作成されます

(`$SPLUNK_HOME/etc/apps/sample_app/`にある)。このテンプレートは、データの様々な特性を明らかにする以下のような複数のカテゴリーのビューを備えています。

- ダッシュボード
- フォーム検索
- 検索ビュー

- 検索とレポート
- 未分類

Splunk Web で App ビルダを使用する

Splunk Web で App ビルダを使用するには、次のステップに従います。

1. Splunk Web にログインし、Splunk マネージャに移動します(右上にある**管理**リンクをクリックします)。
2. **Apps** をクリックします。
3. **App の作成...** ボタンを押し、新規 App を作成します。
4. App の名前(helloworld など)を入力します。この名前は、`$SPLUNK_HOME/etc/apps/`にある Apps 用ディレクトリに関連付けられます。
5. App のラベルを入力します。このラベルはラベル設定 `app.conf` に割り当てられます。ラベルの名前は同じでも構いません。例えば、**Hello world**、など。ラベルはまた、Splunk Web App ドロップダウン(Splunk Web の左上)にも表示されます。
6. **sample_app** app テンプレートを選択します。
7. 画像、アセットまたはファイルを追加します。指定しないファイルについては、App ビルダがテンプレートのデフォルトを使用します。以下が可能です。
 - バナーの背景画像を追加する。この画像はバナーセクションの背景となります。
`$SPLUNK_HOME/share/splunk/appserver/static/` directory に多くのバナー画像があります。
 - 他の画像をアップロードします。HTML モジュールにある画像を参照してください。
 - `application.css` をアップロードする。このファイルを使用して App のスタイルを変更します。
 - アセットをアップロードする。使用する App で参照したい HTML、JavaScript、その他の有効なファイルフォーマットを追加することができます。
8. App を保存します。Splunk を再起動しなくても、App は有効になります。
9. ダッシュボード、フォーム検索、検索ビューをカスタマイズして、App の構築を続けます。

App ディレクトリ構造

App ディレクトリ構造

すべての App は、`$SPLUNK_HOME/etc/apps` 内のカスタムディレクトリにあります。

注意： `appbuilder` で App を構築する際には、これらすべてのディレクトリが自動作成されます。

App の構造

このディレクトリ内には次のサブディレクトリがあります。

- Default/
 - ◆ Appに必要なすべてのSplunk設定ファイルをDefaultに置きます。すべてのAppにはapp.confがなければなりません。また、savedsearches.conf、inputs.confその他の関連する設定ファイルがある場合もあります。
- Local/
 - ◆ 開発者はローカルディレクトリで設定を行いません。ここでは、Appユーザーと管理者がデフォルト設定を上書きします。

デフォルトディレクトリとローカルディレクトリには、さらにUIを設定するためのサブディレクトリがあります。これらは\$SPLUNK_HOME/etc/apps/<App_name>/<default または local>/data/UI/に存在し、以下が含まれます。

- Nav/
 - ◆ このディレクトリにはdefault.xmlのみがあります。このファイルを使用してビューコレクションの作成を行います。
- Views/
 - ◆ 作成したビューはすべてこのディレクトリに置きます。ビューを使用してカスタムUIを構築します。

カスタム設定

カスタムモジュール、画像、CSS、htmlをAppに追加するには、Appのディレクトリ(\$SPLUNK_HOME/etc/apps/<App_name>/)にappserver/staticディレクトリを追加します。固定ディレクトリを使用して以下を行います。

- Appのスタイルの変更
- ウェブリソースの追加

3.X Appのマイグレート

3.X Appのマイグレート

このトピックでは、3.X AppをSplunk 4.xにマイグレートする方法について説明します。この操作は、アプリケーションのコンテンツにより異なります。したがって、まず始めに、マイグレートの設定と、それら設定が4.0でサポートされているかどうかを決定します。4.0にマイグレートして何が出来るかについて、インストールマニュアルの関連トピックを参照してください。

多くの場合、4.x Appでもナレッジ項目(イベントタイプ、ソースタイプなど)を再利用することができます。また、このトピックの情報を使って、Splunkコミュニティで作成された便利な3.x Appsを再構築し、これを4.xで動作させることができます。

入力とその他のバックエンド設定

バックエンドの設定ファイルは、Splunkサーバーの機能およびデータ設定を指定するためのものです。そのほとんどは問題なくマイグレートすることができます。設定ファイルには、authentication.conf、indexes.conf、inputs.conf、

outputs.conf、web.conf があります。

これらのファイルには小さな変更がなされていることに注意してください。特定の設定がマイグレート可能かどうか分からない場合は、仕様ファイルを確認してください。

デプロイメントサーバーの設定は完全に変更されているため、手動でマイグレートする必要があります。

ナレッジと表示の設定

3.X から 4.0 への設定変更のほとんどはナレッジ(イベントタイプ、保存済み検索など)および表示(Splunk ウェブの外観)レイヤに関するものです。ただし、以下のファイルは通常問題なくマイグレートすることが可能です。

props.conf、transforms.conf、eventtypes.conf、tags.conf。

3.X App から 4.0 App にナレッジを単純にコピーする場合は、検索 App をクローニングし、イベントタイプ、タグ、props、変換その他のナレッジ設定にコピーすることができます。保存済み検索は手動でマイグレートする必要がありますのでご注意ください(後述)。

保存済み検索とフォーム検索

保存済み検索とフォーム検索は大幅に変更されているため、手動でマイグレートする必要があります。

savedsearches.conf に上書きコピーする、または Splunk 管理の検索文字列にコピーできます。Splunk はこれら検索をマイグレートしますが、フィールドおよび廃止検索コマンド内に残った::などは編集する必要があります。保存済み検索をダッシュボードで表示させるには、その検索をダッシュボードに追加する必要があります。これにより、新規検索のビューステートが作成されます。フォーム検索はその新しいビューシステムを通じて作成する必要があります。

savedsearches.conf に古いフォーム検索をマイグレートすることはできません。詳細は本書の、「4.0 によるフォーム検索」を参照してください。

権限の設定

4.0 は新しいオブジェクトモデルを導入しています。このモデルは、すべての App とオブジェクト(保存済み検索、レポート、ビュー、イベントタイプなど)の権限を設定します。3.X App を 4.0 にマイグレートしたときは、Splunk 管理を使用する、または default.meta ファイルを App のディレクトリに手動で追加することにより、App の権限を設定してください。App 権限の設定方法の詳細は本書に記載されています。

注意： 設定を Splunk に手動でコピーした場合(Splunk Web を使用しない)、その設定が Splunk Web で表示されるように、権限を設定する必要があります。

使用するアプリケーションが単に、ファイアウォールスクレーパアプリケーションなどその他のアプリケーションで使用するために提供されたデータである場合、その設定をグローバルにエクスポートすることも可能です。

例

この例は、SplunkBase からウェブアクティビティ App を取得します(保存場所：

[http://www.splunkbase.com/apps/All/Technologies/Web/Web_Servers/Data_Sources_\[General_Web\]/NCSA_combined/app:Web+access+reports](http://www.splunkbase.com/apps/All/Technologies/Web/Web_Servers/Data_Sources_[General_Web]/NCSA_combined/app:Web+access+reports))。

この App は、savedsearches.conf および bundle.conf を含んでいます。保存済み検索は 4.0 の新しい App にマイグレートすることができますが、bundle.conf は廃止されていますので、代わりに app.conf を使用してください。次に、この App をマイグレートするための手順を順を追って説明します。

1. 新たに App ディレクトリを作成します。App ビルダを使用することができます。これは自動的に、default.meta、app.conf、その他のファイルならびに App ディレクトリ構造の全体を作成します。必要に応じて、手動でディレクトリを \$SPLUNK_HOME/etc/apps/ に作成することもできます。例えば、ディレクトリ \$SPLUNK_HOME/etc/apps/web_activity_4 を作成します。必須ファイル(app.conf、default.meta)を追加することを忘れないでください。
2. 古い savedsearches.conf を新規 App のデフォルトディレクトリ :
\$SPLUNK_HOME/etc/apps/web_activity_4/default/savedsearches.conf にコピーします。 保存済み検索検索文字列をすべて Splunk 管理に手動でコピーすることもできます。
3. 保存済み検索を編集して 4.0 で動作することを確認します。特に、::のインスタンスを=に変更してください。例えば、sourcetype::access は sourcetype=access になります。保存済み検索には、他にも問題がある場合があります。Splunk Web は問題についてアラートを表示します。検索は Splunk 管理で直接編集できます。
4. 編集した保存済み検索を保存します。新しい App を表示するには Splunk の再起動が必要な場合があります。
5. 新たにマイグレートした保存済み検索を表示するには、新規ダッシュボードを作成する、または既存のダッシュボードを編集します。

デフォルト App の設定

デフォルト App の設定

デフォルト App は、ユーザー単位またはグローバルに設定することができます。この App はデフォルトで、Splunk にログインする毎に表示されます。

設定

1. user-prefs.conf と呼ばれるファイルをユーザーの次のローカルディレクトリに作成します(ユーザーに適用する)。
\$SPLUNK_HOME/etc/users/<user>/user-prefs/local/user-prefs.conf

または、次のデフォルトの user-prefs ディレクトリに作成します(グローバルに適用する)。
\$SPLUNK_HOME/etc/apps/user-prefs/user-prefs.conf.

2. 次を user-prefs.conf ファイルに追加します。
default_namespace = <App>

例えば、管理ユーザーのデフォルト App を検索 App に設定します。

1. \$SPLUNK_HOME/etc/users/admin/user-prefs/local/user-prefs.conf を編集します。テストユーザーについては、\$SPLUNK_HOME/etc/users/test/user-prefs/local/user-prefs.conf にあります。

2. 次を追加します。

```
default_namespace = Search
```

ダッシュボードの構築

ダッシュボードとは何か

ダッシュボードとは何か

カスタムイベントレンダラを作成する

ダッシュボードを使用すれば、データの様々な特性を強調し、重要な検索にリンクし、共通レポートを表示することができます。各ダッシュボードは1つまたは複数のパネルからなり、各パネルは検索とその検索の視覚的なサマリーが表示されます。パネルはそれぞれ左右または上下に配置されます。パネルには次のものが表示されます。

- 検索またはレポート
- チャート、リストその他検索の表示

ウェブリソースをダッシュボードパネルに含めることもできます。

Splunk は、検索 App にいくつかのデフォルトダッシュボードを標準装備しています。ステータスメニューの下にあるリンクのすべてはダッシュボードです。また検索 App のデフォルトビューもダッシュボードであり、次の場所にあります。

<http://localhost:8000/en-US/app/search/dashboard>

注意： localhost:8000 を設定したホストおよびポートで置き換えてください。

簡単と高度

Splunk は、ダッシュボード構築のための2種類の異なる XML 構文を提供します：つまり、簡単および高度です。ほとんどのダッシュボードは、Splunk のビジュアルダッシュボードエディタにより構築することができます。これは、簡単なダッシュボード構文を書き出します。しかし、すべてのモジュールが簡単ダッシュボードで利用できるわけではありません。したがって、より洗練されたものを構築したい場合は、高度構文を使用してください。

必要なもの

ダッシュボードは検索およびレポート上に構築されます。検索およびレポートは、保存済み検索とレポートまたはダッシュボード XML 設定で使用された検索です。保存済み検索をダッシュボードで指定する場合、Splunk は最新の検索結果を使用します。したがって、検索をスケジュールで実行するように設定することができ、ダッシュボードはキャッシュされた結果を使用します。長期に実行される検索が多くある場合、または多くの人と同じダッシュボードを同時に使用することが予想される場合、保存済み検索を使用してください。ダッシュボードを使用して結果をリアルタイムで表示できるようにするには、検索文字列を直接ダッシュボードに構築します。

ダッシュボードを構築する前に、保存済み検索の設定に関する項を参照してください。

ダッシュボードの構築方法

ダッシュボードの構築方法

次にダッシュボードの構築方法の概要を説明します。

新しい App の開始

新しい App を開始するには、次のようにします。

- App ビルダを使って、App ワークスペースを作成します。
 - ◆ ダッシュボード App を構築する最も簡単な方法は、App ビルダで始めることです。
 - ◆ App ビルダの説明を参照し、**app_sample** テンプレートを使用してください。
- インデックス関連データ
 - ◆ 実行していない場合、データを Splunk に投入します。インデックスするデータについてデータ入力を設定します。オプションで、データを App 個別インデックスに送る、またはそれを分離したくない場合は、デフォルトインデックスに送ります。

新規ダッシュボードの設定

ダッシュボードに追加したい App がわかっている場合、次のようにします。

- ナレッジオブジェクトを追加します。
 - ◆ 保存済み検索とレポートはあらゆるダッシュボードで便利に使用できます。個々の App に関する保存済み検索の設定方法の詳細は、関連項目を参照してください。
 - ◆ 必要に応じて、イベントタイプ、フィールド、タグなど他のナレッジオブジェクトを追加します。
- ダッシュボードを構築します。
 - ◆ すぐに始めたい場合は、簡単ダッシュボードを作成します。
 - ◆ XML に詳しい場合は、より洗練されたダッシュボードを構築することもできます。
 - ◆ ほとんどのダッシュボードは、個別の検索または保存済み検索のいずれかを参照します。したがって、構築した検索について把握するようにしてください。
 - ◆ ウェブリソースをダッシュボードに含めることもできます。
- ダッシュボードに対する権限を設定します。
 - ◆ ダッシュボードで保存済み検索、チャート、その他のエレメントをユーザーがカスタマイズすることを許可するかどうかを決定します。App ユーザーは、独自のプライベートディレクトリ内でオブジェクトをいつでも作成することができ、ユーザーがアプリケーションレベルで作成したオブジェクトを共有することもできます。
 - ◆ 「App 権限の設定」の説明に従って App 権限を設定します。

簡単ダッシュボード

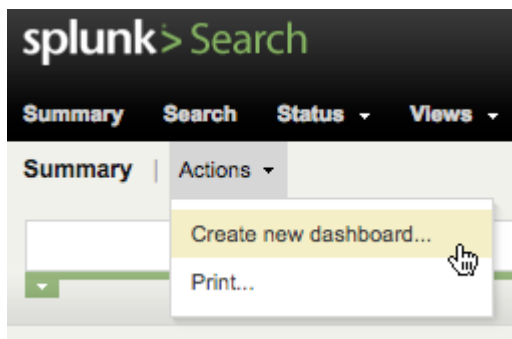
簡単ダッシュボード

簡単ダッシュボード構文を使用してダッシュボードを素早く構築することができます。この構文は高度なビュー構文で構築され、いくつかのモジュールを提供します。より洗練されたダッシュボードが必要な場合は、高度な構文を使う必要があります。尚、簡単構文と高度構文を切り替えることはできません。

簡単ダッシュボードを構築するには、ビジュアルダッシュボードエディタを使用する、または自分の XML 設定を作成します。

ビジュアルダッシュボードエディタ

ビジュアルダッシュボードエディタを使用して簡単ダッシュボードを作成します。始めに、Splunk Web の左上にあるアクションメニューを選択します。



次に、ビジュアルダッシュボードエディタの説明に従います。上述で作成したパネルをドラッグ・アンド・ドロップして、上下左右に動かすことができます。

XML 設定

ビジュアルダッシュボードエディタを使用することなく、簡単ダッシュボード XML を直接編集または作成することもできます。ダッシュボードを編集または作成するには次のようにします：

- Splunk 管理
 - ◆ 管理に移動する。
 - ◆ ビューをクリック。
 - ◆ 新規ボタンを選択して新規ビューを作成する、または、既存のダッシュボードをクリックして編集します。
 - ◆ 新規ビューページの説明に従い、下記で述べるフォーマットで XML を入力します。
- バックエンド
 - ◆ 使用する XML を直接ファイルシステムを使って App にコピーします。

◆ 使用するダッシュボードの XML ファイルが

`$SPLUNK_HOME/etc/apps/<app_name>/default/data/ui/views/`にあることを確認してください。

1. ダッシュボードタグを指定し、ラベルを設定してダッシュボードを開始します。ラベルは Splunk Web のドロップダウンメニューに表示されます。

```
<dashboard>
  <label>My dashboard</label>
</dashboard>
```

これにより、標準 Splunk ヘッダー、ナビゲーションメニュー、および「My dashboard」のラベルが付いた空のダッシュボードページが作成されます。

2. ここで、行を設定してパネルをダッシュボードに追加します。

```
<dashboard>
  <label>My dashboard</label>
  <row>
  </row>
</dashboard>
```

これにより、1 つまたは複数のパネルを格納可能な 1 つの行が作成されます。

3. コンテンツを行に追加します。例えば、「my errors」という保存済み検索からの簡単な結果テーブルを作成します。

```
<dashboard>
  <label>My dashboard</label>
  <row>
    <table>
      <searchName>my errors</searchName>
    </table>
  </row>
</dashboard>
```

注意： `searchName` で参照される保存済み検索は、App のデフォルトローカルディレクトリまたはローカルディレクトリの `savedsearches.conf` に存在する、またはグローバルとして設定されている必要があります。

4. 同じ行に追加のパネルを追加します。この例は 3 つのパネルを並べて表示します。

```
<dashboard>
  <label>My dashboard</label>
  <row>
    <table>
      <searchName>my errors</searchName>
    </table>
    <table>
      <searchName>your possible errors</searchName>
    </table>
    <table>
      <searchName>their definite errors</searchName>
    </table>
  </row>
</dashboard>
```

5. または、パネルを複数行にわたって展開します。この例は、1 つのパネルを自分の行に、2 つのパネルを別の 1 行

に置きます。

```
<dashboard>
  <label>My dashboard</label>
  <row>
    <table>
      <searchName>my errors</searchName>
    </table>
  </row>
  <row>
    <table>
      <searchName>your possible errors</searchName>
    </table>
    <table>
      <searchName>their definite errors</searchName>
    </table>
  </row>
</dashboard>
```

6. 同じヘッディングで1つの行の中にパネルをグループ化します。グループ化属性をその行のノードに追加すると、1行の中に複数のパネルをグループ化することができます。次の例は、*可能性のあるエラー*、とその*確定エラー*を同じグループヘッディングに置いています。

```
<dashboard>
  <label>My dashboard</label>
  <row>
    <table>
      <searchName>my errors</searchName>
    </table>
  </row>
  <row grouping="2">
    <table>
      <searchName>your possible errors</searchName>
    </table>
    <table>
      <searchName>their definite errors</searchName>
    </table>
  </row>
</dashboard>
```

7. さらに、1行でパネルを左側または右側にグループ化することができます。次は、3つのテーブルを左側グループに、2つのテーブルを右側に配置するパネルを1行に作成します。

```
<dashboard>
  <label>My dashboard</label>
  <row grouping="3,2">
    <table>
      <searchName>my errors</searchName>
    </table>
    <table>
      <searchName>your possible errors</searchName>
    </table>
    <table>
      <searchName>their definite errors</searchName>
    </table>
    <table>
      <searchName>known unknown errors</searchName>
    </table>
  </row>
</dashboard>
```

```

    <table>
      <searchName>unknown unknown errors</searchName>
    </table>
  </row>
</dashboard>

```

検索オプション

前述の例はすべて1つのテーブルパネルと1つの保存済み検索を使用します。しかし、パネルには検索を指定するためのオプションが他にもあります。

1. インライン検索を指定する。例えば、

```

<table>
  <searchString>search foo this | timechart that</searchString>
  <earliestTime>-20h</earliestTime>
  <latestTime>-2h</latestTime>
</table>

```

2. 保存済み検索の出力をフィールドで制限する

```

<table>
  <searchName>Errors in the last 24 hours</searchName>
  <fields>host, source, errorNumber</fields>
</table>

```

3. タイトルをパネルに追加する

```

<table>
  <title>Look here for errors that you need to care about</title>
  <searchName>Errors in the last 24 hours</searchName>
  <fields>host, source, errorNumber</fields>
</table>

```

4. パネルの表示オプションを設定する。例えば、

```

<table>
  <title>Look here for errors that you need to care about</title>
  <searchName>Errors in the last 24 hours</searchName>
  <fields>host, source, errorNumber</fields>
  <option name="count">25</option>
  <option name="displayRowNumbers">true</option>
</table>

```

パネル

テーブルのほかにも、検索結果を出力するフォーマットは他にもあります。次に、簡単なXMLで利用できるオプションを紹介します。

table

テーブルパネルは、表形式で検索データを表示します。次のオプションをサポートします。

- count: 整数
 - ◆ 表示する行の最大数を指定します
- displayRowNumbers: true | false

- ◆ 行数を結果行の左側に表示するかどうかを指定します
- showPager: true | false
 - ◆ ページングコントロールをパネルに追加するかどうかを指定します

```
<table>
  <title>Look here for errors that you need to care about</title>
  <searchName>Errors in the last 24 hours</searchName>
  <option name="count">25</option>
  <option name="displayRowNumbers">true</option>
</table>
```

chart

チャートパネルは、図形式で検索データを表示します。パネルが searchName ノードにより保存レポートとペアにされた場合、標準レポートビルダーインタフェースにより生成された既存のフォーマットオプションをロードしようとします。

注意： 保存レポートはチャートフォーマットパラメータを含んでいます。一方、保存済み検索はそれを含みません。詳細は、ユーザーマニュアルの「レポートの保存と共有」を参照してください。

チャートフォーマットはオプションノードよりインラインで上書きすることができます。チャートパネルは次のオプションをサポートします。

- height = *CSS* デイメンジョン
 - ◆ チャートオブジェクトの物理的高さを指定します。
- charting.chart = bar | line | column | area | pie | scatter | bubble
 - ◆ チャートタイプを指定します。
- charting.legend.placement = top | left | bottom | right | none
 - ◆ 凡例の場所をします。
- charting.* = チャートオプション
 - ◆ チャートフォーマットオプションはすべてここでサポートされます。チャートレファレンスを参照してください。

```
<chart>
  <searchString>index=_internal metrics group="pipeline" NOT sendout | head 1000 | timechart
  per_second(cpu_seconds) by processor</searchString>
  <earliestTime>-30h</earliestTime>
  <latestTime>-10h</latestTime>
  <option name="charting.chart">line</option>
  <option name="charting.primaryAxisTitle.text">Time</option>
  <option name="charting.secondaryAxisTitle.text">Load (%)</option>
</chart>
```

event

イベントパネルは検索結果を表示します。このパネルは次のオプションをサポートします。

- count = 整数
 - ◆ 表示する行の最大数を指定します

- displayRowNumbers = true | false
 - ◆ 行数を結果行の左側に表示するかどうかを指定します
- entityName = events | results
 - ◆ イベントまたは結果を表示するかどうかを切り替えます。
 - ◆ デフォルトは results(結果)。
- segmentation = none | inner | outer | full
 - ◆ 行イベントのセグメンテーションモード
- maxLines: 整数
 - ◆ 結果ごとに表示する行の最大数
- showPager: true | false
 - ◆ ページングコントロールをパネルに追加するかどうかを指定します。

```

<event>
  <title>Event view</title>
  <searchString>changelist | head 1000 | dedup changelist</searchString>
  <fields>added deleted changed</fields>
  <option name="showPager">true</option>
  <option name="count">20</option>
  <option name="displayRowNumbers">false</option>
</event>

```

single

シングルパネルは、検索データから 1 つの値(最初の行/最初の列)を表示します。データセットの合計サイズに関わらず、1 つの値のみを表示します。返される検索により、パネルの色を変更することができます。このパネルは次のオプションをサポートします。

- additionalClass = *css クラス名*
 - ◆ 結果コンテナに追加する任意の css クラス名
 - ◆ オプション
- linkView
 - ◆ リンクされている検索を実行するビューを指定します。
 - ◆ デフォルトは dashboard (ダッシュボード)。
- field
 - ◆ 表示するフィールド
 - ◆ デフォルトは最初に返されるフィールド
- linkFields
 - ◆ 結果をリンクするか、ラベルも含めるかを指定します。
 - ◆ 結果と両方のラベルをリンクするには、"result,beforeLabel,afterLabel"を指定します。
 - ◆ デフォルトは result(結果)。
- classField

- ◆ 最初の結果の classField の値を、結果コンテナに追加する css クラスとして追加します。
- ◆ あらかじめ定義されたクラスは、'severe'、'elevated'、'low'および'None'(デフォルト)。
- beforeLabel
 - ◆ 結果の前に表示するラベル。
- afterLabel
 - ◆ 結果の後に表示するラベル。
- linkSearch
 - ◆ 結果をクリック可能リンクに変えるための有効な完全検索クエリを指定します。

```
<single>
  <searchString>| metadata type="sources" | stats count</searchString>
  <option name="afterLabel">sources</option>
</single>
```

1つの色を変更する場合、レンジマップで検索を設定します。

```
<single>
  <searchString>index=_internal 404 source="*web_access.log" earliest=-1h | stats count
  | rangemap field=count low=0-0 elevated=1-100 default=severe</searchString>
  <title>404s this hour</title>
  <option name="classField">range</option>
</single>
```

HTML

HTML パネルはインライン HTML を表示します。HTML タグ間のコンテンツはすべて文字通り解釈され、ページに表示されます。画像などへの参照リンクは、相対的に現在のビューの位置に関連づけられることに注意してください。

```
<html>
  This lists all of the data you have loaded into <strong>your</strong> default indexes
  over all time.
</html>
```

list

リストパネルはデータを一覧表示します。これは、保存済み検索および検索結果などの情報の表示に使用されます。このパネルは次のオプションをサポートします。

- initialSortDir = asc | desc
 - ◆ initialSort フィールドに基づいて結果をソートする向き。
 - ◆ オプション
- labelFieldSearch = *検索文字列*
 - ◆ ユーザーがラベルフィールドをクリックした際に生成される検索文字列。
 - ◆ labelFieldTarget を有効なビューに指定する必要があります。
 - ◆ ラベルフィールドの値は自動的にその検索に追加されます。
- valueField
 - ◆ 結果フィールドの名前で、その値は、リンクリストのラベル部分に表示されます。

- ◆ リンクリストは一般的に、説明ラベルと数字カウントその他(値)フィールドの組み合わせです。
- ◆ 必須
- labelFieldTarget
 - ◆ ラベルフィールドに、検索をディスパッチするクリック可能リンクの生成がセットアップされている場合に、目標とするビュー。
 - ◆ オプション
- initialSort
 - ◆ 結果のフィールドはリンクリストが最初に表示される時にソートされます。
 - ◆ オプション

高度なダッシュボード設定

高度なダッシュボード設定

簡単ダッシュボード設定ではすべてのモジュールを利用することはできません。より洗練されたダッシュボードを作成する場合、簡単ダッシュボード設定では利用できないモジュールを使用する、以下の手順に従います。

設定

次にダッシュボードの構築方法の概要を説明します。

1. データの視覚化および表示方法を決定します。例えば、検索結果をグラフで表示したり、検索結果へのリンクをリストで表示するなどです。
2. 検索を構成します。
3. 各検索についてパネルを構築します。
4. 構築したパネルからダッシュボードを構成します。

最後に、ダッシュボードパネルをレイアウトします。ダッシュボードの構築方法の詳細は、「ダッシュボードレイアウト」を参照してください。

データの視覚化方法を決定する

まず始めに、利用可能モジュールでデータをどのように表示するかを決定します。結果モジュールは、検索結果をダッシュボードで表示するのに最も便利なモジュールです。

検索を構成する

次の2つの方法で検索をダッシュボードパネルに含めることができます。

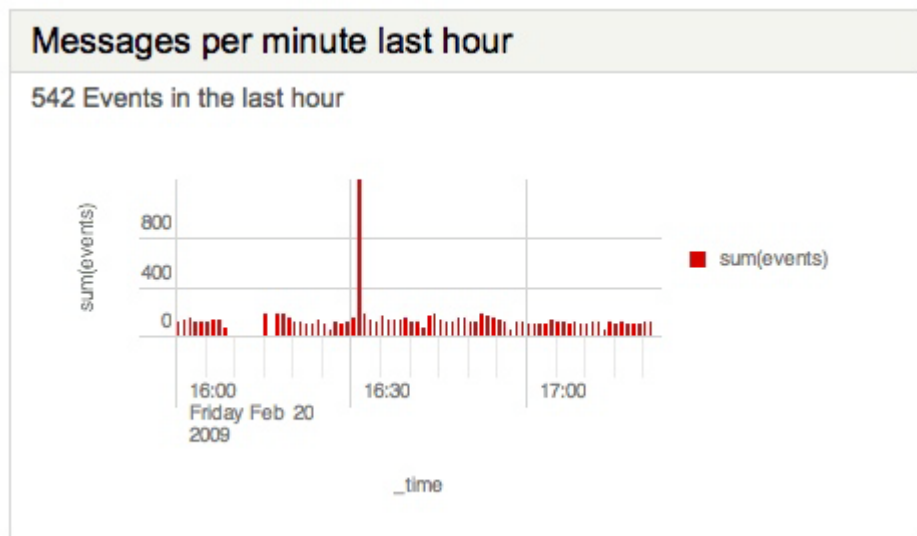
1. 検索を生成し、それを保存し、スケジュールで実行します。次に、HiddenSavedSearch モジュールにより、ダッシュボードから検索結果を参照します。多くのユーザーがダッシュボードにアクセスする場合、または検索が結果を返すのに時間がかかる場合は、これが最良の方法です。

2. HiddenSearch モジュールにより、検索文字列を直接ダッシュボードパネルで参照します。HiddenSearch モジュールは、ダッシュボードがロードするたびに検索を実行します。したがって、検索が結果を素早く返し、少数のユーザーがある与えられた時間にダッシュボードにアクセスする場合には、最も有効に機能します。

パネルの構築

パネルはダッシュボードで表示される結果のセットです。ダッシュボードはパネルをいくつでも含むことができます。各パネルは通常、HiddenSearch または HiddenSavedSearch モジュールを通じ、それに関連する検索を 1 つだけ有します。チャートおよびリンクリストなどの結果モジュールで検索から結果を表示します。

次は、チュートリアルダッシュボードからのパネルの例です。



このパネルの背後にある XML は次のとおりです。

```
<module name="HiddenSearch" layoutPanel="panel_row1_col1" group="Messages per minute last hour" autoRun="True">
  <param name="search">search index=_internal eps group=per_source_thruput NOT filetracker Metrics | eval events=eps*kb/kbps | timechart sum(events)</param>
  <param name="earliest">-1h</param>
  <module name="ResultsHeader">
    <param name="entityName">scanned</param>
    <param name="entityLabel">Events</param>
  <module name="FlashChart">
    <param name="height">180px</param>
    <param name="width">100%</param>
  </module>
</module>
</module>
```

このパネルにはパネルが 3 のみあることに注意してください。3 つのパネルとはつまり、HiddenSearch、ResultsHeader および FlashChart です。HiddenSearch はすべての検索結果を生成し、FlashChart はそれらを表示します。

ResultsHeader は、HiddenSearch が検索したイベント数と共にヘッダーを表示します。

HiddenSearch は親モジュールであるため、layoutPanel、group、autoRun など他の設定をいくつか使用します。

LayoutPanel は、ダッシュボード内のどこにパネルを置くかを示します。Group は、パネルのトップに表示するヘッダ

一です。autoRun は、パネルの検索が、ページのロードの際に実行されることを示します。通常、autoRun = true に設定します。

ダッシュボードの構成

以上で、検索結果を表示するパネルのいくつかができました。これらのパネルを1つのダッシュボードに配置します。次に、ダッシュボードを構成する手順を順を追って説明します。

1. ダッシュボードの XML ファイルを作成します。
2. 表示する順番でパネルをレイアウトします。

ダッシュボードの XML ファイルの作成

ダッシュボードを \$SPLUNK_HOME/etc/apps/<your_app>/default/ui/views/ に追加します。次に、編集する XML ファイルを開きます。まず、次の行を追加します。

```
<view template="dashboard.html">
```

これは、ダッシュボードテンプレートの使用を指定します。ダッシュボードビューは、検索ビューが使用するデフォルトテンプレートとは異なる Mako テンプレートを使用します。したがって、ダッシュボードの XML ファイルの最初にこのテンプレートを指定する必要があります。

refresh=<秒>を追加することにより、リフレッシュの頻度をここで設定することもできます。これは HiddenSearches を返す、または新規 HiddenSavedSearch 結果を取得します。

この例は、ダッシュボードが 30 秒ごとに自動的にリフレッシュするように設定します。次のように、必要な既存 XML を編集します。

```
<view template="dashboard.html">
```

refresh="30" を含めるには、

```
<view refresh="30" template="dashboard.html">
```

次にクロームを追加します。これは検索ビューとまったく同様に機能します。

```
<label>Tutorial Dashboard</label>
<module name="AccountBar" layoutPanel="navigationHeader"/>
<module name="AppBar" layoutPanel="navigationHeader"/>
<module name="Message" layoutPanel="messaging">
  <param name="filter">*</param>
  <param name="clearOnJobDispatch">False</param>
  <param name="maxSize">1</param>
</module>
```

オプションで検索バーを追加できます。これにより、ダッシュボードから検索を実行できるようになります。

```
<module name="SearchBar" layoutPanel="mainSearchControls">
  <param name="useAssistant">true</param>
  <param name="useTypeahead">true</param>
  <module name="TimeRangePicker">
    <param name="selected">This month</param>
  </module>
  <module name="ViewRedirector">
    <param name="viewTarget">simple_search_view</param>
  </module>
</module>
```

```
</module>
</module>
</module>
```

注意：異なるビューで検索を実行するには、ViewRedirector モジュールを追加します。

パネルのレイアウト

次に、パネルをページでどのように表示するかを決定します。パネルは、パネルの親モジュールで `layoutPanel` パラメータを設定することにより、座標系で配列されます。その座標系は、パネルの行と列を指定します。例えば：
`layoutPanel=panel_rowX_colY`。行は何行でも使用することができますが、3 または 4 行に制限することが適当です(2 はデータ表示の標準)。

注意：ダッシュボードは、検索ビューをは異なる `layoutPanels` を使用します。検索ビューは、名前付き `layoutPanels` を使用し、ダッシュボードは座標系を使用します。

例えば、次のように、チュートリアル of のダッシュボードには、2 つのパネル親モジュールがあります。

```
<module name="HiddenSearch" layoutPanel="panel_row1_col1" group="Messages per minute last
hour" autoRun="True">
...
<module name="HiddenSearch" layoutPanel="panel_row1_col2" group="KBps indexed per hour last
2 hours" autoRun="True">
```

また、より大きなパネル内でパネルグループを設定することもできます。次のようになります：

これを設定するには、1 つの親モジュールを指定します。この例は、`StaticContentSample` を使用して、パネルグループ全体のヘッダーを設定します。各パネルは、そのグループ内で置換するための `grp` タグに追加して、`layoutPanel` を指定する 1 つの親モジュールを持ちます。

```
<module name="StaticContentSample" layoutPanel="panel_row2_col1" group="All Indexed Data"
autoRun="True">
  <param name="text">This will show you all of the data you have loaded into index=main over
all time.</param>
<module name="GenericHeader" layoutPanel="panel_row2_col1_grp1">
  <param name="label">Sources</param>
...
  <module name="GenericHeader" layoutPanel="panel_row2_col1_grp2">
  <param name="label">Sourcetypes</param>
...
  <module name="GenericHeader" layoutPanel="panel_row2_col1_grp3">
  <param name="label">Hosts</param>
```

フォーム検索の構築

フォーム検索とは何か

フォーム検索とは何か

フォーム検索は、簡略化した検索インタフェースを表示する Splunk Web のページです。したがって、検索を行うたびにユーザーに検索文字列をすべて入力させる代わりに、フォーム検索を使って、同じユーザーで残っている検索の一部をユーザーにエイリアスすることができます。

Appビルダには3つのフォーム検索の例があります。そのうちのひとつは、sendmail イベントの"from"フィールドに構築された基本検索です。他の2つの例は動的に投入するラジオボタンとドロップダウンを含みます。この2つのフォーム検索ビューは、データに応じて、ラジオボタンとドロップダウンの異なるオプションを提供します。これらの例を使って、各使用ケースに適用してください。

フォーム検索構築方法の概要

フォーム検索は、フィールド上またはその他のデータの識別可能な部分に構築されます。まず始めに、個々のデータと使用用途に適した検索を構築します。次に、この検索のどの部分をエイリアスし、ユーザーから隠すかを特定します。最後に、フォーム検索ビューを構築します。

簡単ダッシュボードと同じ簡略化 XML に構築された簡単フォーム検索を構築することができます。Appビルダの例は、すべてこの簡略化 XML フォーマットを使用します。簡略化 XML の扱い方については、「ダッシュボードの構築方法」を参照してください。動的に投入されるラジオボタンやドロップダウンを持つフォーム検索を構築する場合は、「ダイナミックフォーム検索」を参照してください。

または、検索ビュー内でより高度な ExtendedFieldSearch モジュールを使用することができます。どちらがいいかわからない場合は、簡単フォーム検索をまず使用し、その上に構築します。Appビルダの例を個々のデータに適用することができることを忘れないでください。

簡単フォーム検索

簡単フォーム検索

簡単フォーム検索は、簡単ダッシュボードと同じ簡略化 XML に構築されます。ダッシュボードがフォーム検索 XML で使用するのと同じ XML 設定を使用することができます。

設定

フォーム検索 XML はエレメントをいくつかダッシュボード XML に追加します。これらエレメントはフォームを生成し、適用するフォームエントリーの検索を構築します。次の例は、**username** の値を取得するフォームを作成します。尚、フォームで置き換えられる部分をマークする限り、どんな検索文字列でも入力することができます。

```

<form>
  <label>My form search</label>
  <searchTemplate>${username}$</searchTemplate>
  <fieldset>
    <input type="text" token="username" />
  </fieldset>

  <row>
    <table>
      <title>Top filePaths</title>
      <option name="showPager">true</option>
    </table>
  </row>
</form>

```

- searchTemplate
 - ◆ 検索文字列全体をここに入力します。
 - ◆ ここにはあらゆる種類の検索を置くことができます。ただし、フォームで置換したい語句を\$で囲む必要があります。
- fieldset
 - ◆ 完全な検索文字列を形成するために *searchTemplate* で統合したい1つまたは複数のユーザー入力フィールドを設定します。
 - ◆ トークン属性は、*searchTemplate* からの置換語句を一致させる必要があります。

フォームにラベルを付ける

フォームの前にラベルを表示するようフォーム検索を設定します。この例は、フォームの前に **Enter a user name** を追加します。

```

<input type="text" token="username">
  <label>Enter a user name</label>
</input>

```

デフォルト値を設定する

デフォルトでは、ユーザーが、テキストボックスに入力して値を提供しない場合、トークンが空の文字列で置換されます。デフォルト値を設定するには、次のように設定します。

```

<input type="text" token="username">
  <default>some_user_name</default>
</input>

```

デフォルト値の前に次を設定します。

```

<input type="text" token="username">
  <default>some_user_name</default>
  <prefix>os_username=</prefix>
</input>

```

またはデフォルト値に引用符をつけます。

```

<input type="text" token="username">
  <default>some_user_name</default>
  <prefix>os_username="</prefix>

```

```
<suffix>"</suffix>
</input>
```

ページのロード時にフォームに値を投入するには、seed ノードを使用します。

```
<input type="text" token="username">
  <default>some_user_name</default>
  <prefix>os_username="</prefix>
  <suffix>"</suffix>
  <seed>johnvey</seed>
</input>
```

フォーム出力レイアウト

フォーム検索の出力は、ほとんどの場合ダッシュボードの場合と同じです。ただし、searchTemplate ノードと組み合わせられたときの動作のみが異なります。通常、出力パネルは検索を宣言しません。次の例は、テンプレート検索に一致するイベントを表示する簡単フォーム検索を生成します。

```
<form>
  <label>My form search</label>

  <searchTemplate>sourcetype=p4change $username$ $filePath$</searchTemplate>

  <fieldset>
    <input type="text" token="username" />
    <input type="text" token="filePath">
      <prefix>filePath="</prefix>
      <seed>//current</seed>
      <suffix>"</suffix>
    </fieldset>
  </fieldset>

  <row>
    <event>
      <option name="count">100</option>
    </event>
  </row>
</form>
```

マルチ検索フォームレイアウト

フォーム検索は、複数出力パネルと共通入力セットをペアにした場合に、はるかに便利になる場合があります。searchTemplate ノードをトップレベル階層から各出力パネルに移動することにより、複合フォーム検索を作成することができます。

```
<form>
  <label>Form search example 3 - inverted flow, panel-defined search</label>
  <fieldset>
    <!-- define a common form search input that will be used by all panels
    below that implement a searchTemplate node -->
    <input type="text" token="username">
      <label>Global username</label>
      <default>NON_EXISTENT</default>
      <seed>johnvey*</seed>
    </input>
    <input type="time" />
```

```

</fieldset>
<row>
  <chart>
    <title>Commits over time</title>
    <searchTemplate>sourcetype=p4change OR sourcetype=jira user="$username$" |
timechart count</searchTemplate>
    <option name="charting.chart">area</option>
  </chart>
  <table>
    <title>Top files touched by the user</title>
    <searchTemplate>sourcetype=p4change OR sourcetype=jira user="$username$" | top
filePath</searchTemplate>
  </table>
</row>
<row>
  <table>
    <title>Users vs changetype</title>
    <searchTemplate>sourcetype=p4change OR sourcetype=jira user="$username$" | ctable
user changetype maxcols=4</searchTemplate>
    <option name="count">20</option>
  </table>
  <chart>
    <title>Average lines added by the user</title>
    <searchTemplate>sourcetype=p4change OR sourcetype=jira user="$username$" |
timechart avg(added)</searchTemplate>
    <option name="charting.chart">line</option>
    <option name="charting.legend.placement">none</option>
  </chart>
</row>
</form>

```

上のフォーム検索は4つの異なる検索をディスパッチし、なおかつ、各検索は `fieldset` セクションでユーザーが入力した値を使用します。当然、各検索のトークン属性は、`fieldset` 内で定義された入力ノードの少なくとも1つとマッチする必要があります。

シングル検索、複数後処理

最後に、フォーム検索の特殊置換として、1つの検索を受け取り、後処理で、その検索の異なる側面を表示します。例えば、前述の例の4つの検索は、1つの検索に統合することができます。

```

<form>
  <label>Form search example 4 - inverted flow, panel-defined post-process</label>
  <!-- define a search that returns, in one result set, all of the data that is
needed by the subsequent panels -->
  <searchTemplate>sourcetype=p4change OR sourcetype=jira user="$username$" | head
10000</searchTemplate>
  <fieldset>
    <input type="text" token="username">
      <label>Global username</label>
      <default>NON_EXISTENT</default>
      <seed>johnvey*</seed>
    </input>
    <input type="time" />
  </fieldset>
  <row>
    <chart>
      <title>Commits over time</title>

```

```

        <searchPostProcess>timechart count</searchPostProcess>
        <option name="charting.chart">area</option>
</chart>
<table>
  <title>Top files touched by the user</title>
  <searchPostProcess>top filePath</searchPostProcess>
</table>
</row>
<row>
  <table>
    <title>Users vs changetype</title>
    <searchPostProcess>ctable user changetype maxcols=4</searchPostProcess>
    <option name="count">20</option>
  </table>
  <chart>
    <title>Average lines added by the user</title>
    <searchPostProcess>timechart avg(added)</searchPostProcess>
    <option name="charting.chart">line</option>
    <option name="charting.legend.placement">none</option>
  </chart>
</row>
</form>

```

各パネル内側の searchPostProcess ノードは、最終検索結果を受け取り、別々の検索パイプラインを通じてそれらを返すようにフォーム検索に命令します。基本モデルは、searchTemplate ノードで開始された非変換検索を行い、searchPostProcess ノードで変換検索を適用します。

ダイナミックフォーム検索

ダイナミックフォーム検索

フォーム検索のラジオボタンまたはドロップダウンを検索による値で自動投入します。

ドロップダウン

入力フォーム「ドロップダウン」は、選択可能なオプションをドロップダウンリストで表示します。ドロップダウンのオプションは、内部検索またはユーザーが定義する値により生成することができます。このモジュールは、上述の基本インデックスの type="text" 検索フィールドで定義されたプロパティを引き継ぎます。

- label(必須 : False) 生成されたドロップダウンの横に置くラベル。
- default(必須 : False) 選択すべきデフォルトオプション。デフォルトオプションが見つからない場合、最初のオプションが選択されます。
- prefix(必須 : False) 入力の type="text" の説明を参照。
- suffix (必須 : False) 入力の type="text" の説明を参照。
- choice (必須 : False) 値属性が必要。例 : "bar"。ユーザーが宣言したドロップダウンオプション。これらは、定義した順番で、検索により生成されたオプションの前に表示されます。
- populatingSearch (必須 : False) ドロップダウンのフィールド生成に使用される検索。属性 "fieldForValue" および "fieldForLabel" を必要とする。fieldForValue は populatingSearch から抽出されたフィールド

で、生成されたドロップダウンオプションの値に置かれる。fieldForLabel は populatingSearch から抽出されたフィールドで、生成されたドロップダウンのラベルに置かれる。fieldForLabel は opulatingSearch から抽出されたフィールドで、生成されたドロップダウンのラベルに置かれる。

```
<input type="dropdown" token="username">
  <label>Select Name</label>
  <populatingSearch fieldForValue="suser"
fieldForLabel="suser"><![CDATA[sourcetype=p4change | rex "user=(?<suser>¥w+)@"
  | stats count by suser]]></populatingSearch>
  <default>nagrin</default>
  <choice value="*">Any</choice>
</input>
```

ラジオ

入力フォーム「radio」は、ラジオボタンのリストを表示します。ラジオオプションは、内部検索またはユーザーが定義する値により生成することができます。このモジュールは、上述の基本インデックスの type="text"検索フィールドで定義されたプロパティを引き継ぎます。

- label(必須 : False) 生成されたラジオボタンのセットのトップで置換するラベル。
- default(必須 : False) 選択すべきデフォルトラジオボタン。デフォルトボタンが見つからないばあい、何も選択されません。
- prefix(必須 : False) 入力の type="text"の説明を参照。
- suffix (必須 : False) 入力の type="text"の説明を参照。
- choice (必須 : False) 値属性が必要。例 : "bar"。ユーザーが宣言したラジオボタン。これらは、定義した順番で、検索により生成されたラジオボタンの前に表示されます。
- populatingSearch (必須 : False) ラジオボタンを生成するために使用する検索。"fieldForValue" および "fieldForLabel"が必要。fieldForValue は populatingSearch から抽出されたされたフィールドを定義し、生成されたラジオボタンの値に置かれます。fieldForLabel は populatingSearch から抽出されたされたフィールドを定義し、生成されたラジオボタンのラベルに置かれます。

例 :

```
<input type="radio" token="username">
  <label>Select Name</label>
  <populatingSearch fieldForValue="suser"
fieldForLabel="suser"><![CDATA[sourcetype=p4change | rex "user=(?<suser>¥w+)@"
  | stats count by suser]]></populatingSearch>
  <choice value="*">Any</choice>
</input>
```

高度なフォーム検索

高度なフォーム検索

より洗練されたフォーム検索を作成する場合、検索ビューにある ExtendedFieldSearch モジュールを使用することができます 検索ビューに関する詳細は、「検索ビュー」を参照してください。

設定

検索ビューを開始します。

```
<view onunloadCancelJobs="False" autoCancelInterval="100">
  <!-- autoCancelInterval is set here to 100 -->
  <label>Sample search</label>
  <module name="AccountBar" layoutPanel="appHeader"/>
  <module name="AppBar" layoutPanel="navigationHeader"/>
  <module name="Message" layoutPanel="messaging">
    <param name="filter">*</param>
    <param name="clearOnJobDispatch">False</param>
    <param name="maxSize">1</param>
  </module>
```

次に、作成したいフォーム検索の種類を決定し、次の設定を1つまたは複数選択します。

基本検索置換の例

```
<module name="HiddenSearch" layoutPanel="mainSearchControls">
  <param name="search">sourcetype=$st$</param>
  <module name="ExtendedFieldSearch">
    <param name="intention">
      <param name="name">stringreplace</param>
      <param name="arg">
        <param name="st">
          <param name="default">apache_error</param>
        </param>
      </param>
    </param>
  </module>
  <param name="replacementMap">
    <param name="arg">
      <param name="st">
        <param name="value"></param>
      </param>
    </param>
  </param>
  <param name="field">Sourcetype</param>
  <module name="EventsViewer" layoutPanel="resultsAreaLeft">
    <param name="segmentation">full</param>
  </module>
</module>
</module>
```

ワイルドカードの使用

...

```
<module name="HiddenSearch" layoutPanel="mainSearchControls">
  <param name="search">sourcetype=apache_error *$target$*</param>
  <module name="ExtendedFieldSearch">
    <param name="intention">
      <param name="name">stringreplace</param>
      <param name="arg">
        <param name="target">
          <param name="default">500</param>
        </param>
      </param>
    </param>
  </module>
  <param name="replacementMap">
```

```

    <param name="arg">
      <param name="target">
        <param name="value"></param>
      </param>
    </param>
  </param>
</param>
<param name="field">Wildcard search</param>
<module name="EventsViewer" layoutPanel="resultsAreaLeft">
  <param name="segmentation">full</param>
</module>
</module>
</module>

```

2つの変数を使用

```

<module name="HiddenSearch" layoutPanel="mainSearchControls">
  <param name="search">sourcetype=apache_error $error$ $hours_ago$</param>
  <module name="ExtendedFieldSearch">
    <param name="intention">
      <param name="name">stringreplace</param>
      <param name="arg">
        <param name="error">
          <param name="fillOnEmpty">True</param>
        </param>
      </param>
    </param>
  </param>
  <param name="replacementMap">
    <param name="arg">
      <param name="error">
        <param name="value"></param>
      </param>
    </param>
  </param>
  <param name="field">Multiple replace (apache search)</param>
  <module name="ExtendedFieldSearch">
    <param name="intention">
      <param name="name">stringreplace</param>
      <param name="arg">
        <param name="hours_ago">
          <param name="fillOnEmpty">True</param>
          <param name="prefix">starthoursago=</param>
        </param>
      </param>
    </param>
  </param>
  <param name="replacementMap">
    <param name="arg">
      <param name="hours_ago">
        <param name="value"></param>
      </param>
    </param>
  </param>
  <param name="field">Multiple replace (starthoursago)</param>
  <module name="EventsViewer" layoutPanel="resultsAreaLeft">
    <param name="segmentation">full</param>
  </module>
</module>
</module>
</module>

```

OR の使用

望ましい検索文字列は次のようになります : eventtypetag=authentication tag=cardholder-dest
src_ip="\$SourceIP\$" OR user="\$User\$"

文字列置換を示す"接頭辞"と"接尾辞"パラメータを使用してこれを概算します。 eventtypetag=authentication
tag=cardholder-dest src_ip="\$SourceIP\$" \$User\$

ここで、\$User\$は、'OR user='が語頭に付き、'"が語尾に付きます。

```
<module name="HiddenSearch" layoutPanel="mainSearchControls">
  <param name="search">eventtypetag=authentication tag=cardholder-dest
src_ip="$SourceIP$" $User$</param>
  <module name="ExtendedFieldSearch">
    <param name="field">SourceIP</param>
    <param name="intention">
      <param name="name">stringreplace</param>
      <param name="arg">
        <param name="SourceIP">
          <param name="fillOnEmpty">True</param>
          <param name="value"></param>
        </param>
      </param>
    </param>
  </module>
  <param name="replacementMap">
    <param name="arg">
      <param name="SourceIP">
        <param name="value"></param>
      </param>
    </param>
  </param>
  <module name="ExtendedFieldSearch">
    <param name="field">User</param>
    <param name="intention">
      <param name="name">stringreplace</param>
      <param name="arg">
        <param name="User">
          <param name="fillOnEmpty">True</param>
          <param name="prefix">OR user="</param>
          <param name="suffix">"</param>
        </param>
      </param>
    </module>
  </param>
  <param name="replacementMap">
    <param name="arg">
      <param name="User">
        <param name="value"></param>
      </param>
    </param>
  </param>
  <module name="EventsViewer" layoutPanel="resultsAreaLeft">
    <param name="segmentation">full</param>
  </module>
</module>
</module>
</module>
```

同じ変数の再使用

```
...
<module name="HiddenSearch" layoutPanel="mainSearchControls">
  <param name="search">eventtypetag=config_file source=$File$ OR $File$</param>
  <module name="ExtendedFieldSearch">
    <param name="field">File</param>
    <param name="intention">
      <param name="name">stringreplace</param>
      <param name="arg">
        <param name="File">
          <param name="value"></param>
        </param>
      </param>
    </param>
  </module>
  <param name="replacementMap">
    <param name="arg">
      <param name="File">
        <param name="value"></param>
      </param>
    </param>
  </param>
  <module name="EventsViewer" layoutPanel="resultsAreaLeft">
    <param name="segmentation">full</param>
  </module>
</module>
...
<module name="HiddenSearch" layoutPanel="mainSearchControls">
  <param name="search">* | stats count by $st$</param>
  <module name="ExtendedFieldSearch">
    <param name="field">Count by field</param>
    <param name="intention">
      <param name="name">stringreplace</param>
      <param name="arg">
        <param name="st">
          <param name="value"></param>
        </param>
      </param>
    </param>
  </module>
  <param name="replacementMap">
    <param name="arg">
      <param name="st">
        <param name="value"></param>
      </param>
    </param>
  </param>
  <module name="EventsViewer" layoutPanel="resultsAreaLeft">
    <param name="segmentation">full</param>
  </module>
</module>
</view>
```

検索ビューの構築

検索ビュー

検索ビュー

Splunk Web のページはすべてビューです。ビューの機能に関するより基礎的な事項は、「カスタム UI の構築」を参照してください。検索ビューは、Flash 時間軸などのページです。次の説明に従って、新規検索ビューを作成します。

これは、ビューを設定するための基本概念の一般概要です。検索ビューの設定方法に関する詳細は、「検索ビューを開始する」を参照してください。ダッシュボードの作成については、「ダッシュボードの設定」のページを参照してください。

使用している App の検索ビューを設定するには、次のステップに従います。

1. ビューに含めたいモジュールを決定します。おそらくほとんど場合、検索モジュールを使用します。
2. `<view_name>.xml` で各モジュールを設定します。
3. App ディレクトリ : `$SPLUNK_HOME/etc/apps/<app_name>/data/ui/views/`にあるビューディレクトリに`<view_name>.xml` を置きます。
4. App に複数のビューがある場合、「ビューコレクションの作成」にある説明に従って UI でそれらをアレンジします。

検索ビューの開始

検索ビューの開始

このページでは、簡単検索ビューの構築方法について説明します。このビューをロードするには、このページの例セクションから最終結果を、構築した次の helloworld app にコピーします。

`$SPLUNK_HOME/etc/apps/helloworld/default/data/ui/views/simple_search_view.xml`. 次に、次のページに移動します。

`http://localhost:8000/en-US/app/helloworld/simple_search_view`

(ホストおよびポートを、使用するインストールホストとポートで置き換えてください。)

ビューの開始

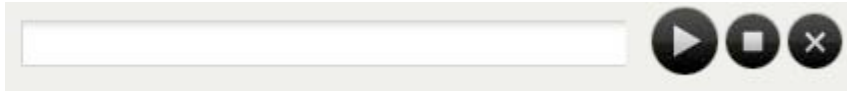
App のビューディレクトリに `view.xml` ファイルを作成します。

編集するファイルを開き、周囲の **view** タグを追加します :

```
<view>
</view>
```

検索バーの追加

最も基本的な検索ビューは、ビューにアクセスすると単に検索バーを表示するだけのものです。



このビューを構築するには、SearchField モジュールが必要です。

このモジュールを検索後であらかじめ投入することができますが、今回は空白にしておきます。

```
<view>
  <!-- This module renders the search box -->
  <module name="SearchField" layoutPanel="mainSearchControls">
</view>
```

layoutPanel を設定して、このビューの表示場所を Splunk に指示します。詳細は、「モジュール概要」の layoutPanels を参照してください。

結果表示エリアの追加

検索結果を表示して相互作用する必要がある場合があります。その場合は、EventsViewer モジュールを追加します。

```
<view>
  <!-- This module renders the search box -->
  <module name="SearchField" layoutPanel="mainSearchControls">
    <!-- This module renders the resulting events from your search -->
    <module name="EventsViewer"/>
  </module>
</view>
```

モジュール継承

SearchField モジュールタグは、EventsViewer モジュールの後まで閉じないことに注意してください。その理由は、EventsViewer は SearchField の子だからです。つまり、EventsViewer は、検索ボックスに入力された検索を引き継ぎます。子はまた、レイアウトパネル設定も引き継ぎます。

子を設定した後、残りのモジュールを閉じることを忘れないでください。

chrome の追加

以上で、このビューの基本的な構造ができました。しかし、ページをスムーズに生成するためには、追加する必要があるモジュールがさらにいくつかあります。まず、chrome を追加する必要があります。特に、トップのナビモジュールである AccountBar と AppBar です。これらがトップナビに表示されるように指定するには、layoutPanel="navigationHeader" を追加します。

```
<view>
  <!-- top nav chrome -->
  <module name="AccountBar" layoutPanel="navigationHeader"/>
  <module name="AppBar" layoutPanel="navigationHeader"/>
  <!-- This module renders the search box -->
  <module name="SearchField" layoutPanel="mainSearchControls">
```

```

    <!-- This module renders the resulting events from your search -->
    <module name="EventsViewer"/>
  </module>
</view>

```

パラメータの指定

次に、ユーザーが結果のページをスムーズに見るための方法が必要になります。特に、検索結果が 1 ページに収まらない場合に必要です。この場合は、**Paginator** モジュールを追加します。

```

<module name="Paginator">
  <param name="entityName">events</param>

```

paginator では **entityName** パラメータを指定していることに注意してください。このパラメータは必須です。Paginator は他にも複数のパラメータをとることができますが、必須ではありません。したがって、本例では使用しません。

この場合、ビューXML は次のようになります。

```

<view>
  <module name="AccountBar" layoutPanel="navigationHeader"/>
  <module name="AppBar" layoutPanel="navigationHeader"/>
  <module name="SearchField" layoutPanel="mainSearchControls">
    <module name="Paginator">
      <param name="entityName">events</param>
      <module name="EventsViewer"/>
    </module>
  </module>
</view> </pre>

```

ビューに名前を付ける

ビューの作成が終わると、それにわかりやすい名前を付けます。名前を付けるには、最初の部分にある<view>タグのすぐ後に<label>タグを追加します。

```

<view>
  <label>Basic Search View</label>

```

例

以下にビュー設定の全体を示します。尚、インデントは重要ではありませんが、継承がわかりやすいように使用することができます。

```

<view>
  <label>Basic Search View</label>
  <!-- Top nav chrome -->
  <module name="AccountBar" layoutPanel="navigationHeader"/>
  <module name="AppBar" layoutPanel="navigationHeader"/>
  <!-- This module renders the search box -->
  <module name="SearchField" layoutPanel="mainSearchControls">
    <!-- You need paginator to page through all the results from the search -->
    <module name="Paginator">
      <param name="entityName">events</param>
      <!-- This module renders the resulting events from your search -->
      <module name="EventsViewer"/>
    </module>
  </module>
</view>

```

</view>

応用

以上で、基本的な検索ビューの設定が終わりました。独自の検索ビューを構築するには、他の検索モジュールを追加します。`$SPLUNK_HOME/etc/apps/search/default/data/ui/views/flashtimeline.xml` にアクセスしてデフォルト検索ビューをいつでも参照することができます。このビューをロードするには、次にアクセスします。

<http://localhost:8000/en-US/app/search/flashtimeline>

(ホストおよびポートを、使用するインストールホストとポートで置き換えてください。)

設定とナレッジオブジェクトの追加

開発者のための設定とナレッジオブジェクトガイド

開発者のための設定とナレッジオブジェクトガイド

App ワークスペースの構築が終わると、使用する App のデータレイヤを設定するための設定を追加することができます。その際には、オブジェクトを App に追加して、App の表示レイヤを設定します。ナレッジオブジェクトおよびコンフィギュレーションを App に追加する場合は、Splunk 管理または Splunk サーバーバンクエンドにある設定ファイルシステムにより行うことができます。

このセクションでは、利用できる主な設定オプションの概要を説明します。設定に関するその他の事項は、管理マニュアルに記載されています。ナレッジオブジェクトの設定に関するその他の事項は、ナレッジマネージャマニュアルを参照してください。

カスタム設定の設定が終わると、関連の App 設定を、使用する App を設定するユーザーが閲覧することができるように、App セットアップ画面の設定が必要です。

設定

App 設定は、使用する App のデータレイヤを設定します。App のデータレイヤはデータ入力とその他、Splunk がデータを扱う方法を指定するための設定を含んでいます。これにより、App が利用できるデータの種類、Splunk インスタンスへのアクセス方法、および Splunk がそれを保存する方法をカスタマイズすることができます。

すべてではありませんが Splunk の多くの App はバンクエンド設定を含んでいます。管理マニュアルの設定ファイルレファレンスページにある設定ファイルリストから設定を使用することができます。詳細は、管理マニュアルの「設定ファイルの機能」を参照してください。

尚、設定はすべてグローバルです。つまり、デフォルトですべての App が利用することができます。それら設定を App のディレクトリに置くことにより、設定を分離することができます。ただし、App がインデックスするデータはすべて常に他の App も利用することができます。

Inputs

App の入力を設定します。特定の種類のデータをインデックスする必要がある場合があります。その場合は、管理マニュアルの「入力の設定」を参照してください。

Indexes

App のデータを保存するために、カスタムインデックスを設定します。必須ではありませんが、データが他の入力と混し合わないよう、これを行うことができます。

props と transforms

カスタムデータ処理のルールを追加します。特殊なデータがある場合、それを処理するためのルールを props.conf に作成し、transforms.conf によりそれをデータにリンクします。App のユーザーが App を分散設定にインストールする場合は、それらユーザーが props.conf の保存場所を知っていることを確認してください。

Restmap

これはカスタムエンドポイントのためのものです。4.0 ではまだサポートされていません。

Authorize

検索コマンドの機能を追加するためのものです。尚、App ユーザーは、使用する機能を、システムに既に存在している役割に割り当てる、または、システムに既に存在しているユーザーに使用する役割を割り当てる必要があります。

User prefs

user prefs によりデフォルト App を設定します。詳細は、本書の「デフォルト App の設定方法」を参照してください。

ナレッジオブジェクト

ナレッジオブジェクトは、使用する App の表示レイヤで利用できるすべてのオブジェクトです。これは、どのデータをどのようにカスタマイズして App ユーザーに表示するかを設定します。権限の設定内容によりませんが、App ユーザーは App に含まれるナレッジオブジェクトとインタラクトし、それを修正することができます。

保存済み検索とレポート

保存済み検索とレポートは、ほとんどの Splunk App に装備されています。保存済み検索とレポートを使用して、重要なデータを動的に取り出します。必要に応じて、それらをダッシュボードに表示したり、実行する Splunk Web の保存済み検索ドロップダウンに追加します。保存済み検索を使用すると、App にロードしたデータに対して必要な検索が簡単に行えます。

App での保存済み検索およびレポートの使用方法は、「保存済み検索」を参照してください。

ビューとダッシュボード

ビューとダッシュボードは、App で構築したナレッジオブジェクトを表示します。ダッシュボードは通常、関連する検索へのリンク、および App をロードする際に表示するレポートへのリンクを含んでいます。検索ビューは、個別に検索を実行できます。

イベントタイプ

イベントタイプを設定して、App のナレッジを取得し共有できるようにします。

フィールド

タグ

コマンド

カスタム検索コマンドを追加します。新しい検索コマンドのそれぞれの機能を忘れずに追加してください。

オブジェクトに対する許可

App のオブジェクトに対するデフォルト権限は、default.meta で設定する、または Splunk 管理で各オブジェクトの権限を修正することにより設定します。

App.conf

App.conf

app.conf を設定して、App が Splunk Web に表示されるようにします。App ビルダを使用する場合、ビルダが代わりにこれを設定します。

設定

App を有効にし、Splunk Web で作成するには、次のスタanzas を \$SPLUNK_HOME/etc/apps/<app_name>/default/app.conf に追加します。

```
[ui]
is_visible = true
label = <name>
```

- スタanzas には [ui] ヘッダーが必要です。
- App が Splunk Web のドロップダウンメニューに表示されるようにするには、is_visible を true に設定します。
- label に使用する App の名前を設定します。

App を app launcher に追加する

次のスタanzas を app.conf に追加し、使用する App が app launcher に追加できるようにします。各属性を次のとおり入力します。

```
[launcher]
author=<author of app>
description=<textual description of app>
version=<version of app>
```

次の画像を、使用する App の ../appserver/static/ ディレクトリに確実に追加してください。

- appIcon_<appname>.png: このアイコンは、Launcher の使用する App の名前の左側に表示されます。

- screenshot_<appname>.png: このスクリーンショットは、Launcher の使用する App の説明の上に表示されます。

固定コンテンツの更新

Splunk の appserver は、使用する App の静的アセット(画像、CSS、JavaScript など)をすべて取得します。App の新規バージョンをリリースする場合、app.conf を設定して更新したアセットをユーザーが利用できるようにすることができます。install スタンザ属性を app.conf ファイルに追加し、ビルドナンバーを指定します。例えば：

```
[install]
build = 2
```

- スタンザには[install]ヘッダーが必要です。
- ビルドナンバーに固有の ID を設定します。こうすると、誰かが App の新規バージョンをインストールしたとき、App にパッケージされた新規アセットをすべて取得できるようになります。

App で作業中に App の固定コンテンツを再ロードし、ブラウザのキャッシュをクリアしたくない場合は、次の URI を入力し、そこにあるボタンをクリックします。

http://localhost:8000/_bump

(使用している Splunk の host:port で置き換える。)

保存済み検索

保存済み検索

保存済み検索は、最も一般的なデベロッパ設定です。これにより共通検索の導入が容易になります。ダッシュボードとフォーム検索はすべて保存済み検索上に構築されます。したがって、Splunk の検索言語に慣れ、データの重要な特性を明らかにできる検索をいくつか作成し、次にそれらをダッシュボードに組み込みます。

ダッシュボードは、チャート、グラフ、リンクなどの形で保存済み検索をわかりやすく表現します。したがって、App ユーザーが何をしようとしているのか、そして、保存済み検索を使用してどのようにその目的を支援することができるかを把握する必要があります。そして、保存済み検索をダッシュボードとビューコレクションに追加します。

例: 使用しているサイトのウェブトラフィックを明確にするための App を構築するとします。参照 URI をフォローし、ダウンロード統計を追跡し、また、ウェブログで利用できるデータを利用して保存済み検索を作成することができます。

検索の構築

これまでに Splunk の検索言語を取り扱ったことがない場合は、ユーザーマニュアルの「検索および調査の方法」を読んでください。データの最も重要な特性を明確にし、App ユーザーの最終目的を支援できるよう、検索を構築します。したがって、例えばヘルプデスク App を構築しようとする場合、そのチームが App から何を得たいのかを把握する必要があります。そして、この情報を収集する検索を構築し、便利な方法でそれを提供します。

検索の保存

構築する検索の構想が決まったら、それを保存して、再実行できるようにします。検索は、Splunk Web から、または Splunk 管理内で保存することができます。または、使用する App のディレクトリにある `savedsearches.conf` を作成することができます。App 開発者が保存済み検索を作成するための最良の方法は、Splunk 管理を、使用する App のコンテキスト内で利用することです。

1. Splunk Web の左上にあるドロップダウンから App を選択します。
2. 次に、右上にある**管理**リンクをクリックします。
3. **App 設定**タブを選択します。これらはすべて 1 つの App を対象にする設定です。保存済み検索またはイベントタイプを作成することができます。または、ビュー、ナビ、検索コマンドを編集します。
4. 新規保存済み検索を構築して保存します。それが App に関する保存済み検索ページに保存され、次の場所からアクセスできます。

```
http://localhost:8000/en-US/manager/<app_name>/saved/searches
```

(localhost:8000 を使用している Splunk ホストおよびインストールポートで置き換えてください。)

検索を App に統合する

保存済み検索を Splunk 管理(または Splunk Web)で最初に作成すると、`$SPLUNK_HOME/etc/users/`のユーザーディレクトリに追加されます。保存済み検索は、その App のディレクトリ、特に `$SPLUNK_HOME/etc/apps/<app_name>/default/savedsearches.conf` にある場合、App に属します。保存済み検索を App ユーザー全員で共有する場合、または App 名前スペースに追加する場合は、その検索に権限を設定します。

1. Splunk 管理で、使用している App の保存済み検索ページにアクセスします。
2. リストビューで保存済み検索を探し、その隣にある**権限**リンクをクリックします。
3. **保存済み検索**の共有ボックスをクリックします。これにより、その検索は、ユーザーディレクトリからその App のディレクトリに移動します。
4. オプションで、ユーザーの読み込み/書き込み権限をアクセスコントロールリストで設定します。

App セットアップ画面の設定

App セットアップ画面の設定

このページでは、使用している App のセットアップ画面を作成する方法を説明します。この画面を使用して、App 管理者が設定する必要があるカスタム設定を表示し、App を利用可能にします。例えば、App を構築する入力を指定したり、保存済み検索の特定の属性をセットアップします。

作業の基本単位は `<input>` です。これは 3 項目(エンドポイント、エンティティ、フィールド)その他、データタイプ、検

証情報、名前/ラベルなどのデータのモデリングに使用する情報を対象にするものです。(エンドポイント、エンティティ、フィールド属性)は、入力が読み込み/書き込みを行うオブジェクトを特定します。例えば、(endpoint=admin/savedsearch entity=saved-search-name field=cron_schedule) - エンドポイント/エンティティは設定される App に対して相対的です。エンドポイント/エンティティは他のブロックから引き継ぐことができます。

入力は 1 つのグループにまとめ、<block>を作成することができます。このブロックは次の目的を持ちます。

1. 同一のオブジェクトグループからの情報を含む場合、そのブロックが含むすべての入力/ブロックにより引き継がれるオプションの(エンドポイントまたはエンティティ、あるいはその両方)属性を含むことができます。
 2. ターゲットが正規表現の場合、反復概念を提供します-正規表現は REST エンティティの名前に一致します。
 3. 類似した設定項目をグループ化します。
 4. 既に存在するエントリを編集する代わりに、新規エントリを作成し、エンティティ名を"_new"に設定します。
- 注意： 必要なフィールド"名"を入力として確実に追加してください。

'<input>'内のノードは、\$field_name\$の記述を使って、それが属するエンティティ内のフィールド値を参照することができます。これは、説明/ラベルの構成に便利です。

ノード属性：

- endpoint: ブロック/入力がアドレス指定するエンティティ/オブジェクトの、
https://hostname:port/servicesNS/nobody/<app-name>/に対する相対 REST エンドポイント。
- entity: アドレスを指定するエンティティの名前。
- field: 設定するフィールド(入力レベルでのみ有効)。
- mode: (bulk|iterate) エンティティ属性が正規表現の場合、バルク更新として取り扱う、または全一致エンティティに反復し、各フィールドについて入力を要求する。デフォルトは iterate。

<setup>

```
<block title="Basic stuff" endpoint="admin/savedsearch" entity="foobar">
  <text> some description here </text>

  <input field="is_scheduled">
    <label>Enable Schedule for foobar</label>
    <type>bool</type>
  </input>

  <input field="cron_scheduled">
    <label>Cron Schedule</label>
    <type>text</type>
  </input>

  <input field="actions">
    <label>Select Active Actions</label>
    <type>list</type>
  </input>

  <!-- bulk update -->
  <input entity="*" field="is_scheduled" mode="bulk">
    <label>Enable Schedule For All</label>
    <type>bool</type>
  </input>
</block>
```

```

<block title="configure the inputs" endpoint="properties/inputs/">
  <block entity="*" mode="iter">
    <input field="disabled">
      <label>disable $name$</label>
      <type>bool</type>
    </input>
  </block>

  <block title="create a new input" entity="_new">
    <input target="name">
      <label>Input stanza</label>
      <type>text</type>
    </input>
    <input target="disabled">
      <label>Disabled</label>
      <type>bool</type>
    </input>
  </block>

</block>

<!-- example config for "Windows setup" defined:
  http://twiki/twiki/bin/view/Main/AppSetup
-->

<block title="Collect Your Event Logs" endpoint="properties/inputs/" >
  <text> some description here </text>

  <block
entity="WinEventLog:(Setup|System|Security|Application|ForwardedEvents)" mode="iter">
    <input field="disabled">
      <label>$disabled_label$</label>
      <type>bool</type>
    </input>
  </block>

</block>

<block title="Index Your Local Registry" endpoint="properties/regmon-filters/">
  <text> some description here </text>

  <block entity="(User|Machine) keys">
    <input field="disabled">
      <label>$disabled_label$</label>
      <type>bool</type>
    </input>
    <input field="baseline">
      <label>$baseline_label$</label>
      <type>bool</type>
    </input>
  </block>

</block>

<block title="Collect Local Statistics" endpoint="properties/wmi/" >
  <text> some description here </text>

  <block entity="WMI:CPUTime|WMI:Memory|WMI:LocalDisk|WMI:FreeDiskSpace">

```

```
    <input field="disabled">
      <label>${disabled_label}</label>
      <type>bool</type>
    </input>
    <input field="interval">
      <label>Poll Every (seconds)</label>
      <type>text</type>
      <validation>¥d+</validation>
    </input>
  </block>
</block>
```

```
</setup>
```

App 権限の設定

App 権限の設定

App のオブジェクトには役割により設定されるアクセスコントロールがあります。アクセスコントロールを使用して、どのユーザーがどのオブジェクトに対し読み込みまたは書き込みをすることができるかを設定します。現時点で、オブジェクトタイプには次のものがあります。

- 保存済み検索
- イベントタイプ
- 検索コマンド
- ビュー
- Splunk 管理のページ
- ダッシュボード
- ビューコレクション

Splunk 管理の権限の設定

Splunk 管理のオブジェクトごとに権限を設定することができます。次の手順に従います。

1. Splunk 管理に移動します。
2. 右側にあるオブジェクトページの 1 つをクリックします。例えば、保存済み検索をクリックします。また、与えられた App の全設定に権限を設定する場合は、すべての設定を選択することもできます。
3. 権限リンクをクリックします。
4. リスト表示されたすべての役割に対して読み込み/書き込みの権限を設定します。
5. 保存をクリックします。

バンクエンドの権限の設定

default.meta を使用して、使用している App のすべてのオブジェクトに対して読み込み/書き込み権限を設定します。

次の手順に従います。

1. default.meta を、使用している App のデフォルトディレクトリ：
\$SPLUNK_HOME/etc/apps/<app_name>/metadata/default.meta に追加します。
2. 次に、このファイルを編集して、App のオブジェクトの権限を設定します。
3. 各オブジェクトのエントリー、またはあるタイプの全オブジェクトを追加します：

```
[/<owner>/<App>/<object_type>/<object_name>]  
access = read : [ <comma-separated list of roles> ], write : [ comma-separated list of roles ]
```

- owner はオブジェクトを作成したユーザーです。デフォルトおよびグローバルオブジェクトでは、大抵が "nobody" となります。
- App は、オブジェクトが存在する App ディレクトリです。
- Object type は、上記にリスト表示されたタイプの 1 つです(保存済み検索、イベントタイプ、ビューなど)。
- object name は、保存済み検索、ビュー、イベントタイプなどに与える名前です。

オブジェクト別の権限の設定

オブジェクトに名前を付けることにより、オブジェクト別に権限を設定することができます。例えば、このエントリーは、"Splunk errors in the last 24 hours"の保存済み検索に対して、管理者の役割、つまり、読み込み/書き込み権限を与えます。

```
[/nobody/search/savedsearches/Splunk%20errors%20last%2024%20hours]  
access = read : [ admin ], write : [ admin ]
```

あるタイプの全オブジェクトに対する権限の設定

タイプを指定して、そのすべてのオブジェクトに権限を設定することができます。このエントリーは、読み込み権限を全員に、書き込み権限を管理者および App の全イベントタイプのパワーユーザーに与えます。

```
[/nobody/search/eventtypes]  
access = read : [ * ], write : [ admin, power ]
```

オブジェクトのグローバル設定

デフォルトでは、オブジェクトは、それを作成した App 内でのみ見ることができます。したがって、helloworld App のイベントタイプを作成する場合、その App でのみ表示されます。オブジェクトをすべての App で利用できるようにするには、次の行を default.meta のオブジェクトのエントリーに追加します。

```
export = system
```

例えば：

```
[/nobody/helloworld/eventtypes]  
access = read : [ * ], write : [ admin, power ]  
export = system
```

これにより、helloworld のすべてのイベントタイプは、使用している Splunk インストール内のすべての App で見ることができるようになります。

App のルック・アンド・フィールのカスタマイズ

App のルック・アンド・フィールのカスタマイズ

App のルック・アンド・フィールのカスタマイズ

ナビゲーションメニューのカスタマイズ

ナビゲーションメニューのカスタマイズ

ここまでで、App のビューをすべて構築できました。次のステップでは、Splunk Web で表示するためにアレンジします。Splunk Web の App バーのドロップダウンメニューに割り当てるコレクションにビューをアレンジします。ビューを表示する順番、およびそれらを表示するメニューを指定します。

このページのステップに従って、App にあるすべてのビュー、検索およびレポートを 1 つにまとめます。また、このステップにより、デフォルトビューを指定します。デフォルトビューは、App を開始した際に最初に表示され、ユーザーが左上隅にあるロゴをクリックした時にロードされるビューです。

設定

Splunk 管理またはファイルシステムを使用して、ナビゲーションメニューを作成および編集します。

1. 使用している App のナビディレクトリ：\$SPLUNK_HOME/etc/apps/<Appname>/default/data/ui/nav/default.xml に "default.xml" と呼ばれるファイルを作成します。

注意： App ビルダを使って App を作成すると、このファイルは自動生成されます。Splunk 管理の外部で編集する場合は、ブラウザで次の URL にアクセスすることによりナビをリフレッシュすることができます。

```
https://<splunkserver>:<splunkmgmtport>/servicesNS/admin/<appname>/data/ui/nav?refresh=1
```

2. Splunk 管理でナビゲーションメニューをクリックしてこのファイルを編集することができます。クリックすると以下にアクセスします。

```
http://localhost:8000/en-US/manager/<appname>/data/ui/nav
```

注意： ホストおよびポートは、使用するインストールホストおよびポートで置き換えてください。appname を、使用している App 名に置き換えてください。

3. 次に、ナビゲーションメニューを設定します。これは、Splunk Web の App のドロップダウンメニューに割り当てられます。

```
<nav>
  <collection label="Dashboards">
    <view name="mydashboard" />
  </collection>
  <collection label="Search views">
    <view name="mysearchview" />
  </collection>
</nav>
```

```
</collection>
</nav>
```

この例は、"mydashboard"という名前のビューを1つ、Splunk Webのダッシュボードドロップダウンに追加し、"chart"という別のビューを検索ドロップダウンに追加します。

メニューのカスタマイズ

ドロップダウンメニューのタイトルは自由に変更できます。例えば、ダッシュボードメニューを **Ponies** に変更するには次のようにします。

```
<nav>
  <collection label="Ponies">
    <view name="mydashboard" />
  </collection>
  <collection label="Search views">
    <view name="mysearchview" />
  </collection>
</nav>
```

デフォルトビューの設定

デフォルトビューを指定します。このビューは、ユーザーが App をロードした際に表示されるビューです。これはまた、ユーザーが左上隅のロゴをクリックした際に表示されるビューです。

ビューをデフォルトに指定するには、`default="true"`タグを追加します。

```
<nav>
  <collection label="Ponies">
    <view name="mydashboard" />
  </collection>
  <collection label="Search views">
    <view name="mysearchview" default="true" />
  </collection>
</nav>
```

デフォルトに指定されたビューがない場合、`default.xml`に記載された最初のビューがデフォルトになります。`default.xml`にビューが記載されていない場合、権限を読み込んだ最初のビュー(アルファベット順)が表示されます。

すべてのビューを動的に含める

ビューコレクションに記載されていないすべてのビューを、明示的に記述せずに含めます。`source="unclassified"`タグのビューを使用します。

```
<nav>
  <collection label="Dashboard">
    <view name="mydashboard" />
  </collection>
  <collection label="Search views">
    <view name="mysearchview" default="true" />
    <view name="anothersearchview" default="true" />
  </collection>
  <collection label="Others">
    <view source="unclassified" />
  </collection>
```

```
</nav>
```

以上で、default.xml に明示的に記載されていなかったビューがすべて、Splunk Web の「その他」ドロップダウンに表示されます。

利用可能なすべてのビュー(記載されているも含める)を含めるには、次のように指定します。

```
<view source="all" />
```

ネスト型メニュー

ネスト型メニューを作成するには、ビューコレクションを子として既存のビューに追加します。

```
<nav>
  <collection label="Dashboard">
    <view name="helloworlddash" />
  </collection>
  <collection label="Views">
    <view name="helloworldview" default="true" />
    <collection label="Others">
      <view source="unclassified" />
    </collection>
  </collection>
</nav>
```

ビューに直接リンクする

ナビゲーションメニューからビューに直接リンクします。このビューは、ドロップダウンメニューに表示される代わりに、ナビゲーションメニューにリンクとして表示されます。*nav* の直下に *view name=mychart* を追加します。

```
<nav>
  <view name="mychart" />
  <collection label="Dashboard">
    <view name="mydashboard" />
  </collection>
  <collection label="Searches">
    <view name="mysearchview" default="true" />
  </collection>
  <collection label="Others">
    <view source="unclassified" />
  </collection>
</nav>
```

ビューを隠す

ビューを隠してナビゲーションメニューで選択できないようにするには、ビューの XML を編集します。

```
<view isVisible="false">
  ...
</view>
```

保存済み検索とレポートの追加

保存済み検索およびレポートをナビゲーションメニューに追加します。この例は保存済み検索を保存済み検索ドロップダウンメニューに追加します。

```

<saved name="MySavedSearch" />
<nav>
  <collection label="Dashboard">
    <view name="mydashboard" />
  </collection>
  <collection label="Searches">
    <view name="mysearchview" default="true" />
  </collection>
  <collection label="Others">
    <view source="unclassified" />
  </collection>
  <collection label="Saved Searches">
    <saved name="mysavedsearch" />
  </collection>
</nav>

```

以上で、保存済み検索 *mysavedsearch* が保存済み検索ドロップダウンに表示されます。

view= タグを *saved* タグに追加することにより、保存済み検索をロードするビューを指定することができます。

```

<nav>
...
  <collection label="Saved Searches">
    <saved name="mysavedsearch" view="mychart" />
  </collection>
</nav>

```

Splunk は、*savedsearches.conf* スタンザにある *'view'* プロパティをチェックします。何も指定しない場合、検索 App の *'flashtimeline'* ビューが使用されます。

保存済み検索はまた、ビューと同様にネストすることができます。

```

<nav>
...
  <collection label="Saved Searches">
    <saved name="Daily indexing volume by server" view="charting" />
    <collection label="Errors">
      <saved source="unclassified" match="error" />
    </collection>
    <saved source="unclassified" />
  </collection>
</nav>

```

保存済み検索を動的に含める

ビューの動的な追加と同様に、無名の保存済み検索を自動的に含めることができます。これには、*saved source="unclassified"* を追加します。

```

<nav>
  <collection label="Dashboard">
    <view name="mydashboard" />
  </collection>
  <collection label="Searches">
    <view name="mysearchview" default="true" />
  </collection>
  <collection label="Others">
    <view source="unclassified" />
  </collection>

```

```
<collection label="Saved Searches">
  <saved source="unclassified" />
</collection>
</nav>
```

この例は、App にある未分類の保存済み検索をすべて保存済み検索メニューにロードし、アルファベット順にソートします。

サブストリングの一致で自動リストを制限する

自動リストはサブストリングの一致により制限することができます。例えば、"match"という語句を含む名前の未分類検索をすべてコレクションに表示するには、saved source="unclassified" match="<term>"を使用します。

一方、App が利用できる検索とレポートをすべて含む自動リストを特定の語句で設定するには、saved source="all" match="<term>"を使用します。

文字列の比較では、大文字と小文字を区別しません。

```
<nav>
...
  <collection label="Errors">
    <saved source="all" match="error" />
  </collection>
</nav>
```

この例は、"Errors"検索コレクションを生成します。これは、"error"というサブストリングをその名前に使って、保存済み検索をすべて自動的にリスト表示します。その検索には、ナビメニューの他の場所に既に表示されたものを含む場合があります。

ウェブリソースをビューに追加

ウェブリソースをビューに追加

Splunk のウェブリソースモジュールを使って、IFrames または HTML をビューに追加することができます。

HTML の追加

ServerSideInclude モジュールを使用して、HTML をビューに追加します。

1. HTML ファイル、例えば **foo.html** を、\$SPLUNK_HOME/etc/apps/<appname>/appserver/static に作成します。
2. ウェブコンテンツのすべてをそのファイルに追加します。
3. ビューXML を更新して、ServerSideInclude モジュールを含めます。

```
<view>
  <module name="ServerSideInclude">
    <param name="src">foo.html</param>
  </module>
</view>
```

注意：ビューはそれぞれの\$SPLUNK_HOME/etc/apps/<appname>/default/data/ui/views/<view>.xml にある必要があります。

4. ブラウザで表示します。

```
http://<hostname>:<port>/app/<appname>/<view>
```

画像の追加

画像を追加するには、Mako テンプレートユーティリティ機能 `make_url` を HTML 内に使用します。

```

```

リンクの追加

ServerSideInclude で含めた HTML テンプレートからリンクを作成する場合には、以下の 2 つの形式があります。

- スタティック：

```
<a href="/manager/foo/bar">click me</a>
```

- ダイナミック：

```
<a href="$ {make_url('/manager/foo/bar')} ">click me</a>
```

その違いは、ダイナミックバージョンでは `'make_url'` メソッドを使用することです。これは、URL に現在の場所を適合させ、スタティックコンテンツまたはプロキシインスタンスに関連するモディファイアを挿入します。 `make_url` を使用して、App 名のようなランタイム情報を挿入することもできます。

```
<a href="$ {make_url(['manager', APP['id'], 'foo', 'bar'])} ">click me</a>
```

尚、この二つ目の形式は、リストを第 1 引数として渡し、標準文字列として渡しません。

HTML テンプレートシステム内で利用できる共通変数のいくつかを次にあげます。

- 現在のネームスペース：

```
APP['id']
```

- 現在の App のわかりやすいラベル：

```
APP['label']
```

- 現在のビュー id (URI に表示)：

```
VIEW['id']
```

- 現在のビューラベル：

```
VIEW['label']
```

ページのスタイル設定

CSS を使用してページのスタイルを設定します。HTML ページを作成したら、対応する CSS ファイルを作成して、ページのスタイルを設定することができます。

注意： Splunk の CSS はスコープされません。したがって、CSS をページに追加する場合、クラス名を確実にスコー

プしてください。スタイルをグローバルセレクタに適用しないでください(例 : body {background:pink;})。代わりに、クラスセレクタを使用して、スコープとコリジョンを制御します(例 : .myclass {background:pink;})。

1. CSS について理解を深めてください。CSS がよくわからない場合は、http://www.w3schools.com/Css/css_syntax.asp の CSS リソースで確認してください。特に、CSS クラスセレクタの機能について理解してください。HTML ドキュメントでは、クラスセレクタは、CSS でスタイル設定する XHTML 要素です。例えば :

```
<div class="bar">
  w00t!
</div>
```

クラスセレクタは、CSS で"."と、それに続くクラス属性値を使って定義されます("."とクラス名の間はスペースなし)。例えば :

```
.bar {
  background:pink;
}
```

つの XHTML ドキュメント内で同じクラス名を複数回使用することができます。

2. CSS ファイル、例えば **foo.css** を、`$SPLUNK_HOME/etc/apps/<appname>/appserver/static` に作成します。
3. CSS ルールをそのファイルに追加します。

```
.bar {
  background:pink;
}
```

すると、"bar"に一致するクラス属性を持つ要素がピンクになります。

```
<div class="bar">
  <h1>I have a pink background now!</h1>
</div>
```

IFrame の追加

IFrameInclude モジュールを使って、IFrame をビューに追加します。

1. リンク先の URI を定めます。
2. ビューXML を更新して、次の URI にリンクする IFrameInclude モジュールを含めます。

```
<view>
  <module name="IFrameInclude">
    <param name="src">myawesomewebsite.com</param>
  </module>
</view>
```

3. オプションで、高さも設定できます。この例では、高さを 42 ピクセルに設定します。

```
<view>
  <module name="IFrameInclude">
    <param name="src">myawesomewebsite.com</param>
    <param name="height">42</param>
  </module>
```

```
</view>
```

4. オプションで、URI が存在するかどうかをまずチェックできます。

```
<view>
  <module name="IFrameInclude">
    <param name="src">myawesomewebsite.com</param>
    <param name="check_exists">True</param>
  </module>
</view>
```

App のスタイルの変更

App のスタイルの変更

CSS を更新するだけで Splunk Web の表示をさまざまに変更できます。各アプリケーションは、アプリケーション全体の CSS を定義する `application.css` ファイルを 1 つ含むことができます。`application.css` ファイルは、Splunk の他の CSS ファイルよりも高い優先度を持っています。したがって、`application.css` に定義された設定はデフォルト設定を上書きします。つまり、各モジュールが個々の CSS ファイルを有していても、`application.css` で一斉に複数のモジュールを上書きすることができます。

CSS レファレンス

注意： ページに表示されるオブジェクトを変更する場合、例えば、ドロップダウンオプションを変更したり、時間軸を追加したりする場合などは、モジュールをビューに追加します。

設定

`application.css` を `$(SPLUNK_HOME)/etc/apps/<appname>/appserver/static/` に作成します。この 1 つのファイルを使用して、Splunk Web のファイルのすべてのスタイル設定を上書きします。例えば、デフォルトフォントを上書きしたり、モジュールのバックグラウンド色を変更することができます。ロゴやレイアウト、その他 Splunk のさまざまな CSS ファイルで設定されたスタイルを変更します。

画像を追加するには、画像を `application.css` ファイルとともに、`../appserver/static/` ディレクトリに置きます。

ベストプラクティス

Firefox の Firebug を使用することを推奨します。Firebug により、ページのエレメントを検査し、変更したいスタイルのクラスを指摘することが可能になります。また、`application.css` に変更を加える前に、さまざまな CSS 設定を試すことができます。

例

このサンプル App は、内蔵された CSS を次の `application.css` で上書きします。

```
/*
 * Top app banner section
 */
```

```

.AccountBar {
  background-image: url(/static/app/samples/samples_header.png);
  background-repeat: no-repeat;
  background-color: #79a60b;
  height: 140px;
}
.AccountBar .appLogo,
.AccountBar p.appName {
  display: none;
}
.AccountBar .accountBarItems {
  background-color: #000;
  opacity: .5;
  filter: alpha(opacity=50);
}
/*
 * view menu system
 */
.AppBar {
  font-family: Arial Black, Arial, sans-serif;
  font-size: 18px;
  font-variant: small-caps;
}
.AppBar a {
  color: #f3df00 !important;
}
.AppBar a:hover {
  color: #6ad7ff !important;
}
/*
 * Body content
 */
body {
  background-image: url(/static/app/samples/body_bg.png);
  background-repeat: repeat-x repeat-y;
}

```

モジュールのしくみ

モジュールのしくみ

Splunk Web のページに表示される要素は、検索バーから検索結果に至るまで、すべてモジュールです。表示されないエレメント、例えば、結果モジュールを投入するためにバックグラウンドで実行されている検索なども同様にモジュールです。

表示したいモジュールおよびそれにリンクしたいモジュールを選択してビューを設定します。ビューの設定に使用できるモジュールはたくさんあります。このページを利用して、モジュールをページに適用する方法について一般的感覚を得てください。これは、モジュールを上手に活用するために、モジュールのカテゴリおよび設定を一覧したものです。全モジュールのリストは、「モジュールレファレンス」を参照する、またはブラウザで次のサイトにアクセスしてください。

<http://<splunkserver>:<splunkwebport>/modules>

検索ビューまたはダッシュボード

ビューを一連のモジュールで構築する場合、検索ビューまたはダッシュボードを作成することができます。検索ビューは、一連の検索モジュールで作成されます(下記参照)。実行した検索は、指定された1つまたは複数のモジュールを通して結果が渡され、1つまたは複数の結果モジュールにより表示されます(下記参照)。ダッシュボードは、様々な検索からの結果を、ほとんどの場合結果モジュールを通じて表示します。必要に応じて、検索ビューおよびダッシュボードビューに他のモジュールを追加することもできます。

モジュールの設定

ビューのモジュールは、ツリー構造です。したがって、検索ビューを構築する場合、検索情報はモジュールからその子モジュールに渡されます。子モジュールは、検索がイベントを返す、または結果に変換されるまでの検索を何らかの方法で修正します。ダッシュボードビューを構築する場合、各パネルは、別の検索から構築します。ダッシュボードビューは、検索ビューに比べ、短い数の多いツリー構造になっています。

params

モジュールの中には **params** と呼ばれる設定があります。例えば：

```
<module name="Message">
  <param name="filter">*</param>
```

Params は、グラフおよびチャートのサイズ、1 ページあたりに表示するイベント数など、モジュール固有の設定を制御します。Params はモジュールレファレンスページまたはモジュールの conf ファイルに記載されています。

params には、必須とオプションの項目があります。params の中には、デフォルト設定もあります。これらはレファレンスページに記載されています。param が必須の場合、そのモジュールに param を含める必要があります。param がオプションの場合、モジュールに含めることもできますが、必ずしも必要ではありません。デフォルトが記載されている場合、それがそのパラメータのデフォルト値であり、設定を行わないかぎりそれが使用されます。Params は、`$SPLUNK_HOME/share/splunk/search_mrsparkle/modules` のモジュールのディレクトリにある各モジュールの conf ファイルに記載されます。

layoutPanel

モジュールがページのどこに表示されるかを指定するには、モジュールタグで `layoutPanel` 属性を設定します。例えば：

```
<module name="SearchBar" layoutPanel="mainSearchControls">
```

注意：

- 検索ビューおよびダッシュボードは、異なる layoutPanels を使用します。検索ビューは、「検索ビューレイアウト」に記載の layoutPanels を使用します。一方、ダッシュボードは、「ダッシュボードの設定」に記載の座標系を使用します。

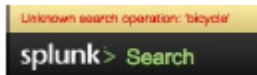
- 階層の上位レベルにあるモジュールには、必ず `layoutPanel` を指定します。`layoutPanel` 設定を持たないツリー内のその他のモジュールはすべて、それぞれの親から引き継ぎます。ページを異なるレイアウトパネルに移動すると、そのパネルの最初のモジュールのレイアウトパネルを設定する必要があります。

検索ビューで利用できる `layoutPanels` には以下があります。

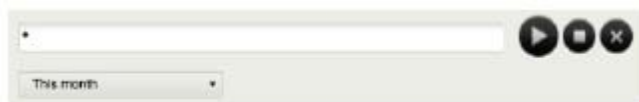
`navigationHeader`



`messaging`



`mainSearchControls`



`graphArea`



`sideBar`



`pageControls` | `pageControls2`



`resultsAreaLeft` | `resultsAreaRight`

Results 1 - 10 for search * (12.000 seconds)

```
01/26/2009:23:44:34.000 Jan 26 23:44:14: --- last message repeated 3 times ---  
sourcetype=syslog * source=/var/log/system.log.0.bz2 *  
  
01/26/2009:23:44:34.000 Jan 26 23:44:14: --- last message repeated 3 times ---  
sourcetype=syslog * source=/var/log/system.log *
```

resultsHeaderPanel

fullWidthControls

モジュール概要

`$SPLUNK_HOME/share/splunk/search_mrsparkle/modules`

ディレクトリのサブディレクトリにグループ化された際のモジュールの概要を以下に記載します。モジュールは、その機能に基づいてグループ化されます。ここでは、指定されたモジュールにどの layoutPanels を置くかに関するヒントを記載しています。利用可能なモジュールとその params に関する詳細は、「モジュールレファレンス」ページを参照してください。

navigationHeader

navigationHeader にモジュールを置くと、トップナビバーに表示されます。

- AccountBar
- AppBar
- TitleBar
- BreadCrumb

messaging

messaging layoutPanel は、メッセージモジュールでのみ使用されます。

static

genericHeader はこのディレクトリにある唯一のモジュールです。一般的ヘッダーを layoutPanel のヘッダーとして使用します。通常、与えられたレイアウトパネルの親モジュールとなります。ページのその部分のヘッダーとして表示されます。

search

Search モジュールは、検索ビューの作成に使用されます。それには以下があります。

- ExtendedFieldSearch
- FieldPicker
- FieldSearch
- HiddenFieldPicker
- HiddenIntention
- HiddenSavedSearch

- HiddenSearch
- PostProcessBar !!
- RadioButtonSearch
- SearchBar
- SubmitButton !!
- TimeRangePicker
- ViewRedirector

Results

Results モジュールは、ダッシュボードの作成、または検索ビューで結果の表示に使用されます。検索ビューでは、これらのモジュールを resultsArea で表示することができます。

以下があります。

- chart controls (listed below)
- flash (listed below)
- page controls (listed below)
- EventsViewer
- FieldViewer
- LinkList
- MultiFieldViewer
- ResultsHeader
- SavedSearches
- SimpleResultsTable
- SingleValue

chart controls

チャートおよびグラフをフォーマットを設定するには chart controls を使用します。

- AxisScaleFormatter
- BaseChartFormatter
- ChartTitleFormatter
- ChartTypeFormatter
- HiddenChartFormatter
- LegendFormatter
- LineMarkerFormatter
- NullValueFormatter
- StackModeFormatter
- XAxisTitleFormatter
- YAxisRangeMaximumFormatter

- YAxisTitleFormatter

flash modules

- FlashChart
- FlashTimeline
- FlashWrapper

page controls

Page は、イベント表示を制御するモジュールを制御します。

- Count
- MaxLines
- Segmentation
- Selector
- SoftWrap
- TextSetting

switchers



Switcher モジュールは特別なモジュールで、複数の派生モジュールセットを持つことができ、それらを切り替えます。

- ButtonSwitcher
- LinkSwitcher
- PulldownSwitcher
- TabSwitcher

Switcher モジュールは、2 つの異なるモードを操作作します(将来は 3 つ)。

- **independent:** ツリーは期待とおりに機能することを意味します。サブツリーA とサブツリーB には相互関係はありません。
- **serializeAll:** 子はすべて基本的に 1 つの長いチェーンにスレッドし直されます。表示側から見ると、それらはグループとして表示/非表示されます。

ただし、検索情報およびフォーマット情報、およびユーザー入力値を渡す限り、それらはすべて、用途、時間範囲、検索変更、フォーマットなどを渡す深いツリーにあるように機能します。それらをどのように使用するかを考えるとき唯一意味をなすのは、switcher の最後の子ツリーが serializeAll モードの場合、絶対に非表示になることがなく、つまり、レポートビルダで、flashchart がどのようにして「各タブ内」に存在するか(実際には最後のタブである)を説明しています。

完全性を確保するために、'serializeActive'がまもなくリリースされます。つまり、N の子がある場合、N 番目の子は常にそのどこかにあることを基本的に意味し、switcher は、1 番目の子から N-1 番目の子のうちのどれを切り替えるかを

選択します。

include

include ディレクトリにあるモジュールを使用して、ウェブリソースをビューに追加します。通常、これらのモジュールは layoutPanel : *fullWidthControls* に表示されます。

- AjaxInclude
- IFrameInclude
- ServerSideInclude

gandalf

gandalf にあるモジュールは使用しないでください。それらはレポートビルダーをサポートします。



モジュールレファレンス

モジュールレファレンス

モジュールの使用法に関する一般的な説明は、「モジュールの概要」を参照してください。

以下に、ビューの構築に必要なすべてのモジュールについて記載します。モジュールは、機能カテゴリー別に整理されています。

Splunk インスタンスの全モジュールの動的記述をロードするには、次の URI をクリックします。

<http://localhost:8000/en-US/modules>

注意： ホストおよびポートを、使用しているインストール仕様で置き換えてください。

Nav

ページレイアウトおよび一般的ページプロパティに関する基本的モジュールは 4 つあります。

AccountBar

ほとんどのビューの最上部にあるバーで、<ユーザー>としてログイン、というロゴと、ログアウトおよび管理者リンクがあります。

AppBar

これは、ほとんどのビューで、上から 2 番目にあるバーです。上位レベルビューカテゴリー(デフォルトでは、**ダッシュ**

ュボードビュー保存済み検索)、および補助リンクセクション(ヘルプ| preferences | about)があります。

TitleBar

コントロールメニュー/アクションメニュー。

このモジュールは常駐し、ダッシュボードの名前、ビューの名前、またはビューおよび関連する保存済み検索の名前を含みます。titlebar は文脈依存アクションの場所として機能します。例えば、ビューをロードした後実行された新規検索の保存など。

パラメータ

必須

なし。

オプション

- showActionsMenu = True | False
 - ◆ アクションメニューを表示するかどうかを指定。
 - ◆ デフォルトは true。

メッセージング

このモジュールは、ユーザーにすべてのメッセージをユーザーに表示します。または、特定のクラスのメッセージのみを表示するように設定することができます。メッセージには、検索、アラート、エラー、インデキシングステータスに関する情報などがあります。各ビューは、バックエンドからメッセージを受け取るために、少なくとも1つのメッセージモジュールを持つ必要があります。

Message

このモジュールは、ユーザーにすべてのメッセージをユーザーに表示します。または、特定のクラスのメッセージのみを表示するように設定することができます。メッセージには、検索、アラート発信、バックエンドの設定間違い、インデキシングステータスに関する情報などがあります。

最も簡単な設定は、フィルタを'*'に設定した1つの Message インスタンスです。これは、メッセージブロードキャストをすべて表示することを意味します。ただし、ビューにある別々のレイアウトパネルに表示される、異なる'filter' params を持つ複数の Message モジュールを使用することができます。

メッセージは、*splunk.search.error* などの定義クラスとともに渡されます。つまり、2つの Message インスタンスがあり、一方のフィルタを'*'に設定し、他方のフィルタを *splunk.search* に設定すると、後者は *splunk.search.error* イベントを受け取り、'*'*のインスタンスは受け取りません。ただし、予期せぬメッセージが、*splunk.indexing.warn* のクラスで渡された場合、*splunk.search* インスタンスはそれを表示しませんが、'*'は表示します。

パラメータ

必須

- filter = 文字列
 - ◆ 必須
 - ◆ 特定のメッセージクラスのみをリッスンするようにフィルタを指定。
 - ◆ 送信されたメッセージをすべてリッスンするには、空白を指定。
- clearOnJobDispatch = True | False
 - ◆ 必須
 - ◆ True に設定すると、新しい検索がディスパッチされたときに、メッセージがクリアされます
- maxSize = 整数
 - ◆ 必須
 - ◆ 最も古いメッセージを捨てる前に表示するメッセージの数。

オプション

- default = 文字列
 - ◆ 任意
 - ◆ 表示するデフォルトメッセージを指定。

例

```
<module name="Message" layoutPanel="messaging">  
  <param name="filter">*</param>  
  <param name="clearOnJobDispatch">False</param>  
  <param name="maxSize">1</param>  
</module>
```

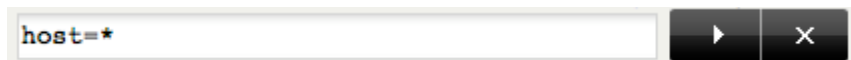
検索

検索アクション用モジュール。

派生モジュールは、親から渡された情報を取得します。したがって、各派生検索モジュールは自らのオプションを元の検索に追加します。検索を絞り込むには、フィールドや時間範囲などのフィルタを追加する子モジュールを含めます。

SearchBar

SearchBar モジュールは、キャンセルボタンと実行ボタンを持つ検索バーを作成します。



パラメータ

必須

なし。

オプション

- `q = 検索文字列`
 - ◆ 検索ボックスに投入するデフォルト検索を指定。
- `useAssistant = true | false`
 - ◆ 検索アシスタントを有効するかどうかを設定。
- `useTypeahead = true | false`
 - ◆ 先行入力をインデックストークンで有効にするかどうかを設定。

例

```
<module name="SearchBar" layoutPanel="main_search_controls">  
  <param name="q">host=*</param>  
  <param name="autoRun">True</param>
```

この例は、検索バーに値を投入して、検索を自動ディスパッチします。

host=*

FieldSearch

特定のフィールドで検索を制限します。このモジュールを使用して、1つフォームフィールドしか持たないフォーム検索を設定します。複数のフォームを持つフォーム検索を設定するには、ExtendedFieldSearch を使用します(後述)。



パラメータ

必須

- `field = フィールド`
 - ◆ 結果の範囲を指定するためのフィールド(`sourcetype`、`clientip` その他の有効なフィールドなど)を指定。

オプション

- `q = 検索文字列`
 - ◆ ページのロード時に表示するデフォルト検索を指定。
- `label = 文字列`
 - ◆ 入力フィールドの横に表示する代替テキストを設定。
 - ◆ 省略した場合、デフォルトは `field param`。

例

```
<module name="FieldSearch">  
  <param name="field">sourcetype</param>
```

ExtendedFieldSearch

特定のフィールドで検索を制限します。このモジュールを使用して、複数のフォームフィールドを持つフォーム検索を設定します。1つのフィールドしか持たないフォーム検索を設定するには、FieldSearchを使用します(前述)。

パラメータ

必須

- field = 文字列
 - ◆ 結果の範囲を指定するためのフィールド(sourcetype、clientip その他の有効なフィールドなど)を指定。

オプション

- intention = 用途
 - ◆ 検索コンテキストオブジェクトに追加する目的。
- replacementMap = 文字列
 - ◆ ユーザーが追加した値で置き換える必要がある用途内の値への最短パスのマップ。
- default = 文字列
 - ◆ テキストフィールドに現れるデフォルト値。
- label = 文字列
 - ◆ 入力フィールドの横に表示する代替テキストを設定。省略した場合、デフォルトは field param。

例

```
<module name="HiddenSearch" layoutPanel="mainSearchControls">
  <param name="search">sourcetype=$st$</param>
  <module name="ExtendedFieldSearch">
    <param name="intention">
      <param name="name">stringreplace</param>
      <param name="arg">
        <param name="st">
          <param name="default">apache_error</param>
        </param>
      </param>
    </param>
  </param>
  <param name="replacementMap">
    <param name="arg">
      <param name="st">
        <param name="value"></param>
      </param>
    </param>
  </param>
  <param name="field">Sourcetype</param>
</module>
</module>
```

ViewRedirector

このモジュールは、コンテキストおよび設定情報で、その祖先から提供されたものを取得し、その検索をディスパッチし、ユーザーをリダイレクトして、指定されたビューでその検索を表示するようにします。ViewRedirector が新しいコンテキストを受け取り、onContextChange()が呼び出された場合、指定されたビューにリダイレクトします。

パラメータ

必須

- viewTarget = ビュー
 - ◆ モジュールがユーザーの検索をリダイレクトするようにビューを設定。

オプション

- mode = automatic | link | button
 - ◆ automatic モードでは、何も表示されず、コンテキスト変更はリダイレクトをトリガします。link および button モードでは、ユーザーは、リダイレクトを開始するためには、モジュールをクリックする必要があります。
 - ◆ デフォルトは automatic。
- label = ラベル
 - ◆ link および ボタンモードでは、リンクまたはボタンのラベルになります。
 - ◆ デフォルトは Go(実行)。
- popup = true | false
 - ◆ ポップアップウィンドウでビューを開始する、または同じウィンドウを使うかを設定。
 - ◆ window preferences に設定するには、true および false 以外の値を設定します。
 - ◆ デフォルトは false(同じウィンドウ)。
- linkType = sid | permalink
 - ◆ リンクを検索 ID に向ける、またはパーマリンクとするかを指定。
 - ◆ デフォルトは sid(ID)。
- sendBaseSID = true | false
 - ◆ ベース検索 ID を送るかどうかの切り替え。
 - ◆ デフォルトは false(送らない)。
- dispatchBeforeRedirect = true | false
 - ◆ リダイレクトの前に検索をディスパッチするかどうかの設定。
 - ◆ デフォルトは false(しない)。

例

```
<module name="ViewRedirector">  
  <param name="viewTarget">flashtimeline</param>
```

</module>

ViewRedirectorLink

このモジュールは、与えられたラベルを持つビューにリンクを付けます。これをクリックすると、その祖先が提供するコンテキスト情報を取得し、その検索をディスパッチし、ユーザーをリダイレクトして、指定されたビューでその検索を表示するようにします。

パラメータ

ViewRedirectorLink は ViewRedirector(上述)の params をすべて継承します。

必須

なし。

オプション

- label = *リンクラベル*
 - ◆ これは、link およびボタンモードで、リンクまたはボタンのラベルになります。
 - ◆ 省略した場合、そのリンクまたはボタンラベルは「ビュー結果」となります。

HiddenSearch

バックグラウンドで検索を実施します。結果を子に渡します。

注意：

- HiddenSearch は、LinkList および TabSwitcher など、多くの下層モジュールで必要です。検索結果の表示に依存するモジュールはほとんどが HiddenSearch を親として持っています。
- 検索を実際に実行するには、autoRun = true を設定する必要があります。

パラメータ

必須

- search = *検索文字列*
 - ◆ リテラルな検索文字列である HiddenSearch はその子に渡します。
 - ◆ **注意：** HiddenSearch を構成する際に、時間モディファイアは最古と最新に設定し、検索自体には設定しないようにします。

オプション

- earliest = *時間文字列*
 - ◆ 時刻範囲の開始の定義に使用。検索の時間範囲の始点を設定。
 - ◆ 'latest'が定義される場合、これを定義する必要がある。

- latest = 時間文字列
 - ◆ 時刻範囲の終了の定義に使用。検索の時間範囲の終点を設定。
 - ◆ 'earliest'を定義する場合、これも定義する必要がある。

注意：「有効な時間文字列の設定」を参照してください。

例

```
<module name="HiddenSearch">
  <param name="searchClass">search</param>
  <param name="searchObject">| metadata type=sources | sort -totalCount</param>
```

HiddenSavedSearch

savedsearches.conf で設定された保存済み検索から結果をロードします。ロード時に保存済み検索を実行する、または最後に実行した保存済み検索の結果を表示します。

パラメータ

必須

- savedSearch = 保存済み検索名
 - ◆ 保存済み検索の履歴で検索を調べるとき、または新規検索をディスパッチする際に使用する保存済み検索の名前。

オプション

- useHistory = None | True | False
 - ◆ デフォルトは None。
 - ◆ ディスパッチする保存済み検索のモジュールの取り扱いを定義します。デフォルトは None。つまり、モジュールは、保存済み検索の履歴から利用可能な以前の実行ジョブを使用します。それ以外は、新規ジョブをディスパッチし、新規ジョブの sid を返すことを意味します。useHistory が True の場合、savedSearch は、保存済み検索がディスパッチした以前の実行ジョブのみを探します。false に設定した場合、モジュールは、ジョブが保存済み検索の履歴にあるかどうかに関わらず、定義された保存済み検索に基づいて、新規ジョブをディスパッチします。

例

この例は、「All local sources」という名前の保存済み検索を使用します。

```
<module name="HiddenSavedSearch" layoutPanel="mainSearchControls" autoRun="true">
  <param name="savedSearch">All local sources</param>
```

HiddenIntention

受信する検索の用途を追加します。

パラメータ

必須

- intention = 文字列
 - ◆ 基本検索のトップにある用途を階層化します。

オプション

なし。

例

HiddenFieldPicker

このモジュールは、ユーザーに表示されるフィールドおよびそのフィールドがある順序を組み込む見えないコントロールを実装します。それらがこのモジュールの派生である場合、イベントとサマリー情報を表示する他のモジュールは、ここで指定されたフィールドリストを取得します。

パラメータ

必須

- fields = フィールドのリスト
 - ◆ ページのロード時にデフォルトでイベントとともに表示されるフィールドのリスト (スペース区切り)

オプション

なし。

例

```
<module name="HiddenFieldPicker">  
  <param name="fields">clientip host uri</param>
```

RadioButtonSearch

このモジュールは、送信およびキャンセルボタンを持つ一連のラジオボタンを生成します。

注意： HiddenSearch が親(または祖先)として必要です。

パラメータ

必須

- label = 文字列
 - ◆ ラジオボタンの横に表示するラベルを追加。
- options = リスト

- ◆ 2つの項目：**text**と**value**のリスト。
- ◆ text=ラジオボタンの横に表示するテキスト。
- ◆ value=ラジオボタンをクリックしたときに選択される値。
- ◆ オプションで、**selected**を追加して、ページロード時にチェックされるボタンを指定できます。

オプション

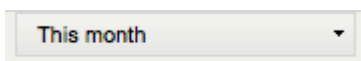
- field = ラジオボタンの値
 - ◆ ある場合、モジュールは、キー値 searchTerms を<field>=<radioButtonValue>の形式で追加します。ない場合、モジュールは単に<radioButtonValue>を検索に追加します。

例

```
<module name="RadioButtonSearchField">
  <param name="options">
    <list>
      <param name="selected">true</param>
      <param name="value">save</param>
      <param name="text">Saves</param>
    </list>
    <list>
      <param name="value">edit</param>
      <param name="text">Edits</param>
    </list>
    <list>
      <param name="value">upload</param>
      <param name="text">Uploads</param>
    </list>
    <list>
      <param name="value">view</param>
      <param name="text">Views</param>
    </list>
    <list>
      <param name="value"></param>
      <param name="text">All</param>
    </list>
  </param>
</module>
```

TimeRangePicker

このモジュールは、ユーザーが時間範囲の変更に使用できるドロップダウンメニューを生成します。時間範囲の値およびラベルは、times.conf の設定から取り込まれます。



パラメータ

必須

なし。

オプション

- selected = *時間範囲ラベル*
 - ◆ times.conf の時間範囲ラベルの 1 つに設定した場合、ページのロード時にその時間範囲オプションが設定されます。
 - ◆ 定着するが持続しない。

オプション

- searchWhenChanged = True | False
 - ◆ 時間範囲が変更された時に検索を起動。
- label = *文字列*

required = False label = 時間範囲ピッカーの上に表示するオプションラベル。

例

```
<module name="TimeRangePicker" layoutPanel="main_search_controls">  
  <param name="selected">This month</param>
```

FieldPicker

このモジュールは、ユーザーが表示するフィールドを選択できるすべての利用可能なフィールドの一覧を表示するフィールドピッカーを起動します。イベントおよびサマリー情報を表示するこのモジュールの派生は、ここで指定または選択されたフィールドリストを取得します。

パラメータ

必須

- fields = *フィールドのリスト*
 - ◆ ページロード時に、デフォルトで選択されるフィールドのリスト (スペース区切り)。
 - ◆ 定着性があり持続する。

オプション

なし。

例

```
<module name="FieldPicker" layoutPanel="side_bar">  
  <param name="fields">host sourcetype source</param>
```

Switchers

Switcher モジュールは、結果表示および子モジュールの別グループなどのオプションを、ビューを切り替えずに切り替えます。子モジュールは 1 つのグループのみが一度に表示されます。Switcher モジュールとは、ボタン、プルダウンメニュー、リンク、タブです。

Switcher モジュールはすべてこれらの params を共有します。

必須

- mode = independent | serializeAll
 - ◆ Mode は現在のところ、'independent'または'serializeAll'のいずれかの値をとります。 ** independent が最も一般的な設定です。independent モードでは、モジュールの N 個のサブツリーはすべて区別され、個別の子ブランチです。
 - ◆ serializeAll は、1つの共有検索またはレポートの異なる特性を切り替えます。このモードでは、a) 最後の子モジュールは、スイッチャのオプションで表示されない、または b) 最後の子ツリーは常に表示されます。

オプション

- selected = グループ名
 - ◆ ページロード時に選択される子のグループを指定。(グループ名は、親モジュールの *group = name* 属性設定)
 - ◆ 存在しない場合、最初の子モジュールとその先祖ツリーが初めに表示されます。

PulldownSwitcher

子の結果が投入されたプルダウンメニューを生成します。一度に 1 セットの子モジュールを表示します。子は、結果を継承する(シリアライズ)、または独立させることができます。

パラメータ

デフォルトの switcher パラメータは上述のとおり。

必須

- label = 文字列
 - ◆ プルダウンの左に表示するテキストラベルを指定。

例

```
<module name="PulldownSwitcher" layoutPanel="main_search_controls">
  <param name="mode">independent</param>
  <param name="selected">Messages per minute last hour</param>
  <param name="label">Show: </param>
```

TabSwitcher

PulldownSwitcher と同様に、このモジュールは一度に 1 つの子を表示します。タブセットで子モジュールの結果を表示します。異なるタブをユーザーがクリックした場合、それに対応する子とその派生モジュールが画面に表示され、他の子モジュール(およびその派生)はすべて隠されます。

パラメータ

デフォルトの switcher パラメータは上述のとおり。

例

```
<module name="TabSwitcher" layoutPanel="main_search_controls">  
<param name="mode">independent</param>
```

ButtonSwitcher

アイコンのクリックに基づいて表示を切り替えます。主に、イベントビューまたはテーブルビューの結果表示を切り替えます。ボタンスタイルは、params のクラス設定で指定します。

異なるボタンをユーザーがクリックした場合、それに対応する子とその派生モジュールが画面に表示され、他の子モジュール(およびその派生)はすべて隠されます。

パラメータ

デフォルトの switcher パラメータは上述のとおり。

LinkSwitcher

異なるリンクをユーザーがクリックした場合、それに対応する子とその派生モジュールが画面に表示され、他の子モジュール(およびその派生)はすべて隠されます。

パラメータ

デフォルトの switcher パラメータは上述のとおり。

必須

- label = 文字列
 - ◆ これは、リンクの左に表示するテキストラベルを指定します。

例

```
<module name="LinkSwitcher" layoutPanel="panel_row4_col2">  
  <param name="mode">serializeAll</param>  
  <param name="label">Format options:</param>
```

結果

検索を結果として渡すには、前の検索モジュールの子として結果モジュールをフォーマットします。

GenericHeader

ヘッダーを表示します。頻繁に Switcher モジュール(TabSwitcher、PulldownSwitcher など)と併用され、子モジュールのラベルを表示します。

パラメータ

必須

- switcherTitle = 文字列
 - ◆ 表示するタイトル。
 - ◆ このモジュールが Switcher モジュール(PulldownSwitcher または TabSwitcher)の最初の子である場合にのみ必要。
- label = 文字列
 - ◆ プルダウンの左に表示するテキストラベルを指定。

オプション

なし。

例

```
<module name="GenericHeader">  
  <param name="switcherTitle">General Options</param>  
  <param name="label">General Options</param>
```

ResultsHeader

ResultsHeader モジュールは結果ラベルを生成します。このモジュールを使用して他のモジュール(ResultsHeader の子であること)のヘッダー生成します。このモジュールは、例えば、**23,420 events**、などのヘッダーを表示します。通常、FlashTimeline の上、またはページングコントロールを実装するモジュールセットの上に置きます。

Timeline 0 Events during this year

パラメータ

必須

- entityLabel = 文字列
 - ◆ ユニットにラベルを付けます。
 - ◆ 通常、events または results に設定することができます。
 - ◆ 他のものにも設定することもできますが、各結果に対し意味のラベルにしてください。
- entityName = events | results | scanned
 - ◆ 表示する数値の取得場所を指定。

オプション

- prefix = 文字列
 - ◆ ある場合、このキーの値は、表示される数値の前に表示されます。
 - ◆ 例えば、'Found'に設定すると、全体の表示は'Found 23,420 events'になります。

- link = ビューへのリンク
 - ◆ ある場合、ビューへのリンクを表示します。
 - ◆ クリックすると、そのビューを、'| top <field>'検索に渡します。これは、対応するチャートを生成します。
 - ◆ 値はキー：'viewTarget'、'label'の辞書です。

これは、モジュールが表示するリンクの動作を指定する設定値の辞書です。この検索結果が代わりに表示される別のビューにユーザーを導きます。リンクのテキストである'label'キー、およびユーザーが導かれる先のビューである'viewTarget'キーを含みます。また、'popup'キーは、True のとき、リンクが新しいポップアップウィンドウを開くようにします。

例

```
<module name="ResultsHeader" layoutPanel="results_area_1">
  <param name="entityName">events</param>
  <param name="prefix">Timeline</param>
  <param name="entityLabel">Events</param>
<module name="ResultsHeader" layoutPanel="results_area_1">
  <param name="entityLabel">Events</param>
  <param name="link">
    <param name="popup">True</param>
    <param name="viewTarget">report_wizard</param>
    <param name="label">Report on results</param>
  </param>
  <param name="prefix">Timeline</param>
  <param name="entityName">events</param>
```

EventsViewer

EventsViewer モジュールは、祖先モジュールと統合して指定する検索から生じるイベントを表示します。このモジュールは SimpleEventsViewer と類似しているため、この 2 つのモジュールは 1 つに統合される予定です。

パラメータ

必須

なし。

オプション

- negateModifier = shiftKey | altKey | ctrlKey | metaKey
 - ◆ キーボードのキーを押下するよう設定し、クリックすると、検索から語句を削除します。
 - ◆ デフォルトは altKey。
- scrollerEnable = True | False
 - ◆ スクロローラーを有効にします。
- scrollerMaxHeight = 整数
 - ◆ scrollerEnable が True の場合、スクローラー最大ピクセル高さ制限を制御します。

- ◆ デフォルトは 10000。
- scrollerMinHeight = 整数
 - ◆ scrollerEnable が True の場合、スクローラー最小ピクセル高さ制限を制御します。
 - ◆ デフォルトは 0。
- displayRowNumbers = true | false
 - ◆ これは行番号を表示/非表示を決定します
 - ◆ デフォルトは true。

例

```
<module name="EventsViewer" layoutPanel="main_search_controls"/>
```

SingleValue

このモジュールは、検索の完了を待ち、その後、結果の最初の行から 1 つのフィールドを表示(レンダリング)します。

パラメータ

必須

なし。

オプション

- label = ラベル
 - ◆ 結果カウントとともに表示するラベル。
- labelPosition = before | after
 - ◆ 表示されたフィールドの前または後にラベルを配置します。
 - ◆ デフォルトは after(後)。
- beforeLabel = ラベル
 - ◆ 結果の前に表示するラベル
- afterLabel = ラベル
 - ◆ 結果の後に表示するラベル
- field = フィールド
 - ◆ 表示するフィールド。
 - ◆ デフォルトは最初に返されるフィールド
- additionalClass = CSS クラス
 - ◆ オプションで、CSS クラスを追加してボタンのスタイルを設定します。
 - ◆ application.css でクラスをスタイル設定します。
- classField = フィールド
 - ◆ 最初の結果の classField の値を、結果コンテナに追加する css クラスとして追加します。

- ◆ あらかじめ定義されたクラスは、'severe'、'elevated'、'low'および'None'(デフォルト)。
- linkSearch = 検索語
 - ◆ 結果をクリック可能リンクに変えるための有効な完全検索を指定します。
- linkView = ビュー
 - ◆ linkSearch を表示するビューを指定。
 - ◆ デフォルトは dashboard(ダッシュボード)。
- linkFields = フィールド
 - ◆ 結果をリンクする、またはラベルも含めるかを指定します。
 - ◆ 結果と両方のラベルをリンクするには、"result,beforeLabel,afterLabel"を指定します。
 - ◆ デフォルトは results(結果のみ)。
- format = number | decimal | percent | none
 - ◆ 値に提供するデータフォーマット方法を指定します。
 - ◆ ロケールに依存します。
 - ◆ デフォルトは none(なし)。

SimpleFieldViewer

このモジュールは、与えられたフィールドの上から N 個の値を表示します。同時に、その与えられた値を持つイベントの数も表示します。

パラメータ

必須

- field = フィールド
 - ◆ 表示する上位値のフィールドを指定。
- count = 整数
 - ◆ モジュールが表示する最も一般的な値の数です。

オプション

- link = ビューへのリンク
 - ◆ ある場合、ビューへのリンクを表示します。
 - ◆ クリックすると、そのビューを、'| top <field>'検索に渡します。これは、対応するチャートを生成します。
 - ◆ 値はキー：'view'、'label'の辞書です。
 - ◆ このキー辞書は、モジュールが表示するリンクの動作を指定します。この検索結果が代わりに表示される別のビューにユーザーを導きます。リンクのテキストである'label'キー、およびユーザーが導かれる先のビューである'viewTarget'キーを含んでいます。また、'popup'キーは、True のとき、リンクが新しくポップアップウィンドウを開くようにします。

例

```
<module name="SimpleFieldViewer" layoutPanel="side_bar">
  <param name="count">5</param>
  <param name="field">twikiuser</param>
</module>
```

MultiFieldViewer

複数フィールドの上位 x 個の値。

一連のフィールド名を表示し、同時にその横に括弧で個別カウントを表示します。ユーザーがフィールド名をクリックすると、ポップアップレイヤが開き、そのフィールドの上位 10 個の値を表示します。そのうちの 1 つをクリックすると、該当する field=value 語句を追加し、検索を再実行します。

パラメータ

必須

なし。

オプション

- label = 文字列
 - ◆ フィールドビューアの横に表示するラベルをオプションで追加します。
- field = フィールド
 - ◆ フィールドが親クラスのパラメータでしたが、このクラスでは不要です。
- count = 整数
 - ◆ これは、モジュールが表示する最も一般的な値の数です。
- link = ビューへのリンク
 - ◆ ある場合、ビューへのリンクを表示します。
 - ◆ クリックすると、そのビューを、'| top <field>'検索に渡します。これは、対応するチャートを生成します。
 - ◆ 値はキー: 'view'、'label'の辞書です。

○モジュールが表示するリンクの動作を指定する設定値の辞書です。この検索結果が代わりに表示される別のビューにユーザーを導きます。リンクのテキストである'label'キー、およびユーザーが導かれる先のビューである'view'キーを含みます。また、'popup'キーは、True のとき、リンクが新しくポップアップウィンドウを開くようにします。

例

```
<module name="MultiFieldViewer">
  <param name="count">10</param>
  <param name="link">
    <param name="view">report_wizard</param>
    <param name="label">Report on this field</param>
  </param>
```

LinkList

リンクのリストを表示(レンダリング)します。リンクリストは通常、主フィールド(`labelField`)と、第2の説明フィールド(`valueField`)の組み合わせます。オプションで、表示 `params` を指定します。

注意： `HiddenSearch` が親に必要です。

Hosts

Sort [alphabetically](#) | [count \(desc\)](#)

- [twiki](#) (1504838)
- [localhost](#) (296808)
- [spacecake](#) (12211)

パラメータ

必須

- `labelField` = フィールド
 - ◆ リンクリストで表示するフィールド名を設定。フィールドの値を表示。
- `valueField` = フィールド
 - ◆ リストで表示するフィールド値を設定。これはほとんどの場合、リンクリストの"label"部分に関連するカウントまたは数値です。

オプション

- `initialSortDir` = desc | asc
 - ◆ `initialSort` フィールドに基づいて結果を並べ替える向き。
 - ◆ デフォルトは asc。
- `initialSort` = 一つのフィールド
 - ◆ 結果のフィールドはリンクリストが最初に表示される時にソートされます。
- `labelFieldSearch` = 検索
 - ◆ ユーザーがラベルフィールドをクリックした際に生成される検索文字列。 `labelFieldTarget` を有効なビューに指定する必要があります。ラベルフィールドの値は自動的にその検索に追加されます。
- `labelFieldTarget` = ビュー
 - ◆ ラベルフィールドが、検索をディスパッチするクリック可能リンクを生成するようにセットアップされている場合に、目標とするビュー。

例

```
<module name="LinkList">
  <param name="initialSortDir">desc</param>
  <param name="labelFieldSearch">*</param>
  <param name="valueField">totalCount</param>
  <param name="labelField">source</param>
  <param name="labelFieldTarget">searchview</param>
  <param name="initialSort">totalCount</param>
```

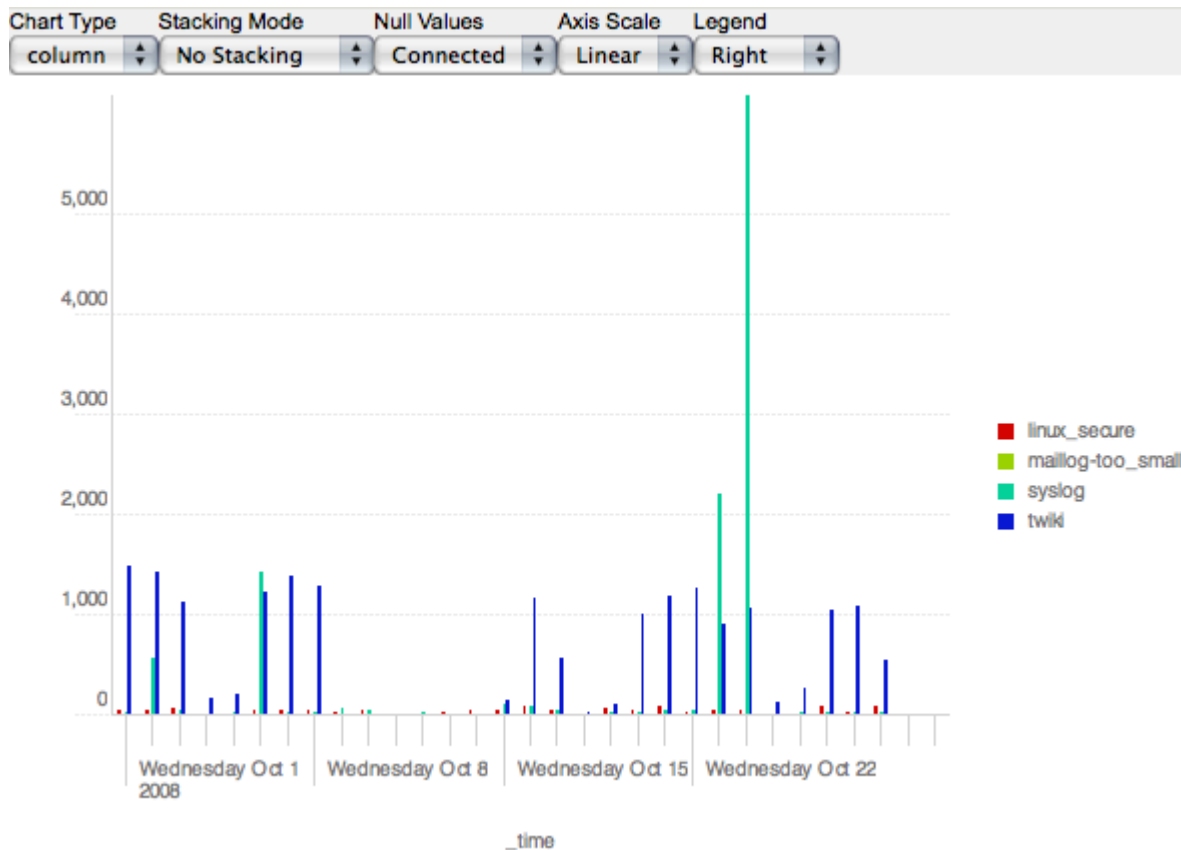
チャートコントロール

結果をチャートで表示する場合、これらのモジュールを使用します。

FlashChart

このモジュールは、Splunk バックエンドで生成可能な検索結果のほとんどをチャートにできる Flash オブジェクトを含みます。

FlashChart モジュールは結果をチャートで表示します。



パラメータ

必須

- width = サイズ
 - ◆ モジュールの幅を指定。
 - ◆ パーセントまたはピクセル数で指定可能。
 - ◆ 通常、"100%"に設定され、利用可能なスペースいっぱいに表示します。 "*"500px"などの値を使ってピクセル数を設定することもできます。
- height = サイズ

- ◆ モジュールの高さを指定。
- ◆ パーセントまたはピクセル数で指定可能。

オプション

- swfFile = パス
 - ◆ swf へのパスを設定。
 - ◆ パスは、\$SPLUNK_HOME/share/splunk/search_mrsparkle/exposed/flash に対する相対パスであること。
 - ◆ swf は、外部使用されない、厳しく制限し、これまで文書化されていない仕様に従うこと。
 - ◆ デフォルトは"charting/bin/charting.swf"。

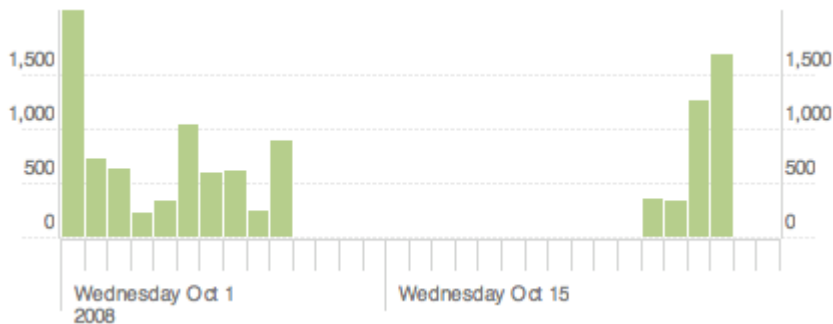
例

```
<module name="FlashChart" layoutPanel="results_area_1">
  <param name="height">300px</param>
  <param name="width">100%</param>
```

FlashTimeline

このモジュールは、時系列でイベント数のチャートを表示可能な Flash オブジェクトを含みます。このチャートは、検索が実行されている間、非同期的に更新されます。

FlashTimeline モジュールは時間軸を生成します。



パラメータ

必須

- width = サイズ
 - ◆ モジュールの幅を指定。
 - ◆ パーセントまたはピクセル数で指定可能。
 - ◆ 通常、これは"100%"に設定され、利用可能なスペースいっぱいに表示します。 "*"500px"などの値を使ってピクセル数を設定することもできます。
- height = サイズ
 - ◆ モジュールの高さを指定。
 - ◆ パーセントまたはピクセル数で指定可能。

オプション

- swfFile = パス
 - ◆ swf へのパスを設定。
 - ◆ パスは、\$SPLUNK_HOME/share/splunk/search_mrsparkle/exposed/flash に対する相対パスであること。
 - ◆ swf は、外部使用されない、厳しく制限し、これまで文書化されていない仕様に従うこと。
 - ◆ デフォルトは"charting/bin/timeline.swf"。
- statusBuckets = 整数
 - ◆ デフォルト= 300
 - ◆ ある場合、300 のデフォルトを上書きします。これは、検索を実行中に、バックエンドが持続可能な時間バケツの最大数です。値を低く設定すると、検索タイムが多少短くなり、ユーザーに表示される情報は減ります。

例

```
<module name="FlashTimeline" layoutPanel="graph_area">  
  <param name="height">180px</param>  
  <param name="width">100%</param>
```

SimpleResultsTable

検索の終了を待ち、その後、最終結果を表形式で生成します。

パラメータ

必須

なし。

オプション

- entityName = events | results | auto
 - ◆ ビューアがイベント、結果、または自動検知に基づいて列/カラムのテーブルを作成するかどうかを決めます。
 - ◆ デフォルトは auto(自動)。
- displayRowNumbers = true | false
 - ◆ true の場合、行数が、テーブルの各行に表示されます。
 - ◆ デフォルトは on(オン)。
- displayMenu = true | false
 - ◆ テーブルのセルに、クリックして表示される検索ヒントのドロップダウンメニューを含めるかどうかを制御します。
 - ◆ デフォルトは false (含めない)。
- dataOverlayMode = heatmap | highlow | none

- ◆ デフォルトデータオーバーレイモードを、heatmap、highlow および none で指定します。
- ◆ デフォルトは none(なし)。

例

```
<module name="SimpleResultsTable" layoutPanel="results_area_1"/>
```

HiddenChartFormatter

このモジュールは、'column'、'line'、'area'、その他さまざまなチャートタイプの変更に使用するプルダウンで構成されています。

パラメータ

必須

なし。

オプション

- chart = area | bar | column | line | pie | scatter
 - ◆ 生成するチャートの全体的タイプを変更するときに使用します。
- chart.stackMode = default | stacked | stacked100
 - ◆ 棒グラフおよび面グラフを'stacked'(積み重ね)モードで表示させるときに使用します。
- chart.nullValueMode = connect | gaps | zero
 - ◆ 'line'チャートおよび'area'チャートで、データにギャップがある場合の対処方法の制御に使用します。
 - ◆ null 値を'0'として扱う、隙間が見えるままにする、または値を補間するのいずれかが行えます。
- secondaryAxis.scale = 'log' | (*empty string*)
 - ◆ Y-軸スケールを対数にする、または線形にするかの指定に使用します。
- legend.placement = right | bottom | left | top | none
 - ◆ チャート自体と相対的なチャート凡例の位置変更を使用します。

例

```
<module name="HiddenChartFormatter">  
  <param name="chart">line</param>  
  <param name="chartTitle">CPU performance in the past 24 hours</param>
```

ページコントロール

ページのイベントをフォーマットするための特別な設定です。

Count

1 ページあたりのイベント数。

注意： Paginator モジュールを連携して使用する場合、Paginator は、カウントモジュールの祖先ではなく派生である

必要があります。

パラメータ

必須

- options = リスト
 - ◆ 2つの項目のリスト : **text** および **value**、および 1つのオプションキー : **selected**。
 - ◆ Text = 表示するテキスト。
 - ◆ Value = 行数(整数)。
 - ◆ 項目がビューのデフォルト設定となるリスト項目とするには、**selected=true** を設定します。

オプション

なし。

例

```
<module name="Count" layoutPanel="page_controls">
  <param name="switcherTitle">As List</param>
  <param name="options">
    <list>
      <param name="text">10</param>
      <param name="value">10</param>
    </list>
    <list>
      <param name="text">20</param>
      <param name="selected">True</param>
      <param name="value">20</param>
    </list>
    <list>
      <param name="text">50</param>
      <param name="value">50</param>
    </list>
  </param>
```

MaxLines

イベント別の最大行数。最大行数を指定するためのドロップダウンを生成します。

パラメータ

必須

- options = リスト
 - ◆ 2つの項目のリスト : **text** および **value**、および 1つのオプションキー : **selected**。
 - ◆ Text = 表示するテキスト。
 - ◆ Value = 行数(整数)。
 - ◆ 項目がビューのデフォルト設定となるリスト項目とするには、**selected=true** を設定します。

オプション

なし。

例

```
<module name="MaxLines" layoutPanel="page_controls">
  <param name="options">
    <list>
      <param name="text">5</param>
      <param name="selected">True</param>
      <param name="value">5</param>
    </list>
    <list>
      <param name="text">10</param>
      <param name="value">10</param>
    </list>
    <list>
      <param name="text">20</param>
      <param name="value">20</param>
    </list>
    <list>
      <param name="text">50</param>
      <param name="value">50</param>
    </list>
    <list>
      <param name="text">100</param>
      <param name="value">100</param>
    </list>
    <list>
      <param name="text">200</param>
      <param name="value">200</param>
    </list>
    <list>
      <param name="text">All</param>
      <param name="value">0</param>
    </list>
  </param>
```

Paginator



```
1 2 3 4 5 6 7 8 9 10 Next
```

イベントをページに分割し、一連のリンクをデータの周りに表示します。ページの設定は、必ず検索の'events'または'results'のいずれかで設定します。

注意： Paginator を持つ EventsViewer(例：EventsViewer は Paginator の子)の前に置いてください。そうでないと、イベントの 1 ページ目のみ表示されます。

パラメータ

必須

- entityName = events | results | settings

- ◆ paginator がイベント数、最終結果、設定マップの変更のいずれかに基づいてリンクを構築すべきか決めます。
- ◆ 多くの場合、これは同じですが、コマンド変換を伴う検索では、これらの数は通常異なります。

オプション

- count = 整数
 - ◆ デフォルトは 10。
 - ◆ 1 ページに表示されるイベント/結果の数。
- maxPages = 整数
 - ◆ デフォルトは 10。
 - ◆ ページの最大数。

例

```
<module name="Paginator" layoutPanel="page_controls">
  <param name="count">100</param>
  <param name="entityName">results</param>
  <param name="switcherTitle">As Table</param>
  <param name="maxPages">10</param>
```

Segmentation

イベントのセグメンテーションタイプを設定します。

パラメータ

[param:options] required = True label = 'text' と 'value' の 2 つの必須キーを持つ項目のリスト。'value' は、raw、inner、outer、full のいずれかです。

必須

- options = リスト
 - ◆ 2 つの項目のリスト : **text** および **value**、および 1 つのオプションキー : **selected**。
 - ◆ Text = 表示するテキスト。
 - ◆ Value = セグメンテーションの値で、inner、outer、full、または raw のいずれかを指定します。
 - ◆ ビューのデフォルトセグメンテーションとなるリスト項目にするには、`selected=true` を設定します。

オプション

- segmentation = inner | outer | full | raw
 - ◆ 定着性があり持続する。
 - ◆ リスニングしているモジュールにブロードキャストするための、セグメンテーションの値を設定します。

例

```
<module name="Segmentation">
  <param name="options">
    <list>
      <param name="text">inner</param>
      <param name="selected">True</param>
      <param name="value">inner</param>
    </list>
    <list>
      <param name="text">outer</param>
      <param name="value">outer</param>
    </list>
    <list>
      <param name="text">full</param>
      <param name="value">full</param>
    </list>
    <list>
      <param name="text">raw</param>
      <param name="value">raw</param>
    </list>
  </param>
</module>
```

SoftWrap

イベントのワードラップを切り替えるチェックボックスを表示します。オフの場合、イベントテキストは、実際のデータに改行がある場合にのみ折り返され、スクロールバーが必要に応じて表示されます。オンの場合、イベントテキストは、ウインドウの端でも折り返されます。

パラメータ

必須

なし。

オプション

- enable = True | False
 - ◆ デフォルトでワードラップをオンにするかどうかを指定。
 - ◆ デフォルトは True(オン)。

例

```
<module name="SoftWrap">
  <param name="switcherTitle">As List</param>
</module>
```

インクルード

次のモジュールを使用して、ウェブリソースをビューに追加します。

ServerSideInclude

カスタムコンテンツのサーバー側のインクルードをサポートします。また、Mako (<http://www.makotemplates.org/>を参照)テンプレート言語をサポートします。

警告： Splunk の CSS は対象となりません。CSS をページに追加する場合は、CSS の対象範囲を確認する、または Splunk のデフォルト CSS を上書きします。

パラメータ

必須

- `src = URI`
 - ◆ 表示する固定 html ファイルを指定。
 - ◆ このファイルは、必ずこのモジュールの `src` サブディレクトリに格納します。

オプション

なし。

例

```
<module name="ServerSideInclude" layoutPanel="full_width_controls">  
  <param name="src">hello_world.html</param>  
</module>
```

IFrameInclude

他の URL からコンテンツを表示するインラインフレーム(iframe)を設定します。

パラメータ

必須

- `src = URI`
 - ◆ IFrame の URL を指定。

オプション

- `check_exists = False | True`
 - ◆ リモート `src` またはローカル `src` が存在するかどうかをチェックします。
 - ◆ 注意： `check_exists` が `True` に設定されている場合は、ローカル App の固定クファイルはスキップします。
 - ◆ デフォルトは `False`(チェックする)。
- `height = auto | number of pixels`
 - ◆ `iframe` を制限する値ピクセル数(数値)。

- ◆ リモートの URI は、fluid height を正しく機能させるために、適切な JavaScript の document.domain 設定が必要です。([http://msdn.microsoft.com/en-us/library/cc196989\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/cc196989(VS.85).aspx) および <https://developer.mozilla.org/en/DOM/document.domain> 参照のこと)
- ◆ デフォルトは auto(自動設定)。

例

```
<module name="IFrameInclude" layoutPanel="full_width_controls">  
  <param name="src">http://www.google.com</param>  
</module>
```

高度なトピック

モジュールの構築

モジュールの構築

この記事はスタブです。

モジュールの構築は、Splunk のモジュールディレクトリで利用可能なすべてのモジュールについて理解してから行ってください。\$SPLUNK_HOME/share/splunk/search_mrsparkle/modules/を参照してください。モジュールは JavaScript および HTML から構築されています。その中には各パラメータについて説明する特別な設定ファイルがあります。また、モジュールは Python を含めることができます。モジュールの JavaScript ファイルは、そのモジュールがフレームワーク全体とどのように相互作用するかを示しています。つまり、モジュールが使用するパラメータの種類と、その検索への適用方法を示します(検索と相互作用する場合)。HTML ファイルは、モジュールをページレイアウトする方法を指定します。

mako テンプレートの構築

mako テンプレートの構築

この記事はスタブです。

テンプレートは、\$SPLUNK_HOME/share/splunk/search_mrsparkle/templates のテンプレートディレクトリに追加できます。テンプレートは、Splunk Web のページをレイアウトする方法を指定します。Splunk 管理セクションについても同様です。テンプレートは Mako テンプレートシステムで構築されています。

Splunk ダッシュボードエレメントを他社製ソフトウェアに埋め込む

Splunk ダッシュボードエレメントを他社製ソフトウェアに埋め込む

Splunk が生成したチャートやグラフを他のウェブアプリケーションに埋め込みます。現時点では、Splunk はシングルサインオンを実装していません。したがって、Splunk に対する認証はインセキュアログインを使用する必要があります。

注意： ページにアクセスする者は誰でも、ページソースを閲覧することによりユーザー資格を得ることができます。

インセキュアログインを有効にする

まず始めに、web.conf でインセキュアログインを有効にします。次のスタンザを追加します。

```
[settings]
enable_insecure_login = true
...
```

その後、Splunk の再起動が必要です。

Splunk の設定ファイルの機能について詳しくない場合は、管理者マニュアルの設定ファイルのトピックを参照してください。

Splunk でビューを作成する

次に、他社製アプリケーションに埋め込む要素を含むビューXML を作成します(例：グラフやチャートなど)。このビューには、簡略化 XML は使用できません。必ず高度 XML で記述してください。

ビューの例

`$SPLUNK_HOME/etc/apps/my_test_app/local/data/ui/views/my_view.xml` からの例を以下に示します。

```
<view template="dashboard.html">
  <module name="StaticContentSample" layoutPanel="panel_row1_col1">
    <param name="text">This is some sample static text.</param>
  </module>
  <module name="StaticContentSample" layoutPanel="panel_row1_col2">
    <param name="text">This is some more sample static text.</param>
  </module>
  <module name="StaticContentSample" layoutPanel="panel_row2_col1">
    <param name="text">This is even more sample static text.</param>
  </module>
</view>
```

ビューに変更を加えた場合は、次の URI をロードしてそのビューをリフレッシュしてください。

`https://myhost.com:8089/servicesNS/myuser/search/data/ui/views?refresh=1`

同じユーザーでリフレッシュする必要はありません。有効なユーザーであれば誰でも可能です。

自分のサイトに iframe を作成する

最後に、サードパーティアプリケーションを閲覧する iframe を次の/insecurelogin エンドポイントにより作成します。

`http://splunkserver:8000/account/insecurelogin?username=admin&password=changeme&return_to=/app/foo/myview`

"return_to"パラメータの値は URI をエスケープしてください(`http://meyerweb.com/eric/tools/dencoder/`)。作成した HTML ページは、iframe 要素内でそのビューを生成します。

HTML の例

iframe と html に追加する例を次に示します。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Splunk Stuff</title>
</head>
<body>
<!-- content starts here -->
<h1>Hey Look At My Pretty Pictures!</h1>
```

```
<iframe src
="http://myhost:8000/account/insecurelogin?username=test_user&password=changeme&return_
to=%2Fapp%2Fmy_test_app%2Fmy_view" width="100%" height="300">
  <p>Your browser does not support iframes.</p>
</iframe>
<!-- content ends here -->
</body>
</html>
```

最終仕上げ

App をパッケージにする

App をパッケージにする

この記事はスタブです。

設定したすべてのコンポーネントが、`$SPLUNK_HOME/etc/apps/<your_App>/default/`にある App のデフォルトディレクトリにあることを確認します。Appbuilder スクリプトを使用している場合は、App 設定はデフォルトまたはローカルのいずれかにあります。

Splunk Base の配布

Splunk Base の配布

この記事はスタブです。

App の構築が終了すると、Splunk Base で配布することができます。

拡張例

拡張例

拡張例

このセクションは、ダッシュボード、フォーム検索、その他のビューの拡張例を記載しています。

ダッシュボードの例

ダッシュボードの例

このセクションはさまざまなダッシュボード設定の例をいくつか記載します。

例 1

```
<view template="dashboard.html">
  <label>Advanced Dashboards - Intro</label>
  <module name="AccountBar" layoutPanel="appHeader" />
  <module name="AppBar" layoutPanel="navigationHeader" />
  <module name="Message" layoutPanel="messaging">
    <param name="filter">*</param>
    <param name="clearOnJobDispatch">False</param>
    <param name="maxSize">1</param>
  </module>
  <module name="TitleBar" layoutPanel="viewHeader">
    <param name="actionsMenuFilter">dashboard</param>
  </module>

  <module name="ServerSideInclude" layoutPanel="panel_row1_col1">
    <param name="src">advanced_dashboard1.html</param>
  </module>

  <module name="StaticContentSample" layoutPanel="panel_row2_col1">
    <param name="text">This is a basic example using HiddenSavedSearch + HiddenChartFormatter
+ FlashChart to show a flash chart from a saved search.</param>
  </module>
  <module name="HiddenSearch" layoutPanel="panel_row2_col1" group="Charting the results of
a saved search" autoRun="True">
    <param name="search">index=_internal source=*metrics.log Component=metrics
group=per_sourcetype_thruput | chart sum(kb) by series | sort sum(kb) desc | head 10</param>
    <param name="earliest">-1h</param>
  <module name="HiddenChartFormatter">
    <param name="chart">bar</param>
    <param name="primaryAxisTitle.text">Sourcetype</param>
    <param name="secondaryAxisTitle.text">KB Indexed</param>
    <param name="legend.placement">none</param>
  </module>
  <module name="JobProgressIndicator" />
  <module name="FlashChart">
    <param name="width">100%</param>
    <param name="height">300px</param>
  </module>
</view>
```

```

    </module>
</module>

<module name="StaticContentSample" layoutPanel="panel_row2_col2" group="Allowing the user
to change the time range">
    <param name="text">This example uses HiddenSearch instead of HiddenSavedSearch, and
includes a TimeRangePicker module to allow the user to change the time range.</param>
</module>
<module name="TimeRangePicker" layoutPanel="panel_row2_col2">
    <param name="searchWhenChanged">True</param>
    <param name="selected">All time</param>
    <module name="HiddenSearch" autoRun="True">
        <param name="search">| dbinspect bins=400</param>
        <module name="HiddenChartFormatter">
            <param name="chart">line</param>
            <param name="primaryAxisTitle.text">Time</param>
            <param name="chartTitle">Distribution of index buckets over time</param>
            <module name="JobProgressIndicator"/>
            <module name="FlashChart"/>
        </module>
    </module>
</module>
<module name="StaticContentSample" layoutPanel="panel_row3_col1">
    <param name="text">This example redirects the user to see the same graph in the charting
view using a simple module called ViewRedirectorLink.</param>
</module>
<module name="HiddenSearch" layoutPanel="panel_row3_col1" group="Simple Redirects - 1"
autoRun="True">
    <param name="search">index=_internal source=*metrics.log Component=metrics
group=per_sourcetype_thruput | timechart sum(kb)</param>
    <param name="earliest">-1h</param>
    <module name="HiddenChartFormatter">
        <param name="chart">line</param>
        <param name="primaryAxisTitle.text">time</param>
        <param name="secondaryAxisTitle.text">KB Indexed</param>
        <param name="legend.placement">right</param>
        <module name="JobProgressIndicator"/>
        <module name="FlashChart">
            <param name="width">100%</param>
            <param name="height">200px</param>
        </module>
    <module name="ViewRedirectorLink" layoutPanel="panel_row3_col1">
        <param name="viewTarget">charting</param>
        <param name="label">Redirect 1</param>
    </module>
</module>
</module>

<module name="StaticContentSample" layoutPanel="panel_row3_col2">
    <param name="text">This example redirects the user to see the same graph in the charting
view using a pair of modules, SubmitButton + ViewRedirector. Just for fun we added a
TimeRangePicker above the SubmitButton which allows the user to change the timerange before
the redirect.</param>
</module>
<module name="HiddenSearch" layoutPanel="panel_row3_col2" group="Simple Redirects - 2"
autoRun="True">
    <param name="search">index=_internal source=*metrics.log Component=metrics
group=per_sourcetype_thruput | timechart sum(kb)</param>
    <param name="earliest">-1h</param>

```

```

<module name="HiddenChartFormatter">
  <param name="chart">line</param>
  <param name="primaryAxisTitle.text">time</param>
  <param name="secondaryAxisTitle.text">KB Indexed</param>
  <param name="legend.placement">right</param>
  <module name="JobProgressIndicator"/>
  <module name="FlashChart">
    <param name="width">100%</param>
    <param name="height">200px</param>
  </module>
  <module name="TimeRangePicker">
    <param name="searchWhenChanged">True</param>
    <param name="selected">Today</param>
    <module name="SubmitButton">
      <param name="label">Redirect 2</param>
      <module name="ViewRedirector">
        <param name="viewTarget">charting</param>
      </module>
    </module>
  </module>
</module>
<module name="ServerSideInclude" layoutPanel="panel_row4_coll">
  <param name="src">advanced_dashboard1_summary.html</param>
</module>

</view>

```

例 2

```

<view template="dashboard.html">
  <label>Advanced Dashboards - Drilldown patterns</label>
  <module name="AccountBar" layoutPanel="appHeader"/>
  <module name="AppBar" layoutPanel="navigationHeader"/>
  <module name="Message" layoutPanel="messaging">
    <param name="filter">*</param>
    <param name="clearOnJobDispatch">False</param>
    <param name="maxSize">1</param>
  </module>
  <module name="TitleBar" layoutPanel="viewHeader">
    <param name="actionsMenuFilter">dashboard</param>
  </module>

  <module name="StaticContentSample" layoutPanel="panel_row1_coll">
    <param name="text">These examples demonstrate a few 'drilldown' patterns for dashboards
that are particularly useful. There is an enormous range of possible configurations and these
are only a couple examples.</param>
  </module>

  <module name="StaticContentSample" layoutPanel="panel_row2_coll">
    <param name="text">This simple example uses a complex SearchSelectLister that generates
the 10 sourcetypes with the most data indexed in the last hour. The user can then pick one
and then we redirect them to flashtimeline and run a search for just the events from that
sourcetype, this time over the past 2 hours. </param>
  </module>
  <!-- our base search will just be "*", over the past 2 hours -->
  <module name="HiddenSearch" layoutPanel="panel_row2_coll" group="Drilldowns - 1"
autoRun="True">
    <param name="search">*</param>

```

```

<param name="earliest">-2h</param>
<module name="SearchSelectLister">
  <param name="settingToCreate">series_setting</param>
  <param name="search">index=_internal metrics NOT source="*web_service.log" NOT
source="*access.log" NOT source="*/searches.log" NOT source="*intentions.log" NOT
source="*splunkd.log" group="per_sourcetype_thruput" | chart sum(kb) over series | sort
sum(kb) desc | head 10 | sort series</param>
  <param name="earliest">-1h</param>
  <param name="label">source</param>
  <param name="searchWhenChanged">True</param>
  <param name="searchFieldsToDisplay">
    <list>
      <param name="label">series</param>
      <param name="value">series</param>
    </list>
  </param>
</module name="ConvertToIntention">
  <param name="settingToConvert">series_setting</param>
  <param name="intention">
    <param name="name">addterm</param>
    <param name="arg">
      <param name="sourcetype">${target}$</param>
    </param>
  </param>
</module name="SubmitButton">
  <param name="label">Drilldown 1</param>
  <module name="ViewRedirector">
    <param name="viewTarget">flashtimeline</param>
  </module>
</module>
</module>
</module>
<module name="StaticContentSample" layoutPanel="panel_row2_col2">
  <param name="text">This example is the same except instead of SearchSelectLister we use
SearchLinkLister and ViewRedirector. The takeaway is that the listers are all interchangeable.
ie if you'd rather have radio buttons just use SearchRadioLister.</param>
</module>
<module name="HiddenSearch" layoutPanel="panel_row2_col2" group="Drilldowns - 2" >
  <param name="search">*</param>
  <param name="earliest">-2h</param>
  <module name="SearchLinkLister">
    <param name="settingToCreate">series_setting</param>
    <param name="search">index=_internal metrics NOT source="*web_service.log" NOT
source="*access.log" NOT source="*/searches.log" NOT source="*intentions.log" NOT
source="*splunkd.log" | chart sum(kb) over series | sort sum(kb) desc | head 10 | sort
series</param>
    <param name="earliest">-1h</param>
    <param name="searchWhenChanged">True</param>
    <param name="searchFieldsToDisplay">
      <list>
        <param name="label">series</param>
        <param name="value">series</param>
      </list>
    </param>
  </module name="ConvertToIntention">
    <param name="settingToConvert">series_setting</param>
    <param name="intention">
      <param name="name">addterm</param>
    </param>
  </module>
</module>

```

```

    <param name="arg">
      <param name="sourcetype">$target$</param>
    </param>
  </param>
  <module name="ViewRedirector">
    <param name="viewTarget">flashtimeline</param>
  </module>
</module>
</module>
</module>
<module name="StaticContentSample" layoutPanel="panel_row3_col1">
  <param name="text">
    This example uses a SearchSelectLister to give the list of indexes. Then a second
    SearchSelectLister generates a link for each sourcetype within the chosen index.
  </param>
</module>
<module name="StaticContentSample" layoutPanel="panel_row3_col1">
  <param name="text">
    Then when the user chooses a sourcetype from the second pulldown we will construct a
    third search for index="foo" sourcetype="bar". When the user clicks the SubmitButton module
    this search will travel down, hit a module called ViewRedirector, and off the user goes.
  </param>
</module>
<module name="StaticContentSample" layoutPanel="panel_row3_col1">
  <param name="text">
    Just for fun we also have configured the ViewRedirector module to launch a popup window,
    and instead of flashtimeline we use a simple stripped down view that we made just for this
    app.
  </param>
</module>

<module name="SearchSelectLister" layoutPanel="panel_row3_col1" group="Drilldowns - 3">
  <param name="label">which index</param>
  <param name="settingToCreate">index_setting</param>
  <param name="search">| eventcount summarize=false index=* | search index!="splunklogger"
  index!="summary" index!="history" | sort index desc</param>
  <param name="searchWhenChanged">True</param>
  <param name="searchFieldsToDisplay">
    <list>
      <param name="label">index</param>
      <param name="value">index</param>
    </list>
  </param>
  <module name="ConvertToIntention">
    <param name="settingToConvert">index_setting</param>
    <param name="intention">
      <param name="name">stringreplace</param>
      <param name="arg">
        <param name="index">
          <param name="fillOnEmpty">True</param>
          <param name="prefix">index=</param>
          <param name="value">$target$</param>
        </param>
      </param>
    </module>
  </param>
  <module name="SearchSelectLister">
    <param name="label">Sourcetype</param>
    <param name="settingToCreate">sourcetype_setting</param>
    <param name="search">| metadata type="sourcetypes" $index$</param>

```

```

<param name="applyOuterIntentionsToInternalSearch">True</param>
<param name="searchFieldsToDisplay">
  <list>
    <param name="label">sourcetype</param>
    <param name="value">sourcetype</param>
  </list>
</param>
<module name="HiddenSearch">
  <param name="search">$index$ $sourcetype$</param>
  <module name="ConvertToIntention">
    <param name="settingToConvert">sourcetype_setting</param>
    <param name="intention">
      <param name="name">stringreplace</param>
      <param name="arg">
        <param name="sourcetype">
          <param name="fillOnEmpty">True</param>
          <param name="prefix">sourcetype=</param>
          <param name="value">$target$</param>
        </param>
      </param>
    </param>
  </module>
  <module name="SubmitButton">
    <param name="label">Search</param>
    <module name="ViewRedirector">
      <param name="viewTarget">really_simple_viewer</param>
      <param name="popup">True</param>
    </module>
  </module>
</module>
</module>
</module>
</module>
</module>
<module name="StaticContentSample" layoutPanel="panel_row3_col2">
  <param name="text">very similar except that the second is a SearchLinkLister. Beware that
prior to 4.0.3 there is a bug in this configuration.(SPL-25464).</param>
</module>

<module name="SearchSelectLister" layoutPanel="panel_row3_col2" group="Drilldowns - 4">
  <param name="label">which index</param>
  <param name="settingToCreate">index_setting</param>
  <param name="search">| eventcount summarize=false index=* | search index!="splunklogger"
index!="summary" index!="history" | sort index desc</param>
  <param name="searchWhenChanged">True</param>
  <param name="searchFieldsToDisplay">
    <list>
      <param name="label">index</param>
      <param name="value">index</param>
    </list>
  </param>
  <module name="ConvertToIntention">
    <param name="settingToConvert">index_setting</param>
    <param name="intention">
      <param name="name">stringreplace</param>
      <param name="arg">
        <param name="index">
          <param name="fillOnEmpty">True</param>
          <param name="prefix">index=</param>
          <param name="value">$target$</param>
        </param>
      </param>
    </param>
  </module>
</module>

```

```

    </param>
  </param>
</param>
<module name="SearchLinkLister">
  <param name="settingToCreate">sourcetype_setting</param>
  <param name="search">| metadata type="sourcetypes" $index$</param>
  <param name="applyOuterIntentionsToInternalSearch">True</param>
  <param name="searchFieldsToDisplay">
    <list>
      <param name="label">sourcetype</param>
      <param name="value">sourcetype</param>
    </list>
  </param>
<module name="HiddenSearch">
  <param name="search">$index$ $sourcetype$</param>
  <module name="ConvertToIntention">
    <param name="settingToConvert">sourcetype_setting</param>
    <param name="intention">
      <param name="name">stringreplace</param>
      <param name="arg">
        <param name="sourcetype">
          <param name="fillOnEmpty">True</param>
          <param name="prefix">sourcetype=</param>
          <param name="value">$target$</param>
        </param>
      </param>
    </param>
  </module>
  <module name="ViewRedirector">
    <param name="viewTarget">flashtimeline</param>
  </module>
</module>
</module>
</module>
</module>
</module>

<module name="StaticContentSample" layoutPanel="panel_row4_coll">
  <param name="text">Now we take a bunch of leaps ahead and put it all together. We put
in a Sorter module, a Paginator module. We put in a HiddenSearch+SimpleResultsHeader pattern
to give us 'Sources (208)'. Then we duplicate the same pattern for both Sourcetypes and Hosts.
(NOTE: prior to 4.0.3 a bug will make this configuration not work, specifically because of
the presence of the Paginator. SPL-25516)</param>
</module>

<module name="SearchSelectLister" layoutPanel="panel_row4_coll" group="Drilldowns - 5">
  <param name="label">which index</param>
  <param name="settingToCreate">index_setting</param>
  <param name="search">| eventcount summarize=false index=* | search index!="splunklogger"
index!="summary" index!="history" | sort index desc</param>
  <param name="searchWhenChanged">True</param>
  <param name="searchFieldsToDisplay">
    <list>
      <param name="label">index</param>
      <param name="value">index</param>
    </list>
  </param>
</module>

```

```

<!-- Here is where we take the generic setting value outputted by the SearchSelectList,
and
    we turn that into an intention.
    Ultimately this one intention will be consumed in 6 different places.
    We will use the combination of HiddenSearch + SimpleResultsHeader in the first 3
child-branches
    to generate the 'Sourcetypes (14)' headers.
    Then we will have 3 SearchLinkLister modules configured with
applyOuterIntentionsToInternalSearch="True"
-->
<module name="ConvertToIntention">
  <param name="settingToConvert">index_setting</param>
  <param name="intention">
    <param name="name">stringreplace</param>
    <param name="arg">
      <param name="index">
        <param name="fillOnEmpty">True</param>
        <param name="prefix">index=</param>
        <param name="value">$target$</param>
      </param>
    </param>
  </param>
</module>
<module name="HiddenSearch">
  <param name="search">| metadata type=sources $index$</param>
  <module name="SimpleResultsHeader" layoutPanel="panel_row4_coll_grp1">
    <param name="entityName">results</param>
    <param name="headerFormat">Sources ( %(count)s )</param>
  </module>
</module>
<module name="HiddenSearch">
  <param name="search">| metadata type=sourcetypes $index$</param>
  <module name="SimpleResultsHeader" layoutPanel="panel_row4_coll_grp2">
    <param name="entityName">results</param>
    <param name="headerFormat">Sourcetypes ( %(count)s )</param>
  </module>
</module>
<module name="HiddenSearch">
  <param name="search">| metadata type=hosts $index$</param>
  <module name="SimpleResultsHeader" layoutPanel="panel_row4_coll_grp3">
    <param name="entityName">results</param>
    <param name="headerFormat">Hosts ( %(count)s )</param>
  </module>
</module>
<!-- from here on inward, it's all just dealing with sources. We put each of these 3
branches
    in its own grp layoutPanel so that they will all float alongside eachother
    within the same panel. -->
<module name="Sorter" layoutPanel="panel_row4_coll_grp1">
  <param name="sortKey">totalCount</param>
  <param name="sortDir">desc</param>
  <param name="fields">
    <list>
      <param name="label">Source</param>
      <param name="value">source</param>
    </list>
    <list>
      <param name="label">Total Count</param>
      <param name="value">totalCount</param>
    </list>
  </param>
</module>

```

```

    <list>
      <param name="label">First Time</param>
      <param name="value">firstTime</param>
    </list>
  </param>
  <module name="Paginator">
    <param name="count">10</param>
    <param name="entityName">settings</param>
    <param name="maxPages">10</param>
    <!-- the module that generates the blue links. Note that it although it configures
its own internal search,
it has a flag that allows it to apply intentions from the main context, to its internal
search.
-->
  <module name="SearchLinkLister">
    <param name="settingToCreate">list1</param>
    <param name="search">| metadata type=sources $index$</param>
    <param name="applyOuterIntentionsToInternalSearch">True</param>
    <param name="searchFieldsToDisplay">
      <list>
        <param name="label">source</param>
        <param name="value">source</param>
      </list>
      <list>
        <param name="label">totalCount</param>
        <param name="labelFormat">number</param>
      </list>
    </param>
    <!-- Further upstream we used ConvertToIntention to make a stringreplace intention
for a metadata search.
Here we use it again, this time to convert the same setting to a simpler addterm
intention.
(the first intention will have been consumed by the searchSelectLister, but the
setting will still be there ) -->
  <module name="ConvertToIntention">
    <param name="settingToConvert">index_setting</param>
    <param name="intention">
      <param name="name">addterm</param>
      <param name="arg">
        <param name="index">$target$</param>
      </param>
    </param>
    <!-- This takes some getting used to, but here is where we take the setting
generated by the SearchLinkLister
when the user clicks a link, and convert that into an intention. -->
  <module name="ConvertToIntention">
    <param name="settingToConvert">list1</param>
    <param name="intention">
      <param name="name">addterm</param>
      <param name="arg">
        <param name="source">$target$</param>
      </param>
    </param>
    <!-- And finally, whenever any context information hits this ViewRedirector
we will instantly redirect.
because of the internals of 'SearchLinkLister', nothing will arrive here
until the moment of the user
clicking the blue link -->
  <module name="ViewRedirector">

```

```

        <param name="viewTarget">flashtimeline</param>
    </module>
</module>
</module>
</module>
</module>
</module>
<!-- from here on inward, it's all just dealing with sourcetypes -->
<module name="Sorter" layoutPanel="panel_row4_coll_grp2">
    <param name="sortKey">totalCount</param>
    <param name="sortDir">desc</param>
    <param name="fields">
        <list>
            <param name="label">Sourcetype</param>
            <param name="value">sourcetype</param>
        </list>
        <list>
            <param name="label">Total Count</param>
            <param name="value">totalCount</param>
        </list>
        <list>
            <param name="label">First Time</param>
            <param name="value">firstTime</param>
        </list>
    </param>
<module name="Paginator">
    <param name="count">10</param>
    <param name="entityName">settings</param>
    <param name="maxPages">10</param>
    <!-- the module that generates the blue links. Note that it although it configures
its own internal search,
it has a flag that allows it to apply intentions from the main context, to its internal
search.
-->
<module name="SearchLinkLister">
    <param name="settingToCreate">list1</param>
    <param name="search">| metadata type=sourcetypes $index$</param>
    <param name="applyOuterIntentionsToInternalSearch">True</param>
    <param name="searchFieldsToDisplay">
        <list>
            <param name="label">sourcetype</param>
            <param name="value">sourcetype</param>
        </list>
        <list>
            <param name="label">totalCount</param>
            <param name="labelFormat">number</param>
        </list>
    </param>
    <!-- Further upstream we used ConvertToIntention to make a stringreplace intention
for a metadata search.
Here we use it again, this time to convert the same setting to a simpler addterm
intention.
(the first intention will have been consumed by the searchSelectLister, but the
setting will still be there ) -->
<module name="ConvertToIntention">
    <param name="settingToConvert">index_setting</param>
    <param name="intention">
        <param name="name">addterm</param>
        <param name="arg">

```

```

        <param name="index">$target$</param>
    </param>
</param>
    <!-- This takes some getting used to, but here is where we take the setting
generated by the SearchLinkLister
        when the user clicks a link, and convert that into an intention. -->
    <module name="ConvertToIntention">
        <param name="settingToConvert">list1</param>
        <param name="intention">
            <param name="name">addterm</param>
            <param name="arg">
                <param name="sourcetype">$target$</param>
            </param>
        </param>
    </module>
    <!-- And finally, whenever any context information hits this ViewRedirector
we will instantly redirect.
        because of the internals of 'SearchLinkLister', nothing will arrive here
until the moment of the user
        clicking the blue link -->
    <module name="ViewRedirector">
        <param name="viewTarget">flashtimeline</param>
    </module>
</module>
</module>
</module>
</module>
</module>
<!-- from here on inward, it's all just dealing with hosts -->
<module name="Sorter" layoutPanel="panel_row4_coll_grp3">
    <param name="sortKey">totalCount</param>
    <param name="sortDir">desc</param>
    <param name="fields">
        <list>
            <param name="label">Host</param>
            <param name="value">host</param>
        </list>
        <list>
            <param name="label">Total Count</param>
            <param name="value">totalCount</param>
        </list>
        <list>
            <param name="label">First Time</param>
            <param name="value">firstTime</param>
        </list>
    </param>
    <module name="Paginator">
        <param name="count">10</param>
        <param name="entityName">settings</param>
        <param name="maxPages">10</param>
        <!-- the module that generates the blue links. Note that it although it configures
its own internal search,
        it has a flag that allows it to apply intentions from the main context, to its internal
search.
-->
    <module name="SearchLinkLister">
        <param name="settingToCreate">list1</param>
        <param name="search">| metadata type=hosts $index$</param>
        <param name="applyOuterIntentionsToInternalSearch">True</param>
        <param name="searchFieldsToDisplay">

```

```

        <list>
            <param name="label">host</param>
            <param name="value">host</param>
        </list>
        <list>
            <param name="label">totalCount</param>
            <param name="labelFormat">number</param>
        </list>
    </param>
    <!-- Further upstream we used ConvertToIntention to make a stringreplace intention
for a metadata search.
    Here we use it again, this time to convert the same setting to a simpler addterm
intention.
    (the first intention will have been consumed by the searchSelectLister, but the
setting will still be there ) -->
    <module name="ConvertToIntention">
        <param name="settingToConvert">index_setting</param>
        <param name="intention">
            <param name="name">addterm</param>
            <param name="arg">
                <param name="index">$target$</param>
            </param>
        </param>
    </module>
    <!-- This takes some getting used to, but here is where we take the setting
generated by the SearchLinkLister
        when the user clicks a link, and convert that into an intention. -->
    <module name="ConvertToIntention">
        <param name="settingToConvert">list1</param>
        <param name="intention">
            <param name="name">addterm</param>
            <param name="arg">
                <param name="host">$target$</param>
            </param>
        </param>
    </module>
    <!-- And finally, whenever any context information hits this ViewRedirector
we will instantly redirect.
        because of the internals of 'SearchLinkLister', nothing will arrive here
until the moment of the user
        clicking the blue link -->
    <module name="ViewRedirector">
        <param name="viewTarget">flashtimeline</param>
    </module>
</module>
</module>
</module>
</module>
</module>
</module>
</module>
</module>
</module>
</module>
</module>

<module name="ServerSideInclude" layoutPanel="panel_row5_col1">
    <param name="src">advanced_dashboard2_summary.html</param>
</module>

</view>

```

フォーム検索の例

フォーム検索の例

例 1

```
<view onunloadCancelJobs="False" autoCancelInterval="100">
  <!-- autoCancelInterval is set here to 100 -->
  <label>Advanced Form Search - 1</label>
  <module name="AccountBar" layoutPanel="appHeader"/>
  <module name="AppBar" layoutPanel="navigationHeader"/>
  <module name="Message" layoutPanel="messaging">
    <param name="filter">*</param>
    <param name="clearOnJobDispatch">False</param>
    <param name="maxSize">1</param>
  </module>
  <module name="TitleBar" layoutPanel="viewHeader">
    <param name="actionsMenuFilter">dashboard</param>
  </module>
  <module name="HiddenSearch" layoutPanel="mainSearchControls" autoRun="True">
    <param name="search">index=_internal metrics</param>
    <module name="StaticSelect">
      <param name="settingToCreate">group</param>
      <param name="label">field:</param>
      <param name="staticFieldsToDisplay">
        <list>
          <param name="label">Index</param>
          <param name="value">per_index_thruput</param>
        </list>
        <list>
          <param name="label">Source</param>
          <param name="value">per_source_thruput</param>
        </list>
        <list>
          <param name="label">Sourcetype</param>
          <param name="value">per_sourcetype_thruput</param>
        </list>
        <list>
          <param name="label">Host</param>
          <param name="value">per_host_thruput</param>
        </list>
      </param>

      <!-- just for this module we need to render him into 'mainSearchControls' or else he'll
take up an odd space in 'splSearchControls-inline' -->
      <module name="ConvertToIntention">
        <param name="settingToConvert">group</param>
        <param name="intention">
          <param name="name">addterm</param>
          <param name="arg">
            <param name="group">${target$}</param>
          </param>
        </param>
      </module>
      <!-- and then in the very next module we return to putting modules into
'splSearchControls-inline' -->
    </module>
  </module>
</view>
```

```

<module name="SearchSelectLister">
  <param name="settingToCreate">series_setting</param>
  <param name="label">value:</param>
  <param name="applyOuterIntentionsToInternalSearch">True</param>
  <param name="search">index=_internal source="*metrics.log" metrics group series |
head 5000 | top limit=200 series | sort series</param>
  <param name="searchFieldsToDisplay">
    <list>
      <param name="label">series</param>
      <param name="value">series</param>
    </list>
  </param>
<module name="TimeRangePicker">
  <!--
  <param name="label">time range:</param>
  -->
  <param name="selected">Last 4 hours</param>
  <param name="searchWhenChanged">True</param>
  <module name="SubmitButton">
    <param name="allowSoftSubmit">False</param>
    <param name="label">Search</param>
  <module name="ConvertToIntention">
    <param name="settingToConvert">series_setting</param>
    <param name="intention">
      <param name="name">addterm</param>
      <param name="arg">
        <param name="series">${target}</param>
      </param>
    </param>
  </module>

  <module name="Message" layoutPanel="graphArea">
    <param name="filter">splunk.search.job</param>
    <param name="clearOnJobDispatch">True</param>
    <param name="maxSize">2</param>

    <module name="StaticContentSample" layoutPanel="resultsAreaLeft">
      <param name="text">Now we use a HiddenSearch module to reset the base search
string to be a timechart of sum(kb). However because HiddenSearch is downstream of the
SubmitButton module, it will still pick up all the stuff the user picked above. We also use
a HiddenChartFormatter here to give us a column chart, suppress the legend and specify the
correct axis titles.</param>
    </module>
    <module name="HiddenSearch" layoutPanel="resultsAreaLeft">
      <param name="search">index=_internal metrics NOT
source="*web_service.log" NOT source="*access.log" NOT source="*/searches.log" NOT
source="*intentions.log" NOT source="*splunkd.log" | timechart sum(kb)</param>
      <module name="HiddenChartFormatter">
        <param name="chart">column</param>
        <param name="primaryAxisTitle.text">(Selected Series)</param>
        <param name="secondaryAxisTitle.text">KB Indexed</param>
        <param name="legend.placement">none</param>
      <module name="JobProgressIndicator"/>
      <module name="FlashChart">
        <param name="width">100%</param>
        <param name="height">200px</param>
      </module>
    </module>
  </module>

```

```

        <module name="StaticContentSample" layoutPanel="resultsAreaLeft">
            <param name="text">Here we do the same thing, also living directly
underneath the SubmitButton module, but instead we reset everything to show the max(eps),
min(eps) and avg(eps) over time.</param>
        </module>
        <module name="HiddenSearch" layoutPanel="resultsAreaLeft">
            <param name="search">index=_internal metrics NOT
source="*web_service.log" NOT source="*access.log" NOT source="*/searches.log" NOT
source="*intentions.log" NOT source="*splunkd.log" | timechart min(eps) avg(eps)
max(eps)</param>
            <module name="HiddenChartFormatter">
                <param name="chart">line</param>
                <param name="primaryAxisTitle.text">(Selected Series)</param>
                <param name="secondaryAxisTitle.text">event throughput</param>
                <param name="legend.placement">bottom</param>
                <module name="JobProgressIndicator"/>
                <module name="FlashChart">
                    <param name="width">100%</param>
                    <param name="height">200px</param>
                </module>
            </module>
        </module>
    </module>
</module>
</module>
</module>
</module>
</module>
</module>
</module>
</module>
</view>

```

例 2

```

<view template="dashboard.html">
    <label>Advanced Form Search - 2</label>
    <module name="AccountBar" layoutPanel="appHeader"/>
    <module name="AppBar" layoutPanel="navigationHeader"/>
    <module name="Message" layoutPanel="messaging">
        <param name="filter">*</param>
        <param name="clearOnJobDispatch">False</param>
        <param name="maxSize">1</param>
    </module>
    <module name="TitleBar" layoutPanel="viewHeader">
        <param name="actionsMenuFilter">dashboard</param>
    </module>
    <module name="SearchSelectLister" layoutPanel="panel_row1_coll" group="Chose group and
series to view sum(kb), avg(kbps), median(eps), max(eps) and min(eps)">
        <param name="settingToCreate">group_setting</param>
        <param name="search">index=_internal source=*metrics.log Component=metrics group
group="*" series="*" | stats count by group</param>
        <param name="earliest">-6h</param>
        <param name="label">Groups</param>
        <param name="searchFieldsToDisplay">
            <list>
                <param name="label">group</param>
                <param name="value">group</param>
            </list>
        </param>
    </module>

```

```

<module name="ConvertToIntention">
  <param name="settingToConvert">group_setting</param>
  <param name="intention">
    <param name="name">stringreplace</param>
    <param name="arg">
      <param name="group">
        <param name="value">${target$}</param>
      </param>
    </param>
  </param>
</module>

<!-- Series selector -->
<module name="SearchSelectLister">
  <param name="settingToCreate">series_setting</param>
  <param name="search">index=_internal source=*metrics.log Component=metrics
group=${group$} | stats count by series</param>
  <param name="earliest">-6h</param>
  <param name="label">Series based on selected group</param>
  <param name="applyOuterIntentionsToInternalSearch">True</param>
  <param name="searchFieldsToDisplay">
    <list>
      <param name="label">series</param>
      <param name="value">series</param>
    </list>
  </param>
  <module name="ConvertToIntention">
    <param name="settingToConvert">series_setting</param>
    <param name="intention">
      <param name="name">stringreplace</param>
      <param name="arg">
        <param name="series">
          <param name="value">${target$}</param>
        </param>
      </param>
    </module>
  <module name="SubmitButton">
    <param name="label">Search</param>
    <!-- Chart for: index=_internal metrics NOT sendout group=<group> series=<series>
| timechart sum(kb) -->
    <module name="HiddenSearch">
      <param name="search">index=_internal source=*metrics.log Component=metrics
group=${group$} series=${series$} | timechart sum(kb)</param>
      <param name="earliest">-6h</param>
      <module name="HiddenChartFormatter">
        <param name="chart">column</param>
        <param name="chart.stackMode">stacked</param>
        <param name="primaryAxisTitle.text">Time</param>
        <param name="secondaryAxisTitle.text">sum(kb)</param>
        <param name="legend.placement">None</param>
        <module name="FlashChart">
          <param name="width">100%</param>
          <param name="height">200px</param>
        </module>
      </module>
    </module>
    <!-- Chart for: index=_internal metrics NOT sendout group=<group> series=<series>
| timechart sum(kb) -->
    <module name="HiddenSearch">

```

```

    <param name="search">index=_internal source=*metrics.log Component=metrics
group=$group$ series=$series$ | timechart avg(kbps)</param>
    <param name="earliest">-6h</param>
    <module name="HiddenChartFormatter">
        <param name="chart">line</param>
        <param name="chart.stackMode">stacked</param>
        <param name="primaryAxisTitle.text">Time</param>
        <param name="secondaryAxisTitle.text">avg(kbps)</param>
        <param name="legend.placement">None</param>
        <module name="FlashChart">
            <param name="width">100%</param>
            <param name="height">200px</param>
        </module>
    </module>
</module>
<!-- Chart for: index=_internal metrics NOT sendout group=<group> series=<series>
| timechart sum(kb) -->
    <module name="HiddenSearch">
        <param name="search">index=_internal source=*metrics.log Component=metrics
group=$group$ series=$series$ | timechart median(eps) max(eps) min(eps)</param>
        <param name="earliest">-6h</param>
        <module name="HiddenChartFormatter">
            <param name="chart">line</param>
            <param name="chart.stackMode">stacked</param>
            <param name="primaryAxisTitle.text">Time</param>
            <param name="secondaryAxisTitle.text">median(eps) max(eps) min(eps)</param>
            <param name="legend.placement">None</param>
            <module name="FlashChart">
                <param name="width">100%</param>
                <param name="height">200px</param>
            </module>
        </module>
    </module>
</module>
</module>
</module>
</module>
</module>
</module>
</module>
</module>

    <module name="SearchSelectLister" layoutPanel="panel_row1_col2" group="Search for cpu
intensive processors in the last 24 hours.">
        <param name="settingToCreate">name_setting</param>
        <param name="search">index=_internal source=*metrics.log Component=metrics
group=pipeline | stats sum(cpu_seconds) as totalCPU by name | where totalCPU > 0 | sort
-totalCPU</param>
        <param name="earliest">-1d</param>
        <param name="label">Name</param>
        <param name="searchFieldsToDisplay">
            <list>
                <param name="label">name</param>
                <param name="value">name</param>
            </list>
        </param>
        <module name="ConvertToIntention">
            <param name="settingToConvert">name_setting</param>
            <param name="intention">
                <param name="name">stringreplace</param>
                <param name="arg">
                    <param name="name">

```

```

        <param name="value">$target$</param>
    </param>
</param>
</param>
<!-- Add the processor -->
<module name="SearchSelectLister">
    <param name="settingToCreate">processor_setting</param>
    <param name="search">index=_internal source=*metrics.log Component=metrics
group=pipeline name=$name$ | stats sum(cpu_seconds) as totalCPU by processor | where totalCPU
> 0 | sort -totalCPU</param>
    <param name="earliest">-1d</param>
    <param name="label">Processor</param>
    <param name="applyOuterIntentionsToInternalSearch">True</param>
    <param name="searchFieldsToDisplay">
        <list>
            <param name="label">processor</param>
            <param name="value">processor</param>
        </list>
    </param>
<module name="ConvertToIntention">
    <param name="settingToConvert">processor_setting</param>
    <param name="intention">
        <param name="name">stringreplace</param>
        <param name="arg">
            <param name="processor">
                <param name="value">$target$</param>
            </param>
        </param>
    </param>
</module>
<module name="SubmitButton">
    <param name="label">Search</param>
    <!-- Chart for: showing the sum cpu_seconds for a given name and processor -->
    <module name="HiddenSearch">
        <param name="search">index=_internal source=*metrics.log Component=metrics
group=pipeline name=$name$ processor=$processor$ | timechart sum(cpu_seconds) </param>
        <param name="earliest">-1d</param>
        <module name="HiddenChartFormatter">
            <param name="chart">column</param>
            <param name="chart.stackMode">stacked</param>
            <param name="primaryAxisTitle.text">Time</param>
            <param name="secondaryAxisTitle.text">sum(kb)</param>
            <param name="legend.placement">None</param>
            <module name="FlashChart">
                <param name="width">100%</param>
                <param name="height">200px</param>
            </module>
        </module>
    </module>
</module>
</module>
</module>
</module>
</module>
</module>
</view>

```