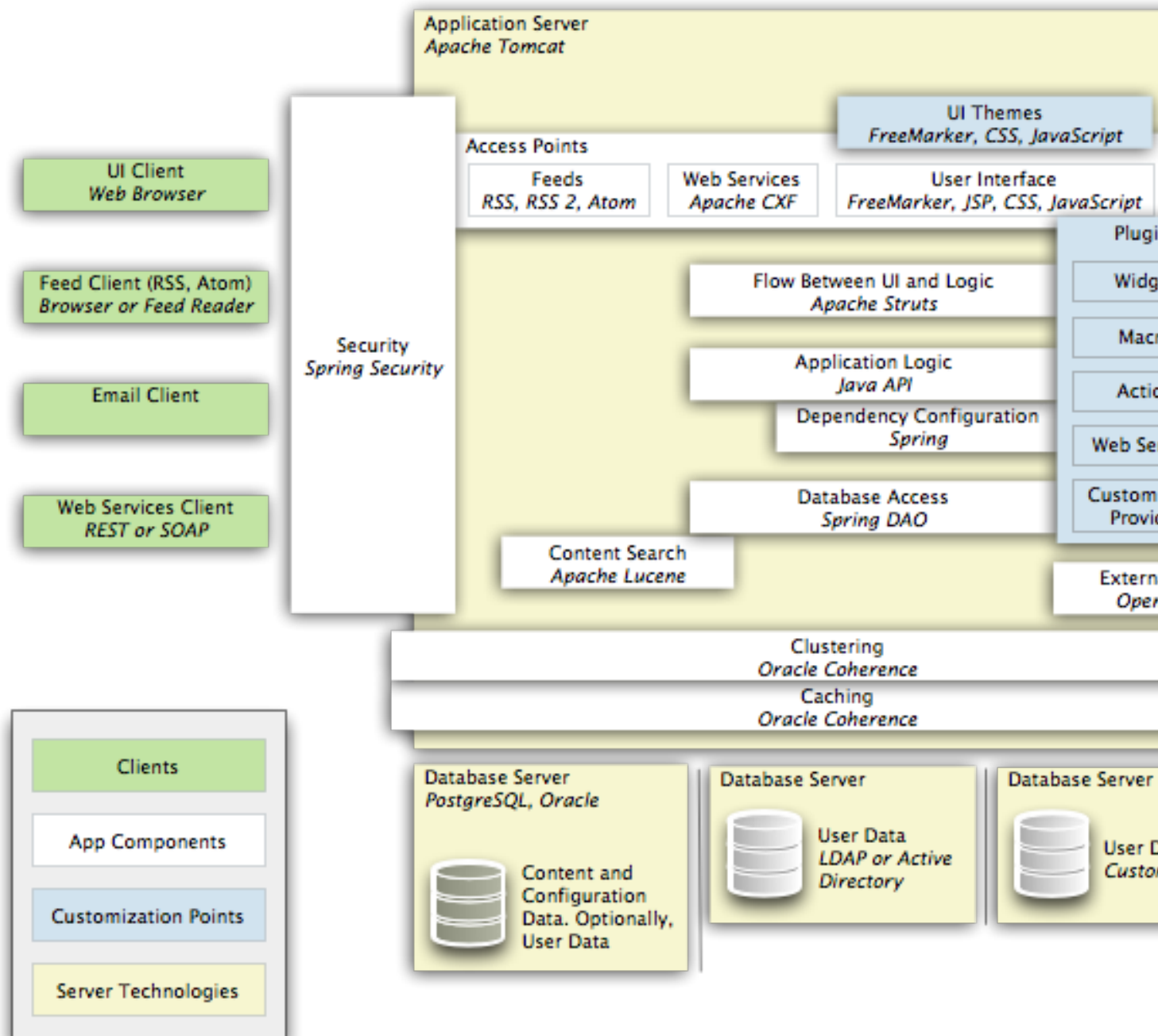


# Application Architecture

The high-level overview describes the technologies that make up Jive SBS. The following illustration lists the key pieces; you'll find descriptions below.



## Access Points

### User Interface

The user interface is rendered from FreeMarker or JavaServer Pages that are essentially templates to display user and content data. Here's a description of the pieces:

- [FreeMarker](#) template (FTL) files. FreeMarker is used for the main user interface. Each FTL file is a template from which a particular part of the user interface is created -- the list of Actions or the area where a blog post is displayed, for example. [FreeMarker is an open source project](#) hosted by [SourceForge](#).
- [JavaServer Pages](#) (JSP) files. JSP is used to render the admin console user interface. JSP was created and is [maintained by Sun Microsystems](#).
- [Cascading style sheets](#) (CSS). CSS styles are used in FreeMarker and JavaServer Pages files to provide positioning of UI element, colors, text styles, and more.
- [JavaScript](#). JavaScript is used to enable certain effects, such as displaying popups and dragging widgets.

### Supported Browsers

For the latest information on supported browsers, see the system requirements in the administrator documentation.

### Customizing

People using the application can customize or personalize the UI by using widgets. Developers can customize the user interface through either themes or plugins.

- Widgets are views on content that people can use to create customized versions of community pages. For example, a widget can display recently changed content or the results of an RSS subscription. For more on widgets, see [Designing Pages with Widgets](#) .
- In a theme, you use custom versions of FreeMarker files and CSS style rules to replace those included. You group customized files into a theme, then map the theme to a part of the application, such as a space or UI reached via a particular URL. You can also merely replace certain words in the user interface, such as "article" for "document." You can't include your own Java code in a theme. For more on developing themes, see the section on themes in this document.
- In a plugin, you can customize the UI by including new commands in existing user interface pages. For example, with an action plugin you can add a new link to the Actions list or a new command to the user menu bar. You can also develop new widgets in a plugin.

### Feeds

[Feeds](#) are a data format through which people can keep up without needing to view the application itself. People subscribing to feeds can view the feed information as a list of content in a [feed reader](#) or web browser.

## Email

A community can send or receive email. It sends email when people choose to receive notifications on particular content, such as when a document has been edited or a discussion has received a new reply.

People can send email to the application by replying to notification messages. For example, if someone receives a notification that a discussion has gotten a new reply, they can reply to the notification message, type their own response, then send the email to have their typed response posted as a new reply to the discussion.

The application can also send email for common reasons, such as to follow up on a request to change a password.

## Web Services

Other applications can access functionality using [web services](#). Most of the application's features are available this way.

The application supports web services in [REST](#) and [SOAP](#) styles. Support for these is provided through [Apache CXF](#), an open source framework.

For more on getting at functionality via web services, see [REST Web Services Reference](#) and [SOAP Web Services Developer Guide](#).

## Flow Between UI and Logic

The application is based in part on what's known as a model-view-controller (MVC) framework. In MVC, a controller layer manages interactions between what people do in the user interface (the view) and how the application's logic (the model) responds. The controller layer is provided by [Apache Struts](#) (the view is FreeMarker or JavaServer Pages, while the model is implemented as Java classes). Application logic responds to the user interface

through Struts actions that return results that depend on the nature of the request. [Struts is an open source framework.](#)

## Customizing

Developers can write their own Struts actions, usually accompanying user interface additions. Actions are written as plugins.

## Application Logic

Application logic is implemented as Java classes. This includes classes for representing the various containers (such as projects) and content types (such as documents), as well as managers for handling them, data access objects for interacting with the database, and classes to support services such as web services, feeds, and so on.

Developers can learn more about the classes exposed as part of the API by taking a look at the [Javadoc](#).

## Dependency Configuration

Java classes rely on many classes and libraries provided either by Jive or by third parties. These dependencies are resolved at run time by using a model called "[inversion of control](#)," or [IOC](#). IOC is supported by the [Spring framework](#). Through the Spring framework, developers can replace default dependencies with their own implementations. For example, a developer could replace a particular security framework with their own.

[Spring](#) is an open source framework.

## Search

People can search for content, of course. The search feature can also be configured to search external content sources. An administrator determines which content should be available for external searches.

## Content Search

For searches of content inside the community, the application uses the [Apache Lucene](#) search engine. [Lucene is an open source tool.](#)

## External Search

When searching content outside the community, the application uses OpenSearch. Other applications that expose their search engines via OpenSearch can be included in community searches by an administrator.

## Caching

Caching is provided by Coherence.

## Clustering

Clustering is provided by Coherence. For more information, see [Clustering Information](#).

## Database Access

The application stores data about content and users in its database. It can also be configured to use another data source, such as an LDAP database, to authenticate user accounts when people log in.

## Application Database

The application database is designed to store content data (such as content, properties for widgets, application settings, and so on) as well as user data (including the contents of user profiles). The application can optionally use its own database to store credentials for authenticating users. See [Application Database Schema](#) for more information.

## Analytics Database

Using analytics (an optional module), community managers can capture and analyze data about their community's use. This database is modeled as a star schema of the sort used by other applications that support data warehousing. For more information, see [Analytics Database Schema](#) or the [Analytics Data Model](#).

## User Data

When installing Jive SBS, you can specify that it should use an external data store for getting the list of users and groups who will be using the application. The application can use this data store to authenticate people when they log in. By default, LDAP and Active Directory are supported.

Note that even if the application is using an external data store for authentication, information about users (profile information and permissions, for example) will be kept in the application database.

## Customizing

Developers can add tables to the application database and access them from within plugin code.

## Security

Authentication and authorization are based on the Spring Security (formerly Acegi) framework. For details on the implementation, along with information on how to customize it, be sure to see [Authentication and Authorization](#).

## Customizing

Developers can customize the security framework by adding J2EE filters through a plugin. [Authentication and Authorization](#) describes what you can do and offers sample code.

## Customization Points

Developers can customize the application, changing its look and feel, adding new widgets or macros, even adding new features. Most of this kind of work is done by developing a plugin that's based on the same technologies that the application itself uses.

## Themes

With themes, you can restyle the user interface. In this way you can add a look and feel that identifies the community with your brand characteristics (logo, colors, and so on). You can

also restyle particular parts of the application, such as to give certain kinds of content or certain spaces their own character. Themes don't require a plugin.

When developing themes, you can take one of three approaches: basic changes through the UI, minor changes to code by [Customizing UI with the Theme Resource Kit](#), and [Advanced Themes Topics](#).

## Plugins

A plugin is the most common way to customize and extend the application.

## Widgets

Widgets are views on content with which people can customize certain pages. Jive SBS includes many widgets by default, but developers can build their own using the plugin framework. [Building Widgets](#) is a good place to start.

## Actions

Building an action plugin is one way to add more involved functionality. For example, a custom action could be behind a feature through which people could choose a number of documents to package into a single PDF file. For more, see [Building Actions](#).

## Custom Authentication Provider

A custom authentication provider knows how to authenticate people against a data source that isn't supported by default (in other words, that isn't the Clearspace database, LDAP, or Active Directory). For more on customizing authentication, authorization, and providing single sign-on, see [Authentication and Authorization](#).

## Server Components

### Application Server

Jive SBS runs on Apache Tomcat. For more information see the [system requirements](#).

## Database Servers

The [system requirements document](#) lists details about the supported DBMSes.