
wingS Whitepaper

1. Introduction

wingS is a Java servlet based toolkit which presents a high level windowing API to the user. Applications created with wingS can be deployed in a standard java servlet container to present a web based interface. Where possible interfaces and concepts are reused from the Java Swing API to minimise the learning curve. A programmer with basic Java knowledge should be able to write their own applications in wingS after a few hours' familiarisation.

The wingS framework takes care of the low level infrastructure needed to manage a Java servlet session, relieving the need to do any special state management. The framework also handles the rendering of the user interface in HTML, and the mapping of HTTP requests into high level events. Thus, apart from being aware of the general constraints imposed by HTTP/HTML, the programmer does not need to be concerned with the messy details of HTML formatting. Instead they deal with a high level windowing API with methods to open windows, layout components, and receive component events. This is in direct contrast to the majority of technologies being used to create web applications which require direct HTML-level formatting at some point.

This clean architecture of wingS makes it suitable for the construction of complex, transactional applications. This is not to say that it cannot be used for simple applications, it has a small footprint and an easy learning curve, so is suitable for any type of web application. However the ability to correctly structure and layer your code will be most beneficial in large applications, where more typical stateless approaches collapse due to the poor structure dictated by the request-/ response- scheme.

As of wingS 3.0, AJAX is used in the background in order to update the page incrementally. This enriches the user experience dramatically while it is totally transparent to the application developer. In fact, an application written with wingS 2.0 will leverage the AJAX capabilities immediately, if only the old JARs are replaced with the new ones. Even custom components will take part in the process of incremental updates without any special treatment.

Apart from requiring a Java servlet container to execute, wingS is non-invasive and imposes no particular technical constraints. The display is always browser based and hence "thin-client", however you can implement many different architectures such as:

- Monolithic "fat client", just code it all as in-process java
- J2EE application by using other standard J2EE facilities
- Classic multi-tier "thin client" by interfacing with a server middle-tier using the technology of your choice (SOAP, CORBA, RMI). Java natively offers plenty of choices

The server-side of a wingS application is inherently cross-platform being pure Java. On the web browser side wingS tries to be as inclusive as possible by adhering to W3C standards, and not requiring any special client-side plug-in. More adventurous users can customise the display rendering if required, using internal APIs, and work is on-going to better modularise these aspects of the toolkit.

wingS applications have been deployed for several years in a variety of industries. The code is stable and mature.

2. Why Choose wingS?

A summary of reasons you might consider wingS for your next project follows.

Clean Architecture

The majority of the API is in Java and presents a high-level interface using familiar concepts such as windows, widgets, and events. Thus you can structure your code in the way you want, rather than the way the technology forces you to. A portion of the toolkit uses cascading style sheets and JavaScript. You are unlikely to be interested in this unless you intend customising the look and feel.

Open Architecture

Use of Java allows a cross platform "back-end", and reliance on basic web standards allows a cross-browser "front-end". No special plug-ins are required.

Open Licence

wingS is licenced using the GNU "LESSER GENERAL PUBLIC LICENSE" (LGPL) which means you can freely modify the source code, as well as safely use it for your own proprietary development.

Stable and Robust

wingS has been in use in commercial production environments for several years. These include vertical domains such as retail, manufacturing, logistics, financial services and local government.

Secure

Since wingS is a pure Java Servlet toolkit, deployment is very clean, this normally being a matter of copying over a few jar files, or a war file. This makes it harder to reverse engineer the application from outside a fire-wall, unlike solutions which rely heavily on scripting. Additionally the way that state management is implemented means that an application that needs to be transactional, will behave so. It is not possible to bookmark an intermediate point in an application's execution, and then later replay from this point. HTTP parameters are posted, rather than URL encoded. If you combine this with a standard SSL connection an application will be very hard to compromise.

Efficient

Use of programmatically generated HTML tends to result in HTML pages that are more efficient to download (requiring fewer round trips to the server) than pages produced by many HTML editors. wingS applications function well over a low bandwidth connection. The toolkit itself is moderately sized and has a modest memory footprint by the standards of typical Java applications.

Scalable

Nothing in the wingS toolkit prevents you from writing applications that can "scale out" to many concurrent users. The scalability limits are likely to depend on the other technologies you use in your application. The stateful architecture by its very nature preserves as much context information as possible between requests, thereby taking load from the business logic and persistence store.

Easily Integrated

wingS uses standard Java Servlet sessions, so if you need to integrate other front-end technologies you can still access the normal Java Servlet API and access data bound to the same session. This should allow its use as part of a larger portal if required.

3. Deployed User Base

Here's a list of companies volunteering to share with you what they have developed.

Table 1. wingS Applications

Application	Company
Financial management application for marketing programs in EMEA	eXXcellent solutions gmbH [http://www.excellent.de]
Research system for electrical wires, connectors and components in the automotive area	
Project and Resource management application	
eCommerce solution, (B2B, B2C)	mercatis information systems GmbH [http://www.mercatis.de]
Corporate LDAP management application	
Web directory [http://www.innovationsregion-ulm.de]	
Warehouse dispatcher software, PDA-based (wireless)	
Retail investment management performance reporting (intranet and extranet). Used by a group company of Close Brothers Group PLC, a FTSE250 company.	Scinapps Ltd [http://www.scinapps.com]
E-Procurement Suite	Wilken GmbH [http://www.wilken.de]

Please contact the project team, if you want a reference to be added.

4. Technology Comparison

4.1. Comparison with JSF

Comparison of wingS with Java Server Faces (JSF), and to a lesser extent Java Server Pages (JSP).

Reasons to Choose JSF

- It is backed by Sun.
- Your application can be structured as a simple pre-determined sequence of pages

Reason to Choose wingS

- You want to program in a single high level language. JSF spreads the application logic across multiple technologies - pure Java, tag-libraries, XML based action configuration and JSP pages.
- You don't want to learn a new syntax for writing HTML (via the JSF tag-libraries), or for binding the tag-library directives to Java code.

- You want more insulation from the low level details of HTML rendering
- You prefer compile time errors to debugging runtime errors. The decoupled structure of the various modules of JSF creates a heavy burden of runtime testing.
- JSF requires extra state management on your part
- Your view layout is more complex, or more dynamic, than can be easily represented using a "static" JSF file specification.
- Your application logic is more complex than can be easily handled using the simple navigation specification mechanism of "action files"
- All debugging can be handled within java code
- You are already familiar with the Java Swing API and related concepts

JSP Comments

JSF was conceived to address the severe shortcomings of JSP. However by taking this as a starting point it inherits most of the negative aspects of programming with JSP. Most of the disadvantages quoted above apply equally to JSP. The current author once spent two months trying to create a simple JSP based application, using tag-libraries and XSLT processing to format the output. The end result was entirely unsatisfactory, and was replaced by a wingS application in a week.

4.2. Comparison with Scripting

The comments here apply to general scripting solutions. This includes Perl-CGI, Python, ASP or ASP.Net. (Strangely Microsoft did not take the opportunity with .Net to provide a high level web API to a web interface. Instead they enhanced ASP (Web Forms), leaving many of the disadvantages, created a new pure windows client library, and require yet another technology for mobile devices).

Reasons to Choose Scripting

- Quick solution for small applications
- Intermediate technology level for less technical implementors

Reasons to Choose wingS

- Single technology
- No embedding and data escaping issues
- Better encapsulation and scoping of modules
- Compile-time checking instead of runtime checking
- Scripting requires major cut and paste of HTML templates
- HTML coding is low-level and too device / browser specific
- Scripting state management solutions tend to be clumsy

- Integration of script with other application code can be tricky. Database bindings are usually available, but if more data processing is required, this is hard to add.
- No special session state management

5. Standard Compliance

wingS is based on Sun's Servlet API version 2.2 or later. It uses Jakarta Commons Logging. Optional features require Direct Web Remoting (DWR), HTTPClient for session recording and playback and bsh-core for scripting support inside template files. There are no other dependencies.

Due to this non-invasive nature wingS is relatively light-weight and can easily be used in combination with other Servlet-based technologies.

6. Principles for Future Development

The wingS team recognizes and subscribes to the following principles:

- Swing API and behaviour - adhere to them as closely as possible and where appropriate
- W3C standards
- Monitor and embrace new developments in the web application arena