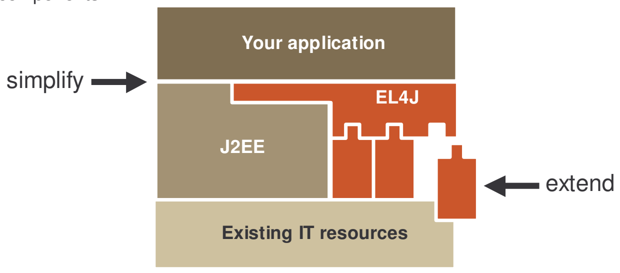
ELCA

EL4J Datasheet

What is EL4J?

The EL4J architecture toolbox simplifies and speeds up the development of J2EE applications.
EL4J is a collection of resources to get the most out of existing J2EE tools. At its core, **EL4J uses the popular Spring Framework**. It adds other common open frameworks and it complements these frameworks with samples, documentation and improvements.

J2EE is a very powerful development platform, with an enormous collection of tools and frameworks. Unfortunately, the price to pay for this vast set of facilities is *complexity*. **EL4J makes it easy to use best-of-breed J2EE technologies** by packaging and pre-configuring a set of leading components. The central component of EL4J is the popular *Spring Framework*, which integrates and configures the other components.



Basic idea: simplify and extend the J2EE

What are the benefits of EL4J?

Developing J2EE applications with EL4J has the following benefits (compared to development with pure J2EE or Spring):

- Get **up to speed quickly** with state-of-the-art application development due to pre-existing technology choices, a model architecture, and **application templates**. Design debate is reduced to the essential and you use the existing best-practices.
- Go with the **winning technologies**: the recommended open frameworks are de-facto standards. Profit from this commoditization of J2EE frameworks and follow the **managed evolution** of EL4J.
- The **lightweight approach** let's you get to the point of your problem immediately, without technology hassle. You work directly with **POJOs** (Plain Old Java Objects), leading to fast development round-trips, simplified tests, and minimal technology lock-in. Migration to the new EJB 3 standard is also simplified as it uses the same approach.
- Effectively **separate business concerns** – developed with standard Java technology – **from technical concerns** - solved orthogonally via interceptors and other AOP mechanisms.

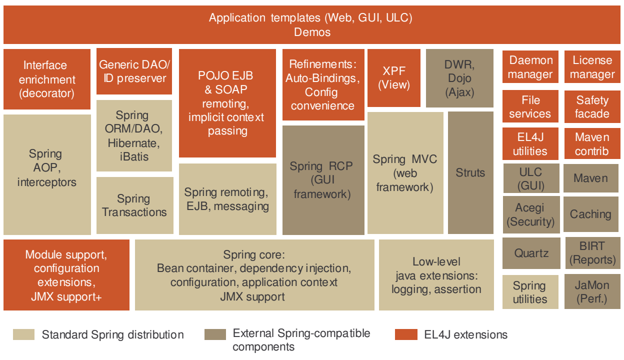
We originally created EL4J for ourselves, to support internal and external projects. Within ELCA, **it has rapidly become the standard approach for J2EE enterprise applications**.

What are the parts of EL4J?

The following standard components are part of the EL4J architecture toolbox:

- Spring Java Framework 2.0
- Hibernate or iBatis
- Maven 2.0, JUnit, JWebUnit
- Commons logging and log4j
- The Spring Security (Acegi) Framework
- Spring Rich Client Platform (Spring RCP)
- The extensions that are part of EL4J

In the following we describe the features of the EL4J architecture toolbox with all its included components. The advantage of EL4J lies in the selection of components, their seamless integration, and punctual improvements.


Application templates (Web, GUI, ULC)
Demos
Interface enrichment (decorator)
Generic DAO/ ID preserver
POJO EJB & SOAP remoting, implicit context passing
Refinements: Auto-Bindings, Config convenience
XPF (View)
DWR, Dojo (Ajax)
Daemon manager
License manager
File services
Safety facade
EL4J utilities
Maven contrib
Spring AOP, interceptors
Spring ORM/DAO, Hibernate, iBatis
Spring Transactions
Spring remoting, EJB, messaging
Spring RCP (GUI framework)
Spring MVC (web framework)
Struts
ULC (GUI)
Maven
Acegi (Security)
Caching
Quartz
BIRT (Reports)
Module support, configuration extensions, JMX support+
Spring core: Bean container, dependency injection, configuration, application context JMX support
Low-level java extensions: logging, assertion
Spring utilities
JaMon (Perf.)
Standard Spring distribution
External Spring-compatible components
EL4J extensions

The components that make up EL4J

What are the features of EL4J?

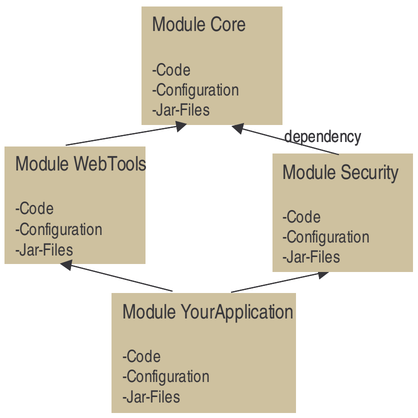
Transparent remoting: all client-server interactions between logical tiers are modeled as method invocations on normal POJOs. POJOs can be either co-located with the client, or deployed remotely in the EJB or Web server or in a standalone application. This is transparent in the code, which treats all invocations in exactly the same way, whether they are in the same process or not.
Advantages: simpler development and better maintainability (local services can be redeployed as remote without changing the client code), agility and investment protection through interoperability with SOAP, CORBA and .NET.

Extensibility and adaptability: in order to cope with the fast pace of change in business requirements, environment and technology, Spring incorporates several extension and adaptation mechanisms. EL4J complements them where appropriate:

- **POJOs can be added, changed and removed, conveniently via configuration**, by simply adapting a running sample configuration. A service is automatically enabled when you add its jar file to the class path; there is no further need for configuration.
- **Interceptors** can complement the behavior of existing services (with Spring's AOP). There are predefined interceptors (e.g. for authorization, tracing or statistics collection), and you can add your own. Interceptors can even be added at run-time, allowing you to intervene in a live application to measure or correct issues.
- **Optional context information** (for example the security principal, the request-ID for performance measures, or the transactional context) can be **passed implicitly with every remote method invocation**, keeping such data out of your business interfaces. This allows also sharing a same application instance between multiple organizations, by passing the principal of the concerned organization with each request.

These features keep the core of EL4J simple, while providing far-going extension potential and giving more power to the developer. **Advantages:** protection against change of external standards and implementations and improved systems agility.

Configuration service: complex applications usually require configuration information. EL4J is based on the simple yet flexible configuration model of Spring. It makes the model of spring even more modular and extensible through patterns for default configurations.
Advantages: get started more easily, better configuration manageability.



A sample set of modules with their dependencies.

Modular build system: Through the abstractions of Maven 2.0, the EL4J code is **split into modules** that can be combined in different ways; each distribution includes only the essential modules. **Dependencies between modules are transitive**: this means you import automatically all the modules your parent modules import. Settings for your environment (database, application server, etc.) are abstracted. We provide seamless integration with Eclipse and JUnit. **Advantages:** modularized and leaner applications, better support for multiple projects with differing needs and lifecycles with very light and standard technologies.

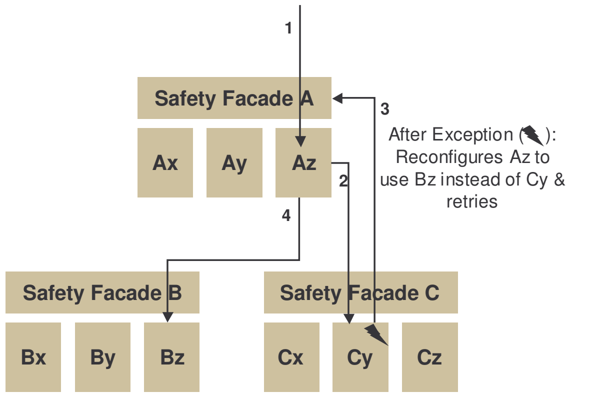
Security service: In order to provide a more flexible and unified model than the JAAS/ J2EE security, EL4J integrates the **Acegi security framework**. Its security model works for the web, simple domain objects and applications that span multiple processes.
Advantages: a more flexible and unified security model; shared security infrastructure reduces operational costs and minimizes the risk of inconsistencies.

Performance measurement: With the proven **Jamon** tool, services can be instrumented at run-time to collect performance information; for tuning or to monitor response times during operation. Performance measurements are displayed via a web interface and via JMX. **Advantages**: on-demand measurement eliminates impact in normal operations, bridges gap between network-level and code-level profiling.

Web applications: EL4J recommends the proven **Spring MVC** framework. It includes an end-to-end template application that helps getting started quickly. **Struts** is an alternative framework when the context demands it. The frameworks are complemented with libraries for Ajax and efficient web presentations.
Advantages: speed up web developments, promotes use of current best practices (e.g. clear separation of presentation and business logic).

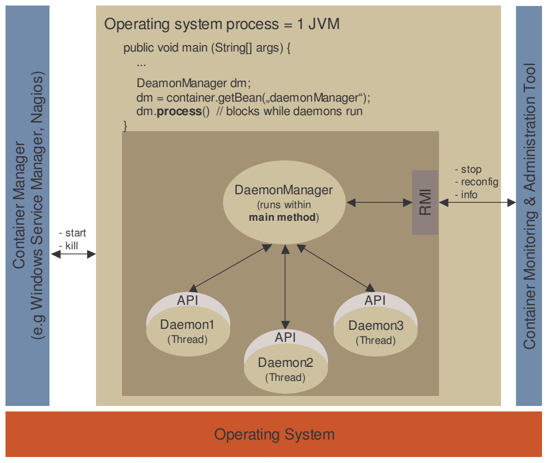
GUI applications: For quick development of professional Swing applications and to adhere to the best practices, EL4J uses the emerging **Spring RCP** framework. EL4J extends and packages the Spring RCP and provides a cleaned-up demo application. Your GUI only needs to declare where its canvas is different from a default GUI canvas, then you add your custom dialogs and views, mostly by extending existing components. **Advantages:** get up to speed quickly with robust and good looking Swing applications, profit from current best practices.

Manageability: Additionally to the **JMX** support of Spring, EL4J can automatically expose all spring beans and their configurations via the Java Management eXtensions (JMX). **Advantages:** gain an instant understanding of the current Spring configuration, seamless and vendor-independent integration in the existing IT-management infrastructure.


1
Safety Facade A
3
After Exception ( ): Reconfigures Az to use Bz instead of Cy & retries
Ax
Ay
Az
2
4
Safety Facade B
Safety Facade C
Bx
By
Bz
Cx
Cy
Cz
The safety facade externalizes error handling and allows for dynamic reconfiguration after exceptions

Safety facade: A safety facade **handles all the technical exceptions for business code**, such that for a user of a component, a business operation either succeeds or completely fails. The facade takes the responsibility of treating abnormal situation away from normal business code. **Advantages:** shorter and centralized exception handling code, clean separation of concerns between business and technical code.

Daemon Manager: The light daemon manager **executes and supervises multiple long-running daemons,** supervised Java thread. The JVM with the daemon manager is typically launched as an operating system service and can also be managed from remote.
Advantages: Implement very reliable daemons with Java and administrate them from remote.



```
public void main (String[] args) {
    ...
    DeamonManager dm;
    dm = container.getBean(„daemonManager“);
    dm.process()  // blocks while daemons run
}
```

Typical use of the daemon manager

Other features: other components complement EL4J with reporting, persistent task scheduling, file operations, a license manager, caching, support for generic DAOs, or with XML merging.
Advantages: Ready to use, well-integrated components for various domains cut the time you need to get started.

More information

http://www.elca.ch/Solutions/Technology_Frameworks/EL4J/EL4J.php

Most of our extensions to Spring and the other components have been published in the open source under http://**EL4J**.sourceforge.net. For a list of distinctive features of EL4J we refer to http://wiki.elca.ch/twiki/el4j/bin/view/EL4J/FeaturesOfEl4j .

Professional support in English, German and French is available from ELCA.

With EL4NET and IIOP.NET ELCA provides similar libraries for other environments as well.