# 

# Health XCEL Platform Tech stack



Tech stack 1

Overview 3

GUI Layer 3

Web Layer 4

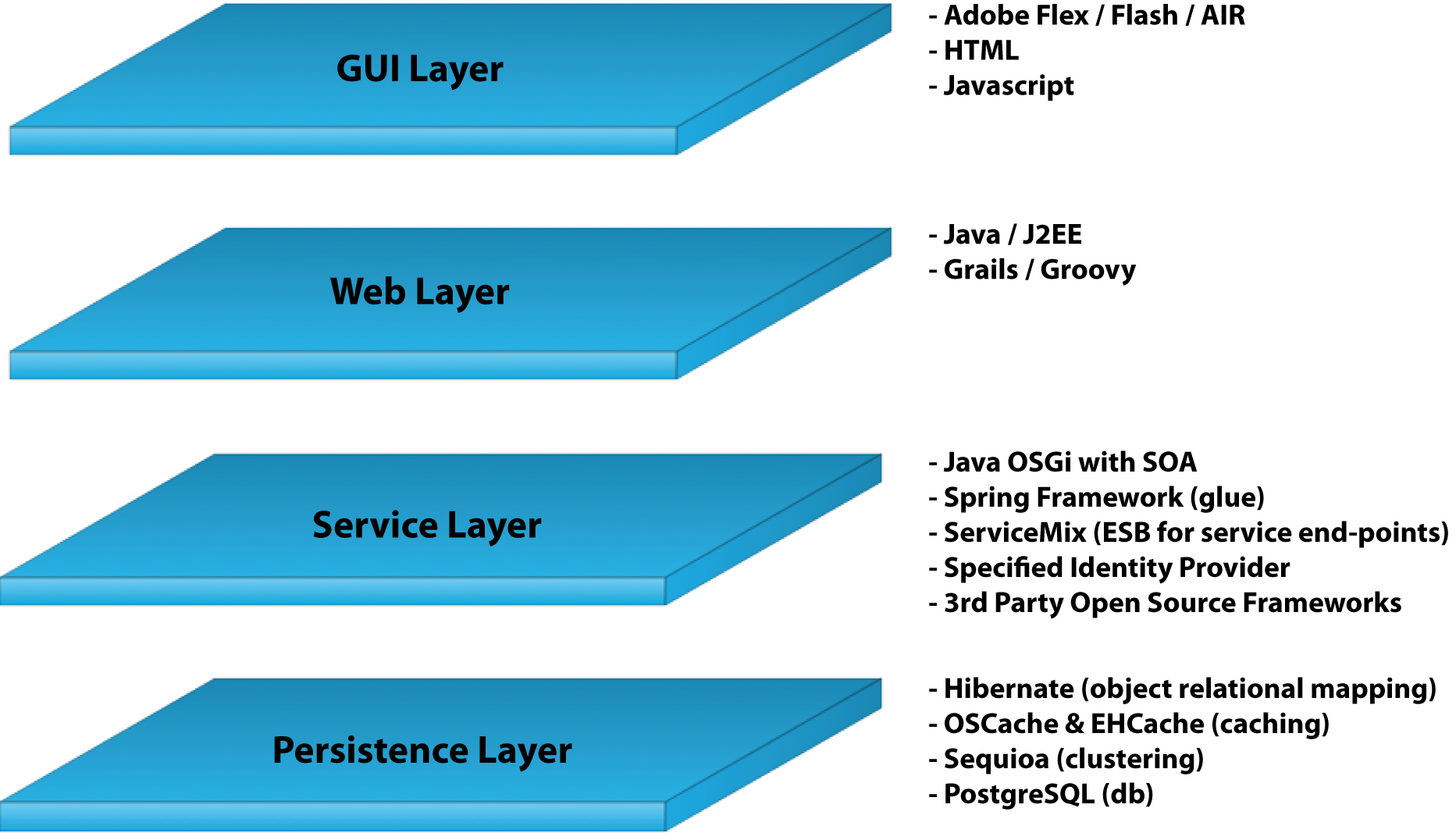
Service Layer 4

Persistence Layer 4

## Overview

Our platform consists of the elements specified in Figure 1. This is a typical N-Tier application software structure created in a modular fashion, with its basis in a Service Oriented Architecture that assumes there are many data providers and data consumers who would like to consume the data differently. We serve up the data by leveraging different protocols and document standards, such as XML, E-mail, FTP over an Enterprise Service Bus.

The GUI and Web layer were created for rapid prototyping and great visual interaction. The service layer and persistence layer live on top of a lightweight OSGi container that supports swapping and versioning of libraries at runtime.



Figure

## GUI Layer

The end-user will be presented with a graphical user interface based on proven, successful technologies from Adobe. Flex and Flash have become synonymous with Rich Internet Application (RIA). It gives the end-user a sleek and appealing interface with which to discover information, play games, collaborate on projects and allows participants to do videoconferencing. Developers of web applications find working with this technology a lot easier and faster than previous web technologies as it comes pre-packaged with existing components, such as well-know HTML form-elements. Developers also get to work in an application environment that is “state-full”, which gives them more freedom to focus on the application and less on the nuts and bolts of how to pass variables over from one page to the next. Flex and Flash supports communicating to its data providers in many different ways, which makes it the ideal candidate for integration work. They understand the AMF binary protocol, XML over HTTP and Web Services using SOAP.

## Web Layer

The graphical user interface needs a place to “live” and that is where the web layer comes in to play. Grails is a rapid prototyping framework based on the best Java open-source tools available such as Spring, Hibernate, AspectJ, Groovy, Sitemesh, Javascript libraries Prototype, Dojo, jQuery and others. It enables the developer to quickly create new web pages without having to focus on the nuts and bolts of choosing and putting together the right technologies. The code is written in Groovy, which is a Java scripting language that compiles at runtime. This not only speeds up developer testing time as creating a Java web application is time consuming because it has to compile its sources every time there is a change, it also gives the developer a wide array of convenience methods that solves 99% of the hurdles usually associated with developing a web application.

## Service Layer

The service layer is the core of the platform. It is created based on a Service Oriented Architecture that assumes there are several consumers of a single service. Therefore, the services need to be able to speak the language of the consumer, be it XML, SOAP over REST or Web Services etc. An example of a service could be the ability to send emails. Many other services would like the ability to send emails but don’t want to write that specific code themselves. Instead, there is one email service that the rest can leverage.

It’s in the service layer that integration work with 3rd party vendors take place. Services can be written to talk to external legacy systems or create new functionality from scratch. An example of an external integration would be to have an identity provider service that can provide single sign-on (SSO) capabilities for the whole system. The identity provider is not located on the same system but the identity service that is set up to talk to the provider makes the connection and handles the authentication that can then be used by all the other services.

The next piece of technology is probably the most useful. All our services reside on an OSGi micro-kernel. The micro-kernel allows all services to be deployed and un-deployed at runtime. This means there is no need to restart the server when new functionality is added which increases server up-time and reliability. OSGi also enables services to be updated, either through versioning or hot-swapping. This means that if a vendor deploys the same service, but with a new version, other services can immediately start using the new version of the service or continue using the old service. It also means that an existing service that might have a bug in it, can be swapped with a service that removed that bug without having to restart the server. This is groundbreaking technology for anyone who has ever worked with maintaining project dependencies and supporting multiple versions of the same library.

## Persistence Layer

The persistence layer is based on best practices for handling large relational databases in an efficient and reliable manner. From the top, Hibernate is the Object Relational Mapping (ORM) tool of choice for every Java developer. It handles retrieving, caching and persisting complex object graphs to a relational database. Hibernate leverages OSCache and EHCache and a plethora of other caching providers. Caching providers enabled frequently access data to be stored in memory for quicker access, thereby speeding up the application.

Between Hibernate and the database lies a clustered database driver that handles persisting data across multiple databases. In case one server goes down, the data is still accessible from other servers.

Finally, PostgreSQL is the database of choice when creating more complex applications but still keeping cost down. PostgreSQL has always been preferred by the science and educational community and non-profit organizations for its reliability, sturdiness and feature-set.

## More information:

* Health XCEL, Inc.: <http://www.hxcel.com>
* Adobe Flex: <http://www.adobe.com/flex>
* Grails: <http://www.grails.org>
* OSGi: <http://www.osgi.org>
* SOA: <http://en.wikipedia.org/wiki/Service-oriented_architecture>
* Hibernate: <http://www.hibernate.org>
* EHCache: <http://ehcache.sourceforge.net>
* OSCache: <http://www.opensymphony.com/oscache>
* PostgreSQL: <http://www.postgresql.org>