eller

Softshell

Application framework

Technical whitepaper

01.01.2023

About Softshell

Softshell is a fully integrated development and production platform for building standards-based web applications. It has been designed for enterprise-grade deployments with high expectations on security, performance and scalability.

The platform was created to support modern rich user interface experience in a manageable, structured way.

The idea behind Softshell utilizes a different way of web page generation. Instead of producing a complete page with data and formatting on the server side, Softshell translates application definition files into dense JavaScript, CSS and HTML5 code and sends it to client on demand, as soon as user might want to need it. The code then generates the display contents directly in the browser.

This approach allows incredible responsiveness of the application interface, combined with faster page loading and server utilization decrease. Softshell supports real-time data updates, automatically distributed to clients, opening new ways of designing collaborative applications.

Applications are defined with XML-formatted templates, which describe presentation and behaviour of each page or component. Templates may be reused and stacked into complex components, decreasing development time and improving application testability. All complex logic is hidden inside of the framework; developers only work with logical objects and parameters.

Softshell uses and supports commonly known libraries such as jQuery, LESS and some others, allowing applications designers to reuse existing modules and web controls.

In addition to the core template processing system and functional libraries, Softshell contains a comprehensive stack of tools and methods to fully support end-to-end process of application analysis, building, testing and operation.

Document index

Framework main highlights

Softshell was built to address the following challenges:

- Bring fast, reliable, error-prone client application development to reality
- Build applications with almost no coding on top of a cross-platform high-level component-based architecture
- Achieve architectural simplicity and governance with reusable components
- Reduce cost if maintenance when upgrading to newer versions or changing the functionality

Product scope

Softshell is a comprehensive product, containing wide-range functionality to cover various enterprise customers’ requirements. Basic framework consists of the following packages:

1. Application server software and utilities for rapid development, testing and production operation
2. Templating framework with templates pre-processor
3. Dynamic data access layer
4. Event based scripting engine
5. Set of provided UI components
6. Remote function call interface
7. Cross-browser interaction libraries
8. Content handling libraries
9. Governance tools and policies
10. Localization tools
11. Security framework and tools
12. Report-generation using embedded BI
13. User friendly data import and export
14. Integration API and 3rd party adapters

These packages are bundled as one deployable Java servlet container application and seamlessly work together.

Platform architecture

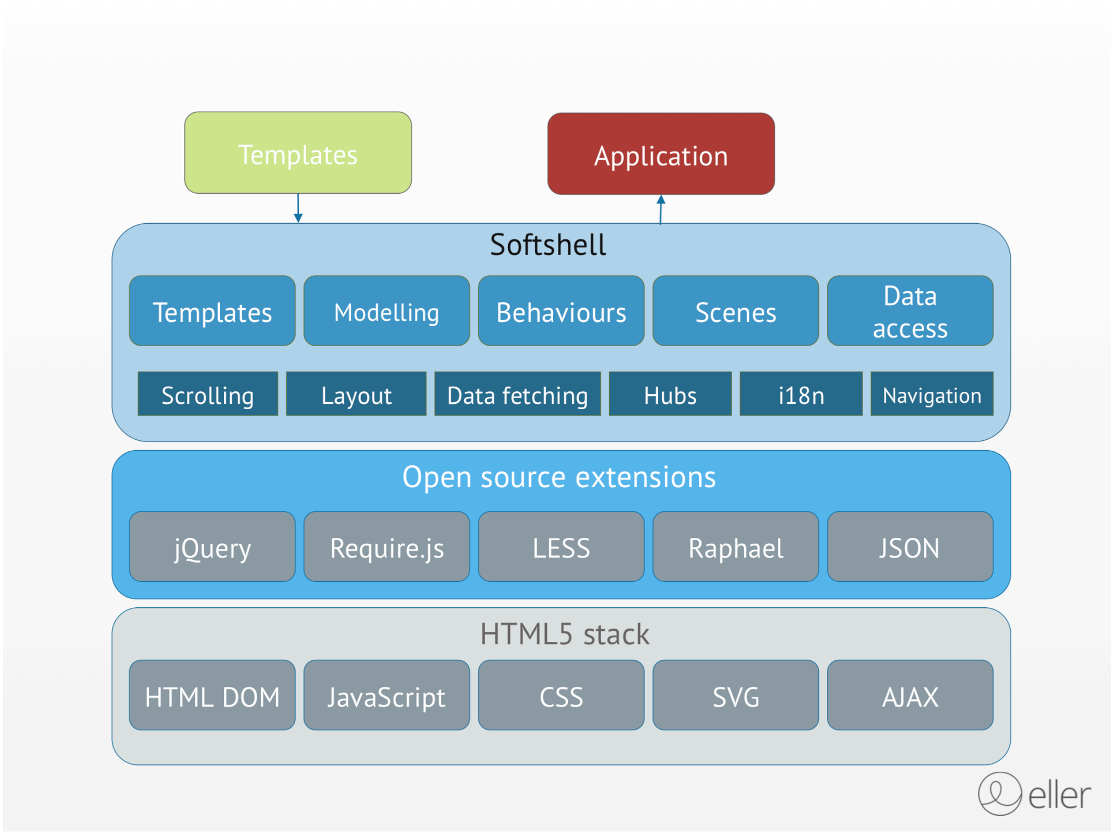

Figure 1 - Softshell architecture

Platform components are split between server and client parts. Server-side components take care of templates processing and are aware of client context. Special tracking mechanism maps preloaded components with each browser window and tab and make sure new code is delivered just in time of next request.

Client and server side both are using same JavaScript libraries, so it is possible to reuse same functions when using a template to build a UI screen or an offine report. Client and server components may use Data Access layer in the same way, ensuring deterministic results of calls with same arguments.

There are different kinds of predefined template types, making it easier to prepare application for various screen types and workflows, without unnecessary duplication of code.

“Scenes” combine multiple templates – (views and controls) to show a complete screen picture to the user. Scenes may use some of preloaded “layouts”, defining layout of components on the screen and their behaviour.

“Behaviours” allow very complex UI functionality to be enabled from an application just by declaring expected behaviours. New behaviours may be added by customer and reused across components.

Scalability

Client response time is dependent on size of data records, fetched from the server. There is no classical “page generation time”, since pages are not build on server. As user clicks forward through the application, response time decreases as browser is caching more and more of application code locally.

Role separation

There are different roles in a modern application development process. By separating application logic, component coding and data definition, Softshell allows all team members working independently on the same application. Design of templates engine allows assignment of templates development to multiple contributors, increasing team productivity.

Security

Security is a major concern in modern web based application. Web browser is out of control of system administrators; therefore security model assumes all data coming to the server are insecure. Client application is connected to a “userapi” interface, ensuring access rights control.

Locking mechanisms

Having multiple users simultaneously edit shared data leads to concurrency and locking issues. Softshell framework contains background techniques to ensure data consistency and access control:

- Record revision identifiers
  All records are marked with revision identifiers, so it is not possible to modify a record, in a version that have not been seen by modifying user. It is always ensured that users modify the most recent version of the record
- Soft locks and explicit locks
  Two locking modes are implemented: implicit and explicit. In traditional systems user acquires explicit lock on a record before he starts editing it.

After editing is finished or cancelled, lock is released. In Eller “soft lock” mechanism locks are acquired automatically, immediately after user starts typing text or making any other changes. Application developer should select appropriate locking model for each case, soft locking is used by default.

System requirements

Server

Software requirements

- Java SE 7,8
- Apache Tomcat 7,8
- Postgres 9.3-10

Server OS support

Linux (CentOS, RedHat, Ubuntu), Windows (Win2000 SP4 or newer), FreeBSD, OpenBSD, NetBSD, Mac OS X, AIX, HP/UX, Solaris, Tru64 Unix, UnixWare.

Client

Supported web browsers

Chrome, Safari (Mac/Windows, iOS), Firefox, Internet Explorer 9, 10, 11, Edge, mobile browsers

List of supported features

Applications

- Desktop application look and feel
- Screen areas, pages, multiple areas in one page, variable separators
- Separate scroll bars in each area, ability to add scrolling to any content part
- Endless scrolling
- Pop-up windows, dialog windows, context menus
- Mouse and touch support
- Main menu, submenus
- Framework automatically remembers and applies user settings for each page
- Data filters, sorting, local filters, grouping, custom views

Data access layer

- Automated on-the-fly SQL query construction from frontend API calls
- Support for data aggregation, sorting, filtering, multi-level grouping
- Support for parent-child structures
- Support for multiple-table queries using metamodel relation scheme
- Entity, field and record level permissions
- Dynamic tracking of changes, real-time view refresh with single value granularity
- Data paging

Dynamic metamodel

- Multiple metamodel definitions in a single app. server (spaces)
- Definition of tables and entities (logical tables), fields, field groups, relations, value lists
- Support for private fields (value is different for each user)
- Support for multiple datatypes, user extendable
- Supports arrays and array operations
- Model changes, updates during application runtime, no restart is needed
- Model changes from WEB browser admin interface
- Model versioning, history

Embedded realtime BI

- Metadata driven
- Online updates to cubes as primary data change
- Data access layer may query BI cubes in the same fashion as data tables
- Support for calculated fields
- Support for time-based axis and multilevel category axis

Backend scripting engine

- ECMA script 1.5 support with extensions, libraries
- User supplied libraries may be loaded on server start
- Remote debugger interface from WEB browser window
- Event-based model with synchronous and asynchronous event loops
- All system functions are available for scripts using a permission-managed API
- Dynamic loading/unloading of scripts

Templates

- Standard XHTML syntax and custom extensions
- Hundreds of API functions to simplify scripting
- Conditions, loops, external references, inheritance
- Multiple versions for different device modes
- Customizable by system administrator
- Auto-template wizards
- Hundreds of reusable screen components- templates, controls, data displays
- Flexible data grids with user-defined columns, resizable headers, customizations

Remote function invocation

- Ability to call a registered server function from a client-side page script (AJAX based)
- Automated mapping of data types (including arrays and complex object types), exceptions, return codes
- Ability to register/deregister callable functions dynamically
- Functions may be implemented in Java or JavaScript
- Java functions may be published automatically with Java annotations
- Security control with annotations
- Preloading of function definitions to allow better error detection
- Network situations failover

Event model

- Multilevel event loops for “before*”, “*”, and “after*” event handlers
- Runtime registrable / deregistrable event handlers
- Event preventing and parameter modification from script
- Some events may be asynchronous
- Event detect and notify all connected users, viewing data modified by event
- Multi-node (cluster installations) event processing

Localization

- Multiple UI languages supported by one instance, user selectable
- Localizable components
- All metamodel fields may be localized
- All HTML tags, attributes, text nodes, CSS data values may be localized
- User may switch languages without logging off

Security

- Component level permissions
- Cross-site scripting filters
- Run-time permissions model based on data dictionary definitions
- Remote callable functions permissions
- Role based and granular (system) permission levels
- Extendable authorization framework with pluggable modules for integration
- Embedded password or certificate authorization capabilities
- Invitation based user registration
- Passwords policy management
- Changes audit

More detailed list of features may be found in Developer and Administration guides.

Frequently asked questions

Q: How does Softshell compare to known frontend development frameworks, such as vue.js, react, angular etc..?
A: Softshell is not a framework, but a complete platform, consisting of many modules, development tools, features and components. Most features that are found in popular frontend frameworks are also available in Softshell, but their usage requires less coding, and they are seamlessly integrated with other modules and APIs

Q: What skills does a developer need to start developing applications using Softshell?
A: A common knowledge of WEB development technologies is required. Developer must have some knowledge of JavaScript, CSS, LESS. Front-end and back-end APIs are both using JavaScript, so there is no need to program in Java, SQL etc. Data modelling requires no coding at all, data queries are build using JSON expressions syntax.