

# ZK

## SIMPLY RICH

The Simplest Way to Make Web Applications Rich.

The Internet has emerged as the default platform for application development. Adoption and usage of Internet has acted as the driving force behind technology spending. According to IDC, Enterprise Information Portals (EIP), B2B and B2C applications, and e-Commerce spending are expected to grow at 41.2% (\$3 billions), 20%, 57.2% (\$5.7 trillions) CAGR in 2006, respectively.

As the Web continues to extend its reach into our daily lives, traditional page-based Web applications face a substantial challenge: the inability to visually represent the complexities in today's applications. The result are two folds: frustrated user experiences and excessive development costs.

Over a decade of evolution, Web applications evolved from static HTML pages, to Dynamic HTML pages, to applets and Flash, and, finally, to Ajax technologies (Asynchronous JavaScript and XML). Illustrated by Google Maps and Netflix, Ajax breathes new life into Web applications by finally combining the Web's ubiquitous 'reach' with the friendly 'richness' of desktop applications.

### Challenges of Ajax Applications

Developing Web application to fulfill today's requirement is never easy. Adding rich user interface by Ajax means adding more cost and risk to the already costly Web development.

Incompatible, sophisticated and even buggy JavaScript DOM API demands more knowledge and countless debugging nights. Asynchronous programming requires more experienced developers, and is prone to errors. Replication of business logics at thousands upon thousands of clients means excessive maintenance efforts.

Since 2005, dozen and more frameworks and libraries are emerging to enable Ajax for Web applications. They range over pure JavaScript libraries and widgets, extension of HTML to embrace widgets, and predefined naming pattern to communicate between clients and server.

After examining closely, these solutions are mainly solving the first challenge: eliminated the JavaScript hassles. The asynchronous communication between clients and servers remains, more or less, the duty of application developers.

### ZK

In response to this challenge, Potix has developed technologies and tools that enable Web applications to have both rich user experiences and lowest development cost ever. The core of the Potix solution is ZK, which includes an Ajax-based event-driven engine to automate

interactivity, a rich set of XUL and XHTML components to enrich usability, and a markup language to design user interfaces without programming.

With ZK, you represent your application in feature-rich XUL and XHTML components, and manipulate them by listening to events triggered by users, as you did for years in desktop applications. All your application codes are running at the server, while the visual representations of components and user activities at the browsers are automatically synchronized by ZK.

### Features and Benefits

**Ajax-based Rich User Interfaces.** Interactive and responsive. Users get the same level of use experience as using desktop applications, without any plugin at the browser.

**Event-driven Model.** Simple and intuitive. The cornerstone of desktop applications in 1990s is sound for its ease to learn and develop.

**XUL-based Components.** Rich and standard. Fastest way to build rich Web applications with over 60 off-the-shelf feature-rich components. No proprietary components secure your investment against vendor lock-in.

**ZK User-interface Markup Language.** Straight-forward and zero programming. Zero configuration. Neutral to components: XUL, XHTML or mixed is up to UI designers. Leverage the resources of XUL and XHTML community.

**Server-Centric Processing.** No replication of business logic at the clients. No asynchronous programming hassles. No RMI nor RPC. Use the same logic and data tiers you are used to.

**Script in Java and EL Expressions.** Quick prototyping and customization. No compilation. No JavaScript. No DOM. Just POJO (Plain Old Java cOde).

**Modal Dialogs.** Most intuitive way to interact with users for alternatives and decisions. Decomposing a sophisticated UI into several manageable dialogs.

**Simple Thread Model.** No thread knowledge required, while the server remains the scalability to handle requests for different pages simultaneously.

**Live Data.** Separating view and data, loading large data effectively, and browsing large data easily.

**GPL.** No vendor lock-in. Encouragement and beneficiary of global collaboration. Verification from large community and deployment.