ZK FrameWork

Developer Preview

By

Jimmy Heller

Under Supervision

Mr. Amirsam Bahador

مقدمه

در دنیا امروز وب توسعه سرعت بسیار زیادی پیدا کرده است. می‌توان کتاب‌ها و مقاله‌های بسیاری را در زمینه توسعه وب و هر هفته یک فریم ورک جدید یافت. با این وجود لازم است تا هر پدیده نو را در زمینه توسعه وب فرا گرفت چرا که نداشتن درک کافی از امکانات یک محصول استفاده از ویژگی‌های آن را نیز محدود خواهد کرد. دلایلی همچون رشد تکنولوژی و تقاضای بیشتر برای توسعه مناسب‌تر وب باعث رشد این فضا بوده است. هر توسعه دهنده روش‌های خود را برای توسعه در اختیار دارد، با این وجود مشکلات در توسعه نیازمند روش‌های جدید است. باید گفت که در افقی دورتر این وب 2.0 و تکنیک‌های زیرین آن از جمله AJAX نیازمند روش‌های نوتری خواهند بود.

ZK با شعار AJAX بدون JavaScript سعی در برطرف کردن بسیاری از مشکلات توسعه نرم‌افزارهای تحت وب است.

قبل از شروع لازم به ذکر است که این مطلب تنها اشاره بسیار کوتاهی به این فریم ورک خواهد داشت و برداشت آزادی از کتاب ZK Developers Guide محصول انتشارات Packtpub است.

همچنین باید اشاره داشت که ZK قابلیت اجرا بر روی مرورگرهای زیر را نیز داراست:

- Internet Explorer 6+
- Firefox 1+
- Safari 2+
- Mozilla 9+
- Opera 9+

و مرورگرهایی که از قابلیت مناسب برای DOM و JavaScript برخوردار نیستند، توانایی اجرای کدهای نوشته شده به کمک این فریم ورک را ندارند.

فریم ورک ZK تحت دو لیسانس ارائه می‌شود که نسخه رایگان آن تحت لیسانس GPL ارائه می‌شود و نسخه تجاری به همراه پشتیبانی از سایت این فریم ورک قابل تهیه است. برای شروع نیاز به اطلاعات کافی از Java و XUL (XML User Interface Language) دارید. توسعه توسط پایگاه داده نیز در این فریم ورک به کمک ابزارهایی نظیر Hibernate صورت می‌گیرد.

ZK چیست؟

این فریم ورک به معنای اجرای AJAX بدون JavaScript است.

در این فریم ورک سه جزء وجود دارد:

1- موتور رویداد محور بر اساس AJAX
2- اجزاء غنی XUL و XHTML
3- ZUML

XHTML

مخفف Extensible Hyper Text Markup Language است. این زبان بسیار شبیه HTML است. ویژگی‌های این زبان :

- حساس بودن به نشانه‌گذاری
- المان‌های بدون محتوا باید بسته شوند مانند

```
<html>
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="de"
xml:lang="de">
<head>
<title>A XHTML example</title>
</head>
<body>
<h1>Testpage</h1>
<p>A paragraph</p>
<p>another<br />paragraph
</p>
</body>
</html>
```

XUL

مخفف User Inteface Markup Lanaguage XML است. این زبان توسط گروه Mozilla ابداع شده است.

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>
<window
id="findfile-window"
title="Find Files"
orient="horizontal"
xmlns="http://www.mozilla.org/keymaster/gatekeeper
/there.is.only.xul">
<button id="find-button" label="Find"/>
<button id="cancel-button" label="Cancel"/>
</window>
```

ZUML

Zk User Interface Markup Language که بر اساس XUL نوشته شده است.

```
<window title="Hello" border="normal">
Hello World!
</window>
```

برای کاربران اپلیکیشن نوشته شده توسط ZK به عنوان دسکتاپ اپلیکیشن شناخته می‌شود. برای توسعه‌دهندگان مانند یک برنامه کامل خواهد بود، ZK از یک پارادایم رویداد محور استفاده می‌کند که درخواست و تقاضا را به صورت اتمیک اجرا خواهد کرد. این زیرساخت به وسیله فریم ورک اجرا می‌شود در نتیجه از کدهای JavaScript بی‌نیاز خواهید بود.

آغاز یک برنامه با ZK

1- در ابتدا نیاز به یک اپلیکیشن سرور است . این وب سرور باید از Servlet 2.4 پشتیبانی کند. در اینجا از Apache Tomcat استفاده خواهیم کرد.

برای استفاده از این سرور به سایت آن مراجعه کنید و نسخه مربوط به سیستم عامل خود را دریافت کنید. دقت داشته باشید که از نسخه نصبی آن استفاده نکنید.

2- کتابخانه ZK را دریافت کنید و از حالت زیپ دربیاورید.
3- برای اجرا شدن برنامه باید طوری فایل‌ها را قرار دهید تا در هر سروری اجرا شود . در نتیجه از پوشه‌سازی به روش زیر استفاده خواهیم کرد.

4- در پوشه کتابخانه فریم ورک خود، تمام JAR فایل‌های شاخه lib را به ترتیب زیر کپی کنید:
- {YOUR_ZK_UNZIP_FOLDER}/dist/lib
- {YOUR_ZK_UNZIP_FOLDER}/dist/lib/ext
- {YOUR_ZK_UNZIP_FOLDER}/dist/lib/zkforge

5- برای معرفی Servlet ها، فیلترها و ... باید از فایلی به نام web.xml استفاده کنیم. این فایل بسیار حیاتی است و هر برنامه J2EE نسخه‌ای از آن را به همراه خواهد داشت. برای این مثال می‌توانید از فایل زیر استفاده کنید.

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

    <description><![CDATA[My ZK Application]]></description>
    <display-name>MyApp</display-name>

    <listener>
        <description>ZK listener for session cleanup</description>
        <listener-
class>org.zkoss.zk.ui.http.HttpSessionListener</listener-class>
    </listener>
    <servlet>
        <description>ZK loader for ZUML pages</description>
        <servlet-name>zkLoader</servlet-name>
        <servlet-
class>org.zkoss.zk.ui.http.DHtmlLayoutServlet</servlet-class>
```

```
        <init-param>
            <param-name>update-uri</param-name>
            <param-value>/zkau</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>zkLoader</servlet-name>
        <url-pattern>*.zul</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>zkLoader</servlet-name>
        <url-pattern>*.zhtml</url-pattern>
    </servlet-mapping>
    <!-- Optional. Uncomment it if you want to use richlets.
    <servlet-mapping>
        <servlet-name>zkLoader</servlet-name>
        <url-pattern>/zk/*</url-pattern>
    </servlet-mapping>
    -->
    <servlet>
        <description>The asynchronous update engine for
ZK</description>
        <servlet-name>auEngine</servlet-name>
        <servlet-
class>org.zkoss.zk.au.http.DHtmlUpdateServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>auEngine</servlet-name>
        <url-pattern>/zkau/*</url-pattern>
    </servlet-mapping>

    <welcome-file-list>
        <welcome-file>index.zul</welcome-file>
        <welcome-file>index.zhtml</welcome-file>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
    </welcome-file-list>
</web-app>
```

6- برای شروع از فایلی به نام hello.zul استفاده خواهیم کرد با محتوای زیر:

```
<window title="My First ZK Application" border="normal">
   Hello World!
</window>
```

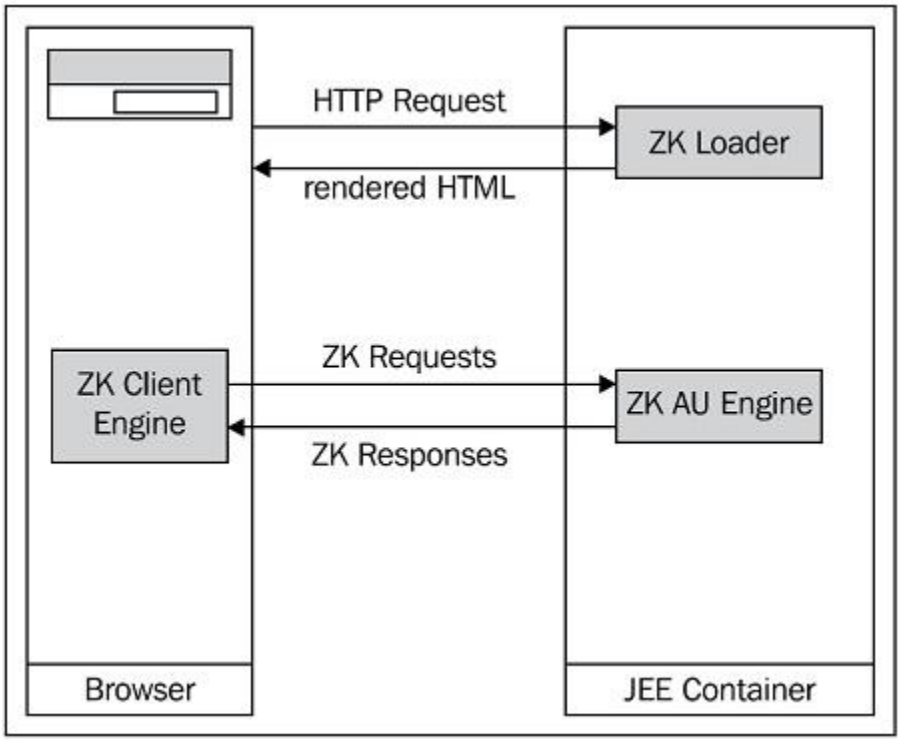
7- در پایان باید پوشه پروژه خود را برای قرار دادن در سرور آماده کنید. در نتیجه تمام پوشه خود را زیپ کنید. سپس پسوند آن را به war تغییر دهید. این فایل را داخل پوشه webapps خود قرار داده و سرور را اجرا کنید. با رفتن به آدرس زیر برنامه شما قابل اجرا خواهد بود.

Localhost:8080/myZK/myZK/hello.zul

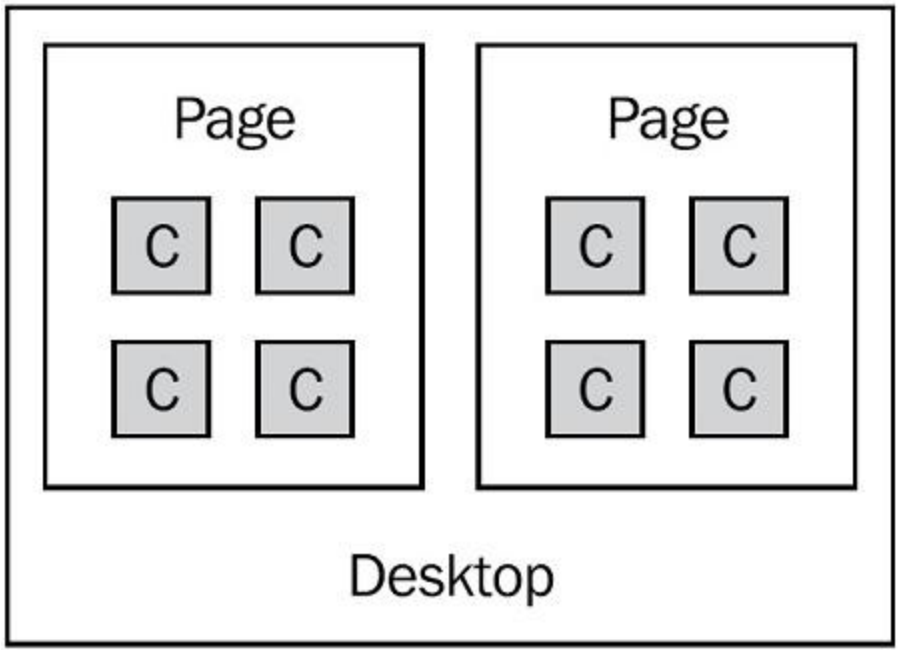
درون فریم ورک ZK

ZK Loader	مسئول بارگذاری و تفسیر یک صفحه ZK است. انتخاب صفحه توسط آدرسی است که از سمت مرورگر کاربر ارائه می‌شود. نتیجه به صورت صفحه HTML به مرورگر ارسال خواهد شد.
ZK Client Engine	این جز درخواست‌های ZK را به سرور ارسال خواهد کرد و پاسخ‌های ZK را دریافت خواهد کرد. به وسیله این پاسخ‌ها DOM در سمت کاربر تسکیل خواهد شد. در نتیجه می‌توان کاربر را بخشی از AJAX دانست.
ZK AU	این بخش، بخش سرور AJAX است.

شکل زیر نمایشی از آنچه در جدول بالا آمده است را ارائه می‌کند.

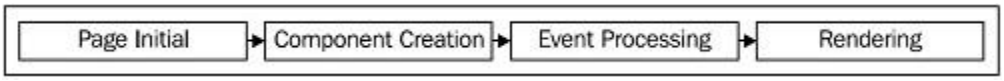


Component	نماد : org.zkoss.zk.ui.Component یک Component شئ UI است مانند دکمه‌ها، برچسب‌ها و ... . این اجزا به جز اینکه در سرور شئ جاوا باشند نمایشی دیداری در مرورگر خواهند داشت. ممکن است فرزندهایی داشته باشد که اگر اینطور نباشد به آن Root گفته می‌شود.
Page	نماد: org.zkoss.ui.page یک صفحه شامل Component ها است. در نتیجه یک Page حامل اجزا است. اکر ZK Loader به تفسیر یک ZUML بپردازد یک page تشکیل شده است.
Desktop	نماد : org.zkoss.zk.us.Desktop یک صفحه ZUML قادر به شامل شدن یک صفحه ZUML دیگر است. این به خاطر این است که این صفحات به وسیله یک URL قابل دسترسی هستند و به یک desktop وصل شده‌اند. Desktop یک حامل برای page ها است.



بارگذاری و به روز رسانی یک صفحه ZK

این قسمت از 4 مرحله تشکیل شده است که به ترتیب زیر اجرا می‌شوند.



فاز Page Initial : در این فاز صفحه مقدار دهی می‌شود. برای کنترل این فاز می‌توانید از یک کلاس یا یک zscritp استفاده کنید. در زیر نمونه‌ای از تعریف یک کلاس برای init می‌بینید:

```
<?init class="sample.InitSample" ?>
```

کلاس InitSampleباید Interface ، org.zkoss.zk.util.util.Initiatior را پیاده سازی کند.

فاز Component Creation: در این فاز ZK Loader مسئول تفسیر صحفه‌ها خواهد بود.

فاز Event Processing: برای هر رویدادی که در صف قرار گرفته، event listeners در یک thread فراخوانی خواهند شد.

فاز Rendering: در مرحله آخر تمام Componentها در یک صفحه HTML قرار خواهند گرفت.

زبان برنامه نویسی ZUML

این زبان در واقع همان XML است با namespace های جدید. Namespaceهای زیر مهمترین هستند در اجرای ZK:

http://www.zkoss.org/2005/zul	The XUL Component set
http://www.w3.org/1999/xhtml	The XHTML component set

ZK-specific attributes http://www.zkoss.org/2005/zk

در یک صفحه XUML می‌توانیم از Java، JavaScript، Ruby و Groovy استفاده کرد. علاوه بر اجزایی نظیر window، textbox و button شما این امکان را دارید تا برای هر صفحه اجزای مخصوص به آن را به وسیله styleها تهیه کنید:

```
<button label="Say Hello" style="border:2px red" />
```

اما اگر لازم باشد تا به دفعات از آن استفاده کنید باید داشته باشید:

```
   <?component name="mybutton" extends="button" style="border:2px red"
label="Say Hello" ?>
```

این به معنای تعریف یک جز به صورت انتخابی خواهد بود که می‌توانید از آن به صورت‌های زیر استفاده کنید:

```
   <mybutton />
<mybutton label="My label" />
```

و این قابلیت را دارید تا آن را override نیز بکنید:

```
   <?component name="button" extends="button" style="border:2px red"
   label="Say Hello" ?>
   <mybutton />
<mybutton label="My label" />
```

تنظیمات و اجرا

تنظیمات فایل Web.xml

مهم‌ترین بخش یک برنامه وب تنظیم فایل web.xml است. بدون اطلاعات صحیح در این فایل برنامه به صورت صحیح کار نخواهد کرد. در ابتدا zkloader را به عنوان اولین servlet خواهیم داشت، این servlet باعث بالا آمدن یک صفحه ZUML خواهد شد (زمانیکه درخواست برای یک صفحه ارسال خواهد شد).

سرولت مورد نظر org.zkoss.zk.ui.http.DHtmlLayoutServlet نام دارد. از آنجایی که اولین ورودی برای فریم ورک خواهد بود، باید اول بار شود.

```
<load-on-startup>1</load-on-startup>
```

این سرولت دو پارامتر دارد:

- **Update-uri:** اجباری است و آدرس منتهی به ZK AU را مشخص خواهد کرد. مرورگر به این آدرس برای ارسال درست درخواست AJAX نیاز دارد. به طور پیش فرض /zkau در نظر گرفته می‌شود.
- **Log_Level:** پارامتری دلخواه که log level را برای org.zkoss package مشخص خواهد کرد. مقادیر مورد نظر OFF,ERROR,DEBUG,INFO و WARNING است.

سرولت مهم و اجباری دوم org.zkoss.zk.au.http.DHtmlUpdateServlet است. این سرولت درخواست‌های AJAX همزمان را به عهده خواهد داشت. آدرس داده شده برای این سرولت باید با آدرس update-uri اولیه برابری داشته باشد.

سرولت اجباری آخر، ZK Session Cleaner خواهد بود و وظیفه پاکسازی thread در هنگام بسته شدن HttpSessionListener حافظه را پاک خواهد کرد.

حال اگر JSP یا JSF در کار باشد ZK سرولتی به عنوان فیلتر در نظر می‌گیرد که پردازش خود را جدا خواهد کرد. این سرولت در org.zkoss.zk.ui.http.DHtmlLayoutFilter قرار دارد و دو پارامتر دارد:

- **Extension:** توصیف اینکه پردازش خروجی چگونه باشد که به صورت پیش فرض HTMLاست.
- **Charset:** تعیین کننده charset خروجی است که به صورت پیش فرض utf-8خواهد بود.

بخشی از یک صفحه web.xml:

```
<listener>
<description>
Used to cleanup when a session is destroyed
</description>
<display-name>ZK Session Cleaner</display-name>
<listener-class>
org.zkoss.zk.ui.http.HttpSessionListener
</listener-class>
</listener>
<servlet>
<description>ZK loader for ZUML pages</description>
<servlet-name>zkLoader</servlet-name>
<servlet-class>
org.zkoss.zk.ui.http.DHtmlLayoutServlet
</servlet-class>
<init-param>
<param-name>update-uri</param-name>
<param-value>/zkau</param-value>
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>zkLoader</servlet-name>
<url-pattern>*.zul</url-pattern>
</servlet-mapping>
<servlet-mapping>
<servlet-name>zkLoader</servlet-name>
<url-pattern>*.zhtml</url-pattern>
</servlet-mapping>
<servlet>
<description>The asynchronous update engine for ZK</description>
```

```
<servlet-name>auEngine</servlet-name>
<servlet-class>
org.zkoss.zk.au.http.DHtmlUpdateServlet
</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>auEngine</servlet-name>
<url-pattern>/zkau/*</url-pattern>
</servlet-mapping>
```