

AJAX UTAN X, MED PROTOTYPE OCH SPRING

Jesper Hammarbäck

Det har skrivits mycket i ämnet AJAX. Det finns webbsidor, how-to:s och en ansevärd mängd ramverk som skapats för att underlätta utveckling av AJAX-applikationer. AJAX är ingen teknik i sig, utan ett samlingsnamn för en mängd tekniker som används för att uppnå mer dynamik på webben. Jesper Hammarbäck visar hur man på enklaste sätt, utan att vare sig behöva traversera DOM-träd eller använda JSON, kan använda AJAX i en webbapplikation.

Prototype är en fin samling JavaScript-funktioner samlade i en fil på 73kb. Den innehåller alla möjliga tänkbara hjälpmedel för webbutvecklare, exempelvis möjligheten att referera till ett id med `$()` istället för det vanliga `document.getElementById()`. Så klart finns också AJAX-stöd vilket vi skall titta närmare på. Prototype är välanvänt, vältestat och fungerar med de flesta webbläsare. I det här exemplet använder jag också Spring Framework som vid det här laget knappast behöver en närmare presentation.

Målet är att bygga en JSP-sida som visar aktuell tid. Inte helt oväntat finns ett krav att tiden ska uppdateras varje sekund, utan att hela webbsidan måste laddas om. Så passande för oss!

Så här kommer det gå till: En gång per sekund anropar vi, med hjälp av Prototype, en Spring-Controller som beräknar aktuell tid och vidarebefordrar värdet till en JSP-vy. Vyn består av några rader JSTL/EL och där har vi möjlighet att göra formateringar genom att bädda in vår data i vanlig HTML-kod. På det sättet får vi en behaglig separation av data och presentation även för våra AJAX-anrop, och en Controller som är lätt att enhetstesta. Resultatet blir en snutt HTML som skrivs in i en av oss definierad DIV-tag i en JSP-sida och vips så är vi i mål.

Vi börjar alltså med att skapa en JSP-sida och en DIV som markerar var vi vill att tiden ska visas. Genom att referera till DIV-taggens id kommer vi kunna ersätta

innehållet med resultatet av vårt ajax-anrop. Eftersom anropet ska ske periodiskt enligt ett givet intervall använder vi Prototypes funktion `PeriodicalUpdater`.

index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <script type="text/javascript" language="javascript"
    src="<c:url value="js/prototype.js"/>">
  </script>
</head>
<body>
  <script type="text/javascript">
    new Ajax.PeriodicalUpdater('timeDiv', 'getTime.ajax',
      {asynchronous: true, frequency: 1});
  </script>

  <div id="timeDiv"></div>
</body>
</html>
```

Lägg märke till hur enkelt själva AJAX-anropet är. Tack vare Prototype behöver vi inte pyssla med vare sig `XMLHttpRequest` eller `ActiveXObject`.

Sedan skapar vi en `Controller` i Spring som returnerar en modell som innehåller aktuell tid. Notera att `ajax/time` alltså betyder `/WEB-INF/jsp/ajax/time.jsp` efter att Springs `viewResolver` gjort sitt:

GetTimeController.java

```
public class GetTimeController extends AbstractController {
  protected ModelAndView handleRequestInternal(
    HttpServletRequest request, HttpServletResponse response) {
```

```
Map<String, Object> model = new HashMap<String, Object>();
model.put("currentTime", new Date());
return new ModelAndView("ajax/time", model);
}
}
```

En Spring-context-fil med mappning måste vi också skapa:

simpleajax-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">

<beans>
  <bean id="getTimeController" class="demo.GetTimeController"/>

  <bean id="urlMapping"
class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
      <props>
        <prop key="/getTime.ajax">getTimeController</prop>
      </props>
    </property>
  </bean>

  <bean id="viewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="viewClass"
value="org.springframework.web.servlet.view.JstlView"/>
    <property name="prefix" value="/WEB-INF/jsp/" />
    <property name="suffix" value=".jsp" />
  </bean>
</beans>
```

... och så behöver vi en web.xml för att sparka igång Spring och mappa .ajax-ändelsen till Springs DispatcherServlet.

web.xml

```
<?xml version="1.0"?>
<web-app 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"
  version="2.4">
  <display-name>simpleajax</display-name>

  <servlet>
    <servlet-name>simpleajax</servlet-name>
    <servlet-class>
      org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>simpleajax</servlet-name>
    <url-pattern>*.ajax</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

Sedan är det dags att skapa vyn som formaterar Date-objektet:

time.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>

<span style="font-weight:bold;">${currentTime}
```

Klart! Paketera resultatet som en webbapplikation och starta den, t.ex. via `http://localhost/simpleajax/`. Om allt fungerar ska tiden visas och uppdateras varje sekund medan servlet-containerns accessloggar fylls med POST-anrop till `/simpleajax/getTime.ajax`.

Det var väl inga svårigheter? Stora delar är uppsättningskod som inte behöver upprepas när man skapar och lägger till fler AJAX-anrop.

Nedan inkluderas en länk till en körbar exempelapplikation. Den innehåller ytterligare en vanligt förekommande AJAX-konstruktion i form av en suggest-funktion vilken presenterar en lista med bilmärken baserat på de bokstäver användaren skrivit i textfältet. ■ ■ ■

LÄR DIG MER OM AJAX

- Exempelapplikation: <http://www.citerus.se/pnehm/1-2007/simpleajax.zip>
- [http://en.wikipedia.org/wiki/Ajax_\(programming\)](http://en.wikipedia.org/wiki/Ajax_(programming))
- <http://www.prototypejs.org/>
- <http://www.sergiopereira.com/articles/prototype.js.html>
- <http://www.springframework.org/>



Jesper Hammarbäck är konsult på Citerus AB och arbetar som utvecklare på javaplattformen. Kontakta honom på jesper punkt hammarback at citerus punkt se.