بسم الله الرحمن الرحيم

اليوم الدرس عن الـ HTTP STATUS CODES

عن رموز الحالة لاستجابة المستعرض لـ سرفلت
ما سنأخذه بهذا الدرس هو:

- هيئة الـ HTTP Response
- وضع قيم STATUS CODES
- وفهم أن STATUS CODES مفيد لـ
  a) اختصار الطرق من أجل أعادة التوجه للمتصفح و من أجل صفحات الخطأ
  b) يجعل السرفلت يقوم بتحويل المستخدم إلى صفحات محددة على المستعرض
  c) صنع واجهة من أجل البحث في محركات البحث المختلفة

وضع قيم لرموز الحالة (STATUS CODES) من السرفلت:

موضع ٣٠٢ و ٤٠٤ من رموز الحالة :

الطريقة العامة من أجل وضع قيمة لرموز الحالة هي عن طريق الطريقة response.setStatus(int)، هناك نوعين من الحالات الشائعة التي تختصر طرق في الصف HttpServletResponse و بأخذ بعين الاعتبار أن هتان الطريقتان ترميان استثناء من نوع IOException في حين أن الطريقة setStatus لا تحتاج إلى استثناء

- **public void sendRedirect(String url)**

رمز الحالة 302يوجه المتصفح للاتصال إلى موقع جديد والطريقة **sendRedirect** تولد الحالة ٣٠٢ مع أعطاء القيم لـ الهيدر Location قيمة الـ URL من الوثيقة الجديدة ولكن يتم وضع قيمة رمز الحالة قبل وضع الهيدر Location.

- **public void sendError(int code, String message)**

رمز الحالة ٤٠٤يستخدم عندما المخدم لا يجد الصفحة التي تبحث عنها والطريقة **sendError** ترسل رمز الحالة (عاداً يكون ٤٠٤) بإضافة إلى رسالة قصيرة من الوسيط **message** من نمط **String** بحيث تتناسق الرسالة مع صفحة HTML وبعدها ترسل إلى المستعرض أو العميل.

وضع قيم لرمز الحالة لا يعني حذف الوثيقة فلا سبيل المثال أن أغلب السيرفرات تقوم بتوليد ملف صغير هو لم يتم التعرف على الرسالة لـ استجابة ٤٠٤،قد ترغب بإنشاء سرفلت مخصص لهذا العمل، ولكن تذكر إذا لم ترسل الخرج عليك استدعاء الطريقتان setStatus أو sendError أولاَ.

هيئة HTTP1.1 لـ رموز الحالة :

في هذا القسم حن نصف أهم رموز الحالة التي تتيح لك استخدام السرفلت للتواصل مع HTTP1.1 الخاص بالعميل، وذالك مع الرسائل الخاصة المرتبطة مع كل رمز من رموز الحالة، وفهمك لهذه الرمز يعزز من قدرات السرفلت الذي سوف تقوم ببرمجته، لذالك يجب أن تعرف الخيارات المقدمة تحت تصرفك كما تكلما في الدرس السابق عن أن هناك متصفحات تدعم بروتوكول HTTP1.0 ولا تدعم الإصدار 1.1 لذالك عليك التأكد من الإصدار قبل العمل ذالك من خلال الطريقة request.getRequestProtocol() ولمزيد من التفاصيل

قم بزيارة الوثيقة RFC 2616 الذي تكلمنا عن نوعية وثائق RFC في الدرس السابق الوثيقة RFC 2616 موجودة في الرابط http://www.rfc-editor.org/rfc/rfc2616.txt

الآن نقوم بوصف جميع رموز الحالة المتاحة في بروتوكول HTTP1.1 وهي مقسمة على خمسة فئات:

- **100–199**

الرموز ضمن هذا المجال تعطي معلومات على العميل أن يستجيب لبعض الإجراءات.

- **200-299**

الرموز ضمن هذا المجال هي دلالة على أن الطلب كان ناجحاً.

- **300-399**

القيم ضمن هذا المجال تستخدم من اجل ملفات التي انتقلت وعادةً تضمن الهيدر `Location` الذي ينقلك إلى عنوان URL جديد.

- **400-499**

القيم ضمن هذا المجال تدل على الأخطاء من طرف العميل (الزبون client).

- **500-599**

القيم ضمن هذا المجال تدل على الأخطاء من طرف السيرفر.

هناك في الصف HttpServletResponse يوجد ثوابت تمثل رموز الحالة المختلفة يتم اشتقاق رسائل قياسية مختلفة مرتبطة بـكل الرموز. ففي السرفلت يوجد ثوابت بدل من الرموز بكل بساطة على سبيل المثال يمكن استخدام الطريقة response.setStatus (response. SC_OK) وهي تقابل بالاستخدام response.setStatus (200) ويمكنك استخدام response.setStatus (response.SC_NO_CONTENT) بدل من response.setStatus (204) وهكذا مع بقية الرموز الحالة فمقابل كل رمز هناك ثابت مقابل له في الصف HttpServletResponse.

100 (Continue)

إذا السرفر تلقى مع request header القيمة 100 (Continue)، هذا يعني أن العميل يسأل في ما إذا كان يمكن إرسال الوثيقة مرفقة بمتابعة الطلب.في هذه الحالة، يجب أن يستجيب السيرفر للحالة رقم 100الثابت في الصف HttpServletResponse هو SC_CONTINUE من أجل العميل يمضي على هذا أو استخدم الحالة رقم ٤١٧ (SC_EXPECTATION_FAILED) لتخبر المستعرض أنها لن تقبل وثيقة. هذا الرمز الحالة جديد في HTTP1.1.

200 (OK)

القيمة 200 (SC_OK) يعني أن كل شيء على ما يرام، والوثيقة يليها طلب post و get. وهو الوضع الافتراضي لكل السرفلتات، وحتى إذا لم تستخدم الطريقة setStatus ستحصل على 200.

202 (Accepted)

القيمة 202 (SC_ACCEPTED) تخبر العميل الذي أرسل الطلب أن الطلب يتم بنائه لكن المعالجة لم تنتهي بعد.

204 (No Content)

رمز حالة من 204(SC_NO_CONTENT) تنص على أنه ينبغي أن يستمر مستعرض لعرض الوثيقة السابقة لأنه لا يوجد وثيقة جديدة متاحة .هذا السلوك هو مفيد إذا كان المستخدم يقوم بالإعادة تحميل بشكل دوري لصفحة عن طريق الضغط على زر تحديث ويمكنك تحديد ذلك في الصفحة السابقة، هذه الرمز هو أيضاً جديد في HTTP 1.1.

205 (Reset Content)

قيمة (SC_RESET_CONTENT) 205يعني أنه لا يوجد أي وثيقة جديدة ولكن المتصفح يجب إعادة تعيين عرض الوثيقة .وبالتالي، يتم استخدام هذا رمز الحالة لإرشاد المتصفحات لمسح حقول النموذج (form) .هذه الرمز هو أيضاً جديد في HTTP 1.1.

301 (Moved Permanently)

الحالة 301 (SC_MOVED_PERMANENTLY)يشير إلى أن الوثيقة المطلوبة هي في مكان آخر، ويعطي عنوان جديد (url) للمستند في الهيدر Location.وينبغي على المستعرض تلقائياً أن يذهب إلى عنوان جديد.

302 (Found)

هذه القيم مشابها لـ 301إلا إنه من حيث المبدأ فأن الـ urlالمقدم من قبل الهيدر Location يجب أن يفسر على أنه بديل مؤقت وليس دائم. في الواقع فأن معظم المستعرضات تتعامل مع 301 و302 الممثلة. ملاحظة في HTTP1.0: يتم نقل الرسالة بشكل مؤقت بدل من العثور على الثابت في الصف HttpServletResponse الثابت هو SC_MOVED_TEMPORARILY وليس SC_FOUND.

303 (See Other)

القيم 303 (SC_SEE_OTHER) هذه الحالة مشابها لـ 301 و 302 إلا أنه إذا كان الطلب الأصلي POST، أن الوثيقة الجديدة (التي حصلنا عليها من الهيدر `Location`) يجب استرجاعها مع GET. انظر إلى رمز الحالة 307. هذا الرمز هو أيضا جديد في HTTP1.1.

304 (Not Modified)

عندما يكون عند العميل وثيقة مؤقتا، فإنه يمكن تنفيد طلب شرطي من خلال الهيدر If-Modified-Since وهي من أجل معرفة إذا الوثيقة قد تم تغيرها منذ تاريخ محدد.فإذاً القيمة 403 (SC_NOT_MODIFIED) تعني أن النسخة المخبأة هي ما تزال جديدة أو حديثة. وإذا حدث خلاف ذالك يجب على السيرفر أن يعود بالوثيقة المطلوبة مع الرمز الحالة (200). ويجب على السيرفلت أعادة تعين هذا الرمز بشكل مباشر. وفي خلاف ذالك يجب أن تنفذ الطريقة getLastModified، والسماح للطريقة الافتراضية service بمعالجة الطلبات استنادا إلى تاريخ التعديل. إذا أردت مثال أرجع إلى المرجع الفصل الثالث القسم السادس في دورة حياة السرفلت تجد مثال اسمه LotteryNumbers.

307 (Temporary Redirect)

القواعد التي يتعامل فيها السيرفر مع المتصفح في الحالة 307 هي مماثلة لـ 302، القيمة 307 هي مضافة إلى HTTP1.1. هناك العديد من المتصفحات تصادف مشكلة أثناء أعادة التوجيه في حالة الاستجابة 302 حتى وأن كانت الرسالة الأصلية هي POST. ومن المفترض أن المتصفحات تقوم بمتابعة إعادة التوجيه الطلب POSTفقط عندما يحصلون على حالة استجابة 303.وهذه الحالة جديدة وهدفها أن تكون الحالة واضحة لا لبس فيها: لمتابعة إعادة توجيه الطلبات POST و GET في الحالة الاستجابة 303 ; ولمتابعة إعادة التوجيه لطلب GET وليس POST في حالة الاستجابة 307. هذا رمز الحالة جديد في HTTP1.1.

400 (Bad Request)

400 (SC_BAD_REQUEST) هذه حالة تشير إلى خطأ في بناء جملة طلب العميل.

401 (Unauthorized)

قيمة ٤٠١ (SC_UNAUTHORIZED) يدل على أن العميل يحاول الوصول إلى صفحة محمية بكلمة مرور ولكن هذا الطلب لم يكن لديهم معلومات السليم وتحديد في الهيدر `Authorization`. ويجب أن يتضمن الرد الهيدر `WWW-Authenticate`. لمزيد من التفاصيل، انظر الفصل المتعلق برنامجي ويب تطبيق الأمن (يتعلق بالسكيورتي وسيكون هناك درس في ذالك).

403 (Forbidden)

رمز حالة 403 (SC_FORBIDDEN) يعني أن المخادم يرفض تزويد الموارد، وبغض النظر عن التصريح. وهذه الحالة غالبا ما تكون نتيجة خطأ من ملف أو أذونات الدليل على المخادم.

404 (Not Found)

الخطأ الشائع من طرف الزبون (SC_NOT_FOUND) 404 هذه الحالة تخبر الزبون أنه لا يمكنه العثور على المورد في العنوان الذي طلبه. والقيم المعيارية لهذا الحالة هي "no such page" تعني أنه لا يوجد مثل الصفحة. الطريقة المفيدة لهذه الحالة هي **sendError("**message**")** الموجودة في الصف HttpServletResponse.

405 (Method Not Allowed)

القيمة 405(SC_METHOD_NOT_ALLOWED) تعني أنه لم يتم السماح لطريقة الطلب (GET ,POST, HEAD,PUT, DELETE,إلخ) لهذا المصدر بالذات. هذا الرمز الحالة هو جديد في HTTP1.1.

415 (Unsupported Media Type)

القيمة 415 (SC_UNSUPPORTED_MEDIA_TYPE) يعني أن في الطلب كان هناك مرفق مع الوثيقة والسيرفر لم يستطع التعامل مع نوع المرفق. هذا الرمز الحالة جديد في HTTP1.1.

417 (Expectation Failed)

إذا كان السيرفر يتلقى توقيع بـ request header مع القيمة 100-continue في رموز الحالة، هذا يعني أن العميل يسأل في ما إذا كان يمكن إرسال الوثيقة مرفقة بمتابعة الطلب. في مثل هذه الحالة، يجب أن يستجيب السيرفر مع هذه الحالة (417) لنقول أن المتصفح أنه لن يقبل الوثيقة أو استخدام 100 (SC_CONTINUE) من أجل العميل يمضي على هذا. هذا رمز الحالة هو جديد في HTTP1.1.

500 (Internal Server Error)

(`SC_INTERNAL_SERVER_ERROR`) 500هذه الحالة تكون نتيجة خطأ في تنفيذ السرفلت نتيجة في سوء برمجة السرفلت. هي غالباً نتيجة أن السرفلت قد أتلف أو أنه أرجعت قيم الهيدر غير مهيأ بشكل صحيح.

501 (Not Implemented)

501 (SC_NOT_IMPLEMENTED) في هذه الحالة تقوم بإعلام العميل بأن السيرفر لا بدعم وظيفة تلبية الطلبات، يتم استخدامها على سبيل المثال عندما العميل يقوم بإصدار أمر مثل PUT فيكون الرد السيرفر لا يدعم.

503 (Service Unavailable)

رمز حالة من (SC_SERVICE_UNAVAILABLE) 503يدل على أن السيرفر لا يستجيب بسبب صيانة أو الحمولة الزائدة .على سبيل المثال، قد يعيد السرفلت هذا الهيدر إذا كانت بعض الصفحات أو تجمع اتصال أو قاعدة البيانات كاملة في الوقت الراهن. يمكن للسيرفر تزويد المستعرض بـ الهيدر Retry-Afterلتخبر العميل بإعادة المحاولة مرة أخرى.

505 (HTTP Version Not Supported)

(SC_HTTP_VERSION_NOT_SUPPORTED) 505هذا الرمز يعني أن السيرفر لا يدعم إصدار HTTPالوارد اسمه في خط الطلب (request line) .هذا الرمز الحالة هو جديد في HTTP 1.1.

ملاحظة: من أجل التوضيح أن كل من الكلمات التالية لها نفس المعنى
العميل الزبون نفس المعنى
السيرفر المخدم الملقم نفس المعنى
المستعرض المتصفح نفس المعنى

مثال الأول: عن إعادة توجيه المستعرض حسب نوع المستعرض أعطينا واجب في الدرس الرابع عن طبعاً اسم المستعرض ما سنقومه تعديل بسيط على الواجب باستبدال الطباعة بالطريقة **response.sendRedirect(url)** وتوجيه إلى مسار محدد حسب المستعرض هذا يعلق برمز الحالة 302.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class WrongDestination extends HttpServlet
 {
public void doGet (HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException
        {
             String userAgent = request.getHeader("User-Agent");
         if ((userAgent != null) &&(userAgent.indexOf("MSIE") != -1))
                    {
                    // Microsoft Minion
                    response.sendRedirect("http://www.microsoft.com");
                    }
         else if ((userAgent != null) &&(userAgent.indexOf("Firefox") != -1))
                    {
                    // Mozilla Firefox
                    response.sendRedirect("http://www.mozilla.org ");
                    }
         else if ((userAgent != null) &&(userAgent.indexOf("Chrome ") != -1))
                    {
                    // Google Chrome
                    response.sendRedirect("http://www.google.com ");
                    }
         else if ((userAgent != null) &&(userAgent.indexOf("Chrome ") != -1))
                    {
                    // Hopeless Netscape Rebel
                    response.sendRedirect("http://home.netscape.com ");
                    }
         }
}
```

المثال الثاني: سنقوم بعمل سرفلت SearchEngineForm يعرض واجها من اجل البحث عن كلمة في عدة محركات بحث الشائعة وننشأ سرفلت لمعالجة الطلب القادم من سرفلت SearchEngineForm وسنعالج حالة 403 (Found) و404(Not Found)وسوف نستخدم الطريقة **response.sendError(int code,String message)** في حال الأخطاء ونستخدم الطريقة **response.sendRedirect(searchURL)** لتوجيه إلى محرك البحث المهم نحن نأخذ البارمترات القادمة من صفحة Html هي اسم محرك البحث و الكلمة التي نريد البحث عنها طبعاً في حالة لم يتم إرسال الرسالة المراد البحث عنها في الويب أو تحديد محرك البحث سوف نستخدم الطريقة **sendError** سوف نقوم بكتابة هذا المثال على مبدأ فرق تسد حيث سنقسم البرنامج إلى أجراء أولاً سوف ننشأ صف SearchSpec وهو من أجل تسهل البرنامج لهذا الصف عضوين بيانيان من نمط String هما رابط محرك البحث URL واسم المحرك البحث name وسوف ننشأ أيضاً صف أخرSearchUtilities وفيه طريقة ستاتيكية (static) لبناء رابط (URL) لأي محرك بحث. الآن السرفلت SearchEngines:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.net.*;
      /** Servlet that takes a search string and a search
      * engine name, sending the query to
      * that search engine. Illustrates manipulating
      * the response status code. It sends a 302 response
      * (via sendRedirect) if it gets a known search engine,
      * and sends a 404 response (via sendError) otherwise.
      */
public class SearchEngines extends HttpServlet
{
      public void doGet(HttpServletRequest request,HttpServletResponse response)
            throws ServletException, IOException {
            String searchString = request.getParameter("searchString");
            if ((searchString == null) || (searchString.length() == 0))
            {
                  reportProblem(response, "Missing search string");
                  return;
            }
                  // The URLEncoder changes spaces to "+" signs and other
                  String searchEngineName =request.getParameter("searchEngine");
            if ((searchEngineName == null) || (searchEngineName.length() == 0))
            {
                  reportProblem(response, "Missing search engine name");
                  return;
            }
                  String searchURL = SearchUtilities.makeURL(searchEngineName,
            searchString);
            if (searchURL != null)
            {
                  response.sendRedirect(searchURL);
            } else
            {
                  reportProblem(response, "Unrecognized search engine");
            }
            }
            private void reportProblem(HttpServletResponse response,String message)
                  throws IOException
            {
                  response.sendError(response.SC_NOT_FOUND, message);
            }
}
```

صف SearchSpec

```
/** Small class that encapsulates how to construct a
* search string for a particular search engine.
*/
public class SearchSpec
{
	private String name, baseURL;
	public SearchSpec(String name,String baseURL)
	{
		this.name = name;
		this.baseURL = baseURL;
	}
/** Builds a URL for the results page by simply concatenating
* the base URL (http://...?someVar=") with the URL-encoded
* search string (jsp+training).
*/
	public String makeURL(String searchString)
	{
		return(baseURL + searchString);
	}
	public String getName()
	{
		return(name);
	}
}
```

صف SearchUtilities

```
/** Utility مع طريقة ستاتيكية لبناء رابط لأي محرك بحث*/
public class SearchUtilities
{
	private static SearchSpec[] commonSpecs =
	{ new SearchSpec("Google","http://www.google.com/search?q="),
	new SearchSpec("AllTheWeb","http://www.alltheweb.com/search?q="),
	 new SearchSpec("Yahoo","http://search.yahoo.com/bin/search?p="),
	new SearchSpec("AltaVista","http://www.altavista.com/web/results?q="),
	new SearchSpec("Lycos","search.lycos.com/default.asp?query="),
	new SearchSpec("HotBot","http://hotbot.com/default.asp?query="),
	new SearchSpec("MSN","http://search.msn.com/results.asp?q="),};
	public static SearchSpec[] getCommonSpecs() {
	return(commonSpecs);
	}
	/** Given a search engine name and a search string, builds
	* a URL for the results page of that search engine
	* for that query. Returns null if the search engine name
	* is not one of the ones it knows about.
	*/
	public static String makeURL(String searchEngineName,
	String searchString)
	{
		SearchSpec[] searchSpecs = getCommonSpecs();
		String searchURL = null;
		for(int i=0; i<searchSpecs.length; i++)
		{
			SearchSpec spec = searchSpecs[i];
			if (spec.getName().equalsIgnoreCase(searchEngineName)) {
			searchURL = spec.makeURL(searchString);
			break;}
		}
		return(searchURL);
	}
}
```

الآن السرفلت SearchEngineForm:الذي يعرض واجها لبحث عن كلمة في عدة من محركات البحث

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
/** Servlet that builds the HTML form that gathers input
* for the search engine servlet. This servlet first
* displays a textfield for the search query, then looks up
* the search engine names known to SearchUtilities and
* displays a list of radio buttons, one for each search
* engine.
*/
public class SearchEngineForm extends HttpServlet
{
	public void doGet(HttpServletRequest request,HttpServletResponse response)
		throws ServletException, IOException
	{
	response.setContentType("text/html");
	PrintWriter out = response.getWriter();
	String title = "One-Stop Web Search!";
	String actionURL = "SearchEngines";
	String docType =
	"<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
	"Transitional//EN\">\n";
	out.println (docType +"<HTML>\n" +"<HEAD><TITLE>" + title + "</TITLE></HEAD>\n" +
	"<BODY BGCOLOR=\"#FDF5E6\">\n" +"<CENTER>\n" +"<H1>" + title + "</H1>\n" +
	"<FORM ACTION=\"" + actionURL + "\">\n" +" Search keywords: \n" +
	" <INPUT TYPE=\"TEXT\" NAME=\"searchString\"><P>\n");
	SearchSpec[] specs = SearchUtilities.getCommonSpecs();
	for(int i=0; i<specs.length; i++)
	{
		String searchEngineName = specs[i].getName();
		out.println("<INPUT TYPE=\"RADIO\" " +"NAME=\"searchEngine\" " +
		"VALUE=\"" + searchEngineName + "\">\n");
		out.println(searchEngineName + "<BR>\n");
	}
	out.println("<BR> <INPUT TYPE=\"SUBMIT\">\n" +
	"</FORM>\n" +"</CENTER></BODY></HTML>");
	}
}
```

اللهم أن استودعناك حمص يا أرحم الراحمين