الفصل الأول

مقدمة في برمجة تطبيقات الانترنت

§ تطبيقات الانترنت

يمكن تعريف تطبيقات الانترنت على أنها برامج تستخدم برتوكول HTTP كبرتوكول أساسي في الاتصال . حيث يقوم بتوصيل المعلومات للمستخدمين بالاعتماد على الشبكة العنكبوتية لتصل إلى المستخدمين وتعرض على المتصفح بلغة HTML . وأيضا يمكن تسميتها بالتطبيقات المعتمدة على الشبكة العنكبوتية .

بمعنى يمكن اعتبار تطبيقات الانترنت كأي تطبيق حاسوبي اعتيادي في شركة أو مؤسسة حكومية تعتمد مفهوم الخادم والعميل Client-Server . حيث يكون الاتصال هنالك في الغالب معتمداً على شبكة محلية تربط المعلومات في تلك المنظومة . يتم إرسال الطلب من خلال العميل Client (**وهو عبارة عن جهاز حاسب يقع في نطاق الشبكة المحلية للمنظومة ويحوي تطبيقات العميل**) إلى الخادم Server (**وهو في العادة عبارة عن جهاز حاسب ذو مواصفات معينة يقع في نطاق الشبكة المحلية للمنظومة ويحوي تطبيقات خاصة بالخادم**) . والذي بدوره يقوم بالعمليات اللازمة لانجاز هذا الطلب وهو طلب العميل . وفي النهاية يقوم الخادم بإرسال النتائج إلى العميل . وكما رأيت السيناريو السابق فان العمليات تتم في نطاق الشبكة المحلية للمنظومة وبوجود برامج مساعدة على كلا من الخادم والعميل .

ولكن باستخدام تطبيقات الانترنت فان نطاق تبادل المعلومات يكون اكبر وذلك بسبب استخدام الانترنت كوسيلة للنقل وهو يمثل نطاقا اكبر مقارنه بنطاق الشبكة المحلية . بالإضافة فان العميل في هذه الحالة أي عند استخدام تطبيقات الانترنت لا يكون مقيد بنظام تشغيل معين أو بتطبيقات وبرامج مساعدة يلزم تحميلها لاستخدام تطبيقات المنظومة . وإنما يكفي وجود برنامج المتصفح أو ما يسمى بمخدوم الويب (Web Client) محملا على جهاز العميل . وفي هذه الحالة يمكن استخدام تطبيقات المنظومة أو جزء منها من أي جهاز متصل بالانترنت

§ الفرق بين الصفحات الثابتة والمتغيرة (Static Pages Vs Dynamic Pages)

في بداية الانترنت كانت صفحات المواقع الموجودة في ذلك الوقت هي صفحات مكتوبة فقط بلغة HTML . وكان كثيرا من المواقع يعتمد على وجود صفحات HTML قليلة تصل في بعض الأحيان إلى صفحة واحدة فقط . تلك الصفحات عادة يطلق عليها اسم الصفحات الثابتة (Static Pages) . وذلك لأنها ثابتة وجامدة حيث لا يمكن تغير محتواها آليا أو في وقت التنفيذ . وإنما يلزم للتغيير في محتواها من خلال استخدام برامج محرر النصوص وإعادة حفظها على خادم الويب مرة أخرى بعد التغيير . تعتبر الصفحات الثابتة هي نتاج لغة البرمجة HTML فقط لبناء صفحات مواقع الانترنت . حيث تعتبر لغة HTML لغة برمجة مناسبة للعرض الثابت فقط إذا ما استخدمت لوحدها . بمعنى أنه يمكن استخدام الصفحات الثابتة مثلا لعرض محتوى كتاب الكتروني على الانترنت . حيث يمكن كتابة صفحات الكتاب باستخدام لغة HTML فقط . وفي النهاية يمكن عرض الكتاب باستخدام برنامج المتصفح والتنقل بين الصفحات باستخدام ارتباط تشعبي معين Hyperlink .

وجود الصفحات الثابتة كان مهم للحصول على محتويات ثابتة ومناسبة للعرض ولكن مع تقدم التقنيات الداعمة للانترنت ولغة HTML ظهرت صفحات يمكن تغييرها في وقت التنفيذ وهي ما يطلق عليها بالصفحات المتغيرة Dynamic Pages

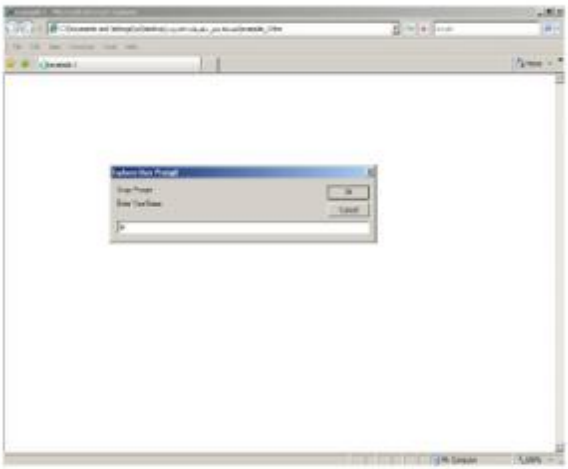
وجود الصفحات المتغيرة هي أيضا صفحات مكتوبة بلغة HTML كما هو الحال في الصفحات الثابتة إلا أنها تستخدم تقنيات ولغات أخرى داعمة للـــHTML . بحيث تعطي صفة التغيير على حسب طلب العميل . ويطلق على الصفحات المتغيرة أيضا اسم الصفحات المتفاعلة (Active Pages) وذلك لأن هذه الصفحات يمكنها التفاعل مع المستخدمين وتغيرها أثناء وقت التنفيذ . وذلك لأنه بإمكان هذه الصفحات التفاعل والتحدث مع المستخدمين .

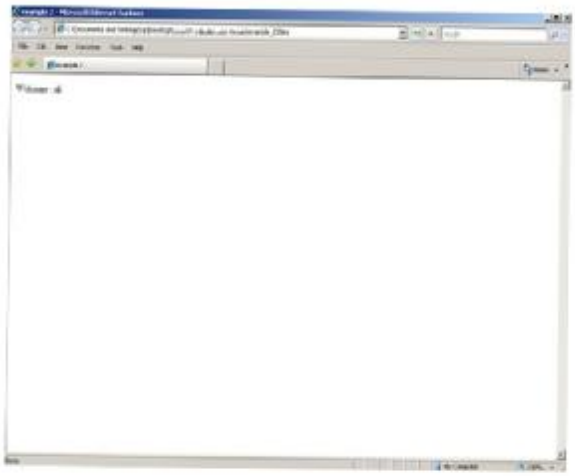
وفيما يلي مثال على الصفحات الثابتة :


```
<html>
<head>
<title>exapmle 1</title>
<head/>
<body>
h1 align="center">Institute of>
<public administration</h1
h2>Java web>
<programming</h2
<h2>Student Guide</h2>
<body/>
<html/>
```

في المثال السابق ترى استخدام لغة HTML فقط في كتابة صفحة ثابتة للعرض فقط وبدون أي تفاعل مع المستخدم . ولكي تميز الفرق بين الصفحات الثابتة و المتغيرة انظر إلى هذا المثال :

```
<html>
<head>
<title>example 2</title>
<head/>
<body>
<"script type="text/JavaScript>
name = prompt("Enter Your Name");
document.writeln("Welcome : " + name");
<script/>
<body/>
<html/>
```





في المثال السابق ترى استخدام HTML مع لغة JavaScript . وأدى ذلك إلى إنشاء صفحة متغيرة أثناء وقت التنفيذ . حيث يتغير محتوى الصفحة على حسب المدخلات من المستخدم . وذلك يظهر التفاعل بين المستخدم والصفحة المتغيرة .

§ مقارنة البرمجة في جهة الخادم والبرمجة من جهة العميل

البرمجة من جهة الخادم : وهي البرامج التي تكون موجودة على الخادم (Server) . ويقوم بتنفيذها باستمرار وذلك للرد على الطلبات القادمة من جهة العميل (Client) . حيث توجد الكثير من اللغات والتقنيات التي تساعد على عمل مثل هذه البرامج ونذكر منها التالي :

- **Php** §
- **Asp** §
- **CGI** §
- **Perl** §

البرمجة من جهة العميل : وهي البرامج التي يتم تنفيذها من خلال العميل حيث يقوم عادة العميل بتحميل (Download) هذه البرامج على شكل ملفات موجودة على الخادم ثم يقوم بتنفيذها . وابسط مثال هي البرامج المكتوبة بلغة HTML . حيث يقوم العميل بتحميل ملفات الـــHTML وتنفيذها باستخدام المتصفح بالإضافة إلى لغة HTML . توجد الكثير من اللغات والتقنيات التي تساعد على عمل مثل هذه البرامج ونذكر منها :

- **JavaScript** §
- **Java Applet** §
- **VB Script** §

§ Java - نظرة عامة على التقنيات

تعتبر لغة Java من اللغات المعروفة في دعم تطبيقات الانترنت وفي الشبكات عموماً . حيث قامت شركة Sun Microsystems وهي الشركة المطور للغة Java بإصدار العديد من البرمجيات على شكل حزم لتطوير البرامج بلغة Java ويطلق عليها اسم **JDK** (Java Development Kit) أو **SDK** (Software Development Kit) . حيث قسمت هذه الحزم إلى أنواع متعددة منها :

١. J2SE (Java 2 Standard Edition) : حيث تحتوي هذه الحزمة على البرامج اللازمة لتطوير تطبيقات Java الشخصية أو المحدودة . حيث تتضمن هذه الحزمة على API اللازمة لتطوير مثل هذه البرامج . وعادة تستخدم J2SE في الأجهزة الشخصية أو التطبيقات على مستوى مؤسسات صغيرة .

٢. J2ME (Java 2 Micro Edition) : حيث تحتوي هذه الحزمة على البرامج اللازمة لتطوير تطبيقات Java التي يمكن استخدامها على الهواتف النقالة أو المحمولة أو الأجهزة الالكترونية الصغيرة

٣. J2EE (Java 2 Enterprise Edition) : حيث تحوي الحزمة على البرامج اللازمة لتطوير تطبيقات Java على مستوى الشركات الكبيرة . حيث تتضمن هذه الحزمة على API اللازمة لتطوير مثل هذه البرامج الداعمة للشبكات عموماً وللانترنت خصوصاً . وعادة تستخدم J2EE في برمجة تطبيقات الويب (Web Application) لتنفيذها على الانترنت . حيث يمكن استخدام J2EE في تطوير العديد من تقنيات لغة Java الداعمة للويب . ومن أهمها :

- § Servlet
- § JSP (Java Server Pages)
- § JB (Java Beans)
- § EJB (Enterprise Java Beans)
- § Applet

§ التركيب العام لتطبيقات الانترنت بلغة JAVA

عادة تكون تطبيقات الانترنت المعتمدة على لغة Java مبنية على أساس تركيب (architectures) معينة من أشهرها :

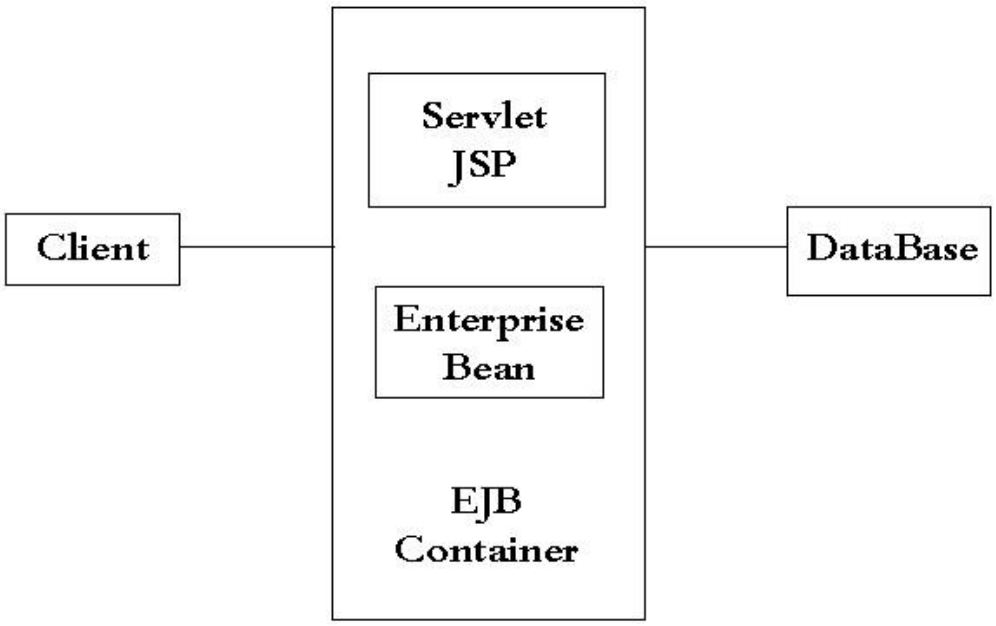
١. تركيب معتمد على وجود تقنيات Java مثل Servlet و JSP كتقنيات بينية بين الخادم والعميل . وفيما يلي رسم توضيحي لهذا البناء :

في هذا البناء يتم توظيف تقنيات Servlet و JSP لتصبح طرف أو طبقة بينية أو ما يسمى (Middle Tier) بين العميل Client و الخادم Server . حيث يطلق على مثل هذا التطبيق اسم تطبيق ثلاثي الطبقات (Three–Tiered Application)

- § طبقة العرض (Presentation Layer) : وتهتم بطريقة العرض للمعلومات عند العميل (Client)
- § طبقة منطقة العمل (Business Logic Layers) : وهي مجموعة من البرامج التي تهتم بمنطق عمل التطبيق توافقاً مع منطق عمل المنظومة .
- § طبقة البيانات (Data Layer) : هنا يكون الاهتمام بطريقة استخراج أو تعديل البيانات في قاعدة البيانات أو الملفات التي تحتوي على البيانات

يعد هذا البناء لتطبيقات الانترنت مناسبا للشركات أو المؤسسات الصغيرة أو متوسطة الحجم

٢ . (J2EE Application Architecture) : هذا النوع من التركيب يعتمد على خصائص إصدار J2EE في خلق تركيب أكثر تعقيداً في تطبيقات الانترنت من حيث الإنشاء . ولكن يتميز بمناسبته لتطبيقات الكبيرة في الشركات والمؤسسات . وقيم يلي رسم توضيحي لهذا البناء



وكما تلاحظ فان هذا التركيب يحوي على تقنيات متعددة من لغة Java مثل JB ، JSP ، Servlet ، EJB . يتناسب هذا التركيب مع التطبيقات الكبيرة التي تحتاج إلى الدقة والاتزان في تنفيذها .

§ النماذج في لغة HTML – HTML Forms

تعتبر النماذج في لغة HTML واجهه سهلة للمستخدم لإدخال البيانات باستخدام مثلاً مربعات النصوص وقوائم الاختيار ومربعات نعم / لا بالإضافة إلى الأزرار. وغير ذلك من الأدوات والعناصر التي توفرها لغة HTML من خلال النماذج . حيث يقوم المستخدم بملأ عناصر نموذج معين ومن ثم الضغط على زر الإرسال حيث تنتقل البيانات من مخدوم الويب (المتصفح) إلى برنامج موجود على خادم الويب وذلك عن طريق استخدام برتوكول HTTP .

يتم إدراج النموذج بالوسم Form حيث يحتوي الوسم Form على مجموعة من الخواص أهمها :
Action : وفيه تحدد عنوان البرنامج الذي سوف يستفيد من البيانات . فمثلاً يمكن أن يكون هذا البرنامج هو عبارة عن Java Servlet موجود على الخادم
http://localhost:8080/YourApplicationName/servlet/ClassName
Method : وفيها تحدد طريقة نقل البيانات . ومن أشهر الطرق المستخدمة طريقة post و get .

```
<html>
<head>
<title>example 3</title>
</head>
<body>
<form action="http://localhost:8080/YourApplicationName/servlet/ClassName" method="get">
</form>
</body>
</html>
```

§ استخدام طريقة POST و GET

عندما تريد إرسال المعلومات من متصفحك (متصفح الويب) إلى الخادم فانك في اغلب الأحيان تستخدم برتوكول HTTP في عمل ذلك . وينص برتوكول HTTP على أن البيانات ترسل في صورة أزواج . كل زوج هو عبارة عن اسم وقيمة . فمثلاً إذا أردت إرسال اسم المستخدم عبر برتوكول HTTP فانك ترسل العبارة التالية name=YourName . وهذا يسمى زوج لأنه يتكون من جزأين . الجزء الأول هو اسم المعلومة (name) والجزء الثاني هو المعلومة نفسها (YourName) . وإذا أردت أن ترسل أكثر من معلومة فانك تفصل بين أزواج المعلومات بعلامة & . فإذا أردت أن ترسل الاسم والبلد فانك ترسل العبارة التالية
. name=YourName&country=Yourcountry

إضافة إلى ما سبق فان هنالك شروطاً على إرسال البيانات في العناوين URL . ومن أهم الشروط هو انك لا تستطيع وضع مسافات بين الأزواج في العناوين URL . ولهذا فإن جميع المسافات في البيانات تحول إلى إشارة (+) . فإذا كان البلد مثلاً هو United States فان سطر البيانات سيكون كالتالي : name=YourName&country=United+States .
وهنالك أيضا شروط أخرى لتحويل الرموز الغير انجليزية والكثير من الأسماء التي يتم تطبيقها على البيانات حتى تصبح جاهزة للإرسال .

ينص برتوكول HTTP أيضا على أن البيانات ترسل بأكثر من طريقة كما ذكرت في السابق . ومن اشرها طريقة get و post . يتم إرسال البيانات بطريقة get بصورة بسيطة جدا . بحيث تكون عبارة عن جزء يضاف على عنوان البرنامج الموجود على الخادم بعد علامة الاستفهام ? . ومثلا إذا لديك برنامج يأخذ الاسم والبلد ويخزنها على الخادم . وكان عنوان البرنامج هو :
http://localhost:8080/YourApplicationName/servlet/ClassName
فان الطريقة get ستقوم فقط بإضافة البيانات إلى العنوان السابق بعد علامة الاستفهام . لصبح عنوان الصفحة الجديدة كالتالي :
……/servlet/ClassName?name=YourName&counrty=Yourcountry

ولابد أنك رأيت شيئا مثل ذلك وربما أكثر تعقيداً في متصفحك عندما تزور المواقع الضخمة وتجري عمليات البحث وغيرها كمثل Google . هذا الأمر يحدث عندما تستخدم طريقة get .

أما الطريقة الثانية فهي طريقة post وفيها يتم إرسال البيانات للخادم عن طريق HTTP ولكن هذه المرة لن تظهر البيانات في شريط العنوان (URL) . ولهذا فان الطريقتين تختلف عن بعضهما البعض ولكن طريقة get تعتبر أسهل وابسط بكثير من طريقة post إلى درجة انك تستطيع أن تركب البيانات بنفسك عن طريق وضعها في العنوان نفسه ورؤية النتائج تلقائياً . ولكن طريقة post لا يمكن لها العمل عن طريق النماذج حيث أنها لا تعمل إلا عن طريق مفتاح النماذج مثل زر input . وفي الحقيقة أن طريقة post تعتبر طريقة جيدة ومناسبة في حالة إرسال كمية كبيرة من البيانات مثل نص رسالة أو في الاتصالات السرية أيضاً . (ملاحظة : هذا لا يعني أنها آمنه جدا عبر الويب . وإنما تحتاج إلى تقنيات أخرى لتجعلها أكثر أمان)

الفصل الثاني

Servlet – نظرة عامة

§ استخدام الـServlet بالإضافة إلى ميزاته

يعتبر الـــServlet برامج تنفذ على جهة الخادم (Web Server) بحيث أنها تكون طبقة بينية (Middle Layer) بين الطلبات القادمة من جهة العميل Client (Web Browser) وقواعد البيانات أو تطبيقات أخرى على الخادم HTTP . وفيما يلي رسم توضيحي للطبقة البينية :

الـــ Servlet يقوم بالمهام التالية :

١ . قراءة البيانات المرسلة من المستخدم
عادة تكون هذه البيانات مدخلة من خلال النماذج (HTML) على صفحة الويب

٢ . البحث عن البيانات المتعلقة بالطلب (القادمة من العميل)
وتتضمن هذه البيانات معلومات عن المتصفح الخاص بالعميل مثل اسم المستضيف الخاص بالعميل وغير ذلك من معلومات أخرى

٣ . توليد النتائج (Generate Results)
عادة يقوم الـــServelt بمعالجة البيانات المرسلة من العميل وإرسال النتائج إليه العميل مرة أخرى . عادة تستخدم في قواعد البيانات وتقنيات أخرى لتوليد بعض النتائج .

٤ . تنسيق النتائج على شكل مستند (HTML)
في الغالب يقوم الـــServlet بتنسيق المخرجات أو النتائج على شكل ملف HTML متضمناً الرد للعميل

٥ . ضبط المعاملات (Parameters) المتعلقة بـــ(HTTP Response)
تتضمن هذه العملية إبلاغ متصفح العميل بنوع المستند المرسل إليه وغير ذلك من معلومات يحتاج إليها المتصفح)

٦ . إرسال المستند إلى العميل
هذا المستند يمكن إرساله بعده تنسيقات مثل نص (TEXT) كما هو الحال في ملفات HTML أو على هيئة صورة (مثل GIF Image) أو على صورة ملفات مضغوطة .

§ إعداد وتهيئة بيئة العمل لتطوير وتنفيذ Servlet و JSP

يلزم الحصول على برامج يمكن من خلالها تكوين بيئة عمل تطويريه على الحاسب الشخصي . حيث يمكن تطوير واختبار هذه التطبيقات أولا ومن ثم استخدامها على الانترنت أو بشكل رسمي :

ويمكن تقسيم هذه البرامج على الشكل التالي :

١ . Web Client : وهذا يتمثل بوجود متصفح الانترنت على الجهاز الشخصي مثل Internet Explorer أو Netscape وغيرها الكثير من المتصفحات . ويمكن الحصول على احد هذه المتصفحات من خلال شبكة الانترنت وبالتحديد من موقع الشركة المنتجة .

٢ . Web or Application Server : فمن المؤكد أن يقوم الجهاز الشخصي بما يقوم به الخادم للويب . وذلك يتمثل بوجود احد البرامج التي تقوم بهذا العمل مثل IIS أو TOMCAT .

٣ . Servlet & JSP Engine : وهي برامج تطبيق Servlet & JSP specification . ويمكن إضافة مثل هذه البرامج إلى خادم الويب . ومن أمثلة هذه المحركات برنامج Allaire JRun والذي يمكن أن يتوافق مع أكثر من خادم ويب مثل IIS و Tomcat .

٤ . Java Development Kit – JDK : حيث تحتوي هذه الحزمة على البرامج اللازمة لتطوير تطبيقات الانترنت .

ملاحظة : استخدام Tomcat يختصر الخطوتين ٢ و ٣

§ التركيب العام لـServlet :

```
؛*.import java.io
؛*.import javax.servlet
؛*.import javax.servlet.http

public class HelloWorld extends HttpServlet
}
، public void doGet(HttpServletRequest request
(HttpServletResponse response
} throws ServletException , IOException
{
{
```

هذا المثال يقوم بمعالجة الطلبات القادمة من العميل (Client) من نوع GET (GET Request) . ونعني بذلك عندما تتم مناداة الـــServlet مباشرة من خلال إدخال عنوان الـــServlet أو من خلال استخدام الوسم FORM وطريقة GET

حتى يكون لدينا Servlet يجب أن نعرف أن class يرث من HttpServlet من خلال استخدام exteands و إعادة تعريف طريقة doGet . ويمكن استخدام طريقة doPost إذا كان طلب العميل من نوع Post (نوع الـــMethod في الوسم Form) .

كلاً من doGet و doPost يستخدمان HttpServletRequest و HttpServletResponse . حيث يحتوي HttpServletRequest الطرق Methods اللازمة للحصول على المعلومات من العميل مثل : بيانات النموذج المرسل ، عناوين الطلب (Http Request Headers) ، اسم المضيف (Hostname).

أما HttpServletResponse فيحتوي Methods تساعد على تحديد المعلومات المرسلة للمخدوم . ومن أمثلة هذه المعلومات : Http Response Headers – HTTP Status Code . ولكن أهم ما يميز HttpServletResponse هو تمكن الحصول على PrintWrite والتي من خلالها يمكن إرسال محتويات المستند للعميل . تحتوي PrintWrite على الطريق Println والتي من خلالها يمكن بناء المحتوى أو الرد للمخدوم (العميل) .

وبما أن doGet و doPost يمكن أن تطرح exceptions فلابد من استخدام جملة throws كما في المثال السابق .

ومن خلال صنع أي Servlet فلابد من استدعاء عدد من الـــpackages الضرورية التي بدونها لن يعمل أي ServletClass . منها :

- Java.io.* من اجل PrintWrite والتي من خلالها إرسال يتم الرد للعميل
- Javax.servlet.* من أجل HttpServlet
- Javax.servlet.http.* من اجل HttpServletRequest و HttpServletResponse

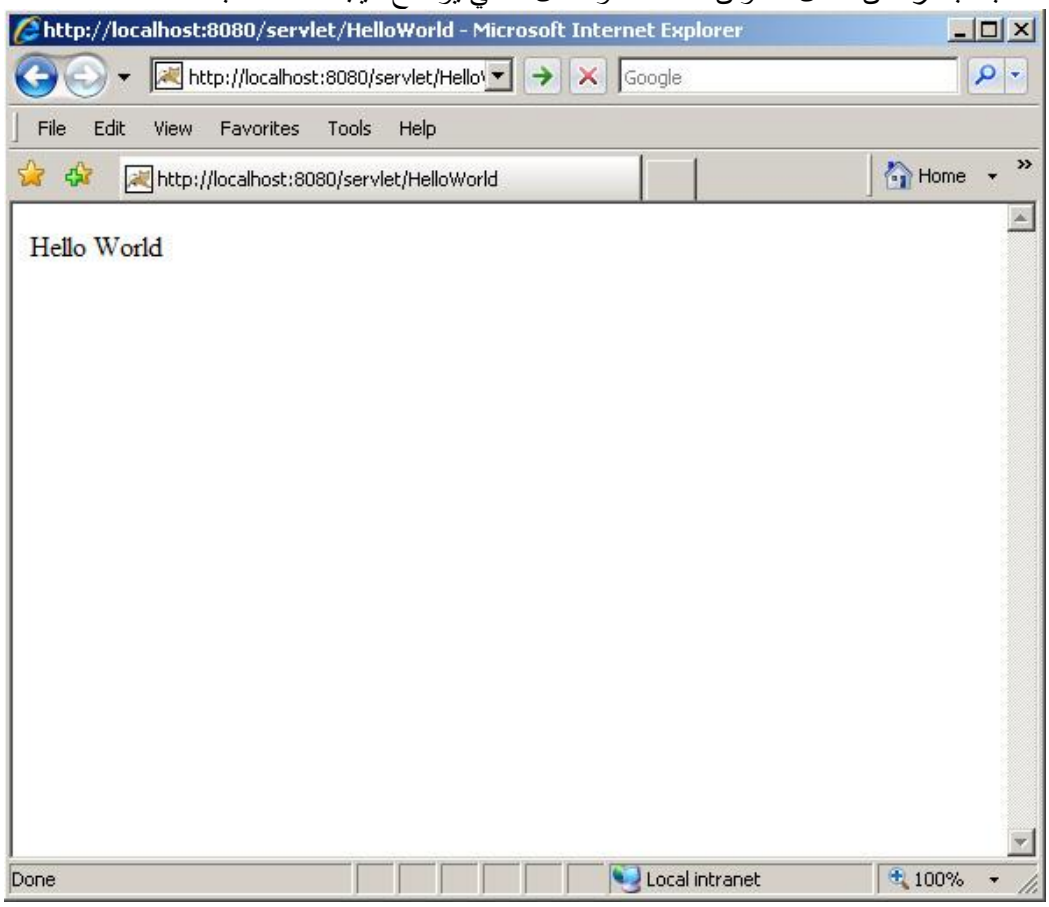
تطبيق ١ : توليد نص من خلال Servlet
HelloWorld.java

```
؛*.import java.io
؛*.import javax.servlet
؛*.import javax.servlet.http

public class HelloWorld extends HttpServlet
}
، public void doGet(HttpServletRequest request
(HttpServletResponse response
} throws ServletException , IOException

؛()PrintWriter out = response.getWriter
؛("out.println("Hello World
{
{
```

يقوم هذا الــServlet بإرسال جملة Hello World كنص للعميل حيث استخدم التركيب العام للــServlet بالإضافة إلى اختيار الطريقة doGet لمعالجة طلب العميل . وذلك حتى يرسل للعميل الطلب مباشرة من خلال العنوان URL . والشكل التالي يوضح نتيجة هذا الطلب



هنا استخدم out.println("") من اجل إعداد مستند للرد . حيث يتم إضافة الجمل إليها . وفي هذا المثال أضفنا جملة Hello World للمستند المرسل للعميل فقط . وفي النهاية يكون الرد محتوي على الجملة فقط .

لكي تقوم بتنفيذ هذا الـــServlet يجب إتباع التالي :

١ . حفظ الـــServlet باسم HelloWorld.java
٢ . ترجمه البرنامج السابق من خلال Javac . سواء كان من خلال Dos Command أو من خلال برامج أخرى IDE
٣ . بعد الترجمة ينشأ ملف من نوع (class.) . وسوف يوضع الملف في مجلد الـــclasses في خادم الويب . وفي هذه الحالة سوف يكون على النحو التالي :
C:\apache-tomcat-5.5.17\webapps\ROOT\WEB-INF\classes
٤ . الآن ومن خلال المتصفح يمكن كتابة العنوان التالي :
http://localhost:8080/servlet/HelloWorld

ملاحظة : عند إنشاء Servlet أو القيام بالتعديل عليه فلابد من إطفاء الـــTomcat وتشغيله مرة أخرى لكي يقوم الخادم بالتعرف عليه و على التعديلات التي حدثت على ملف الـــServlet

الفصل الثالث

استخدام الـServlet

من خلال الفصل السابق قد عرفت التركيب الأساسي للـــServlet بالإضافة إلى توليد النصوص والجمل . وهنا سوف ترى كيف يتم توليد النتائج ولكن هذه المرة عن طريق إرسال صفحات الرد بلغة HTM

تطبيق ٢ : توليد HTML من خلال Servlet
HelloWorld2.java

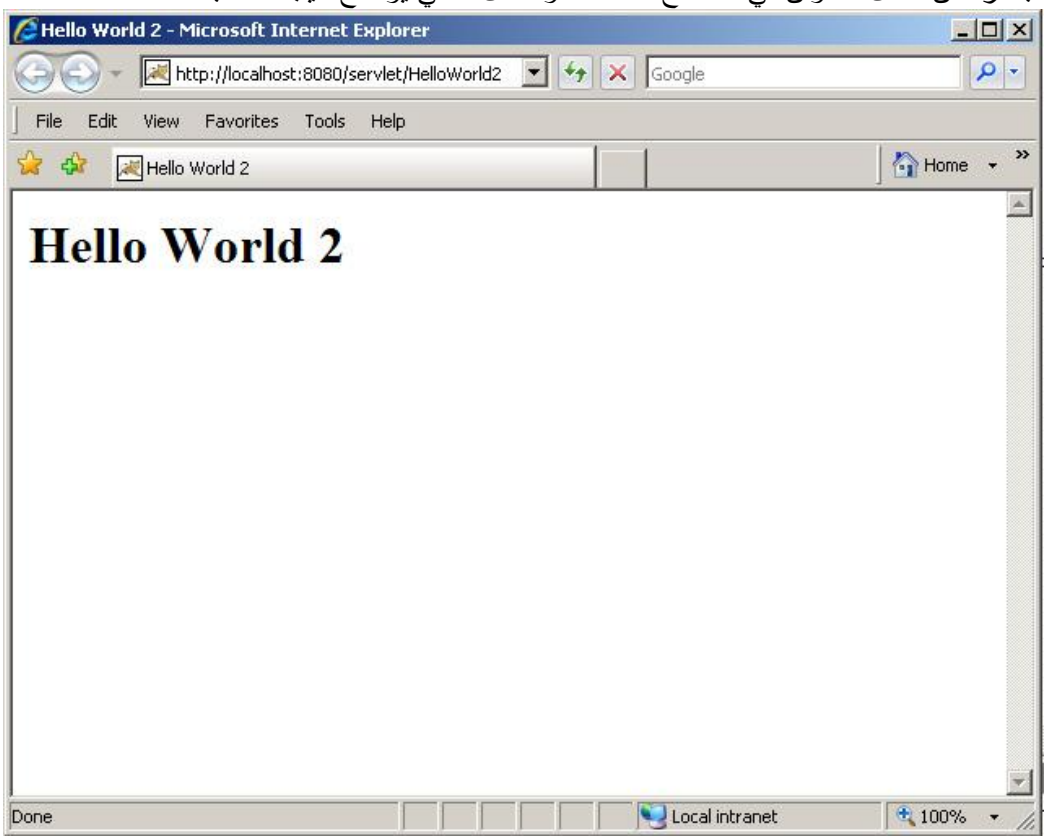
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld2 extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {

response.setContentType("text/html");
PrintWriter out = response.getWriter();
```

```
String docType ="<!DOCTYPE HTML PUBLIC \"-//W3C//DTD
HTML 4.0 " +"Transitional//EN\">\n";
out.println(docType +
"<HTML>\n" +
"<HEAD><TITLE> Hello World 2 </TITLE></HEAD>\n" +
"<BODY>\n" +
"<H1> Hello World 2 </H1>\n" +
"</BODY></HTML>");
}
}
```

يقوم هذا الـــServlet بإرسال جملة Hello World 2 للعميل مع إعطاء تنسيق معين للجملة باستخدام HTML . بالإضافة إلى إرسال نوع المستند للمتصفح . حيث استخدم التركيب العام للـــServlet مع اختيار الطريقة doGet لمعالجة طلب العميل . وذلك حتى نسمح للعميل لطلب الرد مباشرة من خلال العنوان في التصفح URL . والشكل التالي يوضح نتيجة الطلب



وهنا استخدمت reponse.setContentType("text/html") من اجل إعداد المستند للرد لتكون محتوياته نص مجرد و HTML Code . وهنا استخدمت String docType لحفظ نوع المستند المرسل للمتصفح حتى يتمكن المتصفح من معرفة نوع المستند المرسل إليه من اجل ترجمته وتنفيذه على هذا الأساس .

أيضا استخدمت طريقة out.println(); من اجل بناء المستند المرسل للعميل وكذلك تنسيق المستند بشكل صفحة HTML بوجود الوسوم الأساسية لصفحة HTML .

§ دورة حياة الServlet

يمر الــServlet بمراحل منذ إنشاءه إلى نهايته . حيث ينشأ حالة (instance) عند إنشاء الــServlet . وتتم مناداة طريقة init() التي بدورها تحوي code يتم تنفيذه لمرة واحد عند إنشاء الــServlet . وتعتبر مناسبة لإعدادات الــServlet عند إنشاءه . بعد ذلك يتم إنشاء thread لكل طلب (request) من المستخدم . حيث تقوم كل thread بمناداة طريقة service() . بعد ذلك تقوم طريقة service() بمناداة طريقة doPost أو doGet أو غيرها من الطرق (doXxx) . وذلك على حسب نوع طلب http المراد معالجته . أخيرا قبل أن يتم إنهاء الــServlet من قبل الخادم يتم مناداة طريقة Destroy للقيام بالأعمال النهائية للــServlet

§ استخدام الطريقة init()

تتم مناداة init() لمرة واحدة عند إنشاء الــServlet . وتعتبر مناسبة لإعدادات الــServlet عند إنشاءه لأول مرة . ولا تتم مناداتها مرة أخرى عند كل طلب من المستخدم . ويمكن تعريف طريقة init() على الشكل التالي :

```
Public void init () throws ServletException {

}
```

إن احد أهم أسباب استخدام طريقة init() هي قراءة المعلومات عن الخادم المضيف للــServlet

§ استخدام طريقة service()

لكل مرة يستقبل الــServlet طلب من المستخدم تنشأ thread . حيث تقوم كل thread بمناداة طريقة service() . حيث تقوم service() بفحص نوع الطلب من (http request type) حيث يمكن أن تكون GET أو POST أو DELETE أو غيرها . ومن ثم تقوم طريقة service() بمناداة الطريقة المناسبة لنوع الطلب المراد معالجته مثل doGet أو doPost أو doDelete أو غيرها من طرق أخرى .

في غالب الأحيان يقوم الــServlet بمعالجة طلب العميل باستخدام طرق doXxx وقد تكون المعالجة لكلا الطرفين متطابقة . ولهذا يمكن حلة من خلال إعادة كتابة طريقة service() ولكن هذا غير مستحب أبدا . وبدلا من ذلك يمكن إعادة كتابة طريقة doGet ومن مناداتها من خلال طريقة doPost أو بالعكس إذا أمكن . ومثال هذه الطريقة كالتالي :

```
Public void doGet (HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
```

```
//Servlet Code
}
Public void doPost (HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
doGet (request , response );
}
```

§ استخدام طرق doXxx ، doPost ، doGet

هذه الطرق هي الطرق الرئيسية لمعالجة الطلبات عن طريق الـــServlet . وهي شائعة جدا بحيث أنها تستخدم على نحو ٩٩% . وهذا يعني أن معظم الحالات سوف تقوم بإعادة كتابة طرق doGet أو doPost عند إنشاء أي Servlet . ويمكن للـــServlet أيضا أن يخدم أنواع أخرى من الطلبات باستخدام طرق أخرى مثل doDelete أو doPut ولكنها غير شائعة لهذا الحد .

§ استخدام طريقة destroy

الخادم أو الـــServer سوف يقوم بإنهاء Servlet معين من الذاكرة وذلك لأسباب عديدة منها : أن يقوم مدير الخادم (Server administrator) بإصدار أمر لإنهاء الـــServlet أو بسبب وجود غير مبرر للـــServlet في الذاكرة (لا فائدة منه) . ولهذا فان الـــServlet يقوم بمناداة طريقة destroy قبل إنهاءه . وذلك للقيام كما تسمى بعمليات التنظيف اللازمة مثل إغلاق الاتصال بقاعدة البيانات .

§ طلب العميل من خلال نموذج بيانات (Form Data)

معظم البيانات المرسلة من خلال المستخدم تكون عن طريق ملئ نموذج بيانات في صفحة ويب ومن ثم إرسالها إلى الخادم (Server) . حيث يقوم الـــServlet فيما بعد باستقبال هذه البيانات وتحليلها ومعالجتها ومن ثم الرد عليها . وفي الحقيقة شكل هذه البيانات قد يختلف باختلاف طريقة الإرسال المستخدمة في نموذج البيانات المرسل من المستخدم . فعند استخدام GET تكون هذه البيانات متصلة بعنوان الـــServlet (URL) . حيث تكون البيانات المرسلة ف الجزء الواقع بعد علامة الاستفهام ? . وكما أيضا يمكن استخدام طريقة POST ولكن في هذه الحالة يتم إرسال البيانات بشكل مخفي للـــServlet . (راجع استخدام طريقة POST و GET)

قراءة بيانات النموذج من خلال الــServlet

من احد أهم مزايا الــServlet انه يستطيع الحصول على البيانات في أي نموذج مرسل . وهذه العملية سهلة وبسيط وذلك من خلال استخدام طريقة getParameter من HttpServletRequest . سواء كانت البيانات المرسلة بطريقة GET أو POST . ومثال ذلك إذا أردنا أن نرسل الاسم إلى الــServlet فسوف يكون سطر التعليمة كالتالي :

```
String name = request.getParameter (name);
```

وهذه الطريقة سوف تستقبل قيمة واحدة من نوع String وهي المعامل name من خلال getParameter مع الأخذ بالاعتبار تطابق الأسماء أو الأحرف (case sensitive) . ولهذا سوف تقوم هذه الطريقة بإرجاع قيمة واحدة إن وجدت . إذا كان المعامل name موجود ولكن بدون قيمة فان القيمة سوف تكون فارغة (null) . بمعنى انه إذا تم إرسال النموذج إلى الــServlet وكان حقل الاسم فارغ فان الــServlet سوف يستقبل هذا النموذج ويبحث عن المعامل name ويستخرج قيمته . وبما أن النموذج أرسل إلى الــServlet ولكن بدون قيمة فسوف تكون قيمة هذه المعامل (name) فارغة (null) .

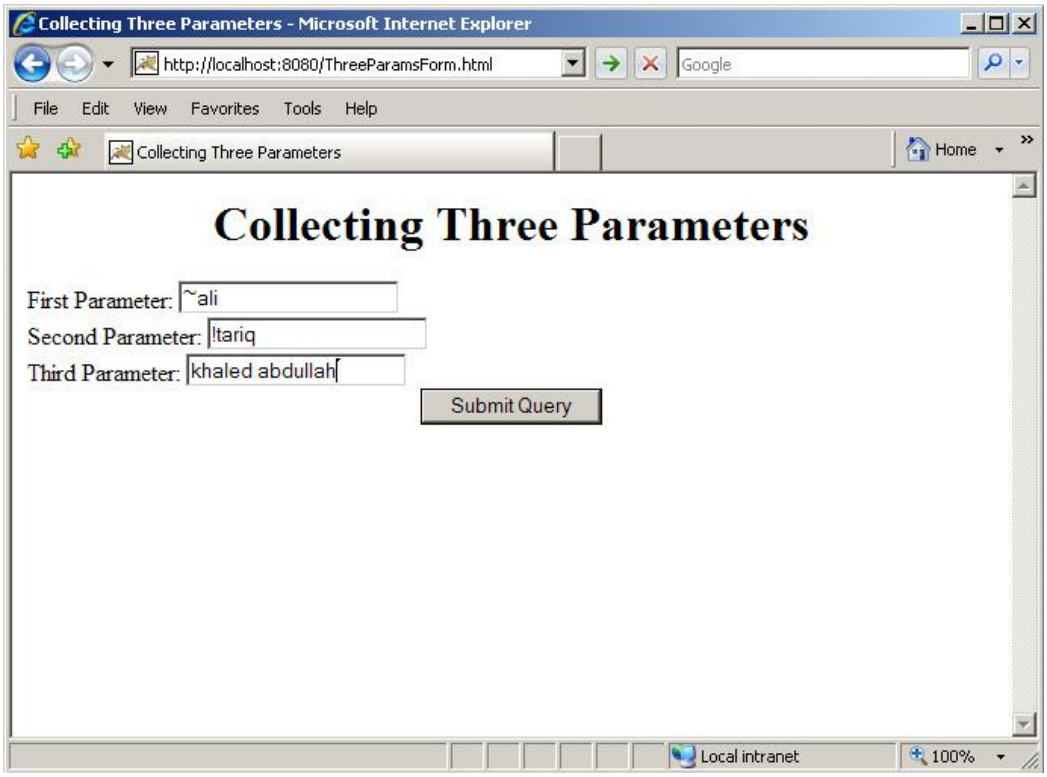
وفيما يلي تطبيق يقوم بقاؤه ثلاث قيم مرسلة من نموذج بيانات يحتوي على ثلاث معاملات وهي : param1 و param2 و param3 . .

تطبيق ٣,١ : قراءة المعاملات المرسلة من نموذج

ThreeParamsForm.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Collecting Three Parameters</TITLE>
</HEAD>
<BODY>
<H1 ALIGN="CENTER">Collecting Three Parameters</H1>
<FORM ACTION="http://localhost:8080/servlet/ThreeParams">
First Parameter: <INPUT TYPE="TEXT" NAME="param1"><BR>
Second Parameter: <INPUT TYPE="TEXT" NAME="param2"><BR>
Third Parameter: <INPUT TYPE="TEXT" NAME="param3"><BR>
<CENTER>
<INPUT TYPE="SUBMIT">
</CENTER>
</FORM>
</BODY>
```

وكما تلاحظ في ملف HTML ومن خلال النموذج Form أضفنا عنوان الــServlet الذي سوف يقوم بمعالجة بيانات النموذج من خلال الإشارة إلى مكان الــServlet الذي سوف يقوم بهذه المهمة وذلك عن طريق إدراج الــURL للصفة Action . بالإضافة إلى إدراج طريقة الإرسال عبر البروتوكول HTTP وهي طريقة Get . وفي الوسم Form تم إدراج ثلاثة حقول من نوع TEXT وأعطيت لها أسماء param1 – param2 – param3 . وفيما يلي تنفيذ هذا البرنامج



بعد النقر على زر Submit Query فان البيانات الموجودة في الحقول كما هو واضح سوف تنتقل إلى الـــServlet وبالتحديد إلى البرنامج المسمى ThreeParameters كما في التالي :

تطبيق ٣,٢ : قراءة المعاملات المرسلة من نموذج
ThreeParameters.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ThreeParams extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html");
PrintWriter out = response.getWriter();
String title = "Reading Three Request Parameters";
out.println(
"<BODY BGCOLOR=\"#FDF5E6\">\n" +
"<H1 ALIGN=CENTER>" + title + "</H1>\n" +
"<UL>\n" +
" <LI><B>param1</B>: "
+ request.getParameter("param1") + "\n" +
" <LI><B>param2</B>: "
+ request.getParameter("param2") + "\n" +
" <LI><B>param3</B>: "
+ request.getParameter("param3") + "\n" +
" <LI><B>param4</B>: "
```

```
+ request.getParameter("param4") + "\n" +
"</UL>\n" +
"</BODY></HTML>");
```

من إعداد

خالد الحامد