قُبلت للنشر في مجلة جامعة البعث (حمص-سوريا) بالمجلد 38 لعام 2016.

تحقيقُ تطبيقات تجارةٍ الكترونية مُتعددةُ الطبقات و تقييم تأثّر الأداء بالتقنيات المستخدمة

الدكتورة زينب راتب خلوف*

الملخّص

تطبيقاتُ التجارة الالكترونية متعددة الطبقات هي تطبيقات موزّعة يقسّمُ فيها منطقُ التطبيق إلى مكوناتٍ تُوزّع على أجهزةٍ عدّة حسب الطبقة التي تنتمي إليها هذه المكونة [2] وتقدّم خدمات تجارة الكترونية للمستخدمين مثل التسّوق عبر الانترنت.

مع اكتساب هذه التطبيقات أهميةً مُتزايدة، ظهرت بيئاتُ تطوير مثل إصدار المؤسسات من جافا(Java EE (Java Enterprise Edition تُبسّط عملَ المبرمج بطرقٍ عدّة ومنها إمكانية استخدام ترميزات (Annotations) لتعريف الاحتياجات الوظيفية للتطبيق من حماية أو إدارة للمُناقلات دون الدخول بالتفاصيل البرمجية لهذه المتطلبات.

الخطوة الأولى في بناء التطبيق هي تصميم بنية التطبيق بشكلٍ صحيحٍ و اختيارُ المكونات الأنسب لضمانِ أداء مقبول. يقدّم هذا البحث في قسمِه الأوّل عرضاً لمُكونات تطبيق تجارة الكترونية متعددُ الطبقات في بيئة JEE 7، وفي قسمِهِ الثاني مقارنة نظرية وتجريبية بين تقنيتين يمكنُ استخدامهما في طبقة العرض: JavaServer Pages (JSP) و JavaServer Faces(JSF).

الكلمات المفتاحية: تطبيقات التجارة الالكترونية متعددةُ الطبقات، تطبيقات الوب، التطبيقات الموزعة، تقييم الأداء، JEE7، Apache JMeter، أداة القياسab، GlassFish، JSP،JSF.

* أستاذ مساعد-قسم هندسة الشبكات و النظم الحاسوبية- كلية الهندسة المعلوماتية-جامعة البعث

Implementing Multi-Tiered E-commerce Applications and Assessing the Impact of the Different Techniques on Performance

Dr. Zainab R. Khallouf*

ABSTRACT

Multitiered ecommerce applications are distributed applications where application logic is divided into components according to function. These components are installed on different machines, depending on the tier to which the application component belongs [2], additionally, these applications provide ecommerce services like online shopping.

With the increasing importance of these applications, new frameworks like Java EE were developed to make the developer's task easier by providing annotations to define functional requirements like security and transactions management.

Correctly designing the application and choosing the suitable components both considered to be the first step in developing these applications.

This paper begins by presenting the parts of multitiered ecommerce application in Java EE7, then compares between two possible presentation layer technologies: JavaServer Pages (JSP) and JavaServer Faces (JSF).

Key words:

Multitiered ecommerce applications, web applications, distributed applications, performance evaluation, JEE7, Apache JMeter, ab measurement tool, GlassFish, JSF, JSP.

* Associate Professor, department of systems and computer networks engineering, faculty of informatics engineering, Al-Baath University, Homs, Syria.

مقدمة والهدف من البحث:

تُعرّف التجارةُ الإلكترونية بأنّها مجموعةُ المناقلات التجارية التي تتمُّ إلكترونياً عن طريق الانترنت و تشمل تطبيقات مثل التسوق عبر الإنترنت، الدفع الإلكتروني، و تطبيقات إدارة العلاقة مع الزبائن (Customer relationship management (CRM)). أمّا تطبيقات التجارة الالكترونية متعددةُ الطبقات فهي تطبيقات موزّعة يقسّم فيها منطق التطبيق إلى مكوناتٍ تعملُ على أجهزةٍ عدّة حسب الطبقة التي تنتمي إليها هذه المكونة وتقدّم خدمات تجارة الكترونية.

يُقسّم التطبيق غالباً إلى أربعة مستويات (طبقات): طبقةُ المستخدم (Client–tier) و تضمّ المكونات التي تعملُ على جهازِ المُستخدم مثل صفحات الوب المُولّدة ديناميكياً، طبقة الوب (Web–tier) وتضمُّ المكونات المسؤولة عن توليد الصفحات و الإستجابة للطلبات التي يُرسلها المستخدم عن طريق المُستعرض، طبقة العمل (Business–tier) وتضمّ مكونات المنطق الوظيفي للتطبيق مثل Enterprise JavaBeans وكلتا الطبقتين تعملان على مخدّم مثل Glassfish يتميز باستضافته لمكونات وب و لمكوناتٍ وظيفية؛ أما الطبقة الرابعة فتضمُ نظم معلومات المؤسسة[2].

يعدّ تصميم بنية التطبيق بشكلٍ صحيح و اختيار المُكونات الأنسب لضمان أداء مقبول الخطوة الأولى في بناء هذه التطبيقات.

يركزُّ البحث على أحد تطبيقات التجارة الالكترونية المستخدمة بشكل كبير ألا وهو التسوق عن طريق الانترنت و يهدف لتسليط الضوء على بنية التطبيق و على مكوناته المختلفة مع مقارنة للأداء مبنية على عدة معايير تُمكّن المطوّر من اختيار البنية الأنسب.

تُقسَّم بقيةُ الورقةِ البحثية كالتالي: سنستعرضُ في القسمِ الثاني لمحة عن أنواع التجارة الالكترونية. نُقدِّمُ في القسم الثالث عرضاً لبنية تطبيق متعدد الطبقات في بيئة JEE 7 ونبيّن مكوناته المُختلفة، أمّا في القسم الرابع فنقدم مقارنةً نظرية بين JSP وJSF، تقنيتين يمكن استخدامهما في طبقة العرض. يُقدّم القسم الخامس مقارنة بين هاتين

التقنيتين من خلال تقييم للأداء باستخدام الأداة Apache JMeter و الأداة ab. في القسم السادس نُلخص نتيجة البحث ونعرض التوجهاتِ المستقبلية.

أنواع التجارة الإلكترونية:

للتجارة الالكترونية أنواعٌ عدّة ولعلّ أهمّها[1]:

- التجارةُ الالكترونية من شركة إلى مستخدم (Business-to-Consumer) و أحد أمثلتها التقليدية شركة Amazon.com التي بدأت أساساً كموقع لبيع الكتب للمستخدمين عن طريق الانترنت.
- التجارة الالكترونية من شركة إلى شركة (Business-to-Business) و المُناقلات التجارية في هذه الحالة تتمّ بين الشركات.
- التجارة الالكترونية من شركة إلى حكومة (Business to Government) ويمكن اعتبارها كحالة خاصة من النوعِ السّابق لكنّ المُناقلات تتمُّ بين شركة وقطاعات عامة.
- التجارة الالكترونية من مستخدم إلى مستخدم (to Consumer Consumer) و هنا تتمُّ المناقلات بين المستخدمين أنفسهم حيث يمكن لمستخدم أن يبيع أو يشتري من مستخدم آخر مُباشرة من خلال موقع وسيط يسمى صانعُ سوقٍ على الانترنت (Online market maker) مثل eBay.

أمّا حسب التكنولوجيا المستخدمة فيوجد أنواع مثل التجارة المحمولة M-commerce وتُعنى بالمُناقلات التجارية التي تتمّ من خلال الأجهزة النقّالة والمحمولة، و F-commerce وتُعنَى بإجراء المُناقلات التجارية من خلال موقع Facebook.

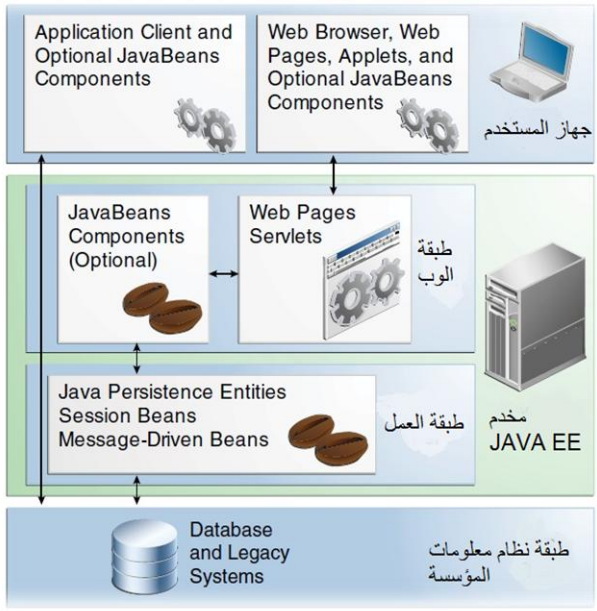
يركّز هذا البحث على النوع الأول وبالتحديد على تطبيق تجارة الكترونية متعدد الطبقات يقدّم وظائف مثل استعراض فئات المُنتجات، سلّة مشتريات و تسجيل لبيانات المستخدم حتى ينتقل إلى الدفع الالكتروني.

نظم التجارة الالكترونية المتعددة الطبقات في بيئة Java EE [2]:

يوضحُ الشكل 1 الطبقات المختلفة في تطبيق Java EE و المكونات في كل طبقة. نميّز بينَ المكونات الآتية:

- مكونات طبقة المستخدم (Client-tier components) تعملُ على جهاز المستخدم مثل صفحات الوب المولّدة ديناميكياً، تطبيقات جافا، أو مكونات مثل التطبيقات المصغرة (Applets).
- مكونات طبقة الوب (Web-tier components) تعملُ على مخدّم Java EE server وتضم مكونات مثل Servlets، (JSF) JavaServer Faces technology،(JSP) JavaServer Faces Facelets، و JavaBeans. يتبع تصميم طبقة الوب عادةً قالبَ التصميم MVC والذي سنبيّن مكوناته وفائدته لاحقاً.
- مكونات طبقة العمل (Business-tier components) تعملُ على مخدّم Java EE server وتضمّ مكونات مثل:(EJBs) Enterprise JavaBeans، خدمات الوب JAX-RS RESTful، و الصفوف المحققة للاستمرارية Java Persistence API.
- برمجيات نظام معلومات المؤسسة (Enterprise information system tier-(EIS)) تعمل على مخدّم نظام المعلومات.

يُعتبر تطبيق Java EE عادة ثلاثيُّ الطبقات لأنّ مكوناته توّزع على ثلاثة أجهزة: جهازُ المستخدم، الجهاز المضيف لمخدّم Java EE و الأجهزة التي تستضيف قواعد البيانات أو نظم المعلومات المتعلقة بالمؤسسة.



الشكل 1: مكونات تطبيق موزّع و متعددُ الطبقات في بيئة Java EE [2]

عرّفت منصةُ JAVA EE مجموعة من المُكونات التي تعمل في طبقة العمل على مخدّم Java EE كما يبيّن الشّكل 1، أما في طبقة الوب ورغم أنّ المعيار يوصي باستخدام إطار JSF إلا أنّ استخدام JSP ممكنٌ أيضاً **ولذلك نركزُ في هذا البحث على المقارنة بينَ هذين التقنيتين اعتماداً على عدة معايير تُمكّن المطوّر من اختيار البنية الأنسب.**

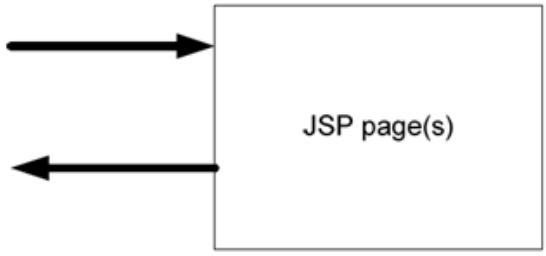
قالب التصميم Model-View-Controller (MVC) [3]:

يعرّف قالب التصميم (Design pattern) بأنه حلٌ عام، قابل لإعادة الإستخدام، لمشكلة شائعة الحدوث في تصميم البرمجيات وهو لايعتبر تصميماً نهائياً يمكن أن يُحوّل مباشرة الى كود وإنمّا توصيف أو قالب لحل مشكلة يمكن أن يُستخدم في حالاتٍ مختلفة. تُظهِر قوالب التصميم الغرضيةُ التوجّه العلاقات والتفاعلات بين الصفوف والأغراض بدون تحديد صفوف أو أغراض التطبيق النهائي.

يُستخدم قالب التصميم MVC أو مايطلقُ عليه أيضاً (نموذج الوب الثاني WEB Model 2) عادة في طبقة الوب في تطبيقات Java EE.

في نموذج الوب الأول (WEB Model 1)، يُعهَدُ إلى صفحة أو مجموعة من صفحات JSP أو صفوف (Servlets) بكلّ المنطقِ الوظيفي للتطبيق فتستقبل وتعالج الطلبات

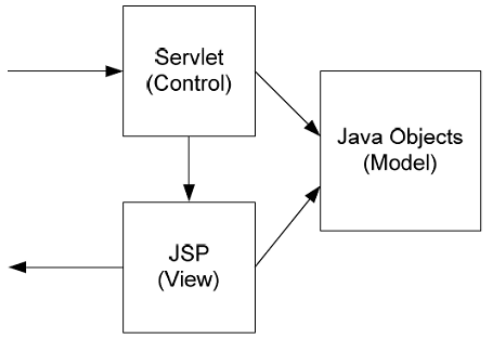
من المستخدم و من ثم تعيد الإجابة كما هو مبين في الشّكل 2. تتضمنُ الصفحات في هذه الحالة وسمات HTML بالإضافة إلى تعليمات جافا. لايوجد فصل واضح بين طبقات التطبيق في النموذج الأول مما يجعل من توسعة التطبيق أو اكتشاف الأخطاء فيه عملية مُعقدة، نظراً لاستخدام HTML مع كود Java في نفس صفحة JSP (أو في نفس Servlet)، كذلك لايتناسب هذا النموذج مع فصل المهام الذي باتت تعمل وفقه مؤسسات تطوير البرمجيات.



الشكل 2: نموذج الوب الأول (WEB Model 1)[3]

أما في النموذج الثاني (MVC) فيتم الفصل بين المَهام والوظائف المختلفة. يتكون MVC من ثلاثة أقسام: النموذج (Model) ويمثّل البيانات و طُرق الوصول إلى هذه البيانات، المتحكم (Controller) ويُمثل صف (Servlet) مسؤول عن سلوكِ التطبيق (Application behavior) وبسلوك التطبيق يقصد وظائف مثل اختيار الصفحة التي سيتمّ عرضها أو اجراء تغييرات على النموذج، في حين يتولى القسم الثالث ألا وهو العرض (View) منطق التقديم (Presentation logic).

في هذه البُنية، يتمّ استلام طلب HTTP القادم من المستعرض من قبل المتحكم (Controller) (Servlet) الذي يتضمن كود جافا فقط. يتواصل المتحكم مع النموذج بهدف الوصول إلى البيانات وعمليات الوصول إلى البيانات تشمل القراءة، الكتابة، التعديل أو الحذف. حالما تتم معالجة الطلب يتم توجيه العرض إلى الصفحة المناسبة والتي تتضمن HTML وبعض وسمات JSP التي تعرضُ المعلومات من خلال الوصول الى النموذج (Model) وبعض متحولات الجلسة إن لزم، كذلك يمكن للنموذج أن يُخطرَ الصفحات بأي تغيير يطرأ على حالته (الشكل 3).



الشكل 3: مكونات قالب التصميمMVC [3]

صفحات(JSP) JavaServer Pages:

تسمحُ تقنية JSP بإنشاء محتوى وب ثابت أو ديناميكي من خلال مجموعة من الميّزات:

- لغة لتطوير صفحات JSP، و هي ملفات نصية توصّف كيفية معالجة طلب من المستخدم و تكوين الرد.
- لغة تعبيرية (Expression language) للوصول إلى الأغراض عند المخدم.
- آلية لتعريف توسعة للغة JSP.
- مجموعة من واجهات برمجة التطبيقات (APIs) يمكن أن تستخدم من قبل مطوّري مستوعب الوب (Web container).

عندما يصلُ طلب من المستخدم لصفحة JSP، تمرّ هذه الصفحة بالمراحل التالية عند المخدّم والتي تُسمى مُجتمعة: دورة حياة JSP (JSP lifecycle).

- الترجمة (Compilation): بداية تتمّ ترجمةُ الصفحة إن لم تكن قد تُرجمت من قبل أو إن كانت قد تغيّرت منذ آخر ترجمة. تتضمن عملية الترجمة ثلاثُ خطوات:
  - تحليل ملف JSP.
  - تحويل صفحة JSP إلى صف (Servlet).
  - ترجمة الصف (Servlet).
- التهيئة (Initialization) من خلال استدعاء الطريقة ()jspInit قبل معالجة أي طلب.

- التنفيذ (Execution) من خلال استدعاء العملية (jspService_() والتي تأخذ وسيطين HttpServletRequest و HttpServletResponse، لتكون مسؤولة عن معالجة طلب المستخدم و توليد الرد لكل طلب.
- تحرير الموارد (JSP Cleanup) من خلال استدعاء الطريقة jspDestroy().

تقنية JavaServer Faces(JSF)[2]:

- تمثّل تقنية JSF إطار مكونات قابلة لتغيير خصائصها (Configurable) و لإعادة الاستخدام (Reusable) لبناء تطبيقات وب مع واجهات للمستخدم (User interface)، يعملُ هذا الإطار عند المخدّم ويعتمد على جافا. تحققُ JSF قالب التصميم MVC حيث أن المتحكم هو صف (Servlet) يدعى FacesServlet، أما النموذج فيمثل مجموعة من الأغراض التي تسمى (Managed beans) أو (Backing beans)، و تضمّ الصفحات (Views) مجموعة من المُكونات و تكتب باستخدام JSP أو Facelets علماً أن المعيار ينصح باستخدام Facelets كونها تقدّم ميزات إضافية لاتقدمها JSP مثل إمكانية التحقق المباشر من مُدخلات المستخدم.

يتكون إطار JSF من جزأين:

1. مجموعة من واجهات برمجة التطبيقات (APIs) لتمثيل وإدارة حالة المكونات، وكذلك لمعالجة الأحداث، التحقق من صحة مدخلات المستخدم عند المخدّم، التحويل بين البيانات (Data conversion)، تعريف التنقل بين الصفحات (Navigation)، دعم اللغات (Internatiolization) و تقديم إمكانية توسعة جميع هذه الميّزات من قبل المطوّر.
2. عدد من المكتبات لاستخدام وَسمات ضمن الصفحات (Tags libraries)، أو لربط الوسمات مع أغراض عند المخدّم.

ضمن تطبيق JSF نجد المكونات التالية:

- مجموعة من صفحات الوب بصيغةXML والمكتوبة كما ذُكر آنفاً باستخدام JSP أو Facelets.
- مجموعة من الأغراض مثل (Managed beans) و تعرّف مجموعة من الخصائص للمكونات على الصفحة.
- ملف لتوصيف خصائص تشغيل التطبيق (web.xml).
- بشكل اختياري، ملف تعريف (faces-config.xml) أو أكثر يمكن أن يُستخدم لتعريف قواعد التنقل (Navigation rules)، لتوصيف الأغراض الموجودة في التطبيق أو تعريف مكونات مُخصصة يعرّفها المستخدم.

عندما يطلبُ مستخدم الصفحةَ myfacelet.xhtml المبنية باستخدام وسمات JSF والتي تتضمن مجموعة من المكونات، يتم بناء العرض (view) المكون من شجرة من المكونات عند المخدم، وبعد ذلك يتم إظهاره (Rendering) إلى المستخدم ويقصد بالإظهار توليد خرج مثل HTML أو XHTML يكون مفهوماً من قبل المستعرض عند المستخدم. بشكل أكثر تفصيلاً، أول محطة في معالجة طلب HTTP من المستخدم هي المتحكم FacesServlet ليمرّ بعد ذلك بخطوات عدة تسمى دورة حياة JSF و تتكون من مرحلتين: التنفيذ و الإظهار (Render).

تتضمن مرحلة التنفيذ الخطوات التالية:

- بناء العرض (View).
- تطبيق قيم وسائط الطلب (Request parameter values).
- التحقق من صحة مُدخلات المستخدم و التحويل بين البيانات.
- استدعاء المنطق الوظيفي للتطبيق.

أمّا في مرحلة الإظهار فيتم إرسال الصفحة المطلوبة كردّ إلى المستخدم.

يتم في JSF تحديد الصفحة التي أتى منها الطلب أو التي ستعاد للمستخدم من خلال متحول يسمى ViewState يُرسل مع كل طلب من المستخدم أو رد من المخدّم و له تأثير على حجم الرسائل المتبادلة كما سنرى في التقييم التجريبي لاحقاً.

لابد من الإشارة إلى أن التنقل بين الصفحات في JSF يمكن أن يتم بشكل ضمني أو بشكل صريح.

1. في التنقل الضمني إلى الصفحات يتم تحديد الصفحة التي سيتم الانتقال إليها من خلال النتيجة التي تعيدها الطريقة Action Method من Managed Bean. في المثال (1)، عندما تتم إعادة سلسلة المحارف "success" يتم الانتقال إلى الصفحة success.xhtml وعندما تعاد سلسلة المحارف "index" يتم الانتقال إلى الصفحة index.xhtml. وفي المثال (2) عندما تتم إعادة 1 يتم الانتقال إلى الصفحة 1.xhtml وعندما يعاد 2 يتم الانتقال إلى الصفحة 2.xhtml.

المثال 1

```
public String guessCommand ()
  {
     if (guess == target)
     {
        return ("success");
     }
     else
     {
        this.attempts++;
        return ("index");
     }
  }
```

المثال 2

```
public int guessCommand ()
  {
     if (guess == target)
     {
        return (1);
     }
     else
     {
        this.attempts++;
        return (2);
     }
  }
```

2. في التنقل الصريح إلى الصفحات تُضاف قواعد التنقّل بشكل صريح إلى ملف التعريفات faces-config.xml، لتحددُ الصفحة التي سيتم الانتقال إليها من أجل نتيجة معينة.

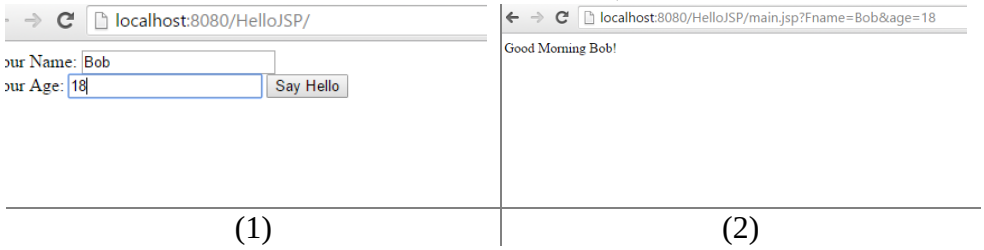
بشكل افتراضي عندما يتم التنقل بين الصفحات يتم استخدام ميزة الانتقال الى الصفحة (Page Forward) والتي لا تغيّر رابط الصفحة (URL) ليتوافق مع الصفحة الجديدة وإنما يبقى عنوان الصفحة المصدر بعكس تقنية إعادة التوجيه إلى صفحة (Page Redirect). لتقنية التنقل هذه دور في تحسين الأداء لأنه في هذه

الحالة يتم تحويل الطلب إلى الصفحة الجديدة بشكل داخلي عند المخدم و لايتمّ إرسال طلب HTTP جديد من المستخدم.

تسمح JSF أيضا باستخدام AJAX لتحقيق مهام غير متزامنة وللتحكم الدقيق (Fine-grain) بسلوك الصفحة دون أن يستخدم المطّور JavaScript وذلك من من خلال استخدام الوسمة f:ajax (التي تستخدم JavaScript بشكل ضمني).

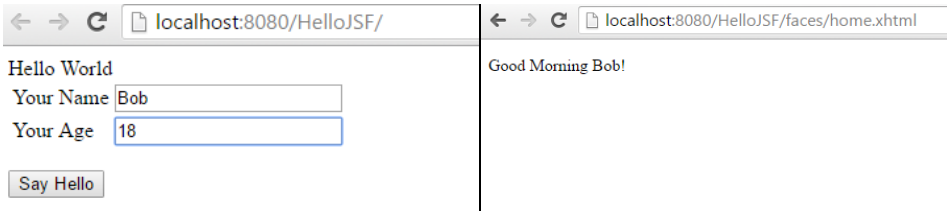
مقارنة بين أداء JSP و JSF في تطبيقات Java EE :

لتقييم ومقارنة أداء التقنيتين اخترنا في البداية تطبيق مكوّن من صفحتين، في الصفحة الأولى نموذج (Form) يتضمن حقلي إدخال للاسم وللعمر وزر أوامر. بعد أن ينقر المستخدم على زر الأوامر يُنقَلُ إلى الصفحة الثانية والتي تُطبَع فيها عبارة حسب العمر الذي أدخله المستخدم في النموذج. لمقارنة الأداء طبقنا خُطة اختبار(Test plan) في الاختبارات الثلاثة الأولى تُحاكي عملَ التطبيق حيث يزور المستخدم الصفحة الأولى، يملأ النموذج، ينقرُ على زرِ الإرسال لتأتيه النتيجة في الصفحة التالية (يوضح الشكلان 4 و 5 تنفيذ التطبيق في الحالتين).



(1) (2)

الشكل 4: تنفيذ تطبيق JSP



الشكل 5: تنفيذ تطبيق JSF

كذلك فحصنا عن كثب الصفحة المرسلة من قبل المخدّم لنجد في حالة JSP:

```
1 <!DOCTYPE html>
2 <html>
3    <head>
```

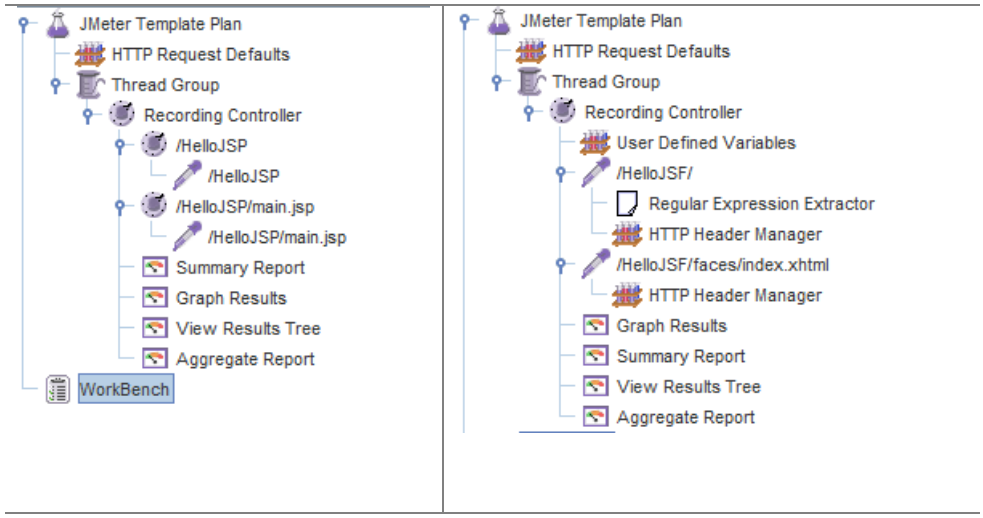
```
 4        <title>TODO supply a title</title>
 5        <meta charset="UTF-8">
 6        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
 7    </head>
 8    <body>
 9        <form action="main.jsp" method="GET">
10            Your Name: <input type="text" name="Fname">
11            <br />
12            Your Age: <input type="text" name="age" />
13            <input type="submit" value="Say Hello" />
14        </form>
15    </body>
16 </html>
```

أما في حالة JSF:

```
1 <?xml version='1.0' encoding='UTF-8' ?>
 2 <!DOCTYPE html>
 3 <html xmlns="http://www.w3.org/1999/xhtml">
 4    <form id="main" name="main" method="post"
action="/HelloJSF/faces/index.xhtml;jsessionid=ea49b2cc1d28a37c23ff6
9a662e9" enctype="application/x-www-form-urlencoded">
 5        <input type="hidden" name="main" value="main" />
 6        <span class="header">Hello World</span>
 7        <table>
 8            <tr>
 9                <td>Your Name</td>
10                <td><input id="main:helloInput" type="text"
name="main:helloInput" />
11                </td>
12            </tr>
13            <tr>
14                <td>Your Age</td>
15                <td><input id="main:helloAge" type="text"
name="main:helloAge" value="0" /></td>
16            </tr>
17        </table><p></p><input type="submit" name="main:j_idt10"
value="Say Hello" /><input type="hidden" name="javax.faces.ViewState"
id="j_id1:javax.faces.ViewState:0" value="-527916370438702537:-
8413648481779620685" autocomplete="off" />
18    </form>
19 </html>
```

نلاحظ أن الصفحة في حالة JSF تتضمن عناصر إضافية مثل الكوكيز (jsessionid) أو معرّف الصفحة (ViewState) الضروري لبناء العَرض بشكل صحيح في دورة حياة JSF. وجود معرف الصفحة (ViewState) يجعل من تقييم أداء التطبيق باستخدام الأداة Apache JMeter [7] مختلف عن تقييم أداء التطبيق في حالة JSP نظراً لضرورة استخراج المتحول (ViewState) من الصفحة الأولى من خلال استخدام تعبير نظامي بطريقة تتيحها الأداة وتضمينه في طلبات تالية ترسل إلى المُخدّم ضمن خطة الاختبار (Test plan) التي تُحاكي العمل الفعلي للتطبيق.

لتقييم أداء هذين التطبيقين تجريبياً، تمّ اجراء الاختبارات على جهاز محمول مزوّد بنظام تشغيل Windows8، معالج Intel Core i5-4200U 1.60GHz و ذاكرة 8 RAM GB باستخدام الأداة Apache JMeter في الاختبار الأول و الثاني والثالث والأداة ab من Apache في الاختبار الرابع. يوضحُ الشكل 6 خطة الاختبار المطبقة في Apache JMeter والتي توضح الروابط التي يطلبها المستخدم؛ القسم الأيسر من الشكل يبيّن JSP والقسم الأيمنJSF. يُلاحظ أيضاً أنّ اختبار JSF يتطلب استخدام مستخرج وفقاً لتعبير نظامي (Regular Expression Execrator) للحفاظ على الحالة بين الطلبات كما ذكرنا سابقاً.



الشكل 6: خطة الاختبار المطبقة لتقييم أداء التطبيق

في الاختبارات الثلاثة الأولى، تم استخدام التقرير المُلخّص (Summary Report) من الأداة Apache JMeter والذي يعرضُ عدد من القيّم و الإحصائيات اخترنا منها:

- حقل العنوان (Label): يعرض طلبات http التي يتم إرسالها خلال الاختبار.
- حقل العينات (Samples): يعرض عدد طلبات http لمستخدم معين. على سبيل المثال إن تضمّن الاختبار خمسة مستخدمين عندئذ كل مستخدم يرسل طلب لكل رابط وبالتالي عدد العينات لكل رابط سيكون 5.
- المتوسط (Average): متوسط زمن الاستجابة لطلب http محدد (millisecond).
- الخطأ: نسبة الخطأ في تنفيذ العينات مثلا الخطأ 404.
- الإنتاجية (Throughput): عدد الطلبات التي أُرسلت إلى المخدّم في وحدة الزمن.
- KB/sec : يعطى بالعلاقة

KB/sec = (Throughput*Average bytes) /1024

- Avg. Bytes: متوسط كميّة البيانات التي يستقبلها المستخدم من المخدّم بوحدة البايت.

الاختبار الأول:

تمّ الاختبار الأول باستخدام الوسائط التالية:

نوع الوسيط	اسم المتحول	قيمة الوسيط
عدد المستخدمين	nb.users	1
القفزة وتمثل زمن انتظار المستخدم التالي قبل ارسال الطلب بالثواني. أي إن اختبرنا عشرة مستخدمين مع قفزة تعادل عشرة عندئذ كل مستخدم جديد سيبدأ بإرسال الطلبات كل ثانية. لكن في حالة	nb.rampup	1

مستخدم وحيد فهذه القيمة ليس لها أهمية.		

نتائج تقييم أداء JSP:

Label	#Samples	Average	Error %	Avg. Bytes
/HelloJSP	1	14	0.00%	860.0
/HelloJSP/main.jsp	1	8	0.00%	187.0
Total	2	11	0.00%	523.5

نتائج تقييم أداء JSF:

Label	#Samples	Average	Error %	Avg. Bytes
/HelloJSF	1	35	0.00	1039.0
/HelloJSP/faces/index.xhtml	1	30	0.00	998.0
/HelloJSP/faces/home.xhtml	1	23	0.00	220.0
total	3	29	0.00	752.3

مناقشة نتائج الاختبار الأول:

تبين النتائج بوضوح أنّ كمية البيانات المتناقلة في حالة JSF (752.3 بايت) أكبر منها في حالة JSP (523.5 بايت) والسبب يعود لنقل حالة الجلسة بين المخدم والمستخدم عند استخدام JSF ولهذا تأثير على الانتاجية في الشبكة ككل، كذلك زمن الاستجابة في تطبيق JSP (11 ms) أقل من زمن الاستجابة في تطبيق JSF (29 ms).

الاختبار الثاني:

تمّ الاختبار الثاني باستخدام الوسائط التالية:

نوع الوسيط	اسم المتحول	قيمة الوسيط
عدد المستخدمين	nb.users	200
القفزة	nb.rampup	200

نتائج تقييم أداء JSP:

Label	#Samples	Average	Error %	Throughput	KB/Sec	Avg. Bytes
/HelloJSP	200	32	0.00%	1.0/sec	0.84	860.0
/HelloJSP/main.jsp	200	16	0.00%	1.0/sec	0.18	187.0
Total	400	24	0.00%	2.0/sec	1.03	523.5

نتائج تقييم أداء JSF:

Label	#Samples	Average	Error %	Throughput	KB/Sec	Avg. Bytes

/HelloJSP	200	31	0.00%	1.0/sec	1.02	1038.8
/HelloJSP/faces/index.xhtml	200	25	0.00%	1.0/sec	0.98	998.8
/HelloJSP/faces/home.xhtml	200	22	0.00%	1.0/sec	0.22	220.0
Total	600	26	0.00%	3.0/sec	2.21	752.5

مناقشة نتائج الاختبار الثاني:

نلاحظ أن نتائج الاختبار الثاني شبيهةٌ بنتائج الاختبار الأوّل: كمية البيانات المتناقلة في حالة JSF (752.3 بايت) أكبر منها في حالة JSP (523.5 بايت) و كذلك زمن الاستجابة في تطبيق JSP (24 ms) أقل من زمن الاستجابة في تطبيق JSF (26 ms).

الاختبار الثالث:

تمّ الاختبار الثالث باستخدام الوسائط التالية:

نوع الوسيط	اسم المتحول	قيمة الوسيط
عدد المستخدمين	nb.users	1000
القفزة	nb.rampup	1000

نتائج تقييم أداء JSP:

Label	#Samples	Average	Error %	Throughput	KB/Sec	Avg. Bytes
/HelloJSP	1000	33	0.00	1.0/sec	0.84	860.0
/HelloJSP/main.jsp	1000	16	0.00	1.0/sec	0.18	187.0
Total	2000	25	0.00	2.0/sec	1.02	523.0

نتائج تقييم أداء JSF:

Label	#Samples	Average	Error %	Throughput	KB/Sec	Avg. Bytes
/HelloJSF	1000	33	0.00 %	1.0/sec	1.02	1038.7
/HelloJSP/faces/index.xhtml	1000	25	0.00 %	1.0/sec	0.98	998.8
/HelloJSP/faces/home.xhtml	1000	21	0.00 %	1.0/sec	0.22	220.0
total	3000	26	0.00 %	3.0/sec	2.21	752.5

مناقشة نتائج الاختبار الثالث:

نلاحظ أن الاختبار الثالث يوصل لنتيجة مماثلة للاختبارين الأول والثاني من حيث أن كمية البيانات المرسلة في JSP أقل و كذلك زمن الاستجابة أفضل.

الاختبار الرابع باستخدام الأداة ab من Apache [9]:

اختبرنا التطبيقين أيضا من خلال الأداة ab مع الوسائط التالية:

نوع الوسيط	اسم المتحول	قيمة الوسيط
عدد الطلبات الكلي	-n requests	10000
عدد المستخدمين الذين سيطلبون الموقع في نفس الوقت	-c concurrency	100

فأعطى الاختبار أن كمية البيانات المنقولة في JSP تعادل (6790000 bytes) في حين أن كمية البيانات المنقولة في حالة JSF (10387514 bytes) وهذا يتوافق مع نتائج الاختبارات باستخدام الأداة Apache JMeter، كما أن متوسط زمن الاستجابة في JSP ([ms] 457.604) بينما في JSP ([ms] 709.410).

المقارنة بين استخدام JSP و JSF في تطبيق Java EE متعددة الطبقات من خلال تقييم أداء التطبيق:

بعد عرض مقارنة بين أداء التقنيتين من خلال تطبيق بسيط، اخترنا في المرحلة الثانية التطبيق AffableBean وهو أحد تطبيقات التجارة الالكترونية المتعددة الطبقات و المقدّمة من خلال موقع NetBeans IDE[5]. يَستخدمُ التطبيقJSP في طبقة العرض **ولذلك أعدنا تحقيق طبقة العرض باستخدام JSF.**

لمحة عن التطبيق AffableBean[5]:

AffableBean هو تطبيق تجارة الكترونية للتسوق عبر الانترنت يتضمن الوظائف التقليدية التي يتضمنها أيّ موقع تجارة الكترونية من استعراض لفئات المنتجات (Categories) واستعراض المنتجات في كل فئة كما هو مبيّن في الشكل 7، بالإضافةِ إلى صفحة سلةٍ مشتريات (Shopping cart) تُضاف إليها المنتجات (Products) التي اختارها المستخدم و تتضمنُ وظائف تعديل لتغيير كميّة منتج معيّن أو حذف منتج من سلة المشتريات. بعد اختيار المنتجات و التحقق من سلة المشتريات يمكن أن ينتقل المستخدم إن لم تكن سلة مشترياته فارغة إلى التسجيل (Checkout) و الدفع الالكتروني الذي يتمّ من خلال اتصال آمن يستخدم برتوكول طبقة المقابس الآمنة (SSL).

الشكل 7: استعراض المنتجات من فئة معينة في تطبيقAffableBean **[5]**

اختبرنا التطبيق في الحالتين مع مستخدم وحيد وباستخدام الأداة Apache JMeter

فحصلنا على النتائج التالية:

نتائج تقييم أداء JSP:

Label	#Samples	Average	Error %	Throughput	KB/Sec	Avg. Bytes
/AffableBean/	1	23	0.00%	43.5/sec	216.75	5105.0
/AffableBean/js/jquery-1.4.2.js	1	31	0.00%	32.3/sec	5161.76	163855.0
/AffableBean/css/affablebean.css	1	8	0.00%	125.0/sec	746.09	6112.0
/AffableBean/img/logo.png	1	4	0.00%	250.0/sec	4313.23	17667.0
/AffableBean/img/cart.gif	1	4	0.00%	250.0/sec	25.39	104.0
/AffableBean/js/jquery-ui-1.8.4.custom.min.js	1	11	0.00%	90.9/sec	936.52	10549.0
/AffableBean/js/jquery.corners.js	1	11	0.00%	90.9/sec	1201.53	13534.0
/AffableBean/img/categories/dairy.jpg	1	5	0.00%	200.0/sec	5134.77	26290.0
/AffableBean/img/categories/fruit%20&%20veg.jpg	1	4	0.00%	250.0/sec	5242.92	21475.0
/AffableBean/img/favicon.ico	1	7	0.00%	142.9/sec	196.15	140.6.0
/AffableBean/img/categories/meats.jpg	1	4	0.00%	250.0/sec	9002.20	36873.0
/AffableBean/img/stalk.png	1	4	0.00%	250.0/sec	3982.18	16311.0
/AffableBean/img/categories/bakery.jpg	1	4	0.00%	250.0/sec	5135.74	21.36.0
Total	13	9	0.00%	99.2/sec	2536.95	26178.2

نتائج تقييم أداء JSP:

Label	#Samples	Average	Error %	Throughput	KB/Sec	Avg. Bytes
/AffableBean/	1	88	0.00%	11.4/sec	61.52	5544.0
/AffableJSF/faces/javax.faces.resource/css/affablebean.css	1	10	0.00%	100.0/sec	649.02	6646.0
/AffableJSF/faces/javax.faces.resource/css/app.css	1	9	0.00%	111.1/sec	39.71	366.0
/AffableJSF/faces/javax.faces.resource/bakery	1	7	0.00%	142.9/sec	2934.71	21036.0

.jpg						
/AffableJSF/faces/javax.faces.resource/dairy.jpg	1	7	0.00%	142.9/sec	3667.69	26290.0
/AffableJSF/faces/javax.faces.resource/meats.jpg	1	7	0.00%	142.9/sec	5144.11	36873.0
/AffableJSF/faces/javax.faces.resource/fruit%20&%20veg.jpg	1	7	0.00%	142.9/sec	2995.95	21475.0
/AffableJSF/faces/javax.faces.resource/img/stalk.png	1	6	0.00%	166.7/sec	2654.79	16311.0
/AffableJSF/faces/javax.faces.resource/jsf.js	1	87	0.00%	11.5/sec	1576.71	140466.0
/AffableJSF/faces/javax.faces.resource/cart.gif	1	8	0.00%	125.0/sec	12.70	104.0
/AffableJSF/faces/javax.faces.resource/logo.png	1	12	0.00%	83.3/sec	1437.74	17667.0
Total	11	22	0.00%	43.1/sec	1121.24	26616.2

مناقشة نتائج الاختبار:

تبين النتائج أن كمية البيانات المتناقلة في حالة JSF (26616.2بايت) أكبر منها في حالة JSP (26178.2 بايت) علماً أنه كما يلاحظ تمّ استخدم jquery-1.4.2 في حالة JSP لتحقيق وظائف إضافية مثل التحقق من مدخلات المستخدم، وكذلك يلاحظ أن استخدام AJAX مع JSF أدى لتوليد الملف AffableJSF/faces/javax.faces.resource/jsf.js والذي كان له تأثير مهم على كمية البيانات المتناقلة، أي حتى دعم JSF لـ AJAX يأتي على حساب زيادة مهمة في كمية البيانات التي يتم تناقلها.

الدراسة المرجعية:

لم نجد أية ورقة بحثية تقدم مقارنة بين JSF و JSP، لكن يوجد روابط على موقع StackOverflow[10] تعرضُ مناقشات متعلقة بمقارنة أداء التقنيتين تعتمد على وجهات نظر شخصية غالباً. في هذه الورقة قدمنا دراسة نظرية وعملية تقارن أداء كل من JSF و JSP وتبيّن محاسن ومحدوديات كل تقنية.

الخلاصة والآفاق المستقبلية:

قدّم هذا البحث في قسمه الأول عرضاً لبنية تطبيق متعدد الطبقات في بيئة JEE 7 وبيّن مكوناته المختلفة، وفي قسمه الثاني عرض مقارنة بين تقنيتين يمكن استخدامهما في طبقة العرض: JSP و JSF. بيّنت النتائج التجريبية لتقييم الأداء تفوّق JSP على JSF من حيث المحافظة على عرض الحزمة في الشبكة و زمن الإستجابة، في المقابل تحقق JSF قالب التصميم MVC مقدمةً ميّزات على مستوى فصل واضح بين العرض

و المنطق الوظيفي لتطبيق الوب. في الآفاق المستقبلية يمكن أخذ عوامل أخرى في التقييم مثل الذاكرة المستهلكة عند المخدّم و استخدام أدوات إضافية لتقييم الأداء.

المراجع:

[1]. E-Commerce: Business, Technology, Society. Keneth C. Laudon and Carol Guercio Traver, Prentice Hall, 3rd Edition (2006).

[2]. The Java EE 7 Tutorial. Release 7 for Java EE Platform, E39031-01, June 2013.

URL: http://docs.oracle.com/javaee/7/tutorial/doc/javaeetutorial7.pdf

[3]. Servlets, JSP, Struts and MVC (Part I). By Venkat Subramaniam.

URL: http://www.agiledeveloper.com/articles/JSPMVC.pdf

Last retrieved: 1/6/2016

[4]. JSF : Page Forward vs Page Redirect. URL:

http://www.mkyong.com/jsf2

Last retrieved: 1/6/2016

[5]. The NetBeans E-commerce Tutorial. URL:

https://netbeans.org/kb/docs/javaee/ecommerce/intro.html

Last retrieved: 1/6/2016

[6]. The NetBeans E-commerce Tutorial - Testing and Profiling.

URL: https://netbeans.org/kb/docs/javaee/ecommerce/test-profile.html

Last retrieved: 1/6/2016

[7]. Apache JMeter™. URL: http://jmeter.apache.org/

[8]. JSP Life Cycle.

URL: http://www.tutorialspoint.com/jsp/jsp_life_cycle.htm

[9]. ab - Apache HTTP server benchmarking tool

https://httpd.apache.org/docs/2.4/programs/ab.html

[10]. http://stackoverflow.com/