
Keywords – JEE 6.

I. INTRODUCTION

Rich characteristics of internet applications draw attention of corporate clients. The main reason of this is that they eliminate the necessity of expensive installation and maintaining of software on users’ terminal services – the only thing necessary for them to run is one of the popular and widely available web browsers. It is almost equally important that any updates or fixes in the software are very easy to apply. The whole process, done on the server side, is completely transparent to the final user. ([11]).

Among the technologies used for creating corporate internet applications Java Enterprise Edition and ASP.NET are in the lead. Version 5 of JEE consists of 23 specifications, which define applications creation standard. Its main purpose is to reduce the complexity and costs of development, deployment and maintaining of server oriented software [2]. Despite exertions of Sun and all Java Community Process participants, JEE 5 platform is still not free of flaws. The proof to that is still unremitting popularity of countless frameworks created to replace the most troublesome standards and to eliminate flaws of previous versions of this specification. The most popular of them are Spring and Hibernate.

At the moment of writing of this article, the date of JEE 6 specification release was planned on September 2009. Spring 3 was in phase of milestone 4 (RC1 version was planned on 21st September 2009).

II. BRIEF HISTORY OF JEE

First JEE version (1.2) was released in December 1999. It has evolved many times since then, increasing amount of specifications and modifying most of them. Special attention should be paid to J2EE 1.4 released in November 2002 (current naming convention JEE 5, is direct continuation of previously used J2EE 1.4). Enterprise Java Beans 2.1 standard that was introduced then (actually the fact that it didn’t fix problems of EJB 2.0), contributed greatly to JEE-Spring community migration. Beside already known problem with complexity of software development based on Servlets and JSP, J2EE 1.4 was also missing abstraction above those technologies, what led programmers to reach for alternative solutions like: Struts, Tapestry, and later Spring MVC etc. The situation was improved greatly by JEE 5 released in 2006. It consisted of, inter alia, completely rebuilt EJB v.3 specification. EJB components started to be based on dependency injection technique from that moment. Entity EJB was replaced by the JPA standard (still being the part of EJB), that provides programmatic interface for data persistence [3]. Use of both of those technologies became easier thanks to relocate the burden of configuration from .xml files to annotations. Next important change introduced with JEE 5 is JSF 1.2 specification. It defines new standard of creation of user interfaces. It introduced essential abstraction over servlets and JSP (using them underneath). The most important feature of JSF components is that they preserve state during session communication with client. Furthermore, standard contains default collection of graphical components of user interface [4].

JEE 5 specification shows clearly following solutions provided earlier by frameworks like Spring or Hibernate.

III. JEE 5 FLAWS

As mentioned earlier JEE 5 is not free of flaws. They are most clearly visible while using alternative technologies, because the creation of corporate applications with the use of tools like Spring, Hibernate and many others is becoming easier and less complex.

In [5] authors pointed a few problems of J2EE 1.4 platform that was present at the time of writing. Despite the time that has passed since then, a part of those problems is still unsolved:

- A tendency to include excessive amount of auxiliary code – situation was improved thanks to dependency injection for objects located in JNDI and relocating part of the configuration burden to annotations, however problem wasn’t eliminated completely and it probably won’t ever be.
- Forcing the use of distributed object model, where it’s not necessary
- Complex EJB components model – thanks to introducing EJB 3 specification, the situation here was improved as well. However the specification has very little to offer compared to business logic layer support offered by ex. Spring
- Unit testing is especially difficult – testing of applications based on JEE specification (especially EJB) outside of the application server is extremely difficult. It affects greatly with test units code coverage ratio and leads to worse quality of testing.

The competition is not sleeping and while consecutive Sun’s specifications are fixing shortcomings and introducing moderate improvements, others are making further major steps forward, defining new quality of Java software development.

Today, the advantages of aspect programming on small and large scale can be clearly seen. JEE offers only a slight support towards that, when modern programmers are expecting a lot more. The most bothersome in JPA specification is surely, lack of the mechanism similar to
Hibernate Criteria API. JPA allows only use of Optimistic Locking mechanism (Pessimistic Locking support is missing). Above problems aren’t of course the only ones.

IV. JEE 6 NOVELTIES [6][7]

JEE platform increases number of specifications it consists of with each consecutive release (fig. 1). Version 6 is supposed to have about 28 of them, and should be released in September 2009.

![Number of specs in another JEE releases](image)

Taking those numbers into consideration, creators of JEE decided to introduce segmentation of specifications. Starting with JEE 6 specifications will be divided to profiles with the first one „Web Profile” – already defined. Adding next one will be the task of Java Community Process. Web Profile beside containing only a part of JEE stack, will be using smaller and more independent specifications. Ex. it will contain EJB Lite specification with message driver beans, rmi, cmp/bmp and few more excluded.

As mentioned earlier quite a few technologies distributed with JEE platform proved to be abortive. Some of them were replaced with newer versions, some are still included to stack in unchanged form. To allow gradual disposing of unwanted technologies, with simultaneous notifying the community about it, the pruning term was introduced. Specification marked as pruned (ex. jax-rpc) can be included in next platform release as optional, meaning that application server provider will be able to decide if he want to implement the specification or not.

JEE 6 will introduce extensibility. Inclusion and configuration of 3rd party libraries (like Spring MVC) is supposed to be easy lie never before. In presentation layer the mechanism consist in putting fragment of existing web.xml inside the jar file of the library. Container will find appropriate configuration file by itself and will know what to do with it. This solution will supposedly be strongly developed in the future.

On the servlets field, the biggest change is the introduction of web.xml as an optional configuration file. Most of the configuration can be done with the use of annotations. What’s more it can consist of multiple smaller configuration files (in compliance with before mentioned functionality of extensibility). Support for asynchronous communication model for servlets will be introduced, which will allow ex. using mechanism similar to AJAX-push (or comet).

JSF attain 2.0 specification. Like in whole JEE 6 the burden of configuration can be relocated to annotations making faces-config.xml file optional. Composite Component term will be introduced (fig. 2).

```xml
<composite:interface>
  <composite:actionSource name="loginEvent"/>
</composite:interface>
<composite:implementation>
  <p>Username: <h:inputText id="usernameInput" /></p>
  <p>Password: <h:inputSecret id="passwordInput" /></p>
  <p><h:commandButton id="loginEvent" value="login" /></p>
</composite:implementation>
```

![Sample composite component](image)

For the definition of this kind of components, Java code is unnecessary. JSF tags and XHTML is all that suffice. Style of writing components, similar to that of Java rest on creating interface and implementation.

EJB acquired lots of useful functionalities. One of them is long awaited support for Singleton. Bean configuration boils down to adding single annotation @Singleton. What should be taken into account is that such singleton has cluster scope!, which is very hard to achieve from the programmer’s point of view without the application server support. Next improvement is the optionality of creating local interface for a bean (up until now such interface was obligatory). From now on one local bean need one and only one class. Other improvements consist of:

Calendar timer based on annotations – programmatic API is also available in case annotations turn to be not sufficient to achieve assumed goal.

Non-persistent timers – up until now timers defined in application with the use of JEE 5 annotations and programmatic API were persistent and started again after system reboot – not every application needed this kind of functionality and for some it was simply wrong.

Asynchronous calls to business methods – achieving the asynchronous call to EJB method was done until now with the use of internal JMS queues. From now on, the only thing necessary will be adding the @Asynchronous annotation.

Global portable JNDI names for EJB – naming format was specified (fig 3.)

EJB Lite – separated collection of EJB specifications, which can be run even in Java SE. Earlier in the article was mentioned its usefulness in Web Profile. It can be also used to easier EJB components testing outside of the application server.

```
java:global/(app)/(module)/(bean)#(intf)
```

![Simple composite component](image)

JPA 2.0 was separated from EJB specification (JSR 317). It was improved with inter alia Criteria API, which allows dynamic query creation with the use of object API. Additionally locking possibilities were extended with
pessimistic locking, which allows explicit blocking of data in database (JPA 1.0 contained support only for optimistic locking that is programmatic data in transaction unchangeability assurance with the use of versioning).

Completely new API in JEE6 is WebBeans 1.0. In JEE5 there wasn’t any prepared bridge that would connect JSF and EJB technologies. WebBeans technology is supposed to cover that hole, ensuring simplicity of using managed beans as logic for JSF. Because of its complete novelty status, the usefulness of technology will be verified by the programmers writing real corporate applications. Investigating possible advantages or flaws is thus really difficult at this time.

Almost new is JAX-RS 1.1 specification. Its main purpose is to provide the support for using REST web services. Already existing jax-ws specification used to deal with traditional web services can help in understanding the principles of jax-rs working. The main characteristic is full configurability with the use of annotations (fig. 4). With even small knowledge of the REST technology the meaning of each annotation and whole code can be intuitively understood. JAX-RS allows web services implementation based on all basic RESTful Services standards.

```java
@Path("/helloworld")
public class HelloWorldRS {
  @GET
  @Produces("text/plain")
  public String hello() {
    return "Hello World";
  }
}
```

Fig.4 Example of jax-rs spec usage [7].

V. SPRING 3 NOVELTIES [9]

> From the structural side, Spring 3 was written in the way allowing best use of the Java 5 language capabilities (despite great exertions for retaining backward compatibility, it means that in some places of the framework it will be broken). Other changes are inter alia:

- Spring modules have been separated and one common spring.jar package will be no more.

Expression language – designed for providing additional possibilities of defining configuration, beans etc. of Spring, and also for introducing one, unified expression language for all modules.

Adding new annotations that provide functionalities present up until now only in files used for configuring Spring context (ex. injection done with @Bean annotation).

Many more...

VI. SUMMARY

It seems that the amount of common problems that were being solved both by JEE and Spring Framework decreases with each consecutive release. Spring and the community that was built around it over years is developing more and more projects putting into effect its own vision of Java software creation like tcServer, dmServer supporting OSGi, support for Grails etc.

REFERENCES

7. What’s new in Java EE 6, Antonio Goncalves, GeeCON 2009