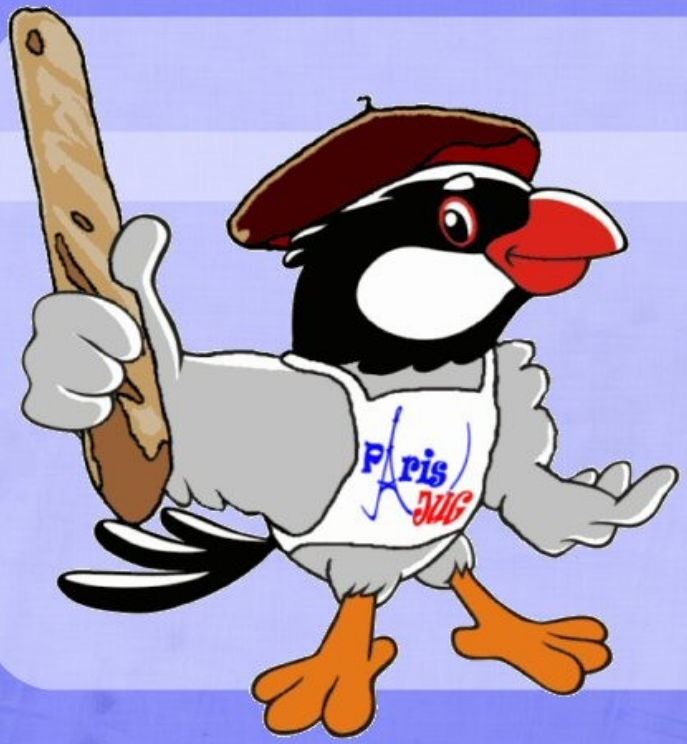


Paris JUG

www.parisjug.org

13/10/2009

www.parisjug.org

13/10/2009

Servlet 3.0

Présentation et Nouveautés

Rémy Maucherat

Pr. Software Engineer

Red Hat

JBoss by Red Hat

www.parisjug.org

Intervenant

- **Rémy Maucherat**
  - Débute avec des Servlets en 99
  - Committer dans Tomcat depuis 2000
  - Membre de l'ASF
  - Employé chez JBoss, reponsable de l'implémentation des APIs Servlets & JSP
  - Membre de l'EG de JSR 315 (Servlets 3.0)
  - Développement de JBoss Web 3 en cours ...

Sommaire

- **Présentation**
- **Servlets 3.0 en détail**
  - Annotations, Fragments et Ordre
  - Async
  - API de configuration
  - Overlays, sécurité, libraries partagées, JSP, etc
- **Conclusion**
- **Questions / Réponses**

Présentation

www.parisjug.org

Besoins

- Intégration de frameworks
- Programmation de Servlets
- Meilleure scalabilité
- Sécurité peu flexible
- Bibliothèques partagées
- Et le reste ...

Intégration

- **Objectif: ne pas éditer web.xml**
- **Mécanismes d'intégration**
  - Ajouts de Servlets, filtres, listeners
  - Configuration de sécurité
  - Interfaces d'administration

Programmation

- Objectif: toujours pas de web.xml
- Annotations
- Utiliser la sécurité du container

Scalabilité

- Objectif: pouvoir gérer des opérations longues
- Ajout de possibilités d'utilisation asynchrones

Bibliothèques partagées

- Objectif: zone grise auparavant, augmenter la portabilité
- Possibilité d'intégration portable entre containers
- Ajouts pour l'intégration d'un container JSP

Le reste

- Objectif: ajouts de fonctions « propriétaires » usuelles
- Envoi de fichiers
- Configuration de la gestion des sessions
- Intégration avec JSP

Annotations, Fragments et ordre

www.parisjug.org

@WebServlet

- Définit un Servlet
- Définit le mapping du Servlet
- Configuration: load-on-startup, init-params, etc

```
@WebServlet(name=”MonServlet”, urlPatterns={"/path1", "/path2"})
public class MonServlet extends HttpServlet{
    public void doGet(HttpServletRequest req, HttpServletResponse
Res) {
        ........
    }
}
```

@WebFilter

- Définit un filtre
- Définit mapping (urlPatterns, servletNames, dispatcherTypes)
- Configration des initParams

```
@WebFilter(“/path”)
public class MonFiltre implements Filter {
     public void doFilter(HttpServletRequest req, HttpServletResponse
res)
     {
          ...
     }
}
```

@WebListener

- Définit un listener
- Sur tous les types de listeners

```
@WebListener
public class MonListener implements ServletContextListener{
    public void contextInitialized(ServletContextEvent sce) {
        ...
    }
}
```

@ServletSecurity

- Définit des contraintes de sécurité
- Attention: attaché à une classe !
- Assez intuitif

```
@WebServlet(name=”MonServlet”, urlPatterns={"/path1", "/path2"})
@ServletSecurity(@HttpConstraint(rolesAllowed={"user", "guest"}),
     httpMethodConstraints=@HttpMethodConstraint("POST",
                          rolesAllowed={"admin"}))
public class MonServlet extends HttpServlet
       ........
```

@MultipartConfig

- Sur un Servlet
- Configuration des fonctions d'envoi de fichiers
- Voir API correspondante

Fragments

- Dans les JAR de /WEB-INF/lib
- /META-INF/web-fragment.xml
- Similaire à web.xml
- Element racine <web-fragment> au lieu de <web-app>
- Fusionné dans web.xml suivant un ordre configurable
- Nom logique associé au fragment (élément <name>)

Ordre absolu

- Dans web.xml, élément <absolute-ordering>
- Liste de noms de fragments
- Element spécial <others/>

```
<web-app>
  <name>MonApplication</name>
  <absolute-ordering>
    <name>Fragment2</name>
    <name>Fragment1</name>
    <others/>
    <name>Fragment3</name>
  </absolute-ordering>
 ....
</web-app>
```

Ordre relatif

- Dans les fragments, élément <ordering>
- Définit l'ordre relatif par rapport aux autres noms de fragments
- Element spécial <others/>

```
<web-fragment>
  <name>A</name>
  <ordering>
    <after><others/></after>
    <before><name>C</name></before>
  </ordering>
 ....
</web-fragment>
```

Overlays

- Ressources statiques incluses dans les JARs
- Dans /META-INF/resources (ex: /META-INF/resources/test.gif correspond à /test.gif)
- Fonctionne avec ServletContext.getResource, getResourceAsStream et getResourcePaths
- Utilisation possibles:
  - interfaces de monitoring, management, etc
  - packaging d'images ou de css

SCI

- JAR service: javax.servlet.ServletContainerInitializer
- Charge la classe désignée
- Appelle onStartup de la classe au démarrage de l'application
- Permet l'intégration de libraries partagées

Async

www.parisjug.org

Principe

- Permet de démarrer un processus asynchrone
- Ensuite, d'utiliser un callback du container
- ... qui va réinvoquer un Servlet pour finir le traitement
- Avantage: meilleure scalabilité
- Attention: ceci n'est pas de l'IO non bloquante, ni spécifique à HTTP (et peut fonctionner avec java.io, AJP, etc)

ServletRequest.startAsync

- **startAsync met le container en mode asynchrone**
- **Retourne un AsyncContext**
  - configuration
  - réinvocation ultérieure
- **Requête toujours active quand le Servlet rend la main**
- **N'utilise pas de thread pendant que la requête est inactive**

AsyncContext

- **Utilisé pour réinvoquer la requête**
  - vers le même Servlet
  - vers un autre Servlet
- **ServletRequest et ServletResponse wapping autorisés**
- **En principe appelé à partir de manière asynchrone**
- **Permet de spécifier des listeners pour nettoyer les objets à la fin de la requête**

AsyncContext API

- dispatch(): réinvoque le Servlet d'origine
- dispatch(String path): réinvoque un autre Servlet
- complete(): termine la requête
- start(Runnable r): exécute une tâche
- addListener(AsyncListener l): ajoute un listener
- setTimeout(long t): timeout en ms

AsyncListener

- **onComplete: appelé quand la requête est terminée**
- **onTimeout: en cas de timeout**
  - mais ne ferme pas la requête automatiquement, AsyncContext.complete doit être appelé
- **onError: en cas d'exception**
- **onStartAsync: appelé par ServletRequest.startAsync; permet aux listeners de se rajouter à nouveau à l'AsyncContext**

Astuces

- **Après un dispatch async, requête fermée**
  - sauf si on réutilise ServletRequest.startAsync
- **Ne pas oublier de rajouter à nouveau les listeners dans onStartAsync**
- **Aucune invocation concurrente**
- **complete() n'est pas synchrone et n'interrompt pas un traitement**
  - la requête sera en fait fermée dès que possible
- **Utiliser un seul thread asynchrone pour gérer l'attente de nombreux AsyncContext**

Prudence ...

- Après avoir appelé startAsync, de préférence pas de traitements longs ou d'IO (le dispatch attendrait)
- IO dans onTimeout
- Toujours préférer faire de l'IO dans un dispatch
- Dans le doute, ne pas utiliser
  - nettement plus complexe
  - économie de threads uniquement dans les phases d'attente de l'application

Example

```
@WebServlet(asyncSupported=true, urlPattern=”/path")
public class TestAsyncServlet extends Httpervlet {
    public void doPost(HttpServletRequest req, HttpServletResponse res) {
        // Traitement
        // ...........
        // Appel de startAsync
        AsyncContext context request.startAsync();
        // Passe l'AsyncContext au traitement asynchrone MonListener
        // ...........
    }
}
```

```
public class MonListener extends Thread {
    public MonListener(AsyncContext context) {
        // .....
    }
    public void run() {
       // Attend un évênement
       // .....
       context.dispatch(); // ou context.complete(), etc
    }
}
```

Configuration programmatique

www.parisjug.org

Dans ServletContext

- Ajout et configuration de Servlets
- Ajout et configuration de filtres
- Ajout de listeners
- Ajouts de contraintes de sécurité
- Uniquement pendant l'initialisation
- Pas à partir de listeners issus de TLDs

Gestion des Servlets

- **Ajout d'un Servlet dynamiquement (retourne ServletRegistration.Dynamic)**
  - addServlet(String servletName, String className)
  - addServlet(String servletName, Servlet servlet)
  - addServlet(String servletName, Class <? extends Servlet> servletClass)
- **Accès aux Servlets**
  - ServletRegistration getServletRegistration(String servletName)
  - Map<String, ? extends ServletRegistration> getServletRegistrations()

ServletRegistration

- **ServletRegistration**
  - Set<String> addMapping(String... urlPatterns): ajout de mappings
  - Collection<String> getMappings()
  - String getRunAsRole()
- **ServletRegistration.Dynamic**
  - void setLoadOnStartup(int loadOnStartup)
  - void setServletSecurity(ServletSecurityElement constraint): ajout de contraintes de sécurité, structure similaire à @ServletSecurity
  - setMultipartConfig(MultipartConfigElement multipartConfig): équivalent à @MultipartConfig
  - setRunAsRole(String roleName)

Gestion des Filtres

- **Ajout d'un filtre dynamiquement (retourne FilterRegistration.Dynamic)**
  - addFilter(String filterName, String className)
  - addFilter(String filterName, Filter filter)
  - addFilter(String filterName, Class <? extends Filter> filterClass)
- **Accès aux filtres**
  - FilterRegistration getFilterRegistration(String filterName)
  - Map<String, ? extends FilterRegistration> getFilterRegistrations()

FilterRegistration

- void addMappingForServletNames (EnumSet<DispatcherType> dispatcherTypes, boolean isMatchAfter, String... servletNames): ajout de mapping par nom de Servlet
- Collection<String> getServletNameMappings()
- void addMappingForUrlPatterns (EnumSet<DispatcherType> dispatcherTypes, boolean isMatchAfter, String... urlPatterns): ajout de mapping par URL
- Collection<String> getUrlPatternMappings()

Servlet/FilterRegistration

- String getName()
- String getClassName()
- boolean setInitParameter(String name, String value): configuration des paramètres
- String getInitParameter(String name)
- Set<String> setInitParameters(Map<String, String> initParameters)
- Map<String, String> getInitParameters()
- void setAsyncSupported(boolean isAsyncSupported)
  - Dynamic uniquement, active le support de async

Gestion des listeners

- **Ajout de listeners**
  - void addListener(String className)
  - <T extends EventListener> void addListener(T t)
  - void addListener(Class <? extends EventListener> listenerClass)

Autres

- Gestion des paramètres d'initialisation
- Création d'instances de Servlets, filtres et listeners, avec injection
- Configuration de session (tracking et cookie)
- ClassLoader getClassLoader(): accès au classloader de l'application
- void declareRoles(String... roleNames): déclaration de roles, équivalent à <security-roles>

Résumé

- **Plutôt complet**
- **Trop détaillé pour parler de tout, pour plus de détails:**
  - chapitre 4.4 de la spécifcation
  - javadoc de ServletContext
- **Exemples d'utilisation**
  - dans un ServletContainerInitializer
  - dans un listener déclaré avec une annotation ou un fragment
  - validation de configuration par l'application ou une framework

Autres nouveautés

www.parisjug.org

Multipart

- Quasi identique à Apache commons-fileupload
- Associer un <multipart-config> au Servlet (également avec une annotation @MultipartConfig)
- Si la requête est une multipart/form-data, dans HttpServletRequest:
  - Collection<Part> getParts()
  - Part getPart(String name)

Part

- InputStream getInputStream()
- String getContentType()
- String getName()
- long getSize()
- void write(String fileName): écrire directement sur le disque
- void delete(): effacer le fichier
- et quelques fonctions additionnelles pour lire les headers des parts

Sécurité

- **Contrôle de la sécurité du container**
- **Sur ServletRequest:**
  - boolean authenticate(HttpServletResponse response): auth directe avec la méthode de login configurée
  - void login(String username, String password): auth directe avec les credentials fournis (nouveau type d'auth: LOGIN)
  - void logout(): enlève le principal actuel de la session

Gestion de Session

- **Configuration du suivi de la session**
- **<session-config>/<tracking-mode>**
- **Valeurs possibles:**
  - COOKIE
  - URL
    - rewriting de l'URL (sécurité faible)
  - SSL: gestion par SSL
    - très bien en théorie, mais ne fonctionne pas vraiment
- **Possibilité de configurer COOKIE uniquement (défaut: URL, COOKIE)**

Cookie de Session

- **<session-config>/<cookie-config> dans web.xml**
- **Configuration de:**
  - <name>: nom du cookie
  - <path>: chemin (attention: par défaut context path)
  - <domain>: masque de host
  - <comment>: commentaire du cookie
  - <secure>: cookie secure
  - <http-only>: cookie non disponible via javascript
  - <max-age>: expiration du cookie

JSP

- Pluggabilité de l'implémentation JSP
- API standard pour représenter <jsp-config> de web.xml
- Disponible par ServletContext.getJspConfigDescriptor()
- Potentiellement utile pour tester plusieurs implémentations JSP
  - ... mais pas en production

Conclusion

www.parisjug.org

Conclusion

- **Enormément d'ajouts pour l'intégration**
  - Mais complexité de déploiement importante
- **APIs intéressantes pour le développeur**
  - Sécurité
  - Async
- **Attention:**
  - Spécification non finale
  - Beacoup de changements depuis le PFD

Bibliographie / liens

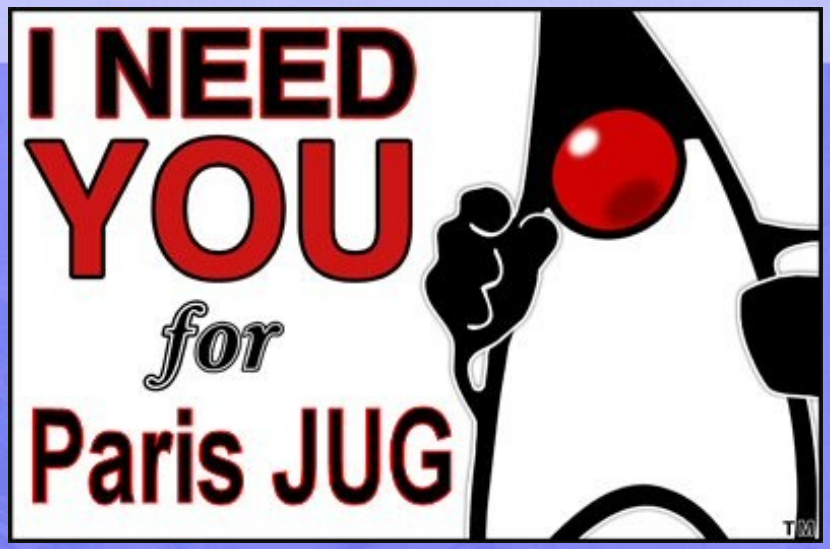
- JSR 315 – Servlet 3.0: http://www.jcp.org/en/jsr/summary?id=315
- JBoss Web: http://jboss.org/jbossweb/
- JBoss AS: http://jboss.org/jbossas/

Questions / Réponses

www.parisjug.org

Sponsors

Merci de votre attention!



www.parisjug.org

Licence

- http://creativecommons.org/licenses/by-nc-sa/2.0/fr/