POLITECNICO
DI MILANO

Politecnico di Milano
A.A. 2015-2016
Software Engineering 2

Code Inspection (CI)

version 1.0

Annalisa Rotondaro (mat. 854268)

Lorenzo Federico Porro (mat. 859093)

Release date: 5 january 2016

Contents

1 Assigned Class and Methods

Class: **WebBundleRuntimeNode** extract from a release of Glassfish 4.1 application server.

Methods: All methods assigned to us are inside WebBundleRuntimeNode:

- **startElement(XMLElement element, Attributes attributes)**

```
279     @Override
280     public void startElement(XMLElement element, Attributes attributes) {
281         if (element.getQName().equals(RuntimeTagNames.PARAMETER_ENCODING)) {
282             SunWebAppImpl sunWebApp = (SunWebAppImpl)getSunDescriptor();
283             sunWebApp.setParameterEncoding(true);
284             for (int i=0; i<attributes.getLength();i++) {
285                 if (RuntimeTagNames.DEFAULT_CHARSET.equals(
286                     attributes.getQName(i))) {
287                     sunWebApp.setAttributeValue(SunWebApp.PARAMETER_ENCODING, SunWebApp.DEFAULT_CHARSET, attributes.getValue(i));
288                 }
289                 if (RuntimeTagNames.FORM_HINT_FIELD.equals(
290                     attributes.getQName(i))) {
291                     sunWebApp.setAttributeValue(SunWebApp.PARAMETER_ENCODING, SunWebApp.FORM_HINT_FIELD, attributes.getValue(i));
292                 }
293             }
294         } else super.startElement(element, attributes);
295     }
```

Figure 1: startElement method from WebBundleRuntimeNode class

- **setElementValue(XMLElement element, String value)**

```
322     /**
323      * receives notification of the value for a particular tag
324      *
325      * @param element the xml element
326      * @param value it's associated value
327      */
328     @Override
329     public void setElementValue(XMLElement element, String value) {
330         if (element.getQName().equals(RuntimeTagNames.CONTEXT_ROOT)) {
331             // only set the context root for standalone war;
332             // for embedded war, the context root will be set
333             // using the value in application.xml
334             Application app = descriptor.getApplication();
335             if (app == null || app.isVirtual()) {
336                 descriptor.setContextRoot(value);
337             }
338         } else if (element.getQName().equals(RuntimeTagNames.KEEP_STATE)) {
339             descriptor.setKeepState(value);
340         } else if (element.getQName().equals(RuntimeTagNames.VERSION_IDENTIFIER)) {
341         } else
342             super.setElementValue(element, value);
343     }
```

Figure 2: setElementValue method from WebBundleRuntimeNode class

- **writeDescription(Node parent, WebBundleDescriptorImpl bundleDescriptor)**

```
345     /**
346      * write the descriptor class to a DOM tree and return it
347      *
348      * @param parent node for the DOM tree
349      * @param bundleDescriptor the descriptor to write
350      * @return the DOM tree top node
351      */
352     @Override
353     public Node writeDescriptor(Node parent, WebBundleDescriptorImpl bundleDescriptor) {
354         Element web = (Element)super.writeDescriptor(parent, bundleDescriptor);
355         SunWebAppImpl sunWebApp = (SunWebAppImpl) bundleDescriptor.getSunDescriptor();
356 
357         // context-root?
358     appendTextChild(web, RuntimeTagNames.CONTEXT_ROOT, bundleDescriptor.getContextRoot());
359     // security-role-mapping
360     SecurityRoleMapping[] roleMappings = sunWebApp.getSecurityRoleMapping();
361     if (roleMappings!=null && roleMappings.length>0) {
362         SecurityRoleMappingNode srmn = new SecurityRoleMappingNode();
363         for (int i=0;i<roleMappings.length;i++) {
364         srmn.writeDescriptor(web, RuntimeTagNames.SECURITY_ROLE_MAPPING, roleMappings[i]);
365         }
366     }
367 
368         // servlet
369         Set servlets = bundleDescriptor.getServletDescriptors();
370         org.glassfish.web.deployment.node.runtime.gf.ServletNode servletNode =
371                 new org.glassfish.web.deployment.node.runtime.gf.ServletNode();
372         for (Iterator itr=servlets.iterator();itr.hasNext();) {
373                 WebComponentDescriptor servlet = (WebComponentDescriptor) itr.next();
374             servletNode.writeDescriptor(web, RuntimeTagNames.SERVLET, servlet);
375         }
376 
```

```
377         // idempotent-url-pattern
378         IdempotentUrlPattern[] patterns = sunWebApp.getIdempotentUrlPatterns();
379         if (patterns != null && patterns.length > 0) {
380             IdempotentUrlPatternNode node = new IdempotentUrlPatternNode();
381             for (int i = 0;i < patterns.length; i++) {
382                 node.writeDescriptor(web, RuntimeTagNames.IDEMPOTENT_URL_PATTERN, patterns[i]);
383             }
384         }
385 
386     // session-config?
387     if (sunWebApp.getSessionConfig()!=null) {
388         SessionConfigNode scn = new SessionConfigNode();
389         scn.writeDescriptor(web, RuntimeTagNames.SESSION_CONFIG, sunWebApp.getSessionConfig());
390     }
391 
392     // ejb-ref*
393     Set<EjbReference> ejbRefs = bundleDescriptor.getEjbReferenceDescriptors();
394     if (ejbRefs.size()>0) {
395         EjbRefNode node = new EjbRefNode();
396         for (EjbReference ejbRef : ejbRefs) {
397         node.writeDescriptor(web, RuntimeTagNames.EJB_REF, ejbRef);
398         }
399     }
400 
401     // resource-ref*
402     Set<ResourceReferenceDescriptor> resourceRefs = bundleDescriptor.getResourceReferenceDescriptors();
403     if (resourceRefs.size()>0) {
404         ResourceRefNode node = new ResourceRefNode();
405         for (ResourceReferenceDescriptor resourceRef : resourceRefs) {
406         node.writeDescriptor(web, RuntimeTagNames.RESOURCE_REF, resourceRef);
407         }
408     }
409 
```

```
410     // resource-env-ref*
411     Set<ResourceEnvReferenceDescriptor> resourceEnvRefs = bundleDescriptor.getResourceEnvReferenceDescriptors();
412     if (resourceEnvRefs.size()>0) {
413         ResourceEnvRefNode node = new ResourceEnvRefNode();
414         for (ResourceEnvReferenceDescriptor resourceEnvRef : resourceEnvRefs) {
415         node.writeDescriptor(web, RuntimeTagNames.RESOURCE_ENV_REF, resourceEnvRef);
416         }
417     }
418 
419     // service-ref*
420     if (bundleDescriptor.hasServiceReferenceDescriptors()) {
421         ServiceRefNode serviceNode = new ServiceRefNode();
422         for (Iterator serviceItr=bundleDescriptor.getServiceReferenceDescriptors().iterator();
423                 serviceItr.hasNext();) {
424         ServiceReferenceDescriptor next = (ServiceReferenceDescriptor) serviceItr.next();
425         serviceNode.writeDescriptor(web, WebServicesTagNames.SERVICE_REF, next);
426         }
427     }
428 
429         // message-destination-ref*
430         MessageDestinationRefNode.writeMessageDestinationReferences(web,
431             bundleDescriptor);
432 
433 
434     // cache?
435     Cache cache = sunWebApp.getCache();
436     if (cache!=null) {
437         CacheNode cn = new CacheNode();
438         cn.writeDescriptor(web, RuntimeTagNames.CACHE, cache);
439     }
440 
```

```
441     // class-loader?
442         ClassLoader classLoader = sunWebApp.getClassLoader();
443         if (classLoader!=null) {
444             ClassLoaderNode cln = new ClassLoaderNode();
445             cln.writeDescriptor(web, RuntimeTagNames.CLASS_LOADER, classLoader);
446         }
447 
448     // jsp-config?
449     if (sunWebApp.getJspConfig()!=null) {
450         WebPropertyNode propertyNode = new WebPropertyNode();
451         Node jspConfig = appendChild(web, RuntimeTagNames.JSP_CONFIG);
452         propertyNode.writeDescriptor(jspConfig, RuntimeTagNames.PROPERTY, sunWebApp.getJspConfig().getWebProperty())
453     }
454 
455     // locale-charset-info?
456     if (sunWebApp.getLocaleCharsetInfo()!=null) {
457         LocaleCharsetInfoNode localeNode = new LocaleCharsetInfoNode();
458         localeNode.writeDescriptor(web, RuntimeTagNames.LOCALE_CHARSET_INFO,
459             sunWebApp.getLocaleCharsetInfo());
460     }
461 
462         // parameter-encoding?
463         if (sunWebApp.isParameterEncoding()) {
464             Element parameter = appendChild(web, RuntimeTagNames.PARAMETER_ENCODING);
465 
466             if (sunWebApp.getAttributeValue(SunWebApp.PARAMETER_ENCODING, SunWebApp.FORM_HINT_FIELD)!=null) {
467                 setAttribute(parameter, RuntimeTagNames.FORM_HINT_FIELD,
468                         sunWebApp.getAttributeValue(SunWebApp.PARAMETER_ENCODING, SunWebApp.FORM_HINT_FIELD));
469             }
470 
471             if (sunWebApp.getAttributeValue(SunWebApp.PARAMETER_ENCODING, SunWebApp.DEFAULT_CHARSET)!=null) {
472                 setAttribute(parameter, RuntimeTagNames.DEFAULT_CHARSET,
473                         sunWebApp.getAttributeValue(SunWebApp.PARAMETER_ENCODING, SunWebApp.DEFAULT_CHARSET));
474             }
475         }
476 
```

```
477         // property*
478         WebPropertyNode props = new WebPropertyNode();
479         props.writeDescriptor(web, RuntimeTagNames.PROPERTY, sunWebApp.getWebProperty());
480 
481         // valve*
482         if (sunWebApp.getValve()!=null) {
483             ValveNode valve = new ValveNode();
484             valve.writeDescriptor(web, RuntimeTagNames.VALVE,
485                                   sunWebApp.getValve());
486         }
487 
488     // message-destination*
489         RuntimeDescriptorNode.writeMessageDestinationInfo(web, bundleDescriptor);
490 
491     // webservice-description*
492         WebServiceRuntimeNode webServiceNode = new WebServiceRuntimeNode();
493         webServiceNode.writeWebServiceRuntimeInfo(web, bundleDescriptor);
494 
495         // error-url
496         if (sunWebApp.getAttributeValue(SunWebApp.ERROR_URL) != null) {
497             setAttribute(web, RuntimeTagNames.ERROR_URL, sunWebApp.getAttributeValue(SunWebApp.ERROR_URL));
498         }
499 
500         // httpservlet-security-provider
501         if (sunWebApp.getAttributeValue(SunWebApp.HTTPSERVLET_SECURITY_PROVIDER) != null) {
502             setAttribute(web, RuntimeTagNames.HTTPSERVLET_SECURITY_PROVIDER,
503                          sunWebApp.getAttributeValue(SunWebApp.HTTPSERVLET_SECURITY_PROVIDER));
504         }
505 
506         // keep-state
507         appendTextChild(web, RuntimeTagNames.KEEP_STATE, String.valueOf(bundleDescriptor.getKeepState()));
508 
509         return web;
510     }
511 }
```

Figure 3: writeDescriptor method from WebBundleRuntimeNode class

2 Functional role of assigned set of classes

The class “WebBundleRuntimeNode” checked, according to the javadoc, is responsible of handling all runtime informations for the web bundle.

org.glassfish.web.deployment.node.runtime.gf

Class WebBundleRuntimeNode

java.lang.Object
    com.sun.enterprise.deployment.node.DeploymentDescriptorNode<T>
        com.sun.enterprise.deployment.node.runtime.RuntimeBundleNode<WebBundleDescriptorImpl>
            org.glassfish.web.deployment.node.runtime.gf.WebBundleRuntimeNode

All Implemented Interfaces:

`RootXMLNode<WebBundleDescriptorImpl>, XMLNode<WebBundleDescriptorImpl>`

Direct Known Subclasses:

`GFWebBundleRuntimeNode`

```
public class WebBundleRuntimeNode
extends RuntimeBundleNode<WebBundleDescriptorImpl>
```

This node is responsible for handling all runtime information for web bundle.

Figure 4: WebBundleRuntimeNode javadoc

The method "startElement (XMLElement element, Attributes attributes)" has no clear javadoc but, from the javadoc of the interface "XMLNode" results that it's role is the notification and the start of an XML element tag in the processed XML source file. Parameters: element (is the XML element type name), attributes (are the specified or default attributes).

startElement

```
public void startElement(XMLElement element,
                         Attributes attributes)
```

Description copied from class: `DeploymentDescriptorNode`

SAX Parser API implementation, we don't really care for now.

Specified by:

`startElement in interface XMLNode<WebBundleDescriptorImpl>`

Overrides:

`startElement in class DeploymentDescriptorNode<WebBundleDescriptorImpl>`

Parameters:

`element - the XML element type name`

`attributes - the specified or defaultted attritutes`

Figure 5: startElement javadoc

The method "setElementValue(XMLElement element, String value)" sets the value of an XML element passed as a parameter.

setElementValue

```
public void setElementValue(XMLElement element,
                            String value)
```

receives notification of the value for a particular tag

Specified by:

`setElementValue in interface XMLNode<WebBundleDescriptorImpl>`

Overrides:

`setElementValue in class RuntimeBundleNode<WebBundleDescriptorImpl>`

Parameters:

`element - the xml element`

`value - it's associated value`

Figure 6: setElementValue javadoc

The method "writeDescriptor (Node parent, WebBundleDescriptorImpl bundleDescriptor)" writes the descriptor to a JAXP DOM node and returns it.

writeDescriptor

```
public Node writeDescriptor(Node parent,
                  WebBundleDescriptorImpl bundleDescriptor)
```

write the descriptor class to a DOM tree and return it

Specified by:

`writeDescriptor in interface XMLNode<WebBundleDescriptorImpl>`

Overrides:

`writeDescriptor in class DeploymentDescriptorNode<WebBundleDescriptorImpl>`

Parameters:

`parent - node for the DOM tree`

`bundleDescriptor - the descriptor to write`

Returns:

`the DOM tree top node`

Figure 7: writeDescriptor javadoc

3 List of issues found by applying the checklist

3.1 Code inspection checklist

3.1.1 Naming Conventions

1) "All class names, interface names, method names, class variables, method variables, and constants used should have meaningful names and do what the name suggests."

- **Fig. 8**

1. The method "getQName()" called by the method "startElement()" has a poor meaningful name: outside his declaration context is not clear what the method actually returns.

```
279     @Override
280     public void startElement(XMLElement element, Attributes attributes) {
281         if (element.getQName().equals(RuntimeTagNames.PARAMETER_ENCODING)) {
282             SunWebAppImpl sunWebApp = (SunWebAppImpl)getSunDescriptor();
283             sunWebApp.setParameterEncoding(true);
```

Figure 8: getQName() method call.

- **Fig. 9**

1. The name of the variable "web" is too generic: it' s easy to mistake it' s meaning.

```
352     @Override
353     public Node writeDescriptor(Node parent, WebBundleDescriptorImpl bundleDescriptor) {
354         Element web = (Element)super.writeDescriptor(parent, bundleDescriptor);
355         SunWebAppImpl sunWebApp = (SunWebAppImpl) bundleDescriptor.getSunDescriptor();
```

Figure 9: "web" variable declaration.

- **Fig. 10**

1. The name of the variable "srmn" is an acronym but not an abvious one, it' s name won't help identify it' s meaning outside it' s declaration context.

```
360     SecurityRoleMapping[] roleMappings = sunWebApp.getSecurityRoleMapping();
361     if (roleMappings!=null && roleMappings.length>0) {
362         SecurityRoleMappingNode srmn = new SecurityRoleMappingNode();
363         for (int i=0;i<roleMappings.length;i++) {
364         srmn.writeDescriptor(web, RuntimeTagNames.SECURITY_ROLE_MAPPING, roleMappings[i]);
365         }
366     }
```

Figure 10: srmn variable declaration.

- **Fig. 11**

1. The name of the variable “patterns” is too generic: it’ s easy to mistake it’ s meaning.

2. The name of the variable “node” is too generic: it’s easy to mistake it’ s meaning.

3. The name of the variable “scn” is an acronym but not an obvious one, it’ s name won’t help identify it’ s meaning outside it’ s declaration context.

```
377         // idempotent-url-pattern
378         IdempotentUrlPattern[] patterns = sunWebApp.getIdempotentUrlPatterns();
379         if (patterns != null && patterns.length > 0) {
380             IdempotentUrlPatternNode node = new IdempotentUrlPatternNode();
381             for (int i = 0;i < patterns.length; i++) {
382                 node.writeDescriptor(web, RuntimeTagNames.IDEMPOTENT_URL_PATTERN, patterns[i]);
383             }
384         }
385
386     // session-config?
387     if (sunWebApp.getSessionConfig()!=null) {
388         SessionConfigNode scn = new SessionConfigNode();
389         scn.writeDescriptor(web, RuntimeTagNames.SESSION_CONFIG, sunWebApp.getSessionConfig());
390     }
```

Figure 11: “patterns”, “node” and “scn” variables declarations.

- **Fig. 12**

1. The name of the variable “serviceItr” is too generic: although is clear that is an iterator, it’ s easy to mistake it’ s meaning.

```
419     // service-ref*
420     if (bundleDescriptor.hasServiceReferenceDescriptors()) {
421         ServiceRefNode serviceNode = new ServiceRefNode();
422         for (Iterator serviceItr=bundleDescriptor.getServiceReferenceDescriptors().iterator();
423                 serviceItr.hasNext();) {
424         ServiceReferenceDescriptor next = (ServiceReferenceDescriptor) serviceItr.next();
425         serviceNode.writeDescriptor(web, WebServicesTagNames.SERVICE_REF, next);
426         }
427     }
```

Figure 12: “serviceItr” variable declaration.

- **Fig. 13**

1. The name of the variable "props" is too generic: it' s easy to mistake it' s meaning.
2. The name of the variable "valve" is too generic: it' s easy to mistake it' s meaning.

```
477         // property*
478         WebPropertyNode props = new WebPropertyNode();
479         props.writeDescriptor(web, RuntimeTagNames.PROPERTY, sunWebApp.getWebProperty());
480
481         // valve*
482         if (sunWebApp.getValve()!=null) {
483             ValveNode valve = new ValveNode();
484             valve.writeDescriptor(web, RuntimeTagNames.VALVE,
485                                   sunWebApp.getValve());
486         }
```

Figure 13: "props" and "valve" variables declaration

2) "If one-character variables are used, they are used only for temporary "throwaway" variables, such as those used in for loops."

No issues found for this point.

3) "Class names are nouns, in mixed case, with the first letter of each word in capitalized. Examples: class Raster; class ImageSprite;"

- **Fig. 14**

1. If "XML" is considered a single word the name of the variable type "XMLElement" should be: "XmlElement".

```
279     @Override
280     public void startElement(XMLElement element, Attributes attributes) {
281         if (element.getQName().equals(RuntimeTagNames.PARAMETER_ENCODING)) {
282             SunWebAppImpl sunWebApp = (SunWebAppImpl)getSunDescriptor();
```

Figure 14: "XMLElement" variable type call

4) "Interface names should be capitalized like classes."

No issues found for this point.

5) "Method names should be verbs, with the first letter of each addition word capitalized. Examples: getBackground(); computeTemperature()."

No issues found for this point.

6) "Class variables, also called attributes, are mixed case, but might begin with an underscore ('_') followed by a lowercase first letter. All the remaining words in the variable name have their first letter capitalized. Examples: _windowHeight, timeSeriesData."

No issues found for this point.

7) "Constants are declared using all uppercase with words separated by an underscore. Examples: MIN_WIDTH; MAX_HEIGHT;"

No issues found for this point.

3.1.2 Indention

8) "Three or four spaces are used for indentation and done so consistently"

- **Fig. 15**

1. All the lines from 358 to 366 should be moved 4 spaces to the right, line 364 should be moved by 4 additional spaces to the right to respect for statement indention.
2. Line 371 should be moved 8 spaces to the left.
3. Line 373 should be moved 4 spaces to the left.

```
345     /**
346      * write the descriptor class to a DOM tree and return it
347      *
348      * @param parent node for the DOM tree
349      * @param bundleDescriptor the descriptor to write
350      * @return the DOM tree top node
351      */
352     @Override
353     public Node writeDescriptor(Node parent, WebBundleDescriptorImpl bundleDescriptor) {
354         Element web = (Element)super.writeDescriptor(parent, bundleDescriptor);
355         SunWebAppImpl sunWebApp = (SunWebAppImpl) bundleDescriptor.getSunDescriptor();
356
357         // context-root?
358     appendTextChild(web, RuntimeTagNames.CONTEXT_ROOT, bundleDescriptor.getContextRoot());
359     // security-role-mapping
360     SecurityRoleMapping[] roleMappings = sunWebApp.getSecurityRoleMapping();
361     if (roleMappings!=null && roleMappings.length>0) {
362         SecurityRoleMappingNode srmn = new SecurityRoleMappingNode();
363         for (int i=0;i<roleMappings.length;i++) {
364     ➔       srmn.writeDescriptor(web, RuntimeTagNames.SECURITY_ROLE_MAPPING, roleMappings[i]);
365         }
366     }
367
368         // servlet
369         Set servlets = bundleDescriptor.getServletDescriptors();
370         org.glassfish.web.deployment.node.runtime.gf.ServletNode servletNode =
371                 new org.glassfish.web.deployment.node.runtime.gf.ServletNode();
372         for (Iterator itr=servlets.iterator();itr.hasNext();) {
373                 WebComponentDescriptor servlet = (WebComponentDescriptor) itr.next();
374             servletNode.writeDescriptor(web, RuntimeTagNames.SERVLET, servlet);
375         }
376
```

Figure 15: Indention issues in highlighted lines

- **Fig. 16**

1. All the lines from 386 to 408 should be moved 4 spaces on the right, lines 397 and 406 should be moved 4 additional spaces to the right to respect for statement indention.

```
377         // idempotent-url-pattern
378         IdempotentUrlPattern[] patterns = sunWebApp.getIdempotentUrlPatterns();
379         if (patterns != null && patterns.length > 0) {
380             IdempotentUrlPatternNode node = new IdempotentUrlPatternNode();
381             for (int i = 0;i < patterns.length; i++) {
382                 node.writeDescriptor(web, RuntimeTagNames.IDEMPOTENT_URL_PATTERN, patterns[i]);
383             }
384         }
385
386     // session-config?
387     if (sunWebApp.getSessionConfig()!=null) {
388         SessionConfigNode scn = new SessionConfigNode();
389         scn.writeDescriptor(web, RuntimeTagNames.SESSION_CONFIG, sunWebApp.getSessionConfig());
390     }
391
392     // ejb-ref*
393     Set<EjbReference> ejbRefs = bundleDescriptor.getEjbReferenceDescriptors();
394     if (ejbRefs.size()>0) {
395         EjbRefNode node = new EjbRefNode();
396         for (EjbReference ejbRef : ejbRefs) {
397         node.writeDescriptor(web, RuntimeTagNames.EJB_REF, ejbRef);
398         }
399     }
400
401     // resource-ref*
402     Set<ResourceReferenceDescriptor> resourceRefs = bundleDescriptor.getResourceReferenceDescriptors();
403     if (resourceRefs.size()>0) {
404         ResourceRefNode node = new ResourceRefNode();
405         for (ResourceReferenceDescriptor resourceRef : resourceRefs) {
406         node.writeDescriptor(web, RuntimeTagNames.RESOURCE_REF, resourceRef);
407         }
408     }
```

Figure 16: Indention issues in highlighted lines

- **Fig. 17**

1. All the lines from 401 to 427 should be moved 4 spaces to the right.
2. Line 423 should be moved 8 spaces to the left to respect for statement indention.
3. Lines 424 and 425 should be moved 4 spaces to the right.
4. Line 431 should be moved 4 spaces to the left.

Figure 17: Indention issues in highlighted lines

- **Fig. 18**

1. All the lines from 434 to 439 should be moved 4 spaces to the right.
2. Line 411 should be moved 4 spaces to the right.
3. All the lines from 448 to 460 should be moved 4 spaces to the right.
4. Line 468 should be moved 4 spaces to the left.

```
434     // cache?
435     Cache cache = sunWebApp.getCache();
436     if (cache!=null) {
437         CacheNode cn = new CacheNode();
438         cn.writeDescriptor(web, RuntimeTagNames.CACHE, cache);
439     }
440
441     // class-loader?
442         ClassLoader classLoader = sunWebApp.getClassLoader();
443         if (classLoader!=null) {
444             ClassLoaderNode cln = new ClassLoaderNode();
445             cln.writeDescriptor(web, RuntimeTagNames.CLASS_LOADER, classLoader);
446         }
447
448     // jsp-config?
449     if (sunWebApp.getJspConfig()!=null) {
450         WebPropertyNode propertyNode = new WebPropertyNode();
451         Node jspConfig = appendChild(web, RuntimeTagNames.JSP_CONFIG);
452         propertyNode.writeDescriptor(jspConfig, RuntimeTagNames.PROPERTY, sunWebApp.getJspConfig().getWebProperty());
453     }
454
455     // locale-charset-info?
456     if (sunWebApp.getLocaleCharsetInfo()!=null) {
457         LocaleCharsetInfoNode localeNode = new LocaleCharsetInfoNode();
458         localeNode.writeDescriptor(web, RuntimeTagNames.LOCALE_CHARSET_INFO,
459             sunWebApp.getLocaleCharsetInfo());
460     }
461
462         // parameter-encoding?
463         if (sunWebApp.isParameterEncoding()) {
464             Element parameter = appendChild(web, RuntimeTagNames.PARAMETER_ENCODING);
465
466             if (sunWebApp.getAttributeValue(SunWebApp.PARAMETER_ENCODING, SunWebApp.FORM_HINT_FIELD)!=null) {
467                 setAttribute(parameter, RuntimeTagNames.FORM_HINT_FIELD,
468                         sunWebApp.getAttributeValue(SunWebApp.PARAMETER_ENCODING, SunWebApp.FORM_HINT_FIELD));
```

Figure 18: Indention issues in highlighted lines

- **Fig. 19**

1. Lines 488 and 491 should be moved 4 spaces to the right.
2. Line 473 should be moved 4 spaces to the left.
3. Line 485 should be moved 22 spaces to the left.
4. Line 503 should be moved 13 spaces to the left.

```
470
471         if (sunWebApp.getAttributeValue(SunWebApp.PARAMETER_ENCODING, SunWebApp.DEFAULT_CHARSET)!=null) {
472             setAttribute(parameter, RuntimeTagNames.DEFAULT_CHARSET,
473                   (sunWebApp.getAttributeValue(SunWebApp.PARAMETER_ENCODING, SunWebApp.DEFAULT_CHARSET));
474         }
475     }
476
477     // property*
478     WebPropertyNode props = new WebPropertyNode();
479     props.writeDescriptor(web, RuntimeTagNames.PROPERTY, sunWebApp.getWebProperty());
480
481     // valve*
482     if (sunWebApp.getValve()!=null) {
483         ValveNode valve = new ValveNode();
484         valve.writeDescriptor(web, RuntimeTagNames.VALVE,
485                               (sunWebApp.getValve());
486     }
487
488 // message-destination*
489     RuntimeDescriptorNode.writeMessageDestinationInfo(web, bundleDescriptor);
490
491 // webservice-description*
492     WebServiceRuntimeNode webServiceNode = new WebServiceRuntimeNode();
493     webServiceNode.writeWebServiceRuntimeInfo(web, bundleDescriptor);
494
495     // error-url
496     if (sunWebApp.getAttributeValue(SunWebApp.ERROR_URL) != null) {
497         setAttribute(web, RuntimeTagNames.ERROR_URL, sunWebApp.getAttributeValue(SunWebApp.ERROR_URL));
498     }
499
500     // httpservlet-security-provider
501     if (sunWebApp.getAttributeValue(SunWebApp.HTTPSERVLET_SECURITY_PROVIDER) != null) {
502         setAttribute(web, RuntimeTagNames.HTTPSERVLET_SECURITY_PROVIDER,
503                     (sunWebApp.getAttributeValue(SunWebApp.HTTPSERVLET_SECURITY_PROVIDER));
504     }
```

Figure 19: Indention issues in highlighted lines

9) "No tabs are used to indent"

1. Inside the method "writeDescriptor()" from the class WebBundleRuntimeNode the following lines contains tabs indention: 358; 359; 360; 361; 362; 363; 364; 365; 366; 367; 386; 387; 388; 389; 390; 392; 393; 394; 395; 396; 397; 398; 399; 400; 401; 402; 403; 404; 405; 406; 407; 408; 410; 411; 412; 413; 414; 415; 416; 417; 419; 420; 421; 422; 423; 424; 425; 426; 427; 433; 434; 435; 436; 437; 438; 439; 440; 441; 448; 449; 450; 451; 452; 453; 454; 455; 456; 457; 458; 459; 460; 461; 477; 478; 479; 488; 491.

3.1.3 Braces

10) "Consistent bracing style is used, either the preferred "Allman" style (first brace goes underneath the opening block) or the "Kernighan and Ritchie" style (first brace is on the same line of the instruction that opens the new block)."

No issues found for this point. Consistent use of Kernighan and Ritchie style.

11) "All if, while, do-while, try-catch, and for statements that have only one statement to execute are surrounded by curly brace."

No issues found for this point.

3.1.4 File organization

12) "Blank lines and optional comments are used to separate sections (beginning comments, package/import statements, class/interface declarations which include class variable/attributes declarations, constructors, and methods)."

- **Fig. 20**

1. There' s a blank space in line 66 between import statements that has no meaning.

- **Fig. 21**

1. Methods are separed by one blank line throughout the class, however before the method "setElementValue(...)" there are two blank lines (320 and 321).

13) "Where practical, line length does not exceed 80 characters."

- **Fig. 22**

1. Lines 287 and 291 in method "startElement()" exceeds character limit.

```
 2 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS HEADER.
40
41 package org.glassfish.web.deployment.node.runtime.gf;
42
43 import com.sun.enterprise.deployment.*;
44 import com.sun.enterprise.deployment.node.XMLElement;
45 import com.sun.enterprise.deployment.node.runtime.*;
46 import com.sun.enterprise.deployment.node.runtime.EjbRefNode;
47 import com.sun.enterprise.deployment.node.runtime.ResourceEnvRefNode;
48 import com.sun.enterprise.deployment.node.runtime.ResourceRefNode;
49 import com.sun.enterprise.deployment.node.runtime.common.SecurityRoleMappingNode;
50 import com.sun.enterprise.deployment.runtime.common.*;
51 import com.sun.enterprise.deployment.runtime.web.*;
52 import com.sun.enterprise.deployment.types.EjbReference;
53 import com.sun.enterprise.deployment.xml.DTDRegistry;
54 import com.sun.enterprise.deployment.xml.RuntimeTagNames;
55 import com.sun.enterprise.deployment.xml.WebServicesTagNames;
56 import org.glassfish.deployment.common.SecurityRoleMapper;
57 import org.glassfish.web.deployment.descriptor.WebBundleDescriptorImpl;
58 import org.glassfish.web.deployment.node.WebBundleNode;
59 import org.glassfish.web.deployment.runtime.ClassLoader;
60 import org.glassfish.web.deployment.runtime.*;
61 import org.glassfish.security.common.Group;
62 import org.glassfish.security.common.Role;
63 import org.w3c.dom.Element;
64 import org.w3c.dom.Node;
65 import org.xml.sax.Attributes;
66
67 import java.util.Iterator;
68 import java.util.List;
69 import java.util.Map;
70 import java.util.Set;
71
72
73 /**
```

Figure 20: Blank line between imports.

```
319     }
320
321
322     /**
323      * receives notification of the value for a particular tag
324      *
325      * @param element the xml element
326      * @param value it's associated value
327      */
328     @Override
329     public void setElementValue(XMLElement element, String value) {
330         if (element.getQName().equals(RuntimeTagNames.CONTEXT_ROOT)) {
331             // only set the context root for standalone war;
332             // for embedded war, the context root will be set
333             // using the value in application.xml
334             Application app = descriptor.getApplication();
335             if (app == null || app.isVirtual()) {
336                 descriptor.setContextRoot(value);
337             }
338         } else if (element.getQName().equals(RuntimeTagNames.KEEP_STATE)) {
339             descriptor.setKeepState(value);
340         } else if (element.getQName().equals(RuntimeTagNames.VERSION_IDENTIFIER)) {
341         } else
342             super.setElementValue(element, value);
343     }
344
345     /**
346      * write the descriptor class to a DOM tree and return it
347      *
348      * @param parent node for the DOM tree
349      * @param bundleDescriptor the descriptor to write
350      * @return the DOM tree top node
351      */
352     @Override
353     public Node writeDescriptor(Node parent, WebBundleDescriptorImpl bundleDescriptor) {
```

Figure 21: Inconsistent method separation

```
284         for (int i=0; i<attributes.getLength();i++) {
285             if (RuntimeTagNames.DEFAULT_CHARSET.equals(
286                 attributes.getQName(i))) {
287                 sunWebApp.setAttributeValue(SunWebApp.PARAMETER_ENCODING, SunWebApp.DEFAULT_CHARSET, attributes.getValue(i));
288             }
289             if (RuntimeTagNames.FORM_HINT_FIELD.equals(
290                 attributes.getQName(i))) {
291                 sunWebApp.setAttributeValue(SunWebApp.PARAMETER_ENCODING, SunWebApp.FORM_HINT_FIELD, attributes.getValue(i));
292             }
293         }
```

Figure 22: Highlighted lines are too long.

- **Fig. 23**

1. Lines 353, 355, 358, 364 and 374 in method “writeDescriptor()” exceeds character limit.

```
345     /**
346      * write the descriptor class to a DOM tree and return it
347      *
348      * @param parent node for the DOM tree
349      * @param bundleDescriptor the descriptor to write
350      * @return the DOM tree top node
351      */
352     @Override
353     public Node writeDescriptor(Node parent, WebBundleDescriptorImpl bundleDescriptor) {
354         Element web = (Element)super.writeDescriptor(parent, bundleDescriptor);
355         SunWebAppImpl sunWebApp = (SunWebAppImpl) bundleDescriptor.getSunDescriptor();
356 
357         // context-root?
358     appendTextChild(web, RuntimeTagNames.CONTEXT_ROOT, bundleDescriptor.getContextRoot());
359     // security-role-mapping
360     SecurityRoleMapping[] roleMappings = sunWebApp.getSecurityRoleMapping();
361     if (roleMappings!=null && roleMappings.length>0) {
362         SecurityRoleMappingNode srmn = new SecurityRoleMappingNode();
363         for (int i=0;i<roleMappings.length;i++) {
364         srmn.writeDescriptor(web, RuntimeTagNames.SECURITY_ROLE_MAPPING, roleMappings[i]);
365         }
366     }
367 
368         // servlet
369         Set servlets = bundleDescriptor.getServletDescriptors();
370         org.glassfish.web.deployment.node.runtime.gf.ServletNode servletNode =
371                 new org.glassfish.web.deployment.node.runtime.gf.ServletNode();
372         for (Iterator itr=servlets.iterator();itr.hasNext();) {
373                 WebComponentDescriptor servlet = (WebComponentDescriptor) itr.next();
374             servletNode.writeDescriptor(web, RuntimeTagNames.SERVLET, servlet);
375         }
376 
```

Figure 23: Highlighted lines are too long.

- **Fig. 24**

1. Lines 382, 389 and 402 in method “writeDescriptor()” exceeds character limit.

- **Fig. 25**

```
377         // idempotent-url-pattern
378         IdempotentUrlPattern[] patterns = sunWebApp.getIdempotentUrlPatterns();
379         if (patterns != null && patterns.length > 0) {
380             IdempotentUrlPatternNode node = new IdempotentUrlPatternNode();
381             for (int i = 0;i < patterns.length; i++) {
382                 node.writeDescriptor(web, RuntimeTagNames.IDEMPOTENT_URL_PATTERN, patterns[i]);
383             }
384         }
385
386     // session-config?
387     if (sunWebApp.getSessionConfig()!=null) {
388         SessionConfigNode scn = new SessionConfigNode();
389         scn.writeDescriptor(web, RuntimeTagNames.SESSION_CONFIG, sunWebApp.getSessionConfig());
390     }
391
392     // ejb-ref*
393     Set<EjbReference> ejbRefs = bundleDescriptor.getEjbReferenceDescriptors();
394     if (ejbRefs.size()>0) {
395         EjbRefNode node = new EjbRefNode();
396         for (EjbReference ejbRef : ejbRefs) {
397         node.writeDescriptor(web, RuntimeTagNames.EJB_REF, ejbRef);
398         }
399     }
400
401     // resource-ref*
402     Set<ResourceReferenceDescriptor> resourceRefs = bundleDescriptor.getResourceReferenceDescriptors();
403     if (resourceRefs.size()>0) {
404         ResourceRefNode node = new ResourceRefNode();
405         for (ResourceReferenceDescriptor resourceRef : resourceRefs) {
406         node.writeDescriptor(web, RuntimeTagNames.RESOURCE_REF, resourceRef);
407         }
408     }
```

Figure 24: Highlighted lines are too long.

1. Lines 411, 415, 422 and 424 in method “writeDescriptor()” exceeds character limit.

```
410     // resource-env-ref*
411     Set<ResourceEnvReferenceDescriptor> resourceEnvRefs = bundleDescriptor.getResourceEnvReferenceDescriptors();
412     if (resourceEnvRefs.size()>0) {
413         ResourceEnvRefNode node = new ResourceEnvRefNode();
414         for (ResourceEnvReferenceDescriptor resourceEnvRef : resourceEnvRefs) {
415         node.writeDescriptor(web, RuntimeTagNames.RESOURCE_ENV_REF, resourceEnvRef);
416         }
417     }
418
419     // service-ref*
420     if (bundleDescriptor.hasServiceReferenceDescriptors()) {
421         ServiceRefNode serviceNode = new ServiceRefNode();
422         for (Iterator serviceItr=bundleDescriptor.getServiceReferenceDescriptors().iterator();
423                 serviceItr.hasNext();;) {
424         ServiceReferenceDescriptor next = (ServiceReferenceDescriptor) serviceItr.next();
425         serviceNode.writeDescriptor(web, WebServicesTagNames.SERVICE_REF, next);
426         }
427     }
428
429         // message-destination-ref*
430         MessageDestinationRefNode.writeMessageDestinationReferences(web,
431             bundleDescriptor);
432
433
434     // cache?
435     Cache cache = sunWebApp.getCache();
436     if (cache!=null) {
437         CacheNode cn = new CacheNode();
438         cn.writeDescriptor(web, RuntimeTagNames.CACHE, cache);
439     }
440
```

Figure 25: Highlighted lines are too long.

- **Fig. 26**

1. Lines 452, 464, 466 and 471 in method “writeDescriptor()” exceeds character limit.

```
441     // class-loader?
442         ClassLoader classLoader = sunWebApp.getClassLoader();
443         if (classLoader!=null) {
444             ClassLoaderNode cln = new ClassLoaderNode();
445             cln.writeDescriptor(web, RuntimeTagNames.CLASS_LOADER, classLoader);
446         }
447
448     // jsp-config?
449     if (sunWebApp.getJspConfig()!=null) {
450         WebPropertyNode propertyNode = new WebPropertyNode();
451         Node jspConfig = appendChild(web, RuntimeTagNames.JSP_CONFIG);
452         propertyNode.writeDescriptor(jspConfig, RuntimeTagNames.PROPERTY, sunWebApp.getJspConfig().getWebProperty());
453     }
454
455     // locale-charset-info?
456     if (sunWebApp.getLocaleCharsetInfo()!=null) {
457         LocaleCharsetInfoNode localeNode = new LocaleCharsetInfoNode();
458         localeNode.writeDescriptor(web, RuntimeTagNames.LOCALE_CHARSET_INFO,
459             sunWebApp.getLocaleCharsetInfo());
460     }
461
462         // parameter-encoding?
463         if (sunWebApp.isParameterEncoding()) {
464             Element parameter = appendChild(web, RuntimeTagNames.PARAMETER_ENCODING);
465
466             if (sunWebApp.getAttributeValue(SunWebApp.PARAMETER_ENCODING, SunWebApp.FORM_HINT_FIELD)!=null) {
467                 setAttribute(parameter, RuntimeTagNames.FORM_HINT_FIELD,
468                         sunWebApp.getAttributeValue(SunWebApp.PARAMETER_ENCODING, SunWebApp.FORM_HINT_FIELD));
469             }
470
471             if (sunWebApp.getAttributeValue(SunWebApp.PARAMETER_ENCODING, SunWebApp.DEFAULT_CHARSET)!=null) {
472                 setAttribute(parameter, RuntimeTagNames.DEFAULT_CHARSET,
473                         sunWebApp.getAttributeValue(SunWebApp.PARAMETER_ENCODING, SunWebApp.DEFAULT_CHARSET));
474             }
475         }
```

Figure 26: Highlighted lines are too long.

- **Fig. 27**

1. Lines 479, 489, 497 and 507 in method “writeDescriptor()” exceeds character limit.

```
477         // property*
478         WebPropertyNode props = new WebPropertyNode();                                    80
479         props.writeDescriptor(web, RuntimeTagNames.PROPERTY, sunWebApp.getWebProperty());
480
481         // valve*
482         if (sunWebApp.getValve()!=null) {
483             ValveNode valve = new ValveNode();
484             valve.writeDescriptor(web, RuntimeTagNames.VALVE,
485                                   sunWebApp.getValve());
486         }
487
488     // message-destination*
489         RuntimeDescriptorNode.writeMessageDestinationInfo(web, bundleDescriptor);
490
491     // webservice-description*
492         WebServiceRuntimeNode webServiceNode = new WebServiceRuntimeNode();
493         webServiceNode.writeWebServiceRuntimeInfo(web, bundleDescriptor);
494
495         // error-url
496         if (sunWebApp.getAttributeValue(SunWebApp.ERROR_URL) != null) {
497             setAttribute(web, RuntimeTagNames.ERROR_URL, sunWebApp.getAttributeValue(SunWebApp.ERROR_URL));
498         }
499
500         // httpservlet-security-provider
501         if (sunWebApp.getAttributeValue(SunWebApp.HTTPSERVLET_SECURITY_PROVIDER) != null) {
502             setAttribute(web, RuntimeTagNames.HTTPSERVLET_SECURITY_PROVIDER,
503                          sunWebApp.getAttributeValue(SunWebApp.HTTPSERVLET_SECURITY_PROVIDER));
504         }
505
506         // keep-state
507         appendTextChild(web, RuntimeTagNames.KEEP_STATE, String.valueOf(bundleDescriptor.getKeepState()));
508
509         return web;
510     }
511 }
```

Figure 27: Highlighted lines are too long.

14) "When line length must exceed 80 characters, it does NOT exceed 120 characters."

- **Fig. 22 (see above)**

1. Lines 287 and 291 in method "startElement()" also exceeds 120 character limit.

3.1.5 Wrapping Lines

15) "Line break occurs after a comma or an operator."

No issues found for this point.

16) "Higher-level breaks are used."

No issues found for this point.

17) "A new statement is aligned with the beginning of the expression at the same level as the previous line."

- **Fig. 28**

1. The beginning of line 358 in method "writeDescriptor()" should be aligned with the beginning of previous line.

```
352    @Override
353    public Node writeDescriptor(Node parent, WebBundleDescriptorImpl bundleDescriptor) {
354        Element web = (Element)super.writeDescriptor(parent, bundleDescriptor);
355        SunWebAppImpl sunWebApp = (SunWebAppImpl) bundleDescriptor.getSunDescriptor();
356
357        // context-root?
358    appendTextChild(web, RuntimeTagNames.CONTEXT_ROOT, bundleDescriptor.getContextRoot());
359    // security-role-mapping
360    SecurityRoleMapping[] roleMappings = sunWebApp.getSecurityRoleMapping();
361    if (roleMappings!=null && roleMappings.length>0) {
362        SecurityRoleMappingNode srmn = new SecurityRoleMappingNode();
363        for (int i=0;i<roleMappings.length;i++) {
364        srmn.writeDescriptor(web, RuntimeTagNames.SECURITY_ROLE_MAPPING, roleMappings[i]);
365        }
366    }
```

Figure 28: Wrong alignement in highlited lines.

- **Fig. 29**

1. The beginning of line 374 in method "writeDescriptor()" should be aligned with the beginning of previous line.

```
368         // servlet
369         Set servlets = bundleDescriptor.getServletDescriptors();
370         org.glassfish.web.deployment.node.runtime.gf.ServletNode servletNode =
371                 new org.glassfish.web.deployment.node.runtime.gf.ServletNode();
372         for (Iterator itr=servlets.iterator();itr.hasNext();) {
373                 WebComponentDescriptor servlet = (WebComponentDescriptor) itr.next();
374             servletNode.writeDescriptor(web, RuntimeTagNames.SERVLET, servlet);
375         }
```

Figure 29: Wrong alignement in highlited lines.

3.1.6 Comments

18) "Comments are used to adequately explain what the class, interface, methods, and blocks of code are doing."

- **Fig. 30**

1. Comment on lines 357 and 368 in method "writeDescriptor()" have poor meanings.

```
357         // context-root?
358     appendTextChild(web, RuntimeTagNames.CONTEXT_ROOT, bundleDescriptor.getContextRoot());
359     // security-role-mapping
360     SecurityRoleMapping[] roleMappings = sunWebApp.getSecurityRoleMapping();
361     if (roleMappings!=null && roleMappings.length>0) {
362         SecurityRoleMappingNode srmn = new SecurityRoleMappingNode();
363         for (int i=0;i<roleMappings.length;i++) {
364         srmn.writeDescriptor(web, RuntimeTagNames.SECURITY_ROLE_MAPPING, roleMappings[i]);
365         }
366     }
367
368         // servlet
369         Set servlets = bundleDescriptor.getServletDescriptors();
370         org.glassfish.web.deployment.node.runtime.gf.ServletNode servletNode =
371                 new org.glassfish.web.deployment.node.runtime.gf.ServletNode();
372         for (Iterator itr=servlets.iterator();itr.hasNext();) {
373                 WebComponentDescriptor servlet = (WebComponentDescriptor) itr.next();
374             servletNode.writeDescriptor(web, RuntimeTagNames.SERVLET, servlet);
375         }
```

Figure 30: Poor meaning of highlighted comments.

- **Fig. 31**

1. Comment on lines 377 and 386 in method “writeDescriptor()” have poor meanings.
2. Comment on lines 392 and 401 in method “writeDescriptor()” have poor meanings and also use acronyms and abbreviations that are not explained.

```
377         // idempotent-url-pattern
378         IdempotentUrlPattern[] patterns = sunWebApp.getIdempotentUrlPatterns();
379         if (patterns != null && patterns.length > 0) {
380             IdempotentUrlPatternNode node = new IdempotentUrlPatternNode();
381             for (int i = 0;i < patterns.length; i++) {
382                 node.writeDescriptor(web, RuntimeTagNames.IDEMPOTENT_URL_PATTERN, patterns[i]);
383             }
384         }
385
386     // session-config?
387     if (sunWebApp.getSessionConfig()!=null) {
388         SessionConfigNode scn = new SessionConfigNode();
389         scn.writeDescriptor(web, RuntimeTagNames.SESSION_CONFIG, sunWebApp.getSessionConfig());
390     }
391
392     // ejb-ref*
393     Set<EjbReference> ejbRefs = bundleDescriptor.getEjbReferenceDescriptors();
394     if (ejbRefs.size()>0) {
395         EjbRefNode node = new EjbRefNode();
396         for (EjbReference ejbRef : ejbRefs) {
397         node.writeDescriptor(web, RuntimeTagNames.EJB_REF, ejbRef);
398         }
399     }
400
401     // resource-ref*
402     Set<ResourceReferenceDescriptor> resourceRefs = bundleDescriptor.getResourceReferenceDescriptors();
```

Figure 31: Poor meaning of highlighted comments.

- **Fig. 32**

1. Comment on lines 410 and 419 in method “writeDescriptor()” have poor meanings and also use acronyms and abbreviations that are not explained.

```
410     // resource-env-ref*
411     Set<ResourceEnvReferenceDescriptor> resourceEnvRefs = bundleDescriptor.getResourceEnvReferenceDescriptors();
412     if (resourceEnvRefs.size()>0) {
413         ResourceEnvRefNode node = new ResourceEnvRefNode();
414         for (ResourceEnvReferenceDescriptor resourceEnvRef : resourceEnvRefs) {
415         node.writeDescriptor(web, RuntimeTagNames.RESOURCE_ENV_REF, resourceEnvRef);
416         }
417     }
418
419     // service-ref*
420     if (bundleDescriptor.hasServiceReferenceDescriptors()) {
421         ServiceRefNode serviceNode = new ServiceRefNode();
422         for (Iterator serviceItr=bundleDescriptor.getServiceReferenceDescriptors().iterator();
423                 serviceItr.hasNext();;) {
424         ServiceReferenceDescriptor next = (ServiceReferenceDescriptor) serviceItr.next();
425         serviceNode.writeDescriptor(web, WebServicesTagNames.SERVICE_REF, next);
426         }
427     }
```

Figure 32: Poor meaning of highlighted comments.

- **Fig. 33**

1. Comment on lines 434 and 441 in method “writeDescriptor()” have poor meanings.

2. Comment on line 448 in method “writeDescriptor()” has poor meaning and also uses acronyms that are not explained.

3. Comment on lines 455 and 462 in method “writeDescriptor()” have poor meanings.

```
434     // cache?
435     Cache cache = sunWebApp.getCache();
436     if (cache!=null) {
437         CacheNode cn = new CacheNode();
438         cn.writeDescriptor(web, RuntimeTagNames.CACHE, cache);
439     }
440
441     // class-loader?
442         ClassLoader classLoader = sunWebApp.getClassLoader();
443         if (classLoader!=null) {
444             ClassLoaderNode cln = new ClassLoaderNode();
445             cln.writeDescriptor(web, RuntimeTagNames.CLASS_LOADER, classLoader);
446         }
447
448     // jsp-config?
449     if (sunWebApp.getJspConfig()!=null) {
450         WebPropertyNode propertyNode = new WebPropertyNode();
451         Node jspConfig = appendChild(web, RuntimeTagNames.JSP_CONFIG);
452         propertyNode.writeDescriptor(jspConfig, RuntimeTagNames.PROPERTY, sunWebApp.getJspConfig().getWebProperty());
453     }
454
455     // locale-charset-info?
456     if (sunWebApp.getLocaleCharsetInfo()!=null) {
457         LocaleCharsetInfoNode localeNode = new LocaleCharsetInfoNode();
458         localeNode.writeDescriptor(web, RuntimeTagNames.LOCALE_CHARSET_INFO,
459             sunWebApp.getLocaleCharsetInfo());
460     }
461
462         // parameter-encoding?
463         if (sunWebApp.isParameterEncoding()) {
464             Element parameter = appendChild(web, RuntimeTagNames.PARAMETER_ENCODING);
```

Figure 33: Poor meaning of highlighted comments.

- **Fig. 34**

1. Comment on lines 477, 481, 495 and 500 in method “writeDescriptor()” have poor meanings.

```
477         // property*
478         WebPropertyNode props = new WebPropertyNode();
479         props.writeDescriptor(web, RuntimeTagNames.PROPERTY, sunWebApp.getWebProperty());
480
481         // valve*
482         if (sunWebApp.getValve()!=null) {
483             ValveNode valve = new ValveNode();
484             valve.writeDescriptor(web, RuntimeTagNames.VALVE,
485                                   sunWebApp.getValve());
486         }
487
488     // message-destination*
489         RuntimeDescriptorNode.writeMessageDestinationInfo(web, bundleDescriptor);
490
491     // webservice-description*
492         WebServiceRuntimeNode webServiceNode = new WebServiceRuntimeNode();
493         webServiceNode.writeWebServiceRuntimeInfo(web, bundleDescriptor);
494
495         // error-url
496         if (sunWebApp.getAttributeValue(SunWebApp.ERROR_URL) != null) {
497             setAttribute(web, RuntimeTagNames.ERROR_URL, sunWebApp.getAttributeValue(SunWebApp.ERROR_URL));
498         }
499
500         // httpservlet-security-provider
501         if (sunWebApp.getAttributeValue(SunWebApp.HTTPSERVLET_SECURITY_PROVIDER) != null) {
502             setAttribute(web, RuntimeTagNames.HTTPSERVLET_SECURITY_PROVIDER,
503                          sunWebApp.getAttributeValue(SunWebApp.HTTPSERVLET_SECURITY_PROVIDER));
504         }
```

Figure 34: Poor meaning of highlighted comments.

19) "Commented out code contains a reason for being commented out and a date it can be removed from the source file if determined it is no longer needed."

No issues found for this point.

3.1.7 Java Source Files

20) "Each Java source file contains a single public class or interface."

No issues found for this point.

21) "The public class is the first class or interface in the file."

No issues found for this point.

22) "Check that the external program interfaces are implemented consistently with what is described in the javadoc."

No issues found for this point.

23) "Check that the javadoc is complete (i.e., it covers all classes and files part of the set of classes assigned to you)."

- **General**

1. The method "setAttributeValue(...)" doesn' t figure in the javadoc document.
2. The methods "getApplication()" and "setKeepState()" used in the method "setElementValue(...)" of class "WebBundleDescriptorImpl" don' t figure in the javadoc document.

- **Fig. 35**

1. The method "starElement" in class "XMLElement" uses the metthod "getQName" in line 281 but this method has no javadoc.

Figure 35: Missing javadoc for method "getQName()"

- **Fig. 36**

1. The class "SunWebAppImpl" used in line 282 to declare the variable "sunWebApp" has no javadoc.

org.glassfish.web.deployment.runtime

Class SunWebAppImpl

java.lang.Object
  java.util.Observable
    org.glassfish.deployment.common.DynamicAttributesDescriptor
      org.glassfish.deployment.common.Descriptor
        com.sun.enterprise.deployment.runtime.RuntimeDescriptor
          org.glassfish.web.deployment.runtime.WebPropertyContainer
            org.glassfish.web.deployment.runtime.SunWebAppImpl

All Implemented Interfaces:

SunWebApp, Serializable

```
public class SunWebAppImpl
extends WebPropertyContainer
implements SunWebApp
```

See Also:

Serialized Form

Figure 36: Missing javadoc for class "SunWebAppImpl"

- **Fig. 37**

1. The method "setParameterEncoding(...)" has no javadoc.

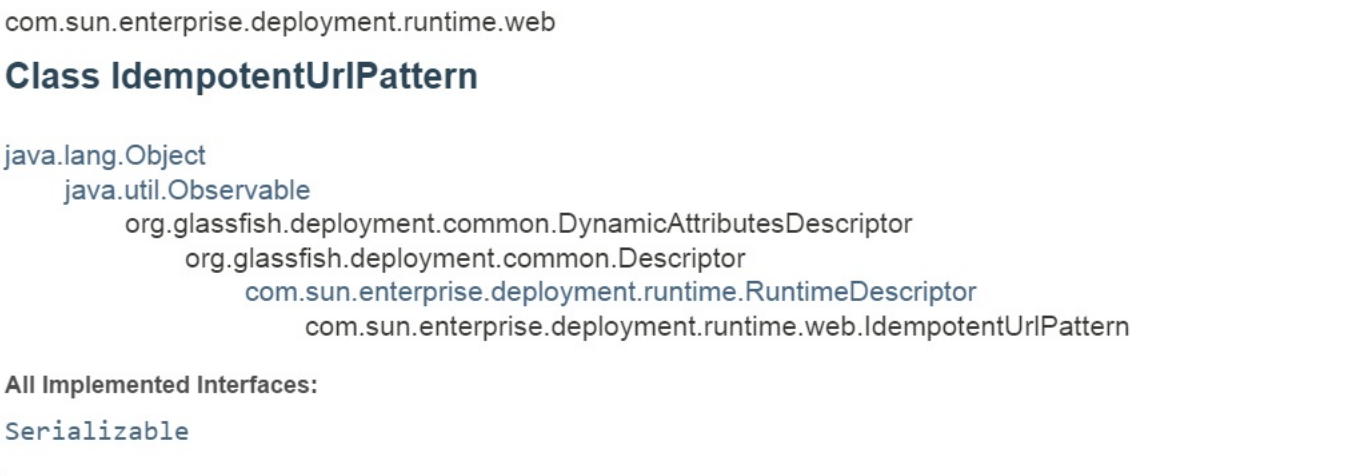
setParameterEncoding

```
public void setParameterEncoding(boolean value)
```

Figure 37: Missing javadoc for method "setParameterEncoding(...)"

- **Fig. 38**

1. The class "IdemPotentUrlPattern" used in line 378 to declare array patterns has no javadoc.


com.sun.enterprise.deployment.runtime.web

Class IdempotentUrlPattern

java.lang.Object
  java.util.Observable
    org.glassfish.deployment.common.DynamicAttributesDescriptor
      org.glassfish.deployment.common.Descriptor
        com.sun.enterprise.deployment.runtime.RuntimeDescriptor
          com.sun.enterprise.deployment.runtime.web.IdempotentUrlPattern

All Implemented Interfaces:

Serializable

---

public class IdempotentUrlPattern
extends RuntimeDescriptor

See Also:

Serialized Form

Figure 38: Missing javadoc for class "IdemPotentUrlPattern"

- **Fig. 39**

1. The class "ClassLoader" used in line 442 to declare the variable "classLoader" has no javadoc.

org.glassfish.web.deployment.runtime

Class ClassLoader

java.lang.Object
  java.util.Observable
    org.glassfish.deployment.common.DynamicAttributesDescriptor
      org.glassfish.deployment.common.Descriptor
        com.sun.enterprise.deployment.runtime.RuntimeDescriptor
          org.glassfish.web.deployment.runtime.WebPropertyContainer
            org.glassfish.web.deployment.runtime.ClassLoader

All Implemented Interfaces:

Serializable

```
public class ClassLoader
extends WebPropertyContainer
```

See Also:

Serialized Form

Figure 39: Missing javadoc for class "ClassLoader"

- **Fig. 40**

1. The interface "SunWebApp" used in line 466 has no javadoc.

Figure 40: Missing javadoc for interface "SunWebApp"

3.1.8 Package and Import Statements

24) "If any package statements are needed, they should be the first non-comment statements. Import statements follow."

No issues found for this point.

3.1.9 Class and Interface Declarations

25) "The class or interface declarations shall be in the following order: A. class/interface documentation comment B. class or interface statement C. class/interface implementation comment, if necessary D. class (static) variables a. first public class variables b. next protected class variables c. next package level (no access modifier) d. last private class variables E. instance variables a. first public instance variables e. next protected instance variables f. next package level (no access modifier) g. last private instance variables F. constructors G. methods"

No issues found for this point.

26) "Methods are grouped by functionality rather than by scope or accessibility."

- **Fig. 41**

1. To respect the grouping needed by the coding convention, the method "getSunDescriptor()" in line 202, should be above the method "registerBundle(...)".

```
171      */
172     public List<String> getSystemIDs() {
173         return null;
174     }
175 
176    /**
177     * register this node as a root node capable of loading entire DD files
178     *
179     * @param publicIDToDTD is a mapping between xml Public-ID to DTD
180     * @param versionUpgrades The list of upgrades from older versions
181     * to the latest schema
182     * @return the doctype tag name
183     */
184     public static String registerBundle(Map<String, String> publicIDToDTD,
185                                         Map<String, List<Class>> versionUpgrades) {
186        publicIDToDTD.put(DTDRegistry.SUN_WEBAPP_230_DTD_PUBLIC_ID, DTDRegistry.SUN_WEBAPP_230_DTD_SYSTEM_ID);
187        publicIDToDTD.put(DTDRegistry.SUN_WEBAPP_231_DTD_PUBLIC_ID, DTDRegistry.SUN_WEBAPP_231_DTD_SYSTEM_ID);
188        publicIDToDTD.put(DTDRegistry.SUN_WEBAPP_240_DTD_PUBLIC_ID, DTDRegistry.SUN_WEBAPP_240_DTD_SYSTEM_ID);
189        publicIDToDTD.put(DTDRegistry.SUN_WEBAPP_241_DTD_PUBLIC_ID, DTDRegistry.SUN_WEBAPP_241_DTD_SYSTEM_ID);
190        publicIDToDTD.put(DTDRegistry.SUN_WEBAPP_250_DTD_PUBLIC_ID, DTDRegistry.SUN_WEBAPP_250_DTD_SYSTEM_ID);
191        publicIDToDTD.put(DTDRegistry.SUN_WEBAPP_300_DTD_PUBLIC_ID, DTDRegistry.SUN_WEBAPP_300_DTD_SYSTEM_ID);
192        if (!restrictDTDDeclarations()) {
193           publicIDToDTD.put(DTDRegistry.SUN_WEBAPP_240beta_DTD_PUBLIC_ID, DTDRegistry.SUN_WEBAPP_240beta_DTD_SYSTEM_ID);
194        }
195 
196        return RuntimeTagNames.S1AS_WEB_RUNTIME_TAG;
197    }
198 
199    /**
200     * @return the descriptor instance to associate with this XMLNode
201     */
202     public Object getSunDescriptor() {
203         return descriptor.getSunDescriptor();
204     }
205 
```

Figure 41: Non grouped methods

27) "Check that the code is free of duplicates, long methods, big classes, breaking encapsulation, as well as if coupling and cohesion are adequate."

1. The method "writeDescriptor()" is way longer than the others in the same class.

3.1.10 Initialization and Declarations

28) "Check that variables and class members are of the correct type. Check that they have the right visibility (public/private/protected)"

No issues found for this point.

29) "Check that variables are declared in the proper scope"

No issues found for this point.

30) "Check that constructors are called when a new object is desired"

No issues found for this point.

31) "Check that all object references are initialized before use"

No issues found for this point.

32) "Variables are initialized where they are declared, unless dependent upon a computation"

No issues found for this point.

33) "Declarations appear at the beginning of blocks (A block is any code surrounded by curly braces "{" and "}"). The exception is a variable can be declared in a 'for' loop."

No issues found for this point.

3.1.11 Method Calls

34) "Check that parameters are presented in the correct order"

No issues found for this point.

35) "Check that the correct method is being called, or should it be a different method with a similar name"

No issues found for this point.

36) "Check that method returned values are used properly"

No issues found for this point.

3.1.12 Arrays

37) "Check that there are no off-by-one errors in array indexing (that is, all required array elements are correctly accessed through the index)"

No issues found for this point.

38) "Check that all array (or other collection) indexes have been prevented from going out-of-bounds"

No issues found for this point.

39) "Check that constructors are called when a new array item is desired"

No issues found for this point.

3.1.13 Object Comparison

40) "Check that all objects (including Strings) are compared with "equals" and not with "=="

No issues found for this point.

3.1.14 Output Format

41) "Check that displayed output is free of spelling and grammatical errors"

No issues found for this point.

42) "Check that error messages are comprehensive and provide guidance as to how to correct the problem"

No issues found for this point.

43) "Check that the output is formatted correctly in terms of line stepping and spacing"

No issues found for this point.

3.1.15 Computation, Comparisons and Assignments

44) "Check that the implementation avoids "brutish programming: (see http://users.csc.calpoly.edu/~jdalbey/SWE/CodeSmells/bonehead.html)"

No issues found for this point.

45) "Check order of computation/evaluation, operator precedence and parenthesizing"

No issues found for this point.

46) "Check the liberal use of parenthesis is used to avoid operator precedence problems."

- **Fig. 42**

1. Line 361 should have the expression "roleMappings.lenght>0" parenthesized to avoid possible issues.

```
357         // context-root?
358     appendTextChild(web, RuntimeTagNames.CONTEXT_ROOT, bundleDescriptor.getContextRoot());
359     // security-role-mapping
360     SecurityRoleMapping[] roleMappings = sunWebApp.getSecurityRoleMapping();
361     if (roleMappings!=null && roleMappings.length>0) {
362         SecurityRoleMappingNode srmn = new SecurityRoleMappingNode();
363         for (int i=0;i<roleMappings.length;i++) {
364         srmn.writeDescriptor(web, RuntimeTagNames.SECURITY_ROLE_MAPPING, roleMappings[i]);
365         }
366     }
```

Figure 42: Parenthesis needed in the highlighted line

- **Fig. 43**

1. Line 379 should have the expression "patterns.lenght>0" parenthesized to avoid possible issues.

```
377         // idempotent-url-pattern
378         IdempotentUrlPattern[] patterns = sunWebApp.getIdempotentUrlPatterns();
379         if (patterns != null && patterns.length > 0) {
380             IdempotentUrlPatternNode node = new IdempotentUrlPatternNode();
381             for (int i = 0;i < patterns.length; i++) {
382                 node.writeDescriptor(web, RuntimeTagNames.IDEMPOTENT_URL_PATTERN, patterns[i]);
383             }
384         }
```

Figure 43: Parenthesis needed in the highlighted line

47) "Check that all denominators of a division are prevented from being zero"

No issues found for this point.

48) "Check that integer arithmetic, especially division, are used appropriately to avoid causing unexpected truncation/rounding"

No issues found for this point.

49) "Check that the comparison and Boolean operators are correct"

No issues found for this point.

50) "Check throw-catch expressions, and check that the error condition is actually legitimate"

No issues found for this point.

51) "Check that the code is free of any implicit type conversions"

No issues found for this point.

3.1.16 Exceptions

52) "Check that the relevant exceptions are caught"

No issues found for this point.

53) "Check that the appropriate action are taken for each catch block"

No issues found for this point.

3.1.17 Flow of Control

54) "In a switch statement, check that all cases are addressed by break or return"

No issues found for this point.

55) "Check that all switch statements have a default branch"

No issues found for this point.

56) "Check that all loops are correctly formed, with the appropriate initialization, increment and termination expressions"

No issues found for this point.

3.1.18 Files

57) "Check that all files are properly declared and opened"

No issues found for this point.

58) "Check that all files are closed properly, even in the case of an error"

No issues found for this point.

59) "Check that EOF conditions are detected and handled correctly"

No issues found for this point.

60) "Check that all file exceptions are caught and dealt with accordingly"

No issues found for this point.

4 Any other problem you have highlighted

No additional issues outside the checklist have been found during the inspection.

5 Supporting information

5.1 Glossary

- **javadoc**: is a documentation generator created by Sun Microsystems for the Java language

5.2 References

- **Assignment 3**: template for the redaction of this document.
- **LaTeX documentation**: to support the redaction of this document.
- **LyX documentation**: to redact this document.

5.3 Tools used

- **LyX**: used to write this document.
- **DropBox**: used to share materials between members of the group.
- **GitHub**: used to share materials between memebers of the group and to deliver this document.
- **Paint**: used to build the pictures.
- **OneNote**: used to work on the code and to highlight lines.

5.4 Workload

The time spent to redact this document is approx:
Lorenzo Federico Porro : 24 hours
Annalisa Rotondaro : 24 hours