Code Inspection

Glassfish v4.1.1

Pozzi Matteo
Scandroglio Stefano

5-01-2016

Assigned Classes and Methods

For this assignement we had to analyze 3 different methods belonging to 3 different classes:

- **setSecurityConfig**(SecurityConfig config) which belongs to the WebModule class;

- **addMapping**(String…urlPatterns) which belongs to the WebServletRegistrationImpl class;

- **DynamicWebServletRegistrationImpl**(StandardWrapper wrapper , WebModule webModule) which is the constructor of the DynamicWebservletRegistrationImpl class.

All those 3 classes were located in the same source file called **WebModule.java** located in “appserver/web/web-glue/src/main/java/com/sun/enterprise/web”.

Functional Role of the Assigned Classes

Here we provide a short description of the functionalities we have identified for our set of assigned classes and methods:

- **WebModule:** this class is an implementation of the *WebModule* interface and extends the J2EE Management Module by adding additional product specific attributes and operations. The J2EE Management Module is a specification of the attributes, operations and architecture of the managed objects required by compliant J2EE platform implementations.

    - **setSecurityConfig:** we could not get a deep understanding of what the method does due to a lack of comments in the code but, what we’ve understood by looking at both the javadoc and the code is that the method receives as input a SecurityConfiguration object from which it extracts the parameters for the login configuration which are used to set the login authentication method and the login error page. Then security constraints are extracted from the SecurityConfiguration input object and are used to set the security parameters of certain web resources.

- **WebServletRegistrationImpl:** this class is an implementation of the ServletRegistration class which is used to configure the parameters of a servlet.

    - **addMapping:** this method is used to propagate the servlet mappings (controls on how a servlet is accessed) to the underlying webBundleDescriptor so that it can update, thanks to a security subsystem, the corresponding security constraints.

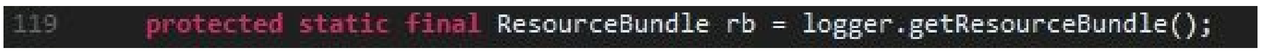
- **DynamicWebServletRegistrationImpl:** this class is an implementation of ServletRegistration.Dynamic class and extends DynamicServletRegistrationImpl class.
    - **DynamicWebServletRegistrationImpl:** this method is the constructor of the class and it initializes both the WebBundleDescriptor got from the WebModule and the WebComponentDescriptor got from the WebBundleDescriptor. If the WebComponentDescriptor is *null*, it will be created and added to the WebBundleDescriptor.

List of Issues

Here we provide a list of the various issues we've found when analyzing the code, using the checklist that was given to us.
For each method we will go through each section of the checklist and highlight the errors we've found inside the code (specifying the line at which such an error is found) with the corresponding checklist identifier.
When it will be possible we will also provide a screenshot of the snippet of code containg the aforementioned error.

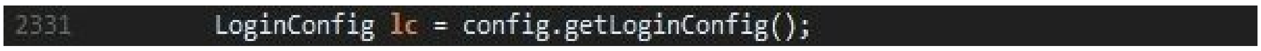
public void setSecurityConfig(SecurityConfig config){

- Naming Conventions
    - 1 :
        - Line 119 : *rb* should have meaningful name (ex. *resourceBundle*).



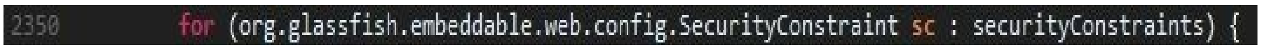
```
119     protected static final ResourceBundle rb = logger.getResourceBundle();
```

        - Line 2331 : *lc* is not a meaningful name (ex: *loginConfig*).



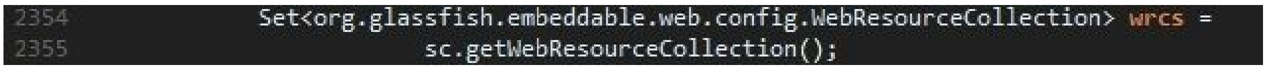
```
2331        LoginConfig lc = config.getLoginConfig();
```

- Line 2350 : *sc* should have meaningful name (ex: *securityConstraint*) even though it is a variable used as index of a for loop, it is widely used in the following portion of the method, so we thought that it would be better if it had a more significant name.



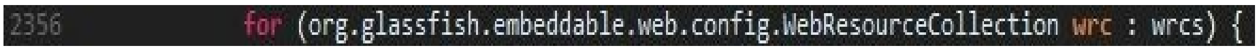
```
2350        for (org.glassfish.embeddable.web.config.SecurityConstraint sc : securityConstraints) {
```

- Line 2354 : *wrcs* is not a meaningful name (ex: *webResourceCollections*).



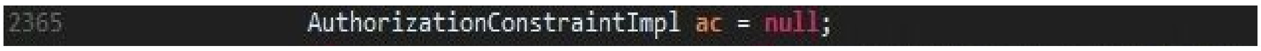
```
2354            Set<org.glassfish.embeddable.web.config.WebResourceCollection> wrcs =
2355                        sc.getWebResourceCollection();
```

- Line 2356 : *wrc* should have meaningful name (ex: *webResourceCollection*) even though it is a variable used as index of a for loop, it is widely used in the following portion of code, so we thought that it would be better if it had a more significant name.



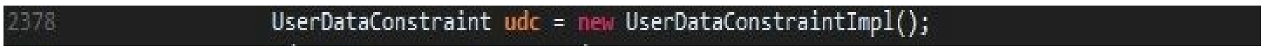
```
2356            for (org.glassfish.embeddable.web.config.WebResourceCollection wrc : wrcs) {
```

- Line 2365 : *ac* is not a meaningful name (ex: *authorizationConstraint*).



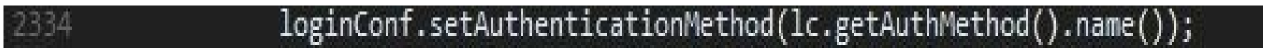
```
2365                AuthorizationConstraintImpl ac = null;
```

- Line 2378 : *udc* is not a meaningful name.



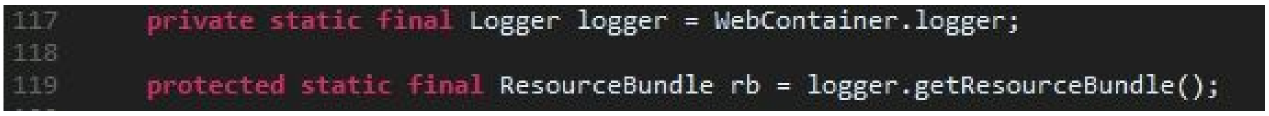
```
2378                UserDataConstraint udc = new UserDataConstraintImpl();
```

- 5 :
  - Line 2334 : method “name()” should be a verb (ex: “getName()”).



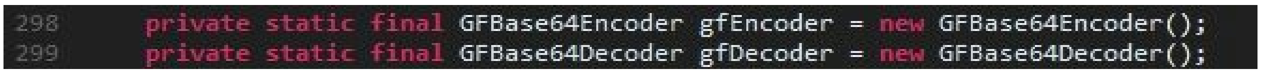
```
2334            loginConf.setAuthenticationMethod(lc.getAuthMethod().name());
```

- 7 :
  - Line 117 : *logger* is final but not in upper case.
  - Line 119 : *rb* is final but not in upper case.



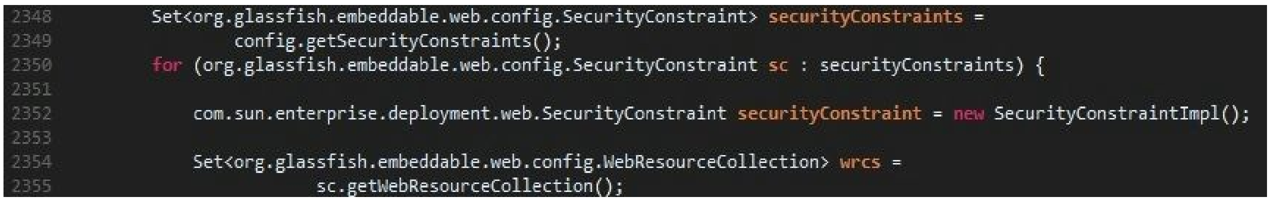
```
117     private static final Logger logger = WebContainer.logger;
118
119     protected static final ResourceBundle rb = logger.getResourceBundle();
```

  - Line 298 : *gfEncoder* is final but not in upper case.
  - Line 299 : *gfDecoded* is final but not in upper case.



```
298     private static final GFBase64Encoder gfEncoder = new GFBase64Encoder();
299     private static final GFBase64Decoder gfDecoder = new GFBase64Decoder();
```

- Indention
    - 8 :
        - Line 2355 : uses a 12 spaces indention instead of a 8 spaces indention as it’s done in line 2349.

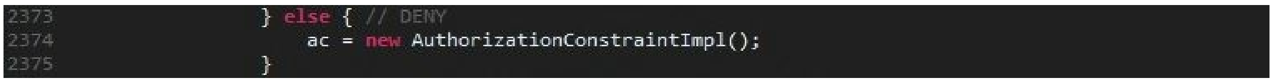


```
2348        Set<org.glassfish.embeddable.web.config.SecurityConstraint> securityConstraints =
2349                config.getSecurityConstraints();
2350        for (org.glassfish.embeddable.web.config.SecurityConstraint sc : securityConstraints) {
2351
2352            com.sun.enterprise.deployment.web.SecurityConstraint securityConstraint = new SecurityConstraintImpl();
2353
2354            Set<org.glassfish.embeddable.web.config.WebResourceCollection> wrcs =
2355                        sc.getWebResourceCollection();
```

- File Organization
    - 12 :
        - Lines 2325, 2330, 2336, 2342, 2347, 2351, 2353, 2357, 2364, 2377, 2384, 2390, 2396, 2401, 2414, 2424, 2426 : blank lines don’t separate sections (see image above as an example).

- Wrapping Lines
        - Lines 2354/2355 : line 2354 is broken after “=” which is useless because line already exceeded length of 80 and without breaking it, it will not exceed length of 120.

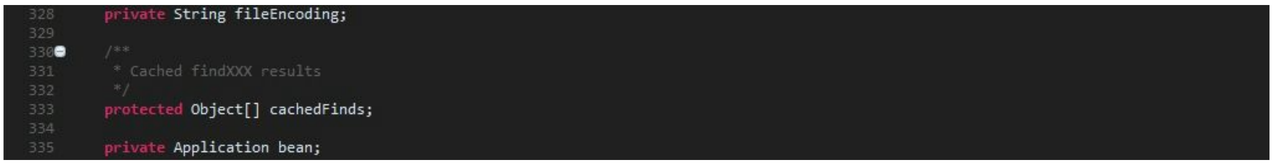
- Comments
    - 18 :
        - No comments throughout the method except for an unexplained *//DENY* (line 2373).



```
2373                } else { // DENY
2374                    ac = new AuthorizationConstraintImpl();
2375                }
```

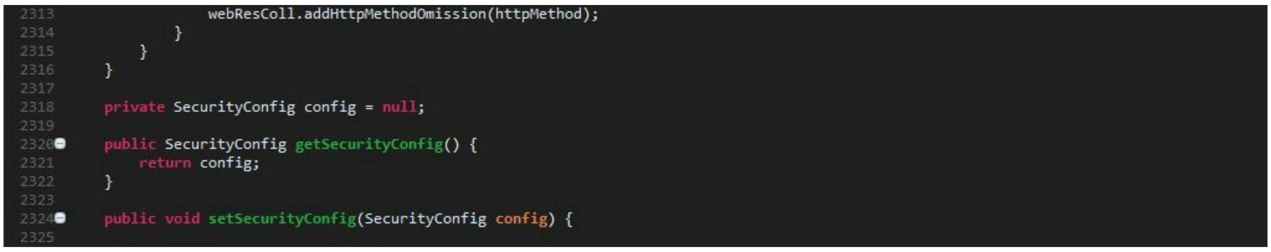
- Class and Interface Declaration
    - 25 :
        - In the class to which this method belongs constants are ordered as:
            - Private
            - Protected
            - Public
            - Private

- Line 333 : a protected attribute declared between to private.



```
328     private String fileEncoding;
329
330     /**
331      * Cached findXXX results
332      */
333     protected Object[] cachedFinds;
334
335     private Application bean;
```

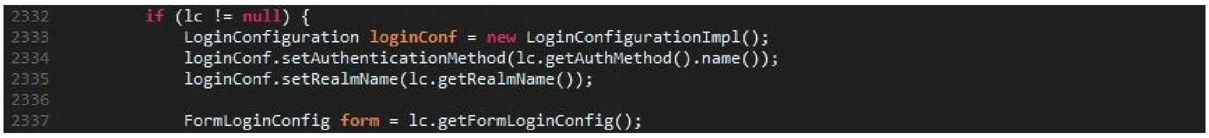
- Line 2318 : attribute declared between two methods.



```
2313                webResColl.addHttpMethodOmission(httpMethod);
2314            }
2315        }
2316    }
2317
2318    private SecurityConfig config = null;
2319
2320    public SecurityConfig getSecurityConfig() {
2321        return config;
2322    }
2323
2324    public void setSecurityConfig(SecurityConfig config) {
2325
```

- 26 :
  - This method, which is a setter, is defined right after a getter.

- <u>Initialization and Declarations</u>
  - 31,32 :
    - Line 2402 : *pipeline* is not declared nor initialized.
    - Line 2415 : variable *realm* is not declared nor initialized anywhere inside this method.
  - 33 :
    - Line 2333 : *loginConf* declared inside an if branch instead of outside it.
    - Line 2337 : *form* declared inside an if branch instead of outside it.

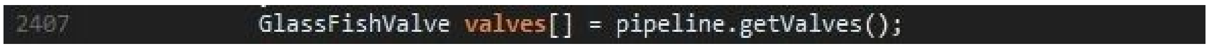


```
2332        if (lc != null) {
2333            LoginConfiguration loginConf = new LoginConfigurationImpl();
2334            loginConf.setAuthenticationMethod(lc.getAuthMethod().name());
2335            loginConf.setRealmName(lc.getRealmName());
2336
2337            FormLoginConfig form = lc.getFormLoginConfig();
```

    - Line 2343 : *decorator* declared inside an if branch instead of outside it.
    - Line 2365 : *ac* declared inside a for loop.
    - Line 2378 : *udc* declared inside a for loop.
    - Line 2403 : *basic* declared inside an if branch instead of outside it.

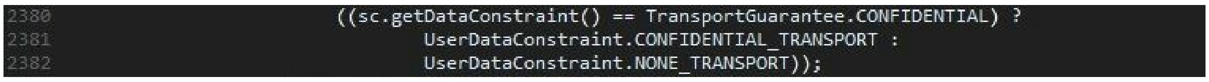
        - Line 2407 : *valves[]* declared inside an if branch instead of outside it.

- Arrays
    - 39 :
        - Line 2407 : doesn’t call a constructor to define the array *valves[]*.



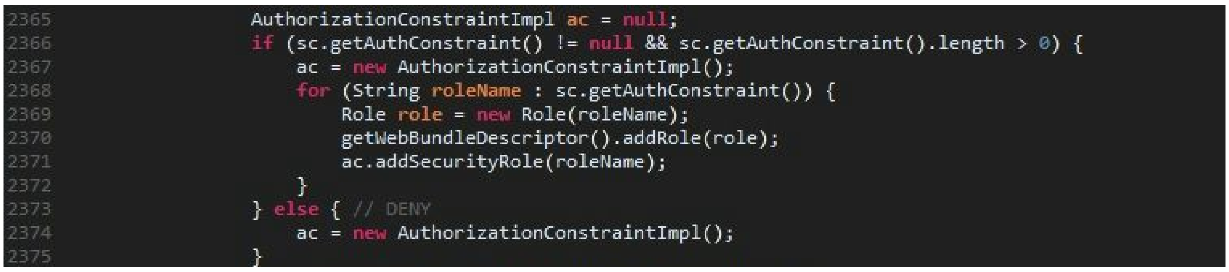
```
2407                GlassFishValve valves[] = pipeline.getValves();
```

- Object Comparison
    - 40 :
        - Line 2380 : two objects are compared with “==” instead of “equals()”.



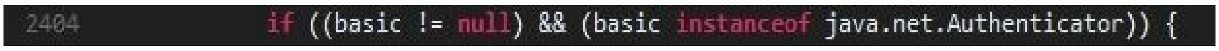
```
2380                ((sc.getDataConstraint() == TransportGuarantee.CONFIDENTIAL) ?
2381                        UserDataConstraint.CONFIDENTIAL_TRANSPORT :
2382                        UserDataConstraint.NONE_TRANSPORT));
```

- Computation, Comparisons and Assignments
    - 44:
        - Line 2373 : useless “else” branch: “ac” must be initialized in line 2365 and must be deleted the initializations in lines 2367 and 2374.



```
2365            AuthorizationConstraintImpl ac = null;
2366            if (sc.getAuthConstraint() != null && sc.getAuthConstraint().length > 0) {
2367                ac = new AuthorizationConstraintImpl();
2368                for (String roleName : sc.getAuthConstraint()) {
2369                    Role role = new Role(roleName);
2370                    getWebBundleDescriptor().addRole(role);
2371                    ac.addSecurityRole(roleName);
2372                }
2373            } else { // DENY
2374                ac = new AuthorizationConstraintImpl();
2375            }
```

    - 46 :
        - Line 2404 : useless parentheses.

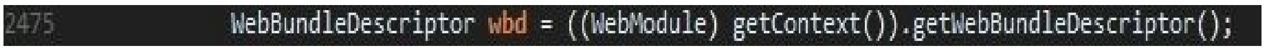


```
2404            if ((basic != null) && (basic instanceof java.net.Authenticator)) {
```

}

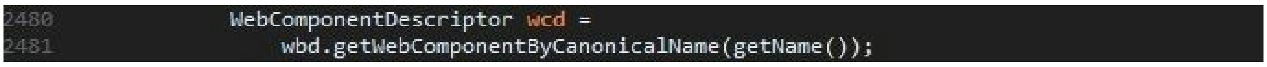
public Set<String> addMapping(String… urlPatterns){

- Naming Conventions
  - 1 :
    - Line 2475 : *wbd* is not a meaningful variable name. (ex: webBundleDesc).



```
2475            WebBundleDescriptor wbd = ((WebModule) getContext()).getWebBundleDescriptor();
```

    - Line 2480 : *wcd* is not a meaningful variable name (ex: webComponentDesc).



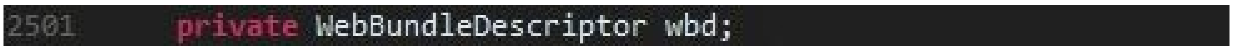
```
2480            WebComponentDescriptor wcd =
2481                wbd.getWebComponentByCanonicalName(getName());
```

- Java Source Files
  - 23 :
    - No documentation for this class is found on http://glassfish.pompel.me/, but only comments in the code.

}

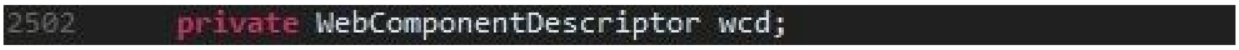
public DynamicWebServletRegistrationImpl(StandardWrapper wrapper, WebModule webModule){

- Naming conventions
  - 1 :
    - Line 2501 : *wbd* should have meaningful name (ex: *webBundleDescriptor*).



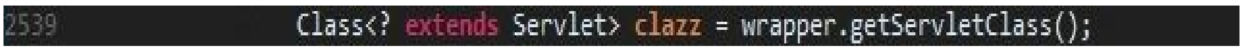
```
2501        private WebBundleDescriptor wbd;
```

    - Line 2502 : *wcd* should have meaningful name (ex: *webComponentDescriptor*).



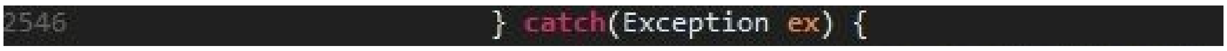
```
2502        private WebComponentDescriptor wcd;
```

    - Line 2539 : *clazz* should have meaningful name (ex: *class* or *servletClass*).



```
2539                Class<? extends Servlet> clazz = wrapper.getServletClass();
```

        - Line 2546 : *ex* should have meaningful name (ex: *exception*).



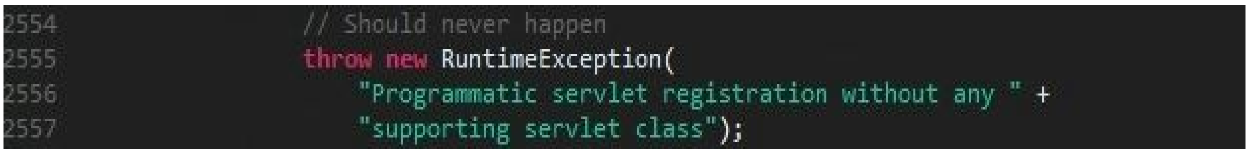
```
2546                        } catch(Exception ex) {
```

- <u>File organization</u>

    - 12 :
        - Line 2504 : blank line doesn't separate sections.

- <u>Comments</u>

    - 18 :
        - Line 2554 : useless comment, it doesn't explain what block of code do.



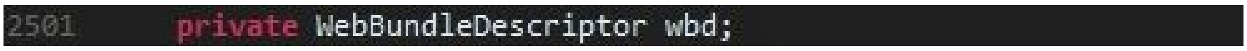
```
2554                // Should never happen
2555                throw new RuntimeException(
2556                    "Programmatic servlet registration without any " +
2557                    "supporting servlet class");
```

- <u>Class and Interface Declarations</u>

    - 26 :
        - Line 2687 : "loadServletClassName()" should be grouped with "servletClass" methods (ex: near "setServletClassName()" and "setServletClass()").

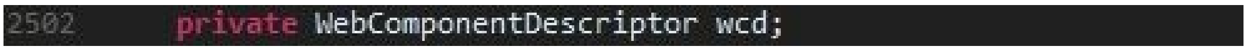
- <u>Initialization and Declarations</u>

    - 32 :
        - Line 2501 : *wbd* should be initialized when declared.



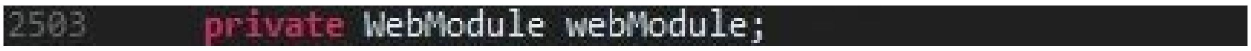
```
2501        private WebBundleDescriptor wbd;
```

        - Line 2502 : *wcd* should be initialized when declared.



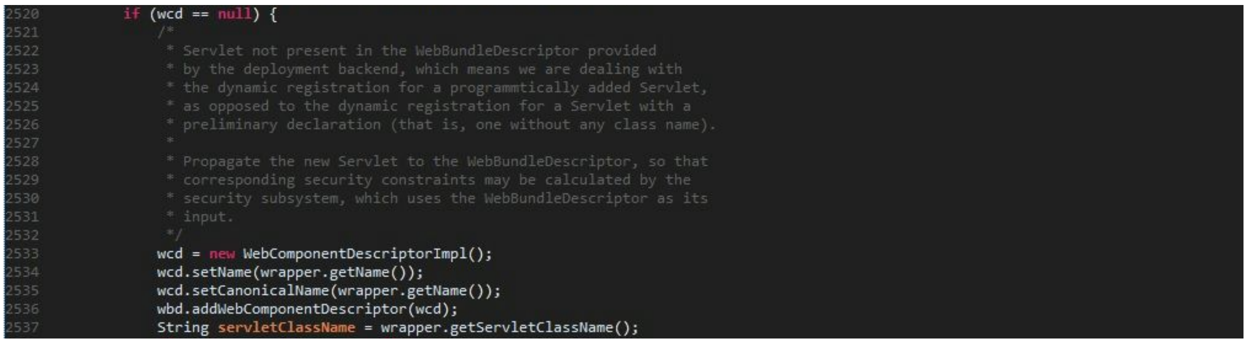
```
2502        private WebComponentDescriptor wcd;
```

        - Line 2503 : *webModule* should be initialized when declared.



```
2503        private WebModule webModule;
```

- 33 :
    - Line 2537 : servletClassName should be declared at the beginning of the block and then initialized inside the “if” block.



```
2520        if (wcd == null) {
2521            /*
2522             * Servlet not present in the WebBundleDescriptor provided
2523             * by the deployment backend, which means we are dealing with
2524             * the dynamic registration for a programmtically added Servlet,
2525             * as opposed to the dynamic registration for a Servlet with a
2526             * preliminary declaration (that is, one without any class name).
2527             *
2528             * Propagate the new Servlet to the WebBundleDescriptor, so that
2529             * corresponding security constraints may be calculated by the
2530             * security subsystem, which uses the WebBundleDescriptor as its
2531             * input.
2532             */
2533            wcd = new WebComponentDescriptorImpl();
2534            wcd.setName(wrapper.getName());
2535            wcd.setCanonicalName(wrapper.getName());
2536            wbd.addWebComponentDescriptor(wcd);
2537            String servletClassName = wrapper.getServletClassName();
```

}