Rule	Enabled	Reason
'ANYONE' Method Permission	No	not relevant
Abstract Specialization	Yes	Enabled
Accessor Method Naming Convention	No	SetXX, GetXX are not the preferred method names as per our coding guidelines
Accessor Usage in Defining Class	Yes	Enabled
Accidental Concatenation	Yes	Enabled
Action Field Revalidated	No	not relevant
Action Method Returns Unknown Value	No	not relevant
Action Revalidated	No	not relevant
Add Method to Interface	Yes	Enabled
Allow compareTo to Throw Exceptions	Yes	Enabled
Always Override toString	No	Do we need this ? If needed we can use lombok to handle this
Always Use Identifier Names	No	not relevant
Annotation Type Naming Convention	Yes	Enabled
Anonymous Authentication in LDAP	No	not relevant
Anonymous Class in Loop	Yes	Enabled
Anonymous Class Member Visibility	Yes	Enabled
Append String	Yes	Enabled
Applet Field Security	No	not relevant
Apply Dialog Font	No	not relevant
Array and non-Array Comparison	Yes	Enabled
Array Comparison	Yes	Enabled
Array Is Stored Without Copying	Yes	Enabled
Assignment In Condition	Yes	Enabled
Assignment to Non-final Static	Yes	Enabled
Attribute Injection	No	not relevant
Authentication By Hostname	No	not relevant
Avoid Accessing the FactoryBean Directly	No	not relevant
Avoid Auto-boxing	No	Should we enable this rule ? There are just too many instances where we are using autoboxing.
Avoid Auto-unboxing	No	Should we enable this rule ? There are just too many instances where we are using autoboxing.
Avoid Building Queries From User Input	No	not relevant
Avoid Class Loaders in EJB	No	not relevant
Avoid Comparing Classes By String Names	Yes	Enabled
Avoid Fields in Action Classes	No	not relevant
Avoid File IO in EJB	No	not relevant
Avoid Finalizers	Yes	Enabled
Avoid Future Keywords	Yes	Enabled
Avoid GUI in EJB	No	not relevant
Avoid Inner Classes	No	I think its fine to use Inner classes, Should we leave this diabled.
Avoid Instance Initializers	Yes	Enabled

Avoid Instantiation in Loops	Yes	Enabled
Avoid Instantiation to Get Class	Yes	Enabled
Avoid Loading Native Libraries	No	not relevant
Avoid Managing Threads	No	not relevant
Avoid Nested Assignments	No	There are number of places where we do things like " buf = readSizeBuf = new byte[4];" to make the code look concise.
Avoid Nested Blocks	Yes	Enabled
Avoid null Return Values	No	I think there are number of places where we return null.
Avoid Octal Literals	Yes	Enabled
Avoid Package Scope	No	This audit flags all inner classes, constructors, methods, and fields that have a package scope. Should we enable this ?
Avoid Passing this Reference as Argument	Yes	Enabled
Avoid Primitive Method Parameters	No	I think this is against our coding style, primitives are preferred over warpper classes.
Avoid Similar Names	No	Not enabled.
Avoid Sockets in EJB	No	not relevant
Avoid StringBuffer Instantiation With Character Literal	Yes	Enabled
Avoid Subtyping Cloneable	Yes	Enabled
Avoid Synchronization in EJB	No	not relevant
Avoid the no-argument String constructor	Yes	Enabled
Avoid Unsafe Array Declaration	No	This audit rule flags an array declared public, static and final. - disabled.
Avoid Using "Field Access" Strategy	No	not relevant
Avoid Using "instanceof"	No	rule conflicts with other rules.
Avoid Using Autowiring	No	not relevant
Avoid Using Enterprise Schemas Version	No	not relevant
Avoid Utility Methods	No	Should we enable this rule ?
Badly Located Array Declarators	Yes	Enabled
Base64-Encoded Password	No	not relevant
Bean Members Should Be Serializable	No	not relevant
Beware of URL equals() and hashCode()	Yes	Enabled
Blank Line Usage	No	Not aligned to our coding standards.
Blank Password	No	not relevant
Block Depth	Yes	Enabled
Boolean Method Naming Convention	No	Not aligned to our coding standards.
Brace Position	No	taken care by code formatter.
Break with Label	Yes	Enabled
Bundle Activation Policy Compatibility (3.3)	Yes	Enabled
Business Logic in ActionForm	No	not relevant
Call Lock Without Unlock	Yes	Enabled

Caught Exceptions	Yes	Enabled
Character Comparison	Yes	Enabled
Check Enabled Before Logging	Yes	Enabled
Check Enabled Before Logging	Yes	Enabled
Check Type In Equals	Yes	Enabled
Class Extends java.security.Policy	No	not relevant
Class getName() Usage	Yes	Enabled
Class Naming Convention	Yes	Enabled
Class Should Be Final	No	This audit rule looks for classes that do not have any subclasses but are not marked as final. - Should we enable this ?
Class Should Define Validate Method	No	not relevant
Class Should Have Final Static Fields	No	not relevant
Class Should Have Private Fields	No	not relevant
Class Should Validate All Fields	No	not relevant
Classes Should be Their Own Proxy	No	not relevant
Client Request Not Encrypted	No	not relevant
Client Request Not Signed	No	not relevant
Client Request Timestamp Not Signed	No	not relevant
Client Response Not Encrypted	No	not relevant
Client Response Not Signed	No	not relevant
Client Response Timestamp Not Signed	No	not relevant
Client Timestamp Does Not Expire	No	not relevant
Client Uses Username Token	No	not relevant
Clone Method Usage	Yes	Enabled
Clone Without Cloneable	Yes	Enabled
Cloneable Without Clone	Yes	Enabled
Close Connection Where Created	Yes	Enabled
Close Elements in Renderer	No	not relevant
Close In Finally	Yes	Enabled
Close Order	Yes	Enabled
Close Result Set Where Created	Yes	Enabled
Close Sessions Where Opened	No	not relevant
Close Statement Where Created	Yes	Enabled
Close Where Created	No	Creates lots flase positive - cases where resources need to be passed around or stored as instance variables we can't close these resources in the same block of code.
Code in Comments	No	Doesn't work correctly.
Code Injection	No	not relevant
Command Execution	Yes	Enabled
Command Injection	No	not relevant
Comment Local Variables	No	Should we enable this ?
Comparison Of Constants	Yes	Enabled
Comparison Of Incompatible Types	Yes	Enabled
Comparison Of Short And Char	Yes	Enabled

Complex Type Element Naming Convention	No	not relevant
Complex Type Naming Convention	No	not relevant
Concatenation In Appending Method	Yes	Enabled
Concurrent Modification	Yes	Enabled
Conditional Operator Use	No	Should we enable this ? We have used conditional operator in many places.
Configure Logging In File	Yes	
Consistent Suffix Usage	Yes	Enabled
Constant Condition	Yes	Enabled
Constant Conditional Expression	Yes	Enabled
Constant Field Naming Convention	Yes	Enabled
Constants in Comparison	No	should we enable this ?
Constructors Only Invoke Final Methods	Yes	Enabled
Container Should Not Contain Itself As Element	Yes	Enabled
Continue with Label	Yes	Enabled
Convert Class to Interface	Yes	Enabled
Create Global Forward	No	not relevant
Cross-Site Scripting	No	not relevant
Cyclomatic Complexity	Yes	Enabled
Dangling Else	Yes	Enabled
Database Connections Should Not Be Static	Yes	Enabled
Date and Time Usage	Yes	Enabled
Debugging Code	Yes	Enabled
Declare Accessors for All ActionForm Fields	No	not relevant
Declare Accessors for Persistent Fields	No	not relevant
Declare As Interface	Yes	Enabled
Declare Default Constructors	No	Should we enable this rule?
Declare Identifier Properties	No	not relevant
Declare Private Identifier Setter	No	not relevant
Declare Setters for Bean Fields	No	not relevant
Declare Type for java.util.Date Property	No	not relevant
Declared Exceptions	Yes	Enabled
Default Namespace	No	not relevant
Default Not Last in Switch	Yes	Enabled
Define Constants in Interfaces	No	Should we enable this ? We have exceptions to this rule in our code.
Define Initial Capacity	No	Should we enable this ? We have exceptions to this rule in our code.
Define Load Factor	No	Should we enable this ? We have exceptions to this rule in our code.
Delete Temporary Files	Yes	Enabled
Denial of Service Threat	Yes	Enabled
Deploy Mappings With Mapped Classes	No	not relevant
Deprecated Method Found	Yes	Enabled

Dereferencing Null Pointer	Yes	Enabled
DeSerializeability Security	No	Rule enforces All classes to implement a readObject method - Not needed in my opinion
Detect Multiple Iterations	Yes	Enabled
Disallow @Test Annotation	No	This rules states that @Test annotation should not be used. - Not sure why ?
Disallow Array Initializers	No	Arrays should not be statically initialized by an array initializer. Should we enable this rule ?
Disallow AST toString()	Yes	Enabled
Disallow Default Package	Yes	Enabled
Disallow Native Methods	Yes	Enabled
Disallow Notify Usage	Yes	Enabled
Disallow Sleep Inside While	Yes	Enabled
Disallow Sleep Usage	Yes	Enabled
Disallow Temporary Sessions	No	not relevant
Disallow ThreadGroup Usage	Yes	Enabled
Disallow Unnamed Thread Usage	Yes	Enabled
Disallow Use of AWT Peer Classes	Yes	Enabled
Disallow Use of Deprecated Thread Methods	Yes	Enabled
Disallow Yield Usage	Yes	Enabled
Disallowed File	Yes	Some files should not exist in a project. Should be enabled at a project level
Dispose Should Be Invoked	No	not relevant
Do not Access/Modify Security Configuration Objects	No	not relevant
Do not Catch IllegalMonitorStateException	Yes	Enabled
Do Not Create Finalizable Objects	No	Doesn’t allow to create objects like FileInputStream etc.
Do Not Declare Bindings	No	not relevant
Do Not Implement Outdated Interfaces	Yes	Enabled
Do Not Implement Serializable	No	This audit rule violates classes and interfaces that implement Serializable. Not releavant to us I think ?
Do Not Invoke setSize()	No	not relevant
Do Not Serialize Byte Arrays	No	This audit rule violates invocations of the method ObjectOutputStream.write(byte[]). - Not relevant to us I think.
Do Not Subclass ClassLoader	Yes	Enabled
Document Closing Braces	No	Should we enable this rule ? I think it creates clutter.
Don't Create Unused Error	No	not relevant
Don't Encode Markup in Renderer	No	not relevant
Don't Instantiate Beans	No	not relevant

Don't Return Mutable Types	No	Don't return mutable types from methods. - should we enable this ?
Don't use concatenation to convert to String	Yes	Should we enable this rule ? We are using string concatenation in many places ?
Don't Use Default Bean Names	No	not relevant
Don't use HTML Comments	No	not relevant
Double Check Locking	Yes	Enabled
Duplicate Import Declarations	Yes	Enabled
Duplicate Property Name	Yes	Enabled
Duplicate Property Value	No	There can be many properties that have same value - e.g. transactional=true and locking=true, this rules flags them as errors.
Duplicate Validation Form	No	not relevant
Dynamic Dependency in Ivy	No	not relevant
Dynamic Dependency in Maven	Yes	Enabled
Dynamically Compose Test Suites	Yes	Enabled
Efficient Expression	Yes	Enabled
Either Nillable Or MinOccurs	No	not relevant
Empty Catch Clause	Yes	Enabled
Empty Class	Yes	Enabled
Empty Do Statement	Yes	Enabled
Empty Enhanced For Statement	Yes	Enabled
Empty Finalize Method	Yes	Enabled
Empty Finally Clause	Yes	Enabled
Empty For Statement	Yes	Enabled
Empty If Statement	Yes	Enabled
Empty Initializer	Yes	Enabled
Empty Method	Yes	Enabled
Empty Statement	Yes	Enabled
Empty String Detection	Yes	Enabled
Empty Switch Statement	Yes	Enabled
Empty Synchronized Statement	Yes	Enabled
Empty Try Statement	Yes	Enabled
Empty While Statement	Yes	Enabled

Enforce Cloneable Usage	No	This audit rule requires almost All classes to override the clone method, Should we enable this ? this rule flags non-anonymous classes that: (1) do not implement Cloneable (so that the rule doesn't flag appropriate uses of Cloneable utilities), (2) are non-final (final classes can't be extended), (3) do not inherit a clone method (since inserting a clone method would be unnecessarily repetitive), (4) and do not override clone():
Enforce Singleton Property with Private Constructor	Yes	Enabled
Entity Bean's Remote Interface	No	not relevant
Entity Beans	No	not relevant
Entry Point Method	Yes	Enabled
Enumerated Type Naming Convention	Yes	Enabled
Enumeration Constant Naming Convention	Yes	Enabled
Environment Variable Access	No	Environment variables should not be accessed because not all platforms have support for environment variables. - We are using environment variable for setting project root path should we remove it ?
Equality Test with Boolean Literal	Yes	Enabled
Exception Creation	Yes	Enabled
Exception Declaration	No	This rule states that Exceptions should be declared to inherit from Exception, but not from either RuntimeException or RemoteException. Not aligned with our coding standards ?
Exception Parameter Naming Convention	Yes	Enabled
Explicit "this" Usage	Yes	Enabled
Explicit Invocation of Finalize	Yes	Enabled
Explicit Subclass of Object	Yes	Enabled
Expression Evaluation	Yes	Enabled
External Dependency in Ant	No	not relevant
External Dependency in Ivy	No	not relevant
External Dependency in Maven	No	Probably it doesn't work correctly, flags "http://maven.apache.org/POM/4.0.0" to be 'external',
Extra Semicolon	Yes	Enabled
Fail Invoked in Catch	Yes	Enabled
Favor Static Member Classes over Non-Static	Yes	Enabled

Field Access Protection	Yes	Enabled
Field Javadoc Conventions	Yes	Enabled
Field Might Have Null Value	No	This is taken care by JSR 305, Should we enable this rule ?
Field Only Used in Inner Class	Yes	Enabled
File Comment	No	File comments are not enabled currently.
File Length	Yes	2000 lines (all lines including comments.)
Filename Given Out	No	not relevant
Final Method Parameter In Interface	Yes	Enabled
Finalize Method Definition	Yes	Enabled
Finalize Should Not Be Public	Yes	Enabled
Float Comparison	Yes	Enabled
Floating Point Use	No	"Floating point values should rarely be used because of the potential for rounding errors." should we eanble this rule ?
Form Does Not Extend Validator Class	No	not relevant
Fully Parenthesize Expressions	No	Not needed I think, add unnecessary noise.
Getter and Setter Methods Should Be Final	No	Getter and setters are not as per our naming conventions, I think we can ingnore this ?
Handle Numeric Parsing Errors	Yes	Enabled
Hardcoded Password	Yes	Enabled
Hiding Inherited Fields	Yes	Enabled
Hiding Inherited Static Methods	Yes	Enabled
HTTP Response Splitting	No	not relevant
Illegal Main Method	Yes	Enabled
Implement a Zero-Argument Constructor	No	not relevant
Implement BeanNameAware Interface	No	not relevant
Implement Iterable	Yes	Enabled
Implicit Subclass of Object	No	it says ""Object" should be explicit specified as a superclass." Not needed I think ?
Import of Implicit Package	Yes	Enabled
Import Order	No	TBD
Import Style	Yes	Enabled
Improper calculation of array hashCode	Yes	Enabled
Improper conversion of Array to String	Yes	Enabled
Improper Use of Thread.interrupted()	Yes	Enabled
Inappropriate Language	Yes	Enabled
Include Implementation Version	No	MF files are auto generated, "Manifest-Version:" is present but this rule still flags error.
Incompatible Renderer Type	No	not relevant
Incompatible types stored in a collection	Yes	Enabled
Incomplete reset()	No	not relevant
Incomplete State Storing	Yes	Enabled
Incomplete Validation Method	No	not relevant
Inconsistent Conversion Using toArray()	Yes	Enabled
Inconsistent Use of Override	Yes	Enabled

Inconsistent Validator Attribute	No	not relevant
Incorrect Argument Type	Yes	Enabled
Incorrect Use of equals() and compareTo()	Yes	Enabled
Indent Code Within Blocks	No	taken care by code formatter.
Index Arrays with Ints	Yes	Enabled
Inefficient use of toArray()	Yes	Enabled
Initialize Static Fields	Yes	Enabled
Instance Field Naming Convention	Yes	Enabled
Instance Field Security	No	disallows non-final public instance fields. - should we enable this rule ? (We'll need to make all the public/protected non-final fields as private and provide accessor methods)
Instance Field Visibility	Yes	Enabled
Integer Division in a Floating-point Expression	Yes	Enabled
Interface Naming Convention	Yes	Enabled
Invalid Check For Binding Equality	No	not relevant
Invalid Check For Java Model Identity	No	not relevant
Invalid DBC Tag Value	No	not relevant
Invalid Form Bean Class	No	not relevant
Invalid Loop Construction	Yes	Enabled
Invalid Property Type Mapping	No	not relevant
Invalid Source for Database Connection	No	not relevant
Invalid Visitor Usage	Yes	Enabled
Invocation of Default Constructor	Yes	Enabled
Invoke super.finalize() from within finalize()	Yes	Enabled
Invoke super.release() Within release()	No	not relevant
Invoke super.setUp() from within setUp()	Yes	Enabled
Invoke super.tearDown() from within tearDown()	Yes	Enabled
Invoke super.validate() in validate()	No	not relevant
Invoke Synchronized Method In Loop	Yes	Enabled
JNDI Naming Standard	No	not relevant
JUnit Framework Checks	Yes	Enabled
Label Naming Convention	Yes	Enabled
Large Number of Constructors	Yes	Enabled
Large Number of Fields	Yes	Max 10
Large Number of Methods	Yes	Max 15
Large Number of Parameters	Yes	Max 7
Large Number of Switch Statement Cases	Yes	Enabled
Lazily Initialize Singletons	Yes	Enabled
Line Length	No	taken care by code formatter.

Local Declarations	Yes	Variable declaration first in block - disabled. Variable declaration first in method - disabled Explicitly initialze each local variable - disabled Declare each local variable in separate statement. - enabled
Local Variable Naming Convention	Yes	Enabled
Log Exceptions	Yes	also taken care by Log4e
Log Forging	Yes	Enabled
Log Level	Yes	Enabled
Loop Variable Naming Convention	Yes	Enabled
Loss of Precision in Cast	No	This rule doesn't allow to down cast from int to byte etc, there are many places in our code where we need to do this.
Manipulation with XPath	No	not relevant
Message Beans	No	not relevant
Message Document Must Have Version Attribute	No	not relevant
Method Chain Length	Yes	Enabled
Method Could Be Final	No	it says "Methods that are not supposed to be overridden should be declared final." - should we enabel this rule ?
Method Invocation in Loop Condition	No	It allows only some methods to be invoked inside loop conditions, there are number of exception to this rule in our code, should we enable this rule ?
Method Javadoc Conventions	Yes	Enabled
Method Naming Convention	Yes	Enabled
Method Parameter Naming Convention	Yes	Enabled
Method Should Be Private	No	It says "Method should be private if it is not member of an interface and it doesn't override method of superclass and isn't overriden by subtype. " should we enable this rule ?
Method Should Be Static	No	"Methods that do not access any instance state or instance methods should be static." should we enable this rule ?
Mime Encoding Method Usage	No	not relevant
Minimize Scope of Local Variables	No	This rule recommends to convert the while-loop to a for-loop, for cases like 'while (beItr.hasNext())' - I think while-loop is looks cleaner in these cases.
Mismatched Notify	Yes	Enabled
Mismatched Wait	Yes	Enabled
Missing Application Context File	No	not relevant

Missing Assert in JUnit Test Method	Yes	Enabled
Missing Bean Description	No	not relevant
Missing Block	No	not consistent with our coding guidelines. I think ?
Missing Catch of Exception	No	not relevant
Missing Class in web.xml	No	not relevant
Missing Constants In Switch	Yes	Enabled
Missing Default in Switch	Yes	Enabled
Missing Error Page	No	not relevant
Missing Image File	No	not relevant
Missing Message	No	not relevant
Missing Message in Assert	Yes	Enabled
Missing Namespace Grouping Identifiers	No	not relevant
Missing Namespace Version	No	not relevant
Missing Or Misplaced Manifest Version	Yes	Enabled
Missing Or Misplaced Section Name	Yes	Enabled
Missing reset() Method	No	not relevant
Missing static method in non-instantiable class	Yes	Enabled
Missing Update in For Statement	Yes	Enabled
Missing Validator Name	No	not relevant
Misspelled Method Name	Yes	Enabled
Modifier Order	Yes	Enabled
More Than One Logger	Yes	Enabled
Multiple Return Statements	No	"Methods should have a single return statement." Should we enable this rule ?
Multiplication Or Division By Powers of 2	No	"Do not multiply or divide by powers of 2." The shift operator is faster and more efficient. Should we enable this rule ?
Mutability Of Arrays	No	"Do not return internal arrays from non-private methods." I don't think this is consistent with our thinking
Mutable Constant Field	No	"Disallows public static final mutable type fields." should we enable this rule ?
Nested Synchronized Calls	Yes	Enabled
Never Use the Identifier in equals() or hashCode()	No	not relevant
Next method invoked without hasNext method	Yes	Enabled
No Abstract Methods	Yes	Enabled
No Action For Validation	No	not relevant
No DB Driver Loading	No	not relevant
No Explicit Exit	Yes	Enabled
No Explicit This Use in EJB's	Yes	Enabled
No Instance Fields In Portlets	No	not relevant
No Instance Fields In Servlets	No	not relevant

No Public Members	Yes	Some classes can have only protected members e.g. classes that implement an abstract class. - This rule is enabled but some classes are excluded.
No Run Method	Yes	Enabled
No Set-up in Constructors	Yes	
No Such Field For Validation	No	not relevant
No Timestamp In Client Request	No	not relevant
No Timestamp In Client Response	No	not relevant
No Timestamp In Server Request	No	not relevant
No Timestamp In Server Response	No	not relevant
No Validation Message	No	not relevant
Non Static Logger	Yes	Enabled

Non-atomic File Operations	No	This rule flags things like (which is valid.)
		if (databaseDir.exists())
		throw new DBException("Database already exists");

Non-blank Final Instance Field	Yes	Enabled
Non-case Label in Switch	Yes	Enabled
Non-conforming Backing Bean Methods	No	not relevant
Non-private Constructor in Static Type	Yes	Enabled
Non-protected Constructor in Abstract Type	Yes	Enabled
Non-serializable Class Declares readObject or writeObject	Yes	Enabled
Non-serializable Class Declares serialVersionUID	Yes	Enabled
Non-terminated Case Clause	Yes	Enabled
Nonce Not Used	No	not relevant
Notify method invoked while two locks are held	Yes	Enabled
Null Pointer Dereference	No	Doesn't seem to work as expected, JSR 305 works better.
Numeric Literals	No	""Numeric literals should not appear in code." - should we enable this rule
Obey General Contract of Equals	No	"The equals method should compare the identity of the receiver and the argument, returning true if they are the same." 'identity' ? not sure what that means
Obsolete Modifier Usage	Yes	Enabled
One Class per Mapping File	No	not relevant
One Statement per Line	No	taken care by code formatter.
Overloaded Equals	Yes	Enabled

Overloaded Methods	No	"Overloading method names can cause confusion and errors." This audit rule finds methods that are overloaded. Overloaded methods are methods that have the same name and the same number of parameters, but do not have the same types of parameters.- Not consistent with our coding
Override both equals() and hashCode()	Yes	Enabled
Override Clone Judiciously	Yes	Enabled
Overriding a Non-abstract Method with an Abstract Method	Yes	Enabled
Overriding a Synchronized Method with a Non-synchronized Method	Yes	Enabled
Overriding Private Method	Yes	Enabled
Package Javadoc	No	TBD
Package Naming Convention	Yes	Enabled
Package Prefix Naming Convention	Yes	Enabled
Package Structure	Yes	All package names should start with 'com.paintedboxes'
Parenthesize Condition in Conditional Operator	Yes	Enabled
Parse Method Usage	No	"The parseXXX() methods for Numerics should not be used in an internationalized environment." should we enable this ?
Password in File	No	"Storing passwords in a configuration file is a security risk." should we enable this ?
Path Manipulation	Yes	Enabled
Plain Text Password	Yes	Enabled
Platform Specific Line Separator	No	Doesn't seem to work as expected, reports error for things like '\t' , '\n'
Pluralize Collection Names	No	should we enable this ?
Possible Null Pointer	No	Taken care by JSR 305 annotations, which is a better way achieving this.
Potential Infinite Loop	Yes	Enabled
Pre-compute Constant Calculations	Yes	Enabled
Prefer Interfaces to Abstract Classes	No	This rule flags all the abstract classes.
Prefer Interfaces To Reflection	Yes	Enabled
Preferred Expression	Yes	Enabled
Prevent Overwriting Of Externalizable Objects	Yes	Enabled
Process Control	Yes	Enabled
Proper Finalize Usage	Yes	Enabled
Proper Servlet Usage	No	not relevant
Property File Must Exist	No	not relevant
Protected Member in Final Class	Yes	Enabled
Public Constructor in Non-public Type	Yes	Enabled

Public Nested Type in Package Visible Type	Yes	Enabled
Questionable Assignment	Yes	Enabled
Questionable Name	Yes	Enabled
Realm Debug Enabled	No	not relevant
Recursive Call With No Check	Yes	Enabled
Redirected With Password	No	not relevant
Redundant Assignment	Yes	Enabled
Referenced Class Not Defined	No	not relevant
Referenced Class Not Defined	No	not relevant
Reflection Injection	Yes	Enabled
Reflection Method Usage	No	"Don't use Class getMethod(), getField(), getDeclaredMethod() or getDeclaredField() methods in production code." should we enable this ?
Relative Access to Enterprise Schemas	No	not relevant
Relative Library Path	No	not relevant
Repeated Assignment	Yes	Enabled
Replace Synchronized Classes	Yes	Enabled
Request Message Naming Convention	No	not relevant
Request Parameters In Session	No	not relevant
Resource Injection	No	not relevant
Resource URL Manipulation	No	not relevant
Response Message Naming Convention	No	not relevant
Restricted Packages	Yes	Enabled
Restricted Superclasses	Yes	Enabled
ReThrown Exceptions	Yes	Enabled
Return Boolean Expression Value	Yes	Enabled
Return Constant From getComponentType	No	not relevant
Return Constant From getFamily	No	not relevant
Return Constant From getRendererType	No	not relevant
Return in Finally	Yes	Enabled
Reusable Immutables	Yes	Enabled
Reuse DataSources for JDBC Connections	No	not relevant
Rollback Transaction on Exception	No	not relevant
Runtime Method Usage	Yes	Enabled
Same Validator Name	No	not relevant
Serializable Usage	Yes	Enabled
Serializeability Security	Yes	Enabled
Server Request Not Encrypted	No	not relevant
Server Request Not Signed	No	not relevant
Server Request Timestamp Not Signed	No	not relevant
Server Response Not Encrypted	No	not relevant
Server Response Not Signed	No	not relevant
Server Response Timestamp Not Signed	No	not relevant
Server Timestamp Does Not Expire	No	not relevant
Server Uses Username Token	No	not relevant

Service Naming Convention	No	not relevant
Service Operation Naming Convention	No	not relevant
Service Provider Request Security Not Configured	No	not relevant
Service Provider Response Security Not Configured	No	not relevant
Service Requester Request Security Not Configured	No	not relevant
Service Requester Response Security Not Configured	No	not relevant
Session Beans	No	not relevant
Session Must Be Synchronized	No	not relevant
Session-scope Form Needs reset()	No	not relevant
Simple Type Element Naming Convention	No	not relevant
Simple Type Naming Convention	No	not relevant
Sleep method invoked in synchronized code	Yes	Enabled
Sockets in Servlets	No	not relevant
Source Length	Yes	Enabled
Space After Casts	No	taken care by code formatter.
Space After Commas	No	taken care by code formatter.
Space Around Operators	No	taken care by code formatter.
Space Around Periods	No	taken care by code formatter.
Specify an Error Page	No	not relevant
Specify Schema Version	No	not relevant
Specify SOAP Actions For WSDL Operations	No	not relevant
Spell Check Comments	Yes	Enabled
Spell Check Identifiers	Yes	Enabled
Spell Check Property Comments	Yes	Enabled
Spell Check Property Names	No	not relevant
Spell Check Property Values	No	not relevant
Spell Check String Literals	Yes	Enabled
Spell Check XML Attribute Names	Yes	Enabled
Spell Check XML Attribute Values	Yes	Enabled
Spell Check XML Body Text	Yes	Enabled
Spell Check XML Comments	Yes	Enabled
Spell Check XML Tag Names	Yes	Enabled
SQL Injection	Yes	Enabled
Start Method Invoked In Constructor	Yes	Enabled
Statement Creation	No	not relevant
Static Field Naming Convention	Yes	Enabled
Static Field Security	No	TBD
Static Instantiation	Yes	Enabled
Static Member Access	Yes	
String Comparison	Yes	Enabled

String Concatenation	No	should we enble this rule which doesn't allow String concatenation anywhere in the code?
String Concatenation in Loop	Yes	Enabled
String Created from Literal	Yes	Enabled
String indexOf Use	Yes	Enabled
String Literals	Yes	Enabled
String Method Usage	No	Disallows usage of String.equals(), String.equalsIgnoreCase() and String.comparesTo()
StringTokenizer Usage	No	Disallows usage of StringTokenizer
Struts Validator Not in Use	No	not relevant
Subclass should override method	yes	Enabled
Synchronization On getClass Method	Yes	Enabled
Synchronization On Non Final Fields	Yes	Enabled
Synchronized In Loop	Yes	Enabled
Synchronized Method	No	"Methods should never be marked as synchronized."
Tag Handler Field Not Cleared	No	not relevant
Tag Handler Should Implement Release	No	not relevant
Tainted Filter	No	not relevant
Tainted Internet Address	No	not relevant
Temporary Object Creation	Yes	Enabled
Test Case Naming Convention	No	TBD
Throw in Finally	Yes	Enabled
Thrown Exceptions	Yes	Enabled
toString() Method Usage	Yes	Enabled
Transient Field in Non-Serializable Class	Yes	Enabled
Type Declarations	Yes	Enabled
Type Depth	Yes	Enabled
Type Javadoc Conventions	Yes	Enabled
Type Member Ordering	No	Should we enable this ? Or we can have eclipse 'sort members' action. ?
Type Names Must Be Singular	No	Flags class names like Columns, Bytes.
Type Parameter Naming Convention	Yes	Enabled
Unassigned Field	No	reports too many false positives
Undefined Message Key	No	not relevant
Undefined Placeholder	No	not relevant
Undefined Property	Yes	Enabled
Unhashable class in hashed collection	Yes	Enabled
Unique Namespace per Schema	No	not relevant
Unknown Cast	Yes	Enabled
Unknown Component Type	No	not relevant
Unknown Renderer Type	No	not relevant
Unnecessary "instanceof" Test	Yes	Enabled
Unnecessary Catch Block	Yes	Enabled
Unnecessary Default Constructor	Yes	Enabled

Unnecessary Exceptions	No	If a method throws an exception which is a subclass of the exception mentioned in the throws clause, this rule flags such cases.
Unnecessary Final Method	Yes	Enabled
Unnecessary Import Declarations	Yes	Enabled
Unnecessary Null Check	Yes	Enabled
Unnecessary Override	Yes	Enabled
Unnecessary Return	Yes	Enabled
Unnecessary Return Statement Parentheses	No	Conflicts with other rules.
Unnecessary toString() Method Invocation	Yes	Enabled
Unnecessary Type Cast	Yes	Enabled
Unregistered Validator	No	not relevant
Unsupported Clone	Yes	Enabled
Unused Assignment	No	Does't work correctly I think - I enabled this and got too many false violitions
Unused Field	Yes	Enabled
Unused Label	Yes	Enabled
Unused Method	Yes	Enabled
Unused Return Value	No	There are many cases where we don't want to use return value e.g. map.put() etc. should we enable this rule ?
Unused StringBuffer	yes	Enabled
Unused StringBuilder	Yes	Enabled
Unused Validation Form	No	not relevant
Unvalidated Action Field	No	not relevant
Unvalidated Action Form	No	not relevant
Unvalidated Action Form Field	No	not relevant
Usage Of Binary Comparison	Yes	Enabled
Usage Of Static Calendar	Yes	Enabled
Usage Of Static Date Format	Yes	Enabled
Use "for" Loops Instead of "while" Loops	No	rule checks for the use of "while" loops rather than "for" loops, I think while loops are cleaner in some cases
Use "Nullable" Type for Identifier Properties	No	not relevant
Use == to Compare With null	Yes	Enabled
Use @After Annotation Rather than tearDown()	Yes	Enabled
Use @Before Annotation Rather than setUp()	Yes	Enabled
Use @RunWith and @SuiteClasses to build test suite	Yes	Enabled
Use @Test Annotation for JUnit test	Yes	Enabled
Use a Valid Schema Namespace	No	not relevant
Use a Valid WSDL Namespace	No	not relevant
Use Action Interface	No	not relevant

Use Additional Attribute in <constructor-arg> Tag	No	not relevant
Use ApplicationContext to Assemble Beans	No	not relevant
Use arraycopy() Rather Than a Loop	Yes	Enabled
Use Available Constants	Yes	Enabled
Use Buffered IO	Yes	Enabled
Use Camel Case in Namespaces	No	not relevant
Use char Rather Than String	Yes	Enabled
Use charAt() Rather Than startsWith()	Yes	Enabled
Use Compound Assignment	Yes	Enabled
Use deep Arrays methods when necessary	Yes	Enabled
Use Domain-specific Terminology	No	not relevant
Use equals() Rather Than ==	Yes	Enabled
Use equals() Rather Than equalsIgnoreCase()	No	We need equalsIgnoreCase() in some cases.
Use Existing Global Forwards	Yes	Enabled
Use ForwardAction Where Possible	No	not relevant
Use html:messages Instead of html:errors	No	not relevant
Use idref Element	No	not relevant
Use Interceptor to Track Duplicate Submits	No	not relevant
Use Interfaces for Collection Attributes	No	not relevant
Use Interfaces Only to Define Types	No	not relevant
Use locale-specific methods	No	not relevant
Use NoSuchElementException in next()	No	not relevant
Use of "instanceof" with "this"	Yes	Enabled
Use Of Broken Or Risky Cryptographic Algorithm	No	not relevant
Use of instanceof in Catch Block	Yes	Enabled
Use of Random	No	Random is used only inside unit testing code.
Use of xs:any	No	not relevant
Use Only Non-Final Persistent Classes	No	not relevant
Use Only Static Inner Classes	No	not relevant
Use Or to Combine SWT Style Bits	No	not relevant
Use Privileged Code Sparingly	No	not relevant
Use Qualified Attributes	No	not relevant
Use Session-per-request Pattern	No	not relevant
Use Setter Injection	No	not relevant
Use Start Rather Than Run	Yes	Enabled
Use StringBuffer length()	Yes	Enabled
Use StrutsTestCase for Unit Testing	No	not relevant
Use SuccessAction Where Possible	No	not relevant
Use Thread-safe Lazy Initialization	Yes	Enabled
Use Type for Constructor Argument Matching	No	not relevant

Use Type-Specific Names	No	Mandates InputStream objects to have name like "in"
Use Valid SWT Styles	No	not relevant
Use Valid Type for Form-property	No	not relevant
Use valueOf() to wrap primitives	Yes	Enabled
Use WSDL First approach	No	not relevant
Use XmlBeanFactory	No	not relevant
Validate XML	Yes	Enabled
Validation Method Naming Convention	No	not relevant
Validation Not Enabled	No	not relevant
Validator Configuration File Does Not Exist	No	not relevant
Validator Disabled	No	not relevant
Variable Declared Within a Loop	No	TBD
Variable Has Null Value	No	Taken care by JSR 305 annotations, which is a better way achieving this.
Variable Should Be Final	Yes	Enabled
Variable Usage	Yes	Enabled
Wait Inside While	Yes	Enabled
Wait method invoked while two locks are held	Yes	Enabled
wait() Invoked Instead of await()	Yes	Enabled
Weblogic Session ID Length	No	not relevant
White Space Before Property Name	Yes	Enabled
White Space Usage	No	taken care by code formatter.
Wrong Family Returned	No	not relevant
Wrong Integer Type Suffix	Yes	Enabled
WSDL Must Specify ESB Endpoints	No	not relevant
WSDL Namespace for Included Schemas	No	not relevant
XSD File Naming Convention	No	not relevant