

GreenMail

Open Source suite of lightweight and sand boxed email servers supporting SMTP, POP3 and IMAP.

Currently [v1.5.11](#)

About GreenMail

Introduction

GreenMail is an open source, intuitive and easy-to-use test suite of email servers for testing purposes.

Typical use cases include mail integration testing or a lightweight sand boxed mail server for development.

- Supports SMTP, POP3 and IMAP including SSL
- Prevents accidental email leaking to real mail servers
- Provides different deployment models, such as a simple [standalone JVM process](#), as a [WAR module](#), as a [JBoss GreenMail Service](#) or [docker image](#)
- Easily embeddable in JUnit tests for integration testing
- Lightweight with few dependencies

GreenMail is the first and only library that offers a test framework for both receiving and retrieving emails from Java.

Scenarios

GreenMail is useful in the following scenarios:

Test Your Sending Code

- System testing an application. GreenMail responds like a regular SMTP server but does not deliver any email, which enables it to be used in real life applications and real test cases. Messages can easily be extracted, verified and modified. Support for SMTPS (SSL) is enabled.
- GreenMail is an excellent choice for unit testing code that needs to send email with a succinct, efficient (non-polling) way to wait for messages to arrive, retrieve, verify, and modify messages.
- Note that retrieval of emails can be made with a simple java method or through a provided POP3, IMAP retriever helper class. Alternatively you can use a local client like [Thunderbird](#).
- The example below is using the GreenMail JUnit rule. See [examples](#) for hints on how to use GreenMail without the rule.

```
@Rule
public final GreenMailRule greenMail = new GreenMailRule(ServerSetupTest.SMTP);

@Test
public void testSend() throws MessagingException {
    GreenMailUtil.sendTextEmailTest("to@localhost.com", "from@localhost.com",
        "some subject", "some body"); // --- Place your sending code here instead
    assertEquals("some body", GreenMailUtil.getBody(greenMail.getReceivedMessages()[0]));
}
```

Test Your Retrieving Code

- Again GreenMail can be used for system or unit testing an application needing to use POP3 or IMAP by responding like a standard compliant POP3 or IMAP server. Support for POP3S and IMAPS (SSL) is also enabled.
- Messages can be placed directly in users mailboxes or by using SMTP.
- GreenMail ships with helper classes for sending and retrieving. See the javadocs for the [Retriever.java](#) class

```

@Rule
public final GreenMailRule greenMail = new GreenMailRule(ServerSetupTest.SMTP_IMAP);

@Test
public void testReceive() throws MessagingException {
    GreenMailUser user = greenMail.setUser("to@localhost.com", "login-id", "password");
    user.deliver(createMimeMessage()); // You can either create a more complex message...
    GreenMailUtil.sendTextEmailTest("to@localhost.com", "from@localhost.com",
        "subject", "body"); // ...or use the default messages

    assertEquals(2, greenMail.getReceivedMessages().length); // // --- Place your POP3 or IMAP retrieve code here
}

```

Sending and Retrieving

GreenMail can easily be configured to use all or a combination of ports, protocols, and bind addresses. For example it's possible to run GreenMail on SMTP, POP3, POP3S and IMAPS ports as easily as only SMTP. Many systems might already be running these servers or don't allow non root users to open the default ports which is why GreenMail ships with a special configuration for testing.

Mocking a mail server for your development environment

GreenMail provides a JBoss GreenMail service for mocking a mail server for development. It saves you the overhead of either installing a full productive server (like [Apache James](#)).

Check out the possible deployments as [webapp](#), [JBoss Service](#), [standalone](#) or [standalone docker image](#).

Implementation

The implementation is in 100% Java with only a few library dependencies:

- javamail.jar - [JavaMail API](#)
- activation.jar - [Java Activation Framework](#), required by JavaMail (provided by JDK since 1.6+)
- slf4j-api.jar - required for logging via [Simple Logging Facade for Java](#)
- junit.jar - required for easy test setup using Rules

Source Code

You find the source on [GitHub](#) including build instructions. GreenMail is open source released under [The Apache 2.0 License](#).

GreenMail's protocol source code is based on a cocktail of [Foedus](#) and [James](#).

Links

The following lists provide helpful links related to GreenMail and email integration testing.

Projects extending/wrapping GreenMail:

- [Grails Plugin for GreenMail](#) including REST-like GreenMail control interface and UI
- [Play Plugin for GreenMail](#) including REST-like control interface and UI
- [JCA Adapter for GreenMail](#)
- Deprecated [SourceForge GreenMail](#) project location before migration to GitHub in 2014

Articles&Blogs:

- [Integration Testing IMAP, SMTP and POP3 with GreenMail](#)
- [Run Your Tests With GreenMail - A simple Mail Server](#)
- [Integration Testing Mail function with Greenmail and Spock in Grails](#)
- [Unit testing Java mail code](#)
- [Using the GreenMail Java-based In-Memory MailServer in Integration Tests](#)
- [JavaEmail testing with GreenMail](#)

Projects using GreenMail include:


- [Alfresco](#)
- [Mule ESB](#)
- ... and many more. [Tell us](#) about your project :-)

Related projects to [JavaMail](#) and testing/development:

- [Dumbster](#), a fake SMTP server
- [Wiser](#) alias Subethasmtp, another test SMTP server
- [Apache James](#), a full blown enterprise mail server
- [DevNullSmtplib](#)
- [Mock-javamail](#)
- [JavaMail Mock2](#)
- [Collection of IMAP related RFCs](#)

Thanks

Special thanks to all contributors for feedback and [patches](#)!

Many thanks to  and [JetBrains](#) for supporting this project with free OSS licenses!

Examples

The source code of the examples can be found in the test package of GreenMail ([greenmail-core/src/test/java/com/icegreen/greenmail/examples/](#)).

Using JUnit (Rule based setup)

```
/** See code on GitHub */
@Rule
public final GreenMailRule greenMail = new GreenMailRule(ServerSetupTest.ALL);

@Test
    public void testSomething() {
        GreenMailUtil.sendTextEmailTest("to@localhost.com", "from@localhost.com", "subject", "body");
        MimeMessage[] emails = greenMail.getReceivedMessages();
        assertEquals(1, emails.length);
        assertEquals("subject", emails[0].getSubject());
        assertEquals("body", GreenMailUtil.getBody(emails[0]));
        // ...
    }
```

Testing your sending code (simple)

```
/** See code on GitHub */
GreenMail greenMail = new GreenMail(); //uses test ports by default
greenMail.start();
GreenMailUtil.sendTextEmailTest("to@localhost.com", "from@localhost.com", "some subject",
                                "some body"); // --- Place your sending code here
assertEquals("some body", GreenMailUtil.getBody(greenMail.getReceivedMessages()[0]));
greenMail.stop();
```

Testing your sending code (advanced)

```
/** See code on GitHub */
GreenMail greenMail = new GreenMail(ServerSetupTest.ALL);
greenMail.start();

//Use random content to avoid potential residual lingering problems
final String subject = GreenMailUtil.random();
final String body = GreenMailUtil.random();

sendTestMails(subject, body); // --- Place your sending code here

//wait for max 5s for 1 email to arrive
//waitForIncomingEmail() is useful if you're sending stuff asynchronously in a separate thread
assertTrue(greenMail.waitForIncomingEmail(5000, 2));

//Retrieve using GreenMail API
Message[] messages = greenMail.getReceivedMessages();
assertEquals(2, messages.length);

// Simple message
assertEquals(subject, messages[0].getSubject());
assertEquals(body, GreenMailUtil.getBody(messages[0]).trim());

//if you send content as a 2 part multipart...
assertTrue(messages[1].getContent() instanceof MimeMultipart);
MimeMultipart mp = (MimeMultipart) messages[1].getContent();
assertEquals(2, mp.getCount());
assertEquals("body1", GreenMailUtil.getBody(mp.getBodyPart(0)).trim());
assertEquals("body2", GreenMailUtil.getBody(mp.getBodyPart(1)).trim());

greenMail.stop();
```

Testing your retrieving code

```
/** See code on GitHub */
//Start all email servers using non-default ports.
GreenMail greenMail = new GreenMail(ServerSetupTest.ALL);
greenMail.start();

//Use random content to avoid potential residual lingering problems
final String subject = GreenMailUtil.random();
final String body = GreenMailUtil.random();
MimeMessage message = createMimeMessage(subject, body, greenMail); // Construct message
GreenMailUser user = greenMail.setUser("wael@localhost.com", "waelc", "soooosecret");
user.deliver(message);
assertEquals(1, greenMail.getReceivedMessages().length);

// --- Place your retrieve code here
greenMail.stop();
```

Testing using plain JavaMail for sending/retrieving code



```
/** See code on GitHub */
@Rule
public final GreenMailRule greenMail = new GreenMailRule(ServerSetupTest.SMTP_IMAP);

@Test
    public void testSendAndReceive() throws MessagingException, ... {
        Session smtpSession = greenMail.getSmtp().createSession();

        Message msg = new MimeMessage(smtpSession);
        msg.setFrom(new InternetAddress("foo@example.com"));
        msg.addRecipient(Message.RecipientType.TO,
            new InternetAddress("bar@example.com"));
        msg.setSubject("Email sent to GreenMail via plain JavaMail");
        msg.setText("Fetch me via IMAP");
        Transport.send(msg);

        // Create user, as connect verifies pwd
        greenMail.setUser("bar@example.com", "bar@example.com", "secret-pwd");

        // Alternative 1: Create session and store or ...
        Session imapSession = greenMail.getImap().createSession();
        Store store = imapSession.getStore("imap");
        store.connect("bar@example.com", "secret-pwd");
        Folder inbox = store.getFolder("INBOX");
        inbox.open(Folder.READ_ONLY);
        Message msgReceived = inbox.getMessage(1);
        assertEquals(msg.getSubject(), msgReceived.getSubject());
        ...

        // Alternative 2: ... let GreenMail create and configure a store:
        IMAPStore imapStore = greenMail.getImap().createStore();
        imapStore.connect("bar@example.com", "secret-pwd");
        inbox = imapStore.getFolder("INBOX");
        inbox.open(Folder.READ_ONLY);
        msgReceived = inbox.getMessage(1);
        ...

        // Alternative 3: ... directly fetch sent message using GreenMail API
        assertEquals(1, greenMail.getReceivedMessagesForDomain("bar@example.com").length);
        msgReceived = greenMail.getReceivedMessagesForDomain("bar@example.com")[0];
        ...
    }
```

Deployment

Run GreenMail as standalone java application

Running GreenMail as a standalone process is useful if you want a simple sand boxed mail server.

All you require is a compatible JRE and the standalone JAR. The standalone JAR already contains all dependencies and a default logging. You can configure GreenMail by setting system properties

Starting GreenMail standalone

```
java [OPTIONS] -jar greenmail-standalone.jar
```

Option	Description
--------	-------------



Option	Description
-Dgreenmail.setup.all	<p>Uses ServerSetup.ALL configuration to start all default mail services using default ports:</p> <ul style="list-style-type: none">• SMTP : 25• SMTPS : 465• IMAP : 143• IMAPS : 993• POP3 : 110• POP3S : 995 <p><code>-Dgreenmail.setup.all</code></p>
-Dgreenmail.setup.test.all	<p>Uses ServerSetupTest.ALL configuration to start all default mail services using default ports including offset of 3000:</p> <ul style="list-style-type: none">• SMTP : 3025• SMTPS : 3465• IMAP : 3143• IMAPS : 3993• POP3 : 3110• POP3S : 3995 <p><code>-Dgreenmail.setup.test.all</code></p>
-Dgreenmail.PROTOCOL.hostname	<p>Configures the hostname (or IP bind address) and activates a server for given protocol. Protocol can be one of</p> <ul style="list-style-type: none">• smtp• smtps• imap• imaps• pop3• pop3s <p><code>-Dgreenmail.smtp.hostname=127.0.0.1 -Dgreenmail.smtp.port=3025</code></p> <div>Note: Requires <code>-Dgreenmail.PROTOCOL.port</code> option!</div>
-Dgreenmail.PROTOCOL.port	<p>Configures the port for given protocol. Protocol can be one of</p> <ul style="list-style-type: none">• smtp• smtps• imap• imaps• pop3• pop3s <p><code>-Dgreenmail.smtp.port=3025 -Dgreenmail.smtp.hostname=127.0.0.1</code></p> <div>Note: Requires <code>-Dgreenmail.PROTOCOL.hostname</code> option!</div>
-Dgreenmail.hostname	<p>Configures the default hostname or ip bind address.</p> <div>Note: Hostnames must be DNS resolvable!</div> <div>Note: Default hostname is 127.0.0.1</div> <p><code>-Dgreenmail.smtp.port=3025 -Dgreenmail.imap.port=3143 -Dgreenmail.hostname=0.0.0.0</code></p>



Option	Description
-Dgreenmail.users=user1[, ..., userN]	<p>Configures the user mail boxes including password. The list of users is comma separated.</p> <p>A single user is of format <i>logon:password[@domain]</i> , where the <i>domain</i> part is optional.</p> <div>Note: Domain must be DNS resolvable!</div> <p><code>-Dgreenmail.users=foo:pwd@bar.com,jabber:wocky@monster.local,foo1:bar</code></p>
-Dgreenmail.auth.disabled	<p>Disables user authentication check, so that any password works. Useful if you do not want to preconfigure user/passwords. GreenMail automatically creates non existent users. Example: <code>-Dgreenmail.auth.disabled</code></p>
-Dgreenmail.verbose	<p>Enables verbose mode. Useful if you want to see debug and protocol level output. Example: <code>-Dgreenmail.verbose</code></p>
-Dgreenmail.startup.timeout	<p>Overrides the default server startup timeout of 1000ms. Useful if you have a slow or overloaded environment. Example with 2s startup timeout: <code>-Dgreenmail.startup.timeout=2000</code></p>
-Dlog4j.configuration	<p>Configures log4j using given configuration file.</p> <p>By default, GreenMail standalone runner uses a provided log4j configuration packed into the standalone ueber JAR. This options allows you to override the default log4j configuration by providing your own log4j configuration file.</p> <p>Example: <code>-Dlog4j.configuration=file:///tmp/log4j.xml</code></p>

GreenMail standalone configuration options

GreenMail standalone examples

Test setup for SMTP/IMAP and one user

Starts GreenMail for SMTP (test port 3025) and IMAP (test port 3143) using localhost/127.0.0.1 and a single user *test1* with password *pwd1* and email *test1@localhost* :

```
java -Dgreenmail.setup.test.smtp -Dgreenmail.setup.test.imap \  
-Dgreenmail.users=test1:pwd1 -jar greenmail-standalone.jar
```

Test setup for SMTP(S)/IMAP(S)/POP3(S) and two user

Starts GreenMail for SMTP (test port 3025), SMTPS (test port 3465), IMAP (test port 3143), IMAPS (test port 3993), POP3 (test port 3110) and POP3S (test port 3995) using localhost/127.0.0.1 :

```
java -Dgreenmail.setup.test.all -Dgreenmail.users=test1:pwd1,test2:pwd2@example.com \  
-jar greenmail-standalone.jar
```

Test setup for SMTP(S)/IMAP(S)/POP3(S), one user and bound to 0.0.0.0

Starts GreenMail for SMTP (test port 3025), SMTPS (test port 3465), IMAP (test port 3143), IMAPS (test port 3993), POP3 (test port 3110) and POP3S (test port 3995) using 0.0.0.0 :

```
java -Dgreenmail.setup.test.all -Dgreenmail.users=test1:pwd1 \  
-Dgreenmail.hostname=0.0.0.0 \  
-jar greenmail-standalone.jar
```

SMTP only

Starts GreenMail with directly configured SMTP on 127.0.0.1:4025

```
java -Dgreenmail.smtp.hostname=127.0.0.1 -Dgreenmail.smtp.port=4025 \  
-jar greenmail-standalone.jar
```

Fork me on GitHub

Deploy as a standalone Docker image

GreenMail provides a pre-configured Docker image running [GreenMail standalone](#). The image is available via [Docker Hub Public Repository](#), but you can also build it yourself using our [Dockerfile](#).

By default GreenMail standalone runs using the GreenMail test setup, using the following ports (default ports plus offset of 3000):

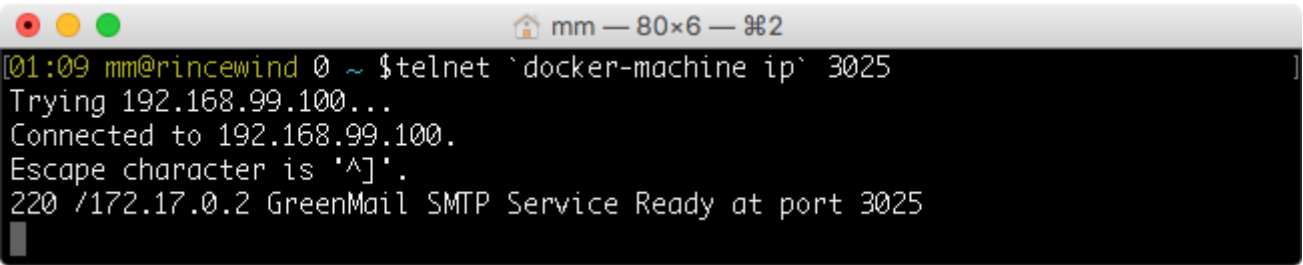
Port	Description
3025	SMTP
3110	POP3
3143	IMAP
3465	SMTPS
3993	IMAPS
3995	POP3S

Using docker to run the image

```
docker pull greenmail/standalone:1.5.10  
docker run -t -i -p 3025:3025 -p 3110:3110 -p 3143:3143 \  
-p 3465:3465 -p 3993:3993 -p 3995:3995 \  
greenmail/standalone:1.5.10
```

You might want to modify your mapped ports

If you want to test it, you can do a quick check with telnet (or any other mail program) by connecting to the exposed SMTP port and checking if GreenMail SMTP server responds:



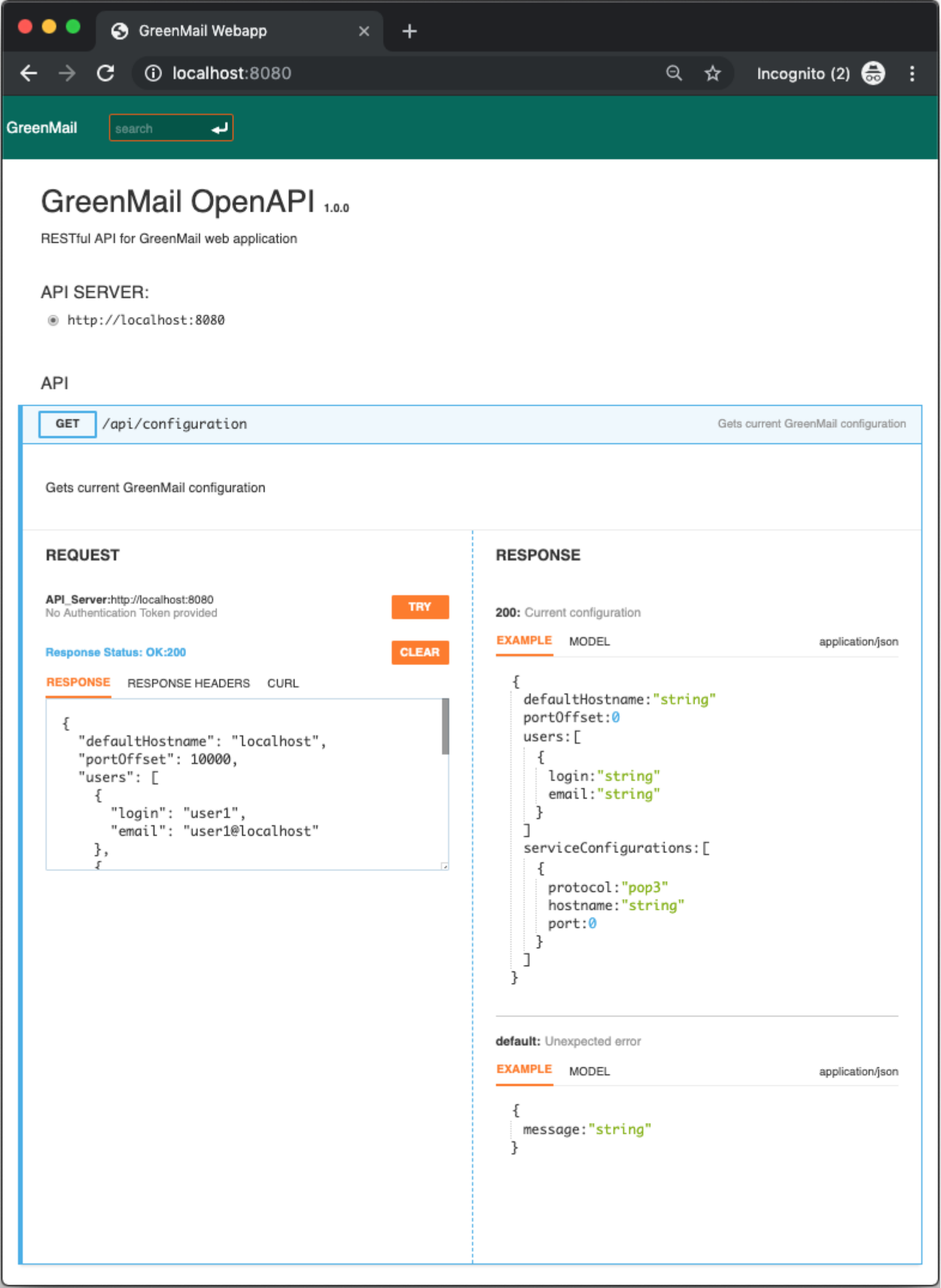
Passing configuration options to GreenMail Standalone Docker image

You can also configure the JVM and GreenMail configuration options via the Docker environment variables `JAVA_OPTS` and `GREENMAIL_OPTS`:

```
docker run -t -i \  
-e GREENMAIL_OPTS='-Dgreenmail.setup.test.all -Dgreenmail.hostname=0.0.0.0 -Dgreenmail.auth.dis' \  
-e JAVA_OPTS='-Djava.net.preferIPv4Stack=true -Xmx512m' \  
-p 3025:3025 -p 3110:3110 -p 3143:3143 \  
-p 3465:3465 -p 3993:3993 -p 3995:3995 \  
greenmail/standalone:1.5.4
```

Deploy as a webapp (WAR)

GreenMail Webapp provides a lightweight Java web application wrapping GreenMail mail server and exposing GreenMail API.



Like the [GreenMail JBoss Service](#), the usage scenario is a development or test environment where a real mail server is too much overhead. The webapp is application server neutral - you should be able to use it on any Java EE 7 application server (Servlet 3.1) running und Java 7 (or greater).

With the GreenMail mail service each developer has its own local mail server sandbox - so there's no danger for accidentally leaking test mails into the Internet.

Deploy the webapp

Simply deploy the webapp like any other Java web application.

For Tomcat, just drop the webapp into \$CATALINA_HOME/webapps directory. Alternatively, you can create a \$CATALINA_HOME/webapps/greenmail, unpack the WAR here and configure the GreenMail service.

Configure the webapp

You can configure the active mail services and available users by editing WEB-INF/web.xml in the WAR file and modifying the context params. A ServletContextListener starts and stops the GreenMail service when deploying/undeploying.



Name	Description
greenmail.defaultHostname	The mail server default hostname (defaults to localhost).
greenmail.portOffset	Offset added to the standard mail ports. Default is 10000 (so an activated SMTP service would start up on port 10025).
greenmail.<PROTOCOL>	Starts a server for this mail protocol (using defaultHostname, port offset and default port). Available protocol names include <ul style="list-style-type: none">• smtp• smtps (SMTP via SSL)• pop3• pop3s (POP3 via SSL)• imap• imaps (IMAP via SSL)
greenmail.<PROTOCOL>.host	Optionally overwrites the default host name (localhost).
greenmail.<PROTOCOL>.port	Optionally overwrites the default port and port offset for the given protocol.
greenmail.users	A whitespace/newline/comma separated list of mail users in the form of USER:PWD@DOMAIN.

Context Parameters

Have a look at the default [web.xml](#).

Deploy as a JBoss Service (SAR)

GreenMail provides a JBoss Service for easy JBoss integration. The usage scenario is a development or test environment where a real mail server is too much overhead. With the GreenMail mail service each developer has its own local mail server sandbox - so there's no danger for accidentally leaking test mails into the Internet.

GreenMail SAR works only up to JBoss AS 6!

JBoss introduced a not backward compatible change to the SAR MBean deployment in JBoss AS 7+. As alternative, try out the [GreenMail WAR deployment](#).

Deploy the service

1. [Download](#) the current service which we distribute as a JBoss SAR (Service Archive).
2. Unjar the service to your server deploy directory.
This example uses the default server, where the service is deployed in exploded mode for follow up configuration.

```
cd ${JBOSS_HOME}/server/default/deploy
mkdir greenmail-jboss-service.sar
cd greenmail-jboss-service.sar
jar xvf /path/to/greenmail-jboss-service-XXX.jar
```

Starting JBoss will now automatically also start the GreenMail JBoss Service. You can see a startup message on the console output, including basic configuration data like active protocols and their ports:



```
greenmail — vagrant@precise64: ~/jboss — 84x10 — %2
22:38:26,051 INFO [GreenMailService] Starting com.icegreen.greenmail:type=service,n
ame=GreenMail,version=1.4.1-SNAPSHOT
22:38:26,062 INFO [GreenMailService] GreenMail configuration: hostname=127.0.0.1,
, smtp=3025 pop3=3110 imap=3143
22:38:26,884 INFO [ServiceEndpointManager] WebServices: jbossws-1.0.3.SP1 (date=200
609291417)
22:38:27,386 INFO [Embedded] Catalina naming disabled
22:38:27,411 INFO [ClusterRuleSetFactory] Unable to find a cluster rule set in the
classpath. Will load the default rule set.
22:38:27,412 INFO [ClusterRuleSetFactory] Unable to find a cluster rule set in the
```

Opening the JBoss Console, you can see the GreenMail Service JMX Bean:

JBoss JMX Management C x

localhost:8080/jmx-console/

JMX Agent View precise64

ObjectName Filter (e.g. "jboss:*", "*:service=invoker,*") : ApplyFilter

Catalina

- [type=Server](#)
- [type=StringCache](#)

JMImplementation

- [name=Default,service=LoaderRepository](#)
- [type=MBeanRegistry](#)
- [type=MBeanServerDelegate](#)

com.icegreen.greenmail

- [loader=greenmail-jboss-service](#)
- [name=GreenMail,type=service,version=1.4.1-SNAPSHOT](#)

jboss

- [database=localDB,service=Hypersonic](#)

You find two entries, of which the second one holds our interest:

- loader=greenmail-jboss-service : A loader for isolated class loading inside the JBoss service (as to prevent the greenmail JavaMail to conflict with the application server version).
- name=GreenMail,type=service,version=... : The GreenMail service bean for managing GreenMail

Configurable user accounts

The GreenMail service is configured by the file [META-INF/jboss-service.xml](#) in the exploded service directory.

Send a mail message via JMX

1. Open the JBoss JMX Console in your browser (see the deployment picture).
2. Select the JMX bean of name GreenMail and type service
3. Invoke the sendMail operation:

void sendMail()

Injects a mail into GreenMail

Param	ParamType	ParamValue	ParamDescription
theTo	java.lang.String	:mas-test-user2@localhost	The TO field
theFrom	java.lang.String	cmas-test-user1@localhos	The FROM field
theSubject	java.lang.String	subject with umlauts: öäü	The SUBJECT field
theBody	java.lang.String	content with umlauts: öÄÜ	The mail content

Invoke

The mail should now be visible by listing the user mails, if there is no automatic polling of the mail box.

Send a mail message using your mail client

You can use your favorite mail client if you configure it for GreenMail by setting the outgoing mail to the GreenMail SMTP port. By default this is localhost:3025. Of course you can also read created mail by configuring your mail client to use GreenMail POP3 (localhost:3110) or IMAP (localhost:3143). You will probably have to adjust these settings for your specific configuration.

List mails for a user via JMX

Instead of using your favourite mail client, you can use the JMX GreenMail service for listing all current mails in of a user in the GreenMail server sandbox.

1. Open your browser with the JMX Console and choose the GreenMail service (see sending mail paragraph)
2. Enter the user email for the listMailsForUserHTML operation:

java.lang.String listMailsForUserHTML()
Lists all mails for an email (HTML list formatted)


Param	ParamType	ParamValue	ParamDescription
email	java.lang.String	cmas-test-user2@localhos	A valid email

Invoke

3. Click the invoke button. The browser now shows all emails for the specified user:

Operation Results

localhost:8080/jmx-console/HtmlAdaptor

**JMX MBean Operation Result**
listMailsForUserHTML()

[Back to Agent View](#) [Back to MBean View](#) [Reinvoke MBean Operation](#)

1 Mails for cmas-test-user2@localhost

From	Subject	Received date	Content
[cmas-test-user1@localhost]	subject with umlauts: öÜÄ	Tue Dec 23 21:21:53 UTC 2014	content with umlauts: öÜÄ

Note that displaying mails does not modify the user mail box.

FAQ

How come I don't have to create any accounts to send/retrieve?

By default GreenMail accepts all incoming emails. If there is no corresponding existing email account, one is automatically created with login and password being the same as the to-address.

What other library dependencies are there?

Check out the [Maven POM](#). Dependencies include

- javamail.jar and activation.jar
- slf4j-api.jar (for logging)
- junit.jar (for test rules)

How come I don't need to install any SSL/TLS related certificates?

GreenMail is designed to be used out of the box with no need to generate, sign or install any certificates into your keystore. GreenMail ships with a builtin keystore with a self signed RSA key. Refer to the source code of [DummySSLServerSocketFactory](#) for details.

How can I use IMAP quotas?

GreenMail supports IMAP quota. Quota capability can be toggled on and off. For details see [ImapServerTest.java](#) quota test

How can I create or delete a mail user?

```
GreenMail greenMail = ...
// Create user with login id equals email
GreenMailUser user1 = greenMail.setUser("foo@localhost", "some secret pwd");
// Create user with login id different than email
GreenMailUser user2 = greenMail.setUser("foo@localhost", "login-id", "some secret pwd");
...
greenMail.getManagers().getUserManager().deleteUser(user1); // Delete user
```



How can I get more verbose output?

Enable the verbose mode, which prints debug output and protocol level communication.
See [ServerSetup.setVerbose\(boolean\)](#) or GreenMail standalone runner option [greenmail.verbose](#) option.

Example of activating verbose mode when configuring via GreenMailRule:

```
@Rule
public final GreenMailRule greenMail = new GreenMailRule(ServerSetup.verbose(ServerSetupTest.SMTP_IMAP));
```

Download

Maven Central contains all [GreenMail artifacts](#).
Docker images are availalbe on [Docker Hub GreenMail Repository](#), such as for the GreenMail standalone runner.

1.5.11 - Oct 22, 2019

This version requires JDK 1.7+ and JavaMail 1.5+ .

Name	GAV
GreenMail Core	com.icegreen:greenmail:1.5.11
GreenMail Standalone	com.icegreen:greenmail-standalone:1.5.11 (Docker Hub)
GreenMail Webapp	com.icegreen:greenmail-webapp:1.5.11:war
GreenMail JBoss Service	com.icegreen:greenmail-jboss-service:1.5.11:jboss-sar
GreenMail Spring	com.icegreen:greenmail-spring:1.5.11

Available via Maven repository or as a [ZIP from GitHub](#).

1.5.10 - Mar 24, 2019

This version requires JDK 1.7+ and JavaMail 1.5+ .

Name	GAV
GreenMail Core	com.icegreen:greenmail:1.5.10
GreenMail Standalone	com.icegreen:greenmail-standalone:1.5.10 (Docker Hub)
GreenMail Webapp	com.icegreen:greenmail-webapp:1.5.10:war
GreenMail JBoss Service	com.icegreen:greenmail-jboss-service:1.5.10:jboss-sar
GreenMail Spring	com.icegreen:greenmail-spring:1.5.10

Available via Maven repository or as a [ZIP from GitHub](#).



1.5.9 - Nov 28, 2018

This version requires JDK 1.7+ and JavaMail 1.5+ .

Name	GAV
GreenMail Core	com.icegreen:greenmail:1.5.9
GreenMail Standalone	com.icegreen:greenmail-standalone:1.5.9 (Docker Hub)
GreenMail Webapp	com.icegreen:greenmail-webapp:1.5.9:war
GreenMail JBoss Service	com.icegreen:greenmail-jboss-service:1.5.9:jboss-sar
GreenMail Spring	com.icegreen:greenmail-spring:1.5.9

Available via Maven repository or as a [ZIP from GitHub](#).

1.5.8 - Aug 26, 2018

This version requires JDK 1.7+ and JavaMail 1.5+ .

Name	GAV
GreenMail Core	com.icegreen:greenmail:1.5.8
GreenMail Standalone	com.icegreen:greenmail-standalone:1.5.8 (Docker Hub)
GreenMail Webapp	com.icegreen:greenmail-webapp:1.5.8:war
GreenMail JBoss Service	com.icegreen:greenmail-jboss-service:1.5.8:jboss-sar
GreenMail Spring	com.icegreen:greenmail-spring:1.5.8

Available via Maven repository or as a [ZIP from GitHub](#).

1.5.7 - Feb 09, 2018

This version requires JDK 1.7+ and JavaMail 1.5+ .

Name	GAV
GreenMail Core	com.icegreen:greenmail:1.5.7
GreenMail Standalone	com.icegreen:greenmail-standalone:1.5.7 (Docker Hub)
GreenMail Webapp	com.icegreen:greenmail-webapp:1.5.7:war
GreenMail JBoss Service	com.icegreen:greenmail-jboss-service:1.5.7:jboss-sar
GreenMail Spring	com.icegreen:greenmail-spring:1.5.7

Available via Maven repository or as a [ZIP from GitHub](#).

1.5.6 - Nov 23th, 2017



This version requires JDK 1.7+ and JavaMail 1.5+ .

Name	GAV
GreenMail Core	com.icegreen:greenmail:1.5.6
GreenMail Standalone	com.icegreen:greenmail-standalone:1.5.6 (Docker Hub)
GreenMail Webapp	com.icegreen:greenmail-webapp:1.5.6:war
GreenMail JBoss Service	com.icegreen:greenmail-jboss-service:1.5.6:jboss-sar
GreenMail Spring	com.icegreen:greenmail-spring:1.5.6

Available via Maven repository or as a [ZIP from GitHub](#).

1.5.5 - May 14th, 2017

This version requires JDK 1.7+ and JavaMail 1.5+ .

Name	GAV
GreenMail Core	com.icegreen:greenmail:1.5.5
GreenMail Standalone	com.icegreen:greenmail-standalone:1.5.5 (Docker Hub)
GreenMail Webapp	com.icegreen:greenmail-webapp:1.5.5:war
GreenMail JBoss Service	com.icegreen:greenmail-jboss-service:1.5.5:jboss-sar
GreenMail Spring	com.icegreen:greenmail-spring:1.5.5

Available via Maven repository or as a [ZIP from GitHub](#).

1.5.4 - April 17th, 2017

This version requires JDK 1.7+ and JavaMail 1.5+ .

Name	GAV
GreenMail Core	com.icegreen:greenmail:1.5.4
GreenMail Standalone	com.icegreen:greenmail-standalone:1.5.4 (Docker Hub)
GreenMail Webapp	com.icegreen:greenmail-webapp:1.5.4:war
GreenMail JBoss Service	com.icegreen:greenmail-jboss-service:1.5.4:jboss-sar
GreenMail Spring	com.icegreen:greenmail-spring:1.5.4

Available via Maven repository or as a [ZIP from GitHub](#).

1.5.3 - Januar 15th, 2017

This version requires JDK 1.7+ and JavaMail 1.5+ .



Name	GAV
GreenMail Core	com.icegreen:greenmail:1.5.3
GreenMail Standalone	com.icegreen:greenmail-standalone:1.5.3 (Docker Hub)
GreenMail Webapp	com.icegreen:greenmail-webapp:1.5.3:war
GreenMail JBoss Service	com.icegreen:greenmail-jboss-service:1.5.3:jboss-sar
GreenMail Spring	com.icegreen:greenmail-spring:1.5.3

Available via Maven repository or as a [ZIP from GitHub](#).

1.5.2 - September 29th, 2016

This version requires JDK 1.7+ and JavaMail 1.5+ .

Name	GAV
GreenMail Core	com.icegreen:greenmail:1.5.2
GreenMail Standalone	com.icegreen:greenmail-standalone:1.5.2 (Docker Hub)
GreenMail Webapp	com.icegreen:greenmail-webapp:1.5.2:war
GreenMail JBoss Service	com.icegreen:greenmail-jboss-service:1.5.2:jboss-sar
GreenMail Spring	com.icegreen:greenmail-spring:1.5.2

Available via Maven repository or as a [ZIP from GitHub](#).

1.5.1 - July 10th, 2016

This version requires JDK 1.7+ and JavaMail 1.5+ .

Name	GAV
GreenMail Core	com.icegreen:greenmail:1.5.1
GreenMail Standalone	com.icegreen:greenmail-standalone:1.5.1 (Docker Hub)
GreenMail Webapp	com.icegreen:greenmail-webapp:1.5.1:war
GreenMail JBoss Service	com.icegreen:greenmail-jboss-service:1.5.1:jboss-sar
GreenMail Spring	com.icegreen:greenmail-spring:1.5.1

Available via Maven repository or as a [ZIP from GitHub](#).

1.5.0 - March 19th, 2016

This version requires JDK 1.7+ and JavaMail 1.5+ .

Name	GAV
GreenMail Core	com.icegreen:greenmail:1.5.0



Name	GAV
GreenMail Standalone	com.icegreen:greenmail-standalone:1.5.0 (Docker Hub)
GreenMail Webapp	com.icegreen:greenmail-webapp:1.5.0:war
GreenMail JBoss Service	com.icegreen:greenmail-jboss-service:1.5.0:jboss-sar
GreenMail Spring	com.icegreen:greenmail-spring:1.5.0

Available via Maven repository or as a [ZIP from GitHub](#).

1.4.1 - April 27th, 2015

This version requires JDK 1.6+ and JavaMail 1.5+ .

Name	GAV
GreenMail Core	com.icegreen:greenmail:1.4.1
GreenMail Webapp	com.icegreen:greenmail-webapp:1.4.1:war
GreenMail JBoss Service	com.icegreen:greenmail-jboss-service:1.4.1:jboss-sar
GreenMail Spring	com.icegreen:greenmail-spring:1.4.1

Download bundle as [ZIP from GitHub](#).

1.4.0 - October 29th, 2014

This version requires JDK 1.6+ and JavaMail 1.5+ .

Name	GAV
GreenMail Core	com.icegreen:greenmail:1.4.0
GreenMail Webapp	com.icegreen:greenmail-webapp:1.4.0:war
GreenMail JBoss Service	com.icegreen:greenmail-jboss-service:1.4.0:jboss-sar
GreenMail Spring	com.icegreen:greenmail-spring:1.4.0

Download bundle as [ZIP from GitHub](#).

1.3.1b - June 14th 2009

This version runs on JDK 1.4+ and JavaMail 1.4 .

Name	GAV
GreenMail Core	com.icegreen:greenmail:1.3.1b
GreenMail JBoss Service	com.icegreen:greenmail-jboss-service:1.3.1b:jboss-sar

Older versions

You can find older version at the old [SourceForge download](#) section