**Compiling Autofetch from Source**

The 1.0 beta release is subversion revision number 10. You should be able to compile the source code using Ant 1.6.x by invoking the dist target. This willl generate the autofetch JAR files in the **dist** subdirectory.

**Downloading Autofetch**

You can download Autofetch [here](https://web.archive.org/web/20080313234108/https:/sourceforge.net/projects/autofetch/). Both the autofetch-[version].jar and autofetch-[connector]-[version].jar are required. The connector is the persistence framework you will use Autofetch with. Right now, there is only a Hibernate connector.

**Using Autofetch with Hibernate**

1. Autofetch has been tested with Hibernate 3.2.5 and likely is compatible with 3.2.x
2. Put the autofetch-[version].jar and autofetch-hibernate-[version].jar files in the classpath of your application. Autofetch does use some external libraries, however, they are a subset of the libraries required by Hibernate. The Autofetch and Hibernate JARs should be loaded with the same classloader.
3. Change how you construct your Hibernate session factory to something like this:
4. Configuration cfg = new AutofetchConfiguration().configure();
5. SessionFactory sf = cfg.buildSessionFactory();
6. Optionally: Wrap instances of Criteria with AutofetchCriteria if you would like Autofetch to automatically add fetch joins to them.
7. Criteria crit = new AutofetchCriteria(sess.createCriteria(Foo.class));
8. Optionally: Remove prefetch specifications from your criteria mappings and make all associations lazy in your hibernate configuration files. The benefit of using Autofetch is that it will take care of producing the right prefetch specifications for you.

**Autofetch Caveats**

Autofetch is not perfect. It may not be able to figure out the best prefetch specifications either because its query cost model is incorrect, or the program context it computes is imprecise. Hopefully this does not happen too often. Please post in the forum any problems you have and the developers will try to help.

**Hibernate Caveats**

1. Autofetch can only monitor traversals through getters and collection methods. Do not use Autofetch if you are using field access for associations you would like to prefetch.
2. Autofetch does not remove the need to keep a session open until all traversals are performed. The standard hibernate solutions apply for traversals far away from the query invocation such as the open session in view pattern.
3. Autofetch does not add prefetch specifications to HQL queries. Hopefully this will be a future improvement.
4. Autofetch monitors traversals by proxying all entities. This does not cause any problems with polymorphism because it only proxies entities for which it knows the concrete class. This means all your entities need default constructors, otherwise they will not be monitored and their associations will not be prefetched. In addition, the Hibernate#getClass method might not return the result you expect.
5. Autofetch does not currently handle map associations.
6. The proxies Autofetch uses for entities cannot currently be serialized. This should be fixed in the next point release.
7. Autofetch extends Hibernate using event listeners and interceptors. If you wish to use your own custom session interceptor please wrap it in an org.autofetch.hibernate.AutofetchInterceptor which will delegate to your interceptor. See the Javadoc API for more details. If you would like to extend the default load and initialize collection interceptors please subclass the appropriate Autofetch listeners instead of the default Hibernate listeners. If you would like to add listeners then make sure the Autofetch listeners are also included.
8. Autofetch may not respect static prefetch specifications in your configuration files for entity loads, however, it will always respect prefetch specifications in your criteria queries.

[Back to Autofetch home page](https://web.archive.org/web/20080313234108/http:/www.cs.utexas.edu:80/~aibrahim/autofetch/index.html)