[Difference between Eager Initialization and Lazy Initialization](http://way2java.com/hibernate/difference-between-eager-initialization-and-lazy-initialization/" \o "Difference between Eager Initialization and Lazy Initialization)

To increase the performance while retrieving records from a database table, **Hibernate** uses two styles – **Eager Initialization** and **Lazy Initialization**. Concept-wise, a good programming practice is lazy initialization; but both have their own importance in coding.

**Lazy initialization (or instantiation)**

There are two types of initializations in a programming language – **eager** initialization and **lazy** initialization. In eager initialization, the object is created in the program execution, in a normal way, which sometimes the programmer may not use it in the program. This looks waste of memory and processor time. Other way is to create the object when the programmer really requires. This is called lazy initialization (one of the design patterns). That is, **if the object is created at runtime, only when the programmer requires is known as lazy initialization.**So, which is preferred, definitely you say lazy initialization. Lazy initialization is basically meant for improving the performance by avoiding unnecessary computation (load on the microprocessor) and reducing memory requirements.

**Some examples:**

**1.** Suppose you have **Customer** object having a big array of orders to display. To display the list to the customer it is very expensive as it requires **lot of hits** to database. It the **Customer** object is initialized at the beginning of the program which later found that Customer never asks to display his orders. Now, as you can guess, **it is the time for lazy initialization**.

**2.** Another scenario is when an object is very expensive (takes long time to create) to create, if you would like to defer (delay) its creation due to other more expensive operations takes place. Say, your program creates many object instances when it starts, but a few are required immediately. Now **lazy initialization**is the time to think to increase the startup performance of the program by delaying the initialization after the required and preferred objects have been created. Let us take a small code:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | **public** **class** Demo  {  **private** Player newPlayer = **new** Player(); *// object is created well in advance*    **public** Player getPlayer()  {  **return** newPlayer;  }  **public** **static** **void** main(String args[])  {  Demo d1 = **new** Demo();  Player p1 = d1.getPlayer();  }  } |

When you see this above code, after knowing what is **lazy initialization**, definitely you say it is meaningless, as the **newPlayer** object is created at the startup of the program itself which may not be required later (wasting its creation time).

The above code is modified as follows for lazy initialization.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | **public** **class** Demo  {  **private** Player newPlayer;    **public** Player getPlayer()  {  **if**(newPlayer == **null**)  {  newPlayer = **new** Player(); *// object is created when asked*  }  **return** newPlayer;  }  **public** **static** **void** main(String args[])  {  Demo d1 = **new** Demo();  Player p1 = d1.getPlayer();  }  } |

In the above code, the creation of **newPlayer** object is delayed making sure we do not create it until it is required.

Knowledge of the initialization differences is useful in understanding [get() vs load()](http://way2java.com/hibernate/difference-bewteen-session-get-and-session-load-in-hibernate/) and [list() vs iterate()](http://way2java.com/hibernate/difference-between-query-list-and-query-iterate-hibernate/).

- See more at: http://way2java.com/hibernate/difference-between-eager-initialization-and-lazy-initialization/#sthash.pPg9wqhq.dpuf