

# Performance

## General Performance Tuning

For an overview, please see the slides of the performance session at [JavaOne : PerformanceBOFatJavaOne](#).

One performance factor is the [MyFacesExtensionsFilter](#). The [MyFacesExtensionsFilter](#) buffers and parses the response on every request. You can disable this, and still gain all functionality the [ExtensionsFilter](#) is providing, by doing two things:

- 1.) Set the context-param

```
<context-param>
    <param-name>org.apache.myfaces.ADD_RESOURCE_CLASS</param-name>
    <param-value>org.apache.myfaces.component.html.util.StreamingAddResource</param-value>
</context-param>
```

for more info, see [StreamingAddResource](#).

- 2.) Get rid of the <HEAD> tag in your HTML, and instead use Tomahawk's <t:documentHead/> tag. Of course, your <f:view/> tag has to enclose your <t:documentHead/> tag for this to work.

## Server Side State Performance Tuning

There are a many settings you can enable in your web.xml file to make [MyFaces](#) perform well.

Server side state saving performs better:

```
<context-param>
    <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
    <param-value>server</param-value>
</context-param>
```

Next step - disable compression of state in server:

```
<context-param>
    <param-name>org.apache.myfaces.COMPRESS_STATE_IN_SESSION</param-name>
    <param-value>false</param-value>
</context-param>
```

Very important, too, is to disable the serialization of state, serialization and deserialization of the component tree is a major performance hit.

```
<context-param>
    <param-name>org.apache.myfaces.SERIALIZE_STATE_IN_SESSION</param-name>
    <param-value>false</param-value>
</context-param>
```

If you find that memory is a constraining factor, then reducing the number of views stored in the session might help. The setting is controlled by:

```
<context-param>
    <param-name>org.apache.myfaces.NUMBER_OF_VIEWS_IN_SESSION</param-name>
    <param-value>20</param-value>
    <description>Only applicable if state saving method is "server" (= default).
        Defines the amount (default = 20) of the latest views are stored in session.
    </description>
</context-param>
```

## Client Side State Performance Tuning

[permalink](#)

The JSF spec mandates a large amount of serialization during the lifecycle of each request. This of course cannot be avoided when using client side state saving. [MyFaces](#) gives the application developer the opportunity to specify the serialization mechanism for an application. Here are instructions on how one would realize the performance increases available via [JBoss serialization](#).

Place the following context parameter in web.xml :

```
<context-param>
    <param-name>org.apache.myfaces.SERIAL_FACTORY</param-name>
    <param-value>org.apache.myfaces.JbossSerialFactory</param-value>
</context-param>
```

Compile the following classes, or an equivalent of each, and place them in the classpath.

```
package org.apache.myfaces;

import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import org.apache.myfaces.shared_impl.util.serial.SerialFactory;
import org.jboss.serial.io.JBossObjectOutputStream;

public class JbossSerialFactory implements SerialFactory {

    public ObjectOutputStream getObjectOutputStream(OutputStream stream) throws IOException {
        return new JBossObjectOutputStream(stream);
    }

    public ObjectInputStream getObjectInputStream(InputStream stream) throws IOException {
        return new MyFacesJBossObjectInputStream(stream);
    }

}
```

```
package org.apache.myfaces;

import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectStreamClass;
import org.apache.myfaces.shared_impl.util.ClassUtils;
import org.jboss.serial.io.JBossObjectInputStream;

public class MyFacesJBossObjectInputStream extends JBossObjectInputStream {

    public MyFacesJBossObjectInputStream(InputStream stream) throws IOException {
        super(stream);
    }

    protected Class resolveClass(ObjectStreamClass desc) throws ClassNotFoundException, IOException{
        try{
            return ClassUtils.classForName(desc.getName());
        }
        catch (ClassNotFoundException e){
            return super.resolveClass(desc);
        }
    }
}
```

Place [jboss-serialization.jar](#) and [trove.jar](#) in your classpath.