

---

# Bimbo

The Bimbo Tutorial

Frank Cornelis

1.0

Copyright © 2008 Frank Cornelis

1. Introduction .....	1
2. Getting Started .....	2
3. Tables .....	7
4. Error Handling .....	10
5. Custom Rendering .....	12
6. Style .....	12
7. Authentication .....	12
8. Getting Involved .....	12

## 1. Introduction

Bimbo is a Java web framework to construct web applications using only annotated POJO's. First of all Bimbo is meant as web application framework to construct functionality centered applications. The idea behind it is that a developer should (initially) not bother about look and feel of an application, but to let the framework handle all these details. The look and feel is less important but should be allowed to be tweaked later on by some web designer.

If you're looking for some fancy looking web framework, or want to have full control over your layout, you'll probably be better of using frameworks like JSF, Facelets, JBoss RichFaces and JBoss Seam.



### Bimbo Online

The Bimbo project site is located at: [Bimbo Project Site](http://sourceforge.net/projects/bimbo/)  
[<http://sourceforge.net/projects/bimbo/>]

Important concepts of the Bimbo framework are:

- Ease of use within a servlet container. Thus the required `web.xml` configuration should be limited.
- No more `xhtml/jsp/jsf/facelets` pages, all we want is an annotated POJO to get us started.
- A web designer should never be able to break the work of a web application Java developer since the latter is in general more expensive.

- The framework should have inherent notion of user login/logout.
- Testability of the framework together with the resulting web applications is very important.
- Easy integration with JavaEE applications (EJB3 session bean injection)
- Lightweight deployment (i.e. outside of a full-blown JavaEE container) within for example Tomcat or Jetty.
- Lightweight integration of other frameworks (JPA entity manager injection).



### Why is it called Bimbo?

We called the project Bimbo since it makes web application development as easy and sleazy as it possibly gets.

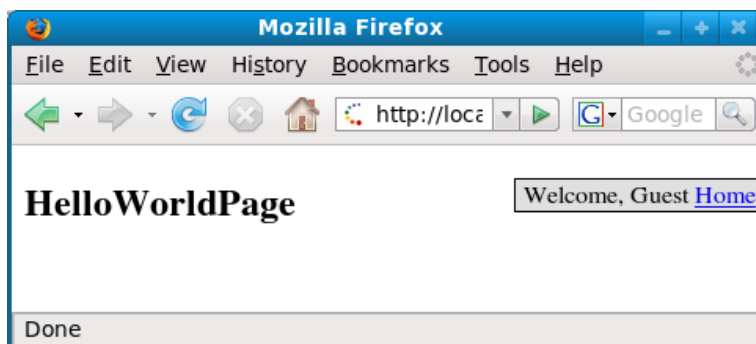
## 2. Getting Started

Let's start with the most simple Bimbo web application possible. *Example 1, "Hello World"*

```
package net.sf.bimbo.demo.tutorial;  
  
public class HelloWorldPage {  
  
}
```

### Example 1. Hello World

When interpreted by the Bimbo framework this will give us the following result page within a web browser *Figure 1, "Hello World"*.



**Figure 1. Hello World**

As you can see Bimbo makes web application development very easy. All you need to get started is an empty class from where on you can build further.

Since the Bimbo framework requires a servlet container you need to package your web application as a WAR. Your `WEB-INF/web.xml` file should contain the following configuration:

```
<servlet>
  <servlet-name>BimboServlet</servlet-name>
  <servlet-class>net.sf.bimbo.BimboServlet</servlet-class>
  <init-param>
    <param-name>WelcomePage</param-name>

    <param-value>net.sf.bimbo.demo.tutorial.HelloWorldPage</param-
value>
  </init-param>

</servlet>

<servlet-mapping>
  <servlet-name>BimboServlet</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>BimboServlet</servlet-name>
  <url-pattern>*.bimbo</url-pattern>
</servlet-mapping>
```

Basically you just need to configure the `WelcomePage` initial parameter of the Bimbo runtime servlet to point to your welcome page class. Your page class files should live under the `WEB-INF/classes/` directory.



### Important

The Bimbo runtime JAR should be present under the `WEB-INF/lib/` directory.

Now that we've got our very basic web application up and running, let's enhance it a little bit. First of all we want to add some human-readable page title.

```
package net.sf.bimbo.demo.tutorial;

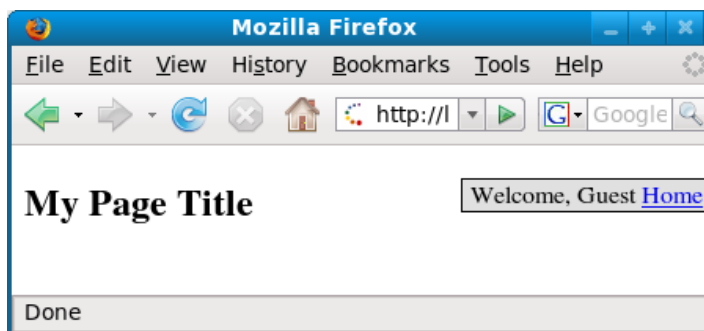
import net.sf.bimbo.Title;

@Title("My Page Title")
public class TitlePage {

}
```

### Example 2. A Page Title

This will give us the following page within a web browser:



**Figure 2. A Page Title**

Basically everything in a Bimbo web application is driven by Java annotations.

Now let's do some basic input/output magic.

```
package net.sf.bimbo.demo.tutorial;

import net.sf.bimbo.Action;
import net.sf.bimbo.Input;
import net.sf.bimbo.Title;

@Title("Input Page")
public class InputPage {

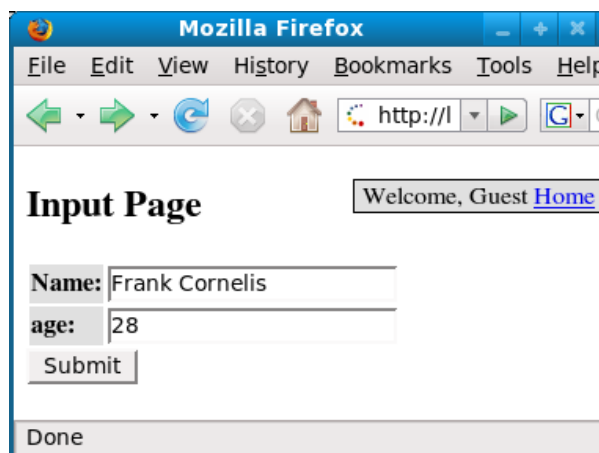
    @Input("Name")
    private String name;

    @Input
    private int age;

    @Action("Submit")
    public OutputPage action() {
        return new OutputPage("Hello " + this.name + ", age " +
            this.age);
    }
}
```

### Example 3. An Input Page

This gives us the following web page:



**Figure 3. An Input Page**

When you click on the `Submit` button the `action` method will be invoked on an `InputPage` object, which will result in the construction of an `OutputPage` object as listed below.



## Note

An action method can return a result page or `null` which means that the action method took care of the response rendering itself. A page action can render the response itself via an `@Resource` annotated field of type `HttpServletRequest`.

```
package net.sf.bimbo.demo.tutorial;

import net.sf.bimbo.Output;
import net.sf.bimbo.Title;

@Resource
@Title("Output Page")
public class OutputPage {

    @Output
    private String output;

    public OutputPage() {
        super();
    }

    public OutputPage(String message) {
        this.output = message;
    }
}
```

## Example 4. An Output Page

For which the resulting screen looks as follows:



Figure 4. An Output Page

That's it! If that isn't easy I don't know.



### Tip

You can always navigate back to the welcome page by clicking the `Home` link in the upper-right box.

## 3. Tables

The Bimbo framework provides a very intuitive model for visualizing and manipulating tables.

Suppose that we defined a table entry as shown in [Example 5, “The Table Entry”](#).

```
package net.sf.bimbo.demo.tutorial;

import net.sf.bimbo.Output;

public class TableEntry {

    @Output("Name")
    private String name;

    @Output("Description")
    private String description;

    public TableEntry() {
        super();
    }

    public TableEntry(String name, String description) {
        super();
        this.name = name;
        this.description = description;
    }
}
```

### Example 5. The Table Entry

Now we can create a page that uses this `TableEntry` to construct a table. This page is defined as shown in [Example 6, “The Table Page”](#).

```
package net.sf.bimbo.demo.tutorial;

import java.util.LinkedList;
import java.util.List;

import javax.annotation.PostConstruct;

import net.sf.bimbo.Action;
import net.sf.bimbo.Output;
import net.sf.bimbo.Title;

@Title("Table")
public class TablePage {

    @Output
    private List<TableEntry> table;

    @PostConstruct
    public void postConstruct() {
        this.table = new LinkedList<TableEntry>();
        this.table.add(new TableEntry("Bimbo", "Java Web Framework"));
        this.table.add(new TableEntry("Java", "Programming Language"));
    }

    @Action("View")
    public ViewTableEntryPage view(TableEntry entry) {
        return new ViewTableEntryPage(entry);
    }
}
```

## Example 6. The Table Page

This gives us the following result within the web browser:

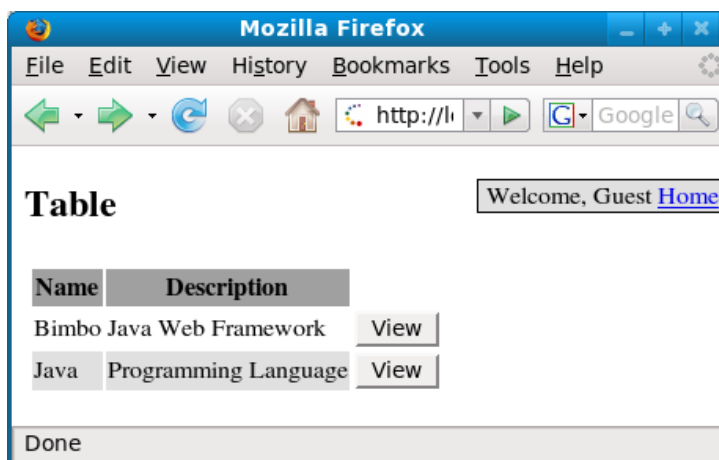
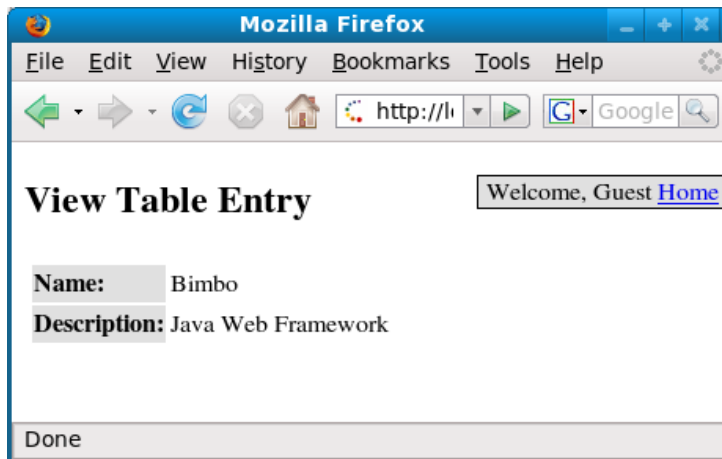


Figure 5. A Table Page



As you can see the `view @Action` method has been interpreted by the Bimbo framework as being a table action instead of a regular page action. This is because the `view` action method has a parameter of the same `TableEntry` type as the table.

When you click on the first `view` button you'll get to see the following page:



**Figure 6. Viewing a table entry**

For which the code is listed in [Example 7, "The View Table Entry Page"](#).

```
package net.sf.bimbo.demo.tutorial;

import net.sf.bimbo.Output;
import net.sf.bimbo.Title;

@Title("View Table Entry")
public class ViewTableEntryPage {

    @Output
    private TableEntry entry;

    public ViewTableEntryPage() {
        super();
    }

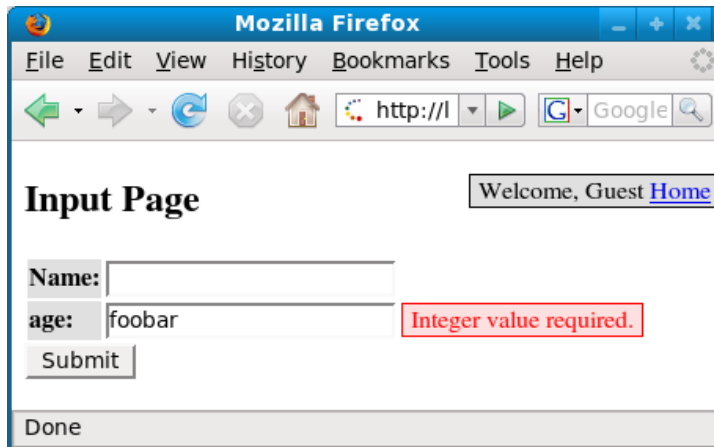
    public ViewTableEntryPage(TableEntry entry) {
        this.entry = entry;
    }
}
```

**Example 7. The View Table Entry Page**

## 4. Error Handling

**Bimbo** has inherent support for web application error handling. In this section we will elaborate on the different types of error handling that are supported by the **Bimbo** framework.

First of all the **Bimbo** framework supports input validation error handling as shown in the following screenshot *Figure 7, "Input Error Handling"*.



**Figure 7. Input Error Handling**

**Bimbo** also supports action method error handling. If an action method throws an exception, the **Bimbo** framework will catch this exception and convert it into an error message.

For example when clicking the `Submit` action button of *Example 8, "An Action Error Page"*.

```
package net.sf.bimbo.demo.tutorial;

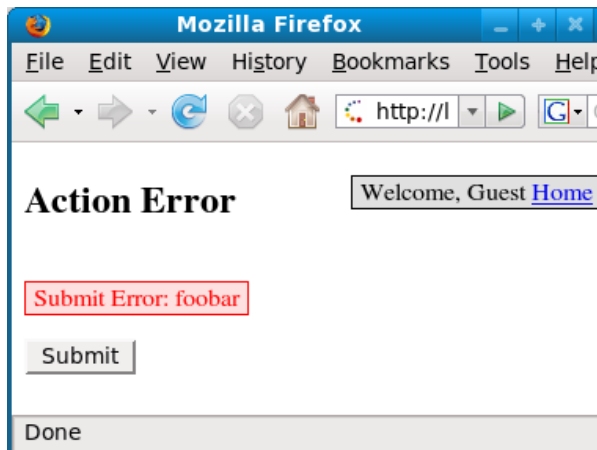
import net.sf.bimbo.Action;
import net.sf.bimbo.Title;

@Title("Action Error")
public class ActionErrorPage {

    @Action("Submit")
    public ActionErrorPage action() {
        throw new RuntimeException("foobar");
    }
}
```

**Example 8. An Action Error Page**

You'll get an error message as shown in [Figure 8, "Action Error Handling"](#).



**Figure 8. Action Error Handling**

Another very interesting feature of the **Bimbo** framework is the possibility to blame an `@Input` field for (business) exceptions thrown by page actions. This feature has been shown in [Example 9, "A Page with Input Field Error Blaming"](#).

```
package net.sf.bimbo.demo.tutorial;

import java.io.IOException;

import net.sf.bimbo.Action;
import net.sf.bimbo.BlameMe;
import net.sf.bimbo.Input;
import net.sf.bimbo.Title;

@Title("Blame Me")
public class BlameMePage {

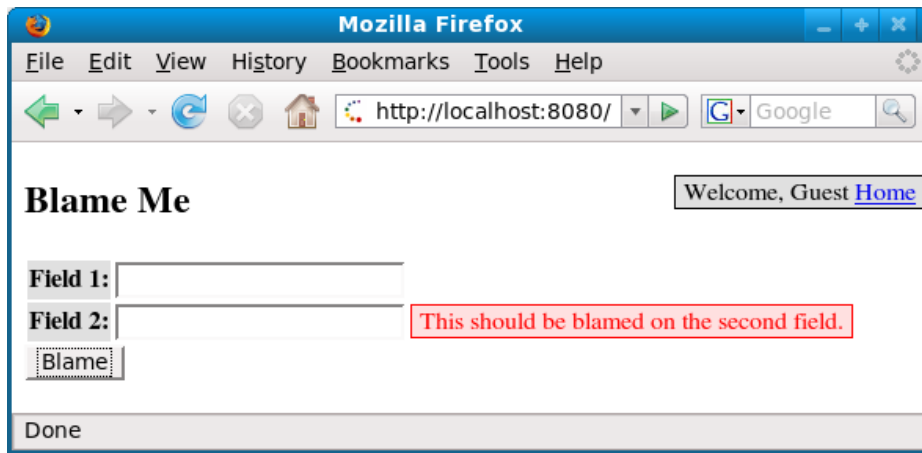
    @Input("Field 1")
    private String field1;

    @Input("Field 2")
    @BlameMe(IOException.class)
    private String field2;

    @Action("Blame")
    public BlameMePage action() throws IOException {
        throw new IOException("This should be blamed on the second
        field.");
    }
}
```

**Example 9. A Page with Input Field Error Blaming**

Which yields the following *Figure 9, "A Page with Input Field Error Blaming"* within a web browser.



**Figure 9. A Page with Input Field Error Blaming**

## 5. Custom Rendering

TODO

## 6. Style

TODO


## 7. Authentication

TODO

## 8. Getting Involved

The **Bimbo** project has been released under the Apache Software License, Version 2.0.

Given the open source license we invite other developers to join in on the development of the Bimbo web application framework.



### Bimbo Project Site

The Bimbo project is hosted at [SourceForge](http://www.sourceforge.net/projects/bimbo/) [http://www.sourceforge.net/projects/bimbo/].

To get you started asap checkout the source code using **subversion** .



```
svn co https://bimbo.svn.sourceforge.net/svnroot/bimbo/trunk  
bimbo
```

The `README.txt` file should be a good entry point.

