

**Tntnet users guide 1.8,, ccccc s**

**Table of contents**

Concept.....3

Installing Tntnet.....3

Create a sime tntnet-application .....

**Concept**



```

name;
street;
city = "New York";
int age;          // content of text-input is converted to int
int sport[];      // multiple checkboxes with the same name on my form
</%args>
<# use the vector like this: #>
% for (sport_type::const_iterator it = sport.begin(); it !=
sport.end(); ++it) {
<$ *it $>
% }

```

## Components

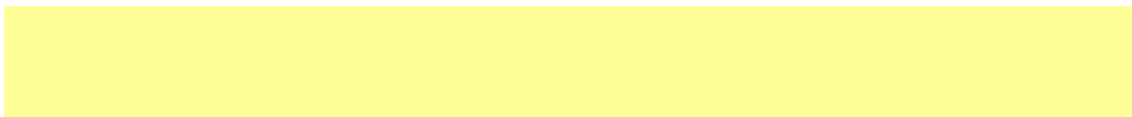
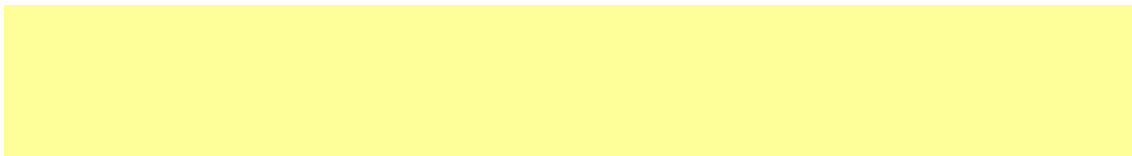
Ec++-pages are called **components**. They are identified by their name. The name is composed of the class-name and the library-name divided with '@'.

Components, which are called by Tntnet are called top-level-components.

Top-level-components return the http-response-code. If a explicit return statement is not specified the constant HTTP\_OK is returned. Constants, which define http-response-codes are defined in the file `http.h`.

Basically there are several possibilities to call a component.

In the simplest case the component-~~ame~~





```
}>  
<& greeting name=(nameValue) lastname=(lastnameValue)  
repeat=(repeatNum)>
```

```
<%def greeting>  
<%at%name;  
lastname;  
name;  
unc6âme0h%! i ... f}õ +Væ{â
```

```
õ ÷VæAt=(repea  
õ ,Qp'ßQ lassG
```



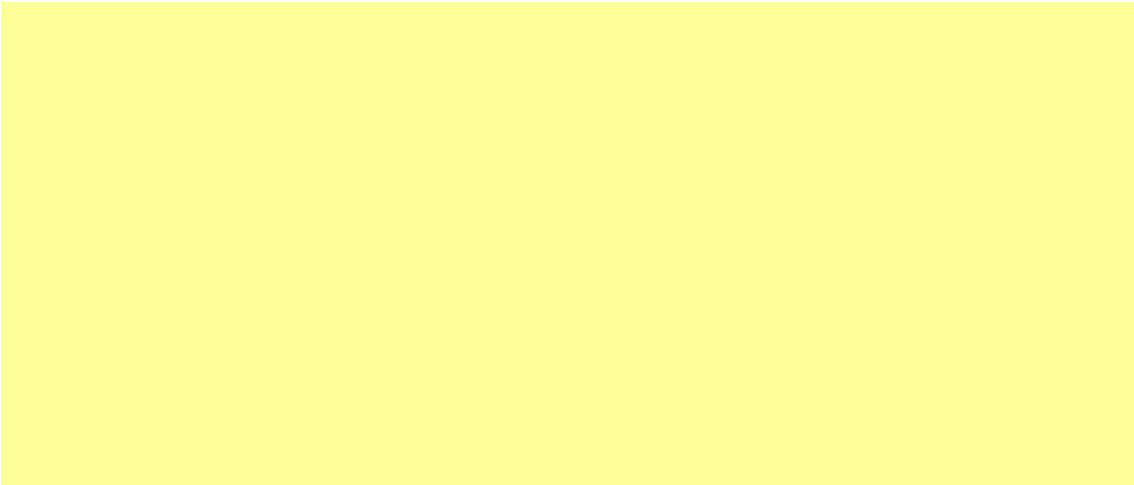


```
std::string currentUser;  
</%session>
```

## Request-variables



variable tntnet answers every request with 404 – HTTP\_NOT\_FOUND. MapUrl” takes at least 100ms



passed to a outputstream and you





