



KNX@HOME

JSP AND AJAX BASED VISUALIZATION OF BUILDING CONTROLS AND
VIEWS FOR EIB/KNX

Supervisor:

Prof. Dr.-Ing. Andreas Grzempa

Authors:

Richard Wurm
Andreas Reichert
Johannes Bayer

Based on a diploma thesis of Stephan Wuchterl

Contact:

iks@fh-deggendorf.de

University of Applied Sciences Deggendorf
Centre for innovative communication systems
Edlmairstr. 6+8 / E207
D-94469 Deggendorf

Get more information on:

<http://www.hto.fh-deggendorf.de/komm/>



Table of Contents

1.	Overview.....	3
2.	Basic conditions for the live CD.....	3
3.	The Teach-In-Module	4
4.	The Communication	5
a.	Persistor	5
b.	Calimero	5
5.	The web server	6
a.	Used Technologies.....	6
b.	Decision for HTML, JSP and Struts	6
c.	JSP with the Struts Framework.....	7
d.	Working modes of the application	7
e.	Ajax	8
f.	Security.....	8
6.	Images and resources	8

1. Overview

The main goal of the application is to allow the user to control his KNX-BUS installation with his home desktop computer. This goal is to be achieved by using the KNXLive! from the Vienna University of Technology, which is an operating system that starts directly from CD. There is also a Teach-In-Module that allows the user to simply add elements from the KNX-Bus-System to the KNX@Home-Application.

The application should:

- Be able to control a KNX installation of arbitrary size
- Be Controlled and presented web based
- Be flexible and adapted to the users' needs
- provide full functionality to the user for manual configuration
- be simple

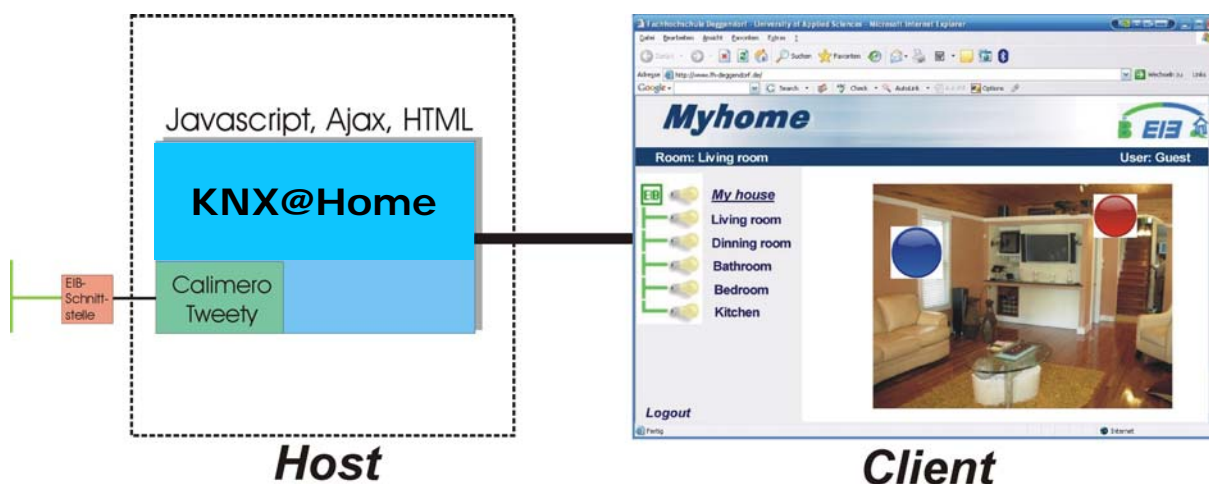


Image 1: Overview of the application

2. Basic conditions for the live CD

The whole applications (Calimero, Persister, web server) should be integrated into a GNU Linux Live CD, like Knoppix.

Because of this, the programs have to get by without a hard disc and with the RAM of an average computer. In fact this is not a big problem, except the database, which requires the access on a hard disc, USB-stick, SD-card, etc.

Several Live distributions can handle an available empty partition on a data medium to store the database.

Once the Live CD has started a complete Linux with a graphical user interface like KDE, the user will be able to use the main application immediately by opening a browser like Konqueror or Firefox. As home page, the browser will show the welcome screen of the web interface.

Due to this, it will be very easy for the user to launch our application-bundle, without installing any programs permanently to his system.

3. The Teach-In-Module

The "Teach-In-Module" should handle the easy way of teaching in devices of a building. Additionally it should get information of available devices through the logged bus-messages.

It will be made like an installation-routine, completely menu-guided. The teaching of a switch happens for example by pressing three times a button. The Teach-In-Module will recognize this action and will show the group address and the possible type (i.e. DPT, Switch, etc.). The user can choose the adequate type and store it as a data point. Thus, it will be stored in the database and for now on, it will be available for the web front end.

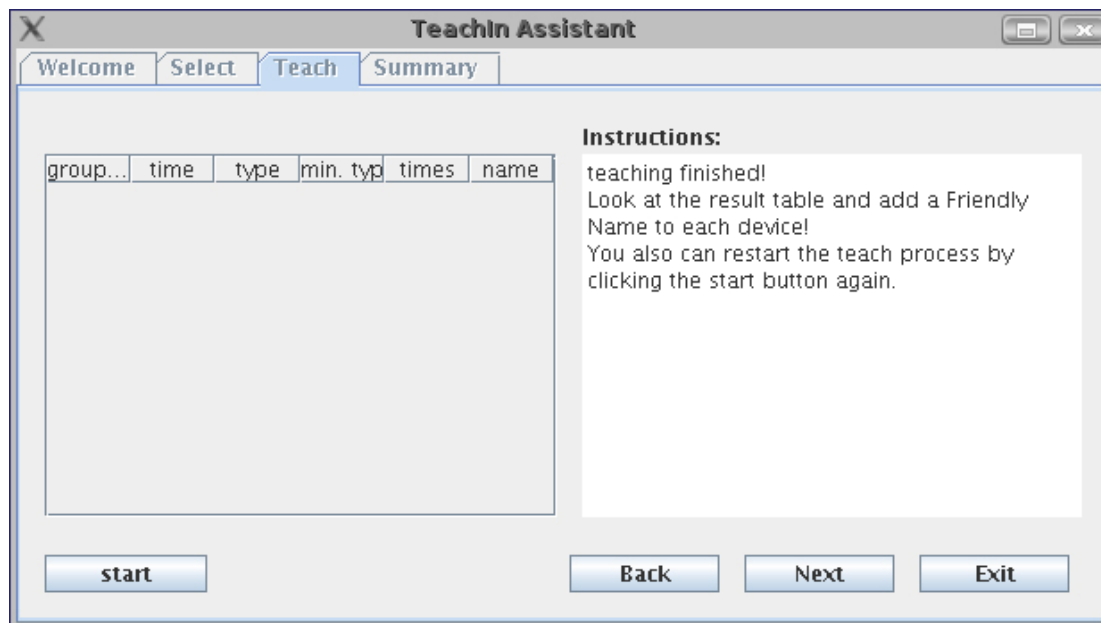


Image 2: The Teach-In user interface

The action of teaching in a new device can be divided into four steps:

1. The **welcome** window

When you start the Teach-In process, this screen welcomes the user and introduces him to the further steps.

2. The **select** window

In the dialog of the select screen, the user will be asked, which device he would like to teach in. The user will be able to choose a device out of a list.

3. The **teach** window

This screen will tell the user, what to do. For example the user has to push five times the light button, if he wants to teach in the group address of the lamp. The Teach-In module will watch the bus monitor especially for a group address, which appears five times in a predetermined period of time. Is a group address with this attributes found, it will be displayed to the user. Every possible data point will be listed and the user can choose the right one.

4. The **summary** window

This window shows the learned data point and will ask the user for affirmation. If the user confirms, the data point will be stored in the database.

4. The Communication

a. Persister

The module – so called „Persister“ – controls the access to the complete database. Thereby the module offers functions to read and write through Java’s interprocess communication interface (RMI). Thus Calimero and the web-front-end can access the persistence-layer.

The Persister consists in the main part of the Hibernate-framework, which provides a persistence-layer itself. For this, the database access can happen completely object-oriented.

Persister starts the RMI registry, the RMI server and provides hereupon all the methods, which are relevant for the persistence.

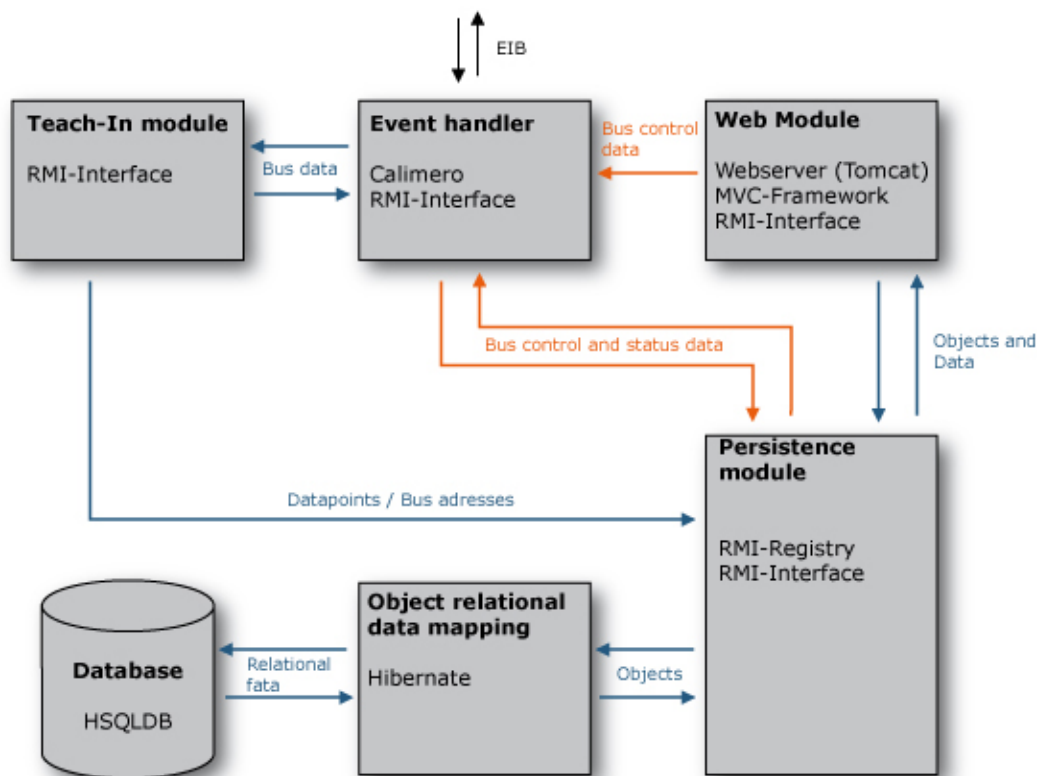


Image 3: The modules in action

b. Calimero

Calimero establishes the connection to an EIBNet/IP device and acts as an event-handler for the web application.

The Calimero API was developed at the TU Wien (<http://www.tuwien.at/>) and students of the University Of Applied Sciences in Deggendorf have adjusted it for their needs.

Functions:

- establishing a connection to the EIBNet/IP device, connection into both directions, read and write
- "monitor-mode", every event on the bus is logged and stored in a database
- event-handler for the web application, i.e. activate switches via web interface,
- remote access to the Teach-In module

Through RMI Calimero can access the “Persister”, from where it achieves the database. The other way round you can control Calimero over the RMI, to switch the light on or off for example.

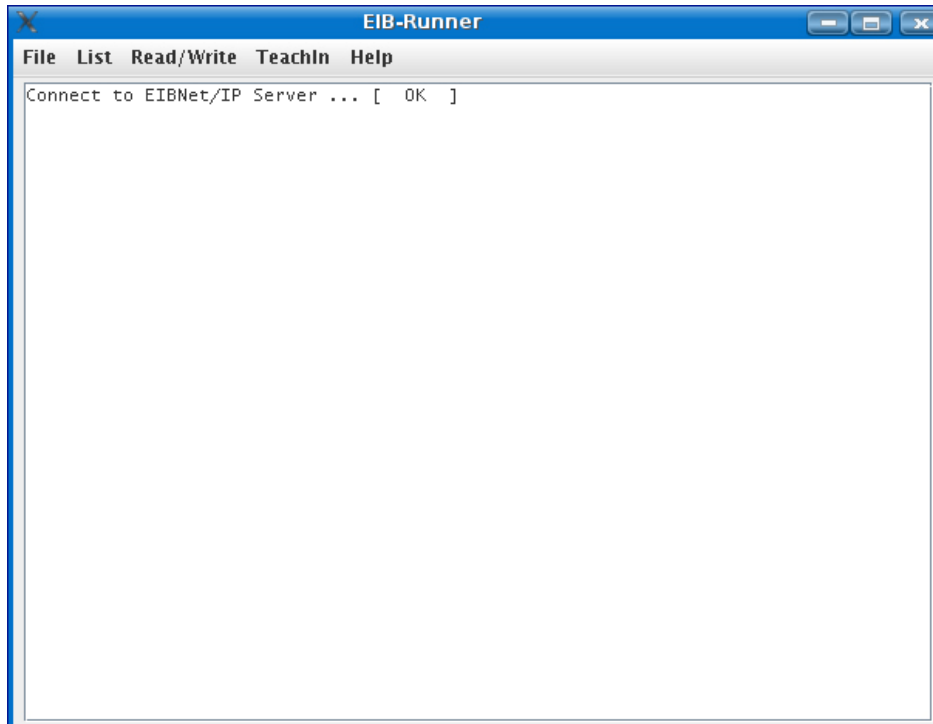


Image 3: Screenshot of the adjusted Calimero called “EIB-Runner”

5. The web server

a. Used Technologies

We use several modern technologies for the application:

- The Apache Tomcat Server as servlet container
- JavaServerPages as the basic concept
- HSQLDB with Hibernate a modern Java database
- The Apache STRUTS MVC-Framework for development and security features
- AJAX as live-communication layer between the client and the server

b. Decision for HTML, JSP and Struts

There are numerous different ways to perform a live-communication - for example the communication could be established by Flash Remote - but working with JSP and Struts has many advantages: the only thing necessary is a browser that supports HTML, CSS and JavaScript. You don't need any special Plug-Ins or even ActiveX components.

A larger independence, maintainability, and sustainability can be provided for by using standardized components and protocols, which would not be possible if using proprietary software.

Because the Calimero application from the TU Wien is Java based it is better to also use a Java based data base (HSQL) Java for the programming of the server.

The MVC framework Struts has been selected to provide security and an easy maintenance of the application.

c. JSP with the Struts Framework

JSP stands for Java Server Pages and provides an easy and fast programming of web applications that are independent from servers and platforms. Java code is implemented into already existing HTML by using JSP standard tags or XML-like tags. This Java code makes it possible to generate dynamic contents and therefore establish the business logic independently from the design of the website.

JSP files are transformed into Java servlets through a JSP compiler and the created servlet is transformed again by the Java compiler into byte code all of which is carried out by the servlet container.

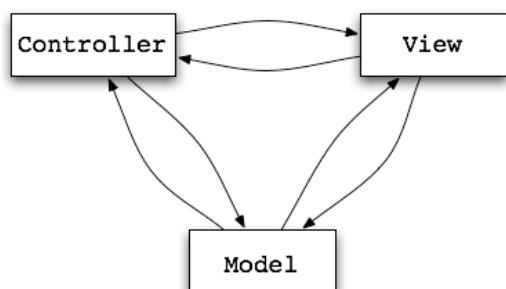


Image 4: The MVC pattern

Struts is an open source framework and the basis for creating web applications according to the MVC (Model View Controller) pattern. It uses open standards and already established design patterns: JSP, servlets, JavaBeans, resource bundles and XML files.

Struts connects all these Java technologies in the background in order to allow

professional development of web applications.

It's common to think of an application as having three main layers: presentation (UI), domain, and data access. In MVC, the presentation layer is split into controller and view:

- the model contains the data used for the presentation
- the view is responsible for viewing the data from the model
- the controller makes the connection between model and view and receives the user requests

d. Working modes of the application

The application consists of two different working modes, the administration mode and the controlling mode.

In the administration mode you can set the basic options of the system as e.g. user roles, users, data point types and group addresses.

Additionally within this working mode the physical and the logical structure of the building (the data points and control elements) is established. For example a ground plan of the building can be implemented and equipped with KNX symbols.

The second working mode controls the KNX bus and visualizes the current status of the building.

e. Ajax

Ajax is a relatively new technology based on standard components (HTML, CSS, JavaScript), that enables the client (browser) to reload and change single parts of an HTML site without having to build up the whole site again.

The intent is to make web pages feel more responsive by exchanging small amounts of data with the server behind the scenes, so that the entire web page does not have to be reloaded each time the user makes a change. This is meant to increase the web page's interactivity, speed, and usability.

In our case the constantly changing status of the KNX bus in the building can be displayed almost in real time.

Therefore it is e.g. possible that the use of a switch within the building and its effects are displayed on the site in an extremely short time, by changing the corresponding symbol. It functions through a timer on the client site (website in the browser) that requests the current status of the bus from the data base every second.

f. Security

Both, the visualization as well as the control of the KNX bus are very sensitive areas that mustn't be accessible openly. That's why it is necessary to implement a user authentication as well as a comprehensive user right management into the system. For that the MVC framework Struts is quite suitable and provides a solid code basis.

Later one can add additional security measures by using SSL encryption with the help of the Apache web server.

6. Images and resources

Images taken from

- Wikipedia [Image 4]
- adaptivepath.com [Image 5]

More information:

- KNXLive!: <http://www.auto.tuwien.ac.at/Projects/hba/knxlive.html>
- JSP: <http://java.sun.com/products/jsp/>
- Struts: <http://struts.apache.org/>
- Ajax: http://en.wikipedia.org/wiki/Ajax_%28programming%29

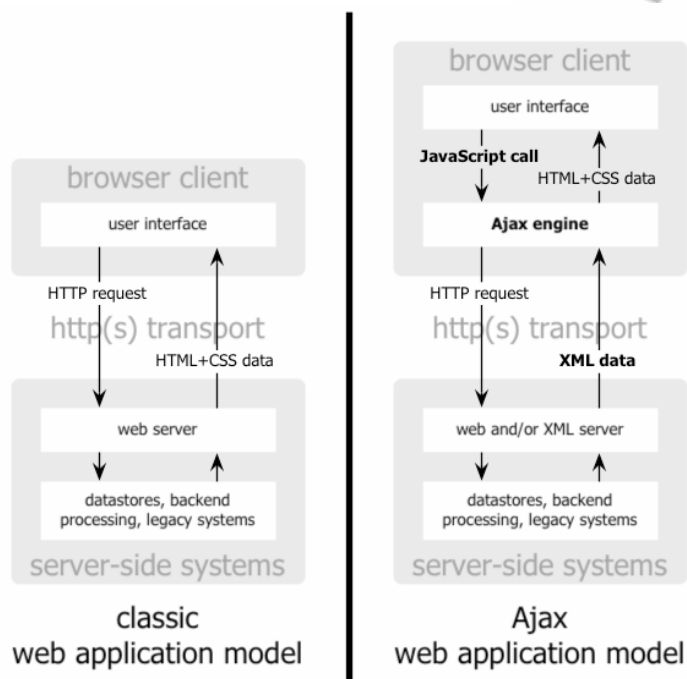


Image 5: Difference between Ajax and classical web development