Base Distribution - Work Package # 45584

Status:	New	**Priority:**	Should have
Author:	Sebastian Kurfuerst	**Category:**	
Created:	2013-02-18	**Assigned To:**	
Updated:	2014-01-10	**Due date:**	
Subject:	Access Control for TYPO3CR Nodes (Concept)		

Description

Access Control on TYPO3CR Nodes

This concept discusses how to do access control on nodes. It has been created by Andi Förthner, Sebastian Kurfürst, and Helmut Hummel during T3BOARD13.

Security Framework in a nutshell

In this section, we quickly describe how authorization in the security framework works with **Policies**,
to again have a common basis for later.

Access is granted or denied for **Resources**, according to the following rules:

1. not-explicitely-specified resources are implicitely ALLOWed.
2. (roughly)if AT LEAST ONE of the current roles defines a GRANT on a resource, access is GRANTED. The full rules follow below:
    1. DENY always wins, so if ONE role is DENIED access, this will win.
    2. otherwise, if at least one role says GRANT, you are allowed to access the resource
3. if a resource is explicitely defined, but none of the current roles defines an ACL on it, this defaults to DENY ("Abstain Case")
4. if multiple resources match on a given code point, ALL resources must be evaluated to ALLOW (ANDed together)

Mapping to nodes

We want to apply as many as possible of the above concepts to **nodes** as well, meaning the following:

- we need to think about what a "resource" is for nodes
- we will use roles as usual
- we will use ACLs (== mapping between roles and resources) as usual

So we essentially need to solve the question: **What are resources on nodes, and how to define them?**

A **resource** on a set of nodes could be defined in various ways: For instance, we could explicitely enumerate the nodes
which are part of this resource. However, this is quite cumbersome, as this list would need to be manually maintained.

Instead, people often want to assign permissions based on certain properties of nodes, for example "I want to ALLOW
creation if TEXT elements" (here, the **node type** of the node should be matched on).

Because of that, we do **not** want to explicitely enumerate a resource based on nodes, but instead
want to **create a boolean function which decides whether a certain node is inside the resource or not**

For creating this boolean function, we want to use the **Eel language**, as it provides predefined syntax,
making it possible to focus on the semantics for us.

To sum it up: A **resource** on a set of nodes is an **Eel Expression** matching these nodes. Example:

```
    // Resource for all text nodes
        q(node).is('instanceof Text')
    // resource for all nodes of a site
        q(node).inSite('TYPO3.NeosDemo')
    // Resource for all nodes whose "text" property starts with "hi"
        // -> a little dangerous, but nevertheless possible
        q(node).is('text ~= Hi')
    // ALLOW creation of "Text" child node underneath three column node
        ( creation() || deletion() )
        && q(node).is('instanceof Text')
```

&& q(node).parent('insanceof ThreeColumn')

Properties of the node mapping

- the whole evaluation is order independent, meaning we do not need priorities, !important or anything like that

Related issues:

related to Base Distribution - Work Package # 45010: [WIP] Fine-Grained Acces...	**Closed**	**2013-01-31**
related to Base Distribution - Story # 55920: TYPO3CR ACL	**Closed**	**2014-02-12**

History

#1 - 2013-03-06 13:22 - Christopher Hlubek

I like the idea of having flexible expressions for node resources. But I cannot yet see a way to implement that efficiently. Especially for read access we need a way to map the resource expression to a query over nodes (well, unless FlowQuery is used exclusively and allows lazy evaluation, which limits the flexibility). Also if we define a lot of node resources (n) we have to check the authorization for a specific node action against the complete list, which will scale only with O(n).

One idea could be to cache the resource information on a node if no external context (user, time, server, etc.) is used in the expression (we can control that through Eel). So the expressions need to be re-evaluated if the node properties or structure changes like in the last example (which would need pretty complex cache invalidation logic).

A real question is how to do ACLs for read access on nodes efficiently. And what do we do if one node on a page is not accessible?